# Oracle® Cloud

# Using Visual Builder Studio

Release 25.04.0
G17612-01
January 2025

Oracle Cloud Using Visual Builder Studio, Release 25.04.0

G17612-01

Copyright © 2020, 2025, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

# Contents


## Preface

Audience xv
Documentation Accessibility xv
Diversity and Inclusion xviii
Related Resources xviii
Conventions xviii

## Part I Overview

### 1 The Basics

What Is Oracle Visual Builder Studio? 1-1
What Are the VB Studio Building Blocks? 1-1
What Are My Identity Roles? 1-3
How Do I Access VB Studio? 1-4
- Access VB Studio from the Oracle Cloud Home Page 1-6

Who Should Use This Guide? 1-7

### 2 Get Yourself Set Up

Update Your Display Name 2-1
Update Your Email Address 2-2
Add Your Avatar Picture 2-2
Set Your Preferred Color Theme 2-3
Configure Your Global Email Notifications 2-3
Set Up a Git Client 2-4
- Git Command-Line Interface 2-5
- Upload Your Public SSH Key 2-5
  - Generate an SSH Key 2-6
  - Add the Public SSH Key to Your VB Studio Account 2-6

Set Up Token-Based Authentication 2-7
See the News Banner 2-12

# Part II Common Tasks

## 3 Work with Projects


What Is a Project? 3-1
- What Are Project Memberships? 3-1
- Check Your Project Membership 3-2
- Request Membership in a Project You Can't Access 3-3
- What Can I Do at the Project Level? 3-5
- Permissions for the Project Home Page 3-8
- What Else Can I Do Within a Project? 3-11
  - Permissions for Working with Git Repositories 3-11
  - Permissions for Merge Requests 3-12
  - Permissions for Maven 3-13
  - Permissions for Docker 3-13
  - Permissions for Jobs, Builds, and Pipelines 3-14
  - Permissions for Releases 3-15
  - Permissions for Environments 3-15
  - Permissions for Workspaces 3-16
  - Permissions for Issues 3-16
  - Permissions for Agile Boards and Sprints 3-16
  - Permissions for Wikis 3-17
  - Permissions for Snippets 3-17
- Open a Project 3-18

Create a Project 3-20
- Empty Project 3-20
- With an Initial Git Repository 3-21
- From an Imported Project 3-22
- From a Project Template 3-24
- Without an Environment 3-27

Set Project Protection 3-28

Manage Project Users and Groups 3-29
- Manage Local VB Studio Groups 3-30
- Export the Users List 3-33

What Else Can You Do with Your Project? 3-35

Manage Projects 3-35
- Delete a Project 3-35
  - Delete a Project from the Project Administration Properties Page 3-36
  - Delete a Project from the Organization Page 3-36
- Export and Import Project Data 3-37
  - What Project Data Does VB Studio Automatically Export? 3-38

Export to and Import from an OCI Object Storage Bucket 3-39
View Export and Import History of the Project 3-44
Edit a Project’s Name, Description, or Visibility 3-44
See When a Project Was Created and Last Updated 3-45
Get a Project's Unique Identifier 3-45
Manage Announcements 3-46
Manage Project Tags 3-47
View Your Project’s Usage Metrics 3-47
Display RSS/ATOM Feeds 3-48
Configure Link Rules 3-49
Set Up Issue Products and Custom Fields 3-51
Create and Configure Issue Products 3-51
Create and Configure Custom Fields in an Issue 3-52
Configure Project Templates 3-53
Visibility, Rules, and Variables 3-54
Define and Manage a Project Template 3-55
Define Project Template Rules 3-56
Add and Manage Variables 3-58
Perform Build Administration Tasks 3-59
Manage Repositories 3-59

4 Manage Source Code Files with Git

Git Concepts and Terms 4-1
Migrate to Git 4-3
Set Up a Git Repository 4-4
Create a Git Repository 4-4
Best Practices for Storing Large Files and Binaries 4-4
Default Repository Size Limits in VB Studio 4-5
Enable and Use Git LFS to Version Large Files 4-5
Create an Empty Git Repository 4-6
Import an External Git Repository 4-7
Mirror an External Git Repository 4-7
Upload Files From Your Computer to the Project’s Git Repository 4-8
Push a Local Git Repository to the Project’s Git Repository 4-9
Access a Git Repository Using SSH 4-10
Access a Git Repository Using Token-Based Authentication 4-11
Add and Manage a Git Repository's Files 4-12
Manage Files from the Git Page 4-12
Use Git Commands to Manage Files 4-14
Associate a VB Studio Issue with a Commit 4-16
Work with Git from the Designer 4-16

Use Branches 4-18
Create a Branch 4-18
Protect a Branch 4-19
Manage a Branch 4-21
Manage a Git Repository 4-22
Copy a Git File/Repository's URL 4-23
View File/Repository History 4-24
Use Tags 4-25
Create and Manage Tags 4-25
Compare Revisions 4-26
Add Comments to a File 4-27
Watch a Git Repository 4-27
Search a Git Repository 4-28
Download a Git Repository's Archive 4-29
Review Source Code with Merge Requests 4-29
Merge Requests Concepts and Terms 4-29
Merge Request States 4-31
Create and Update a Merge Request 4-31
Create a Merge Request 4-31
Create a Merge Request from the Command Line 4-34
Set a Merge Request to Draft State 4-37
Use Templates to Improve MR Descriptions 4-37
Add or Remove Reviewers 4-40
Link an Issue to a Merge Request 4-41
Link a Build Job to a Merge Request 4-41
Add an Attachment (Image or File) to a Merge Request Comment 4-42
Use the Toolbar to Work with Comments 4-46
Watch a Merge Request 4-46
Merge Request Email Notifications 4-47
Review a Merge Request 4-47
Open a Merge Request 4-47
View Commits and Changed Files 4-48
Add a General Comment 4-50
Add an Inline Comment to a File 4-51
Filter a Merge Request Conversation 4-52
Navigate Multiple Comments 4-53
Approve or Reject a Merge Request 4-55
Merge Branches and Close the Merge Request 4-55
Merge Branches 4-55
Resolve a Merge Conflict 4-57
Close a Merge Request 4-59
Reopen a Closed Merge Request 4-59

Cherry-pick a Commit 4-59
Reapply a Merge Request's Commits to a New Branch 4-65
Retarget a Merge Request's Commits to Another Branch 4-67
Revert a Merge Request 4-67
Merge Request and Branch Administration 4-69
Manage Merge Requests 4-69
Assign Owners to Files in the Project's Git Repository 4-69
Set Review and Merge Restrictions on a Repository Branch 4-71

5 Create and Use Environments

Define Your Environments 5-1
Authorize VB Studio to Access Visual Builder and Integration Instances 5-2
Set Up an Environment 5-3
Add an Oracle Integration Instance to an Environment 5-6
Manage Your Instance in Another Identity Domain Using OAuth 5-8
Manage Trust Certificates in an Environment Definition 5-9
Create and Edit VB Studio Custom Backends for Visual Applications 5-10
Manage an Environment 5-13
Use Service Instance Statuses to Troubleshoot Problems 5-14

6 Use Workspaces and the Designer

What Is the Designer? 6-1
What Is a Workspace? 6-1
Create a Workspace 6-3
What is a Scratch Repository? 6-4
Push Your Scratch Repository to the Remote Repository 6-4
Understanding Default Branch Names 6-5
Manage Workspaces 6-5

7 Build Your Applications

Configure and Run Project Jobs and Builds 7-1
What Are Jobs and Builds? 7-1
What Are Software Packages? 7-1
Create and Manage Jobs 7-2
Configure a Job 7-6
Configure Job Protection Settings 7-6
Access a Project's Git Repositories 7-10
Trigger a Build Automatically on a Schedule 7-18
Use Build Parameters 7-20

Use a Named Password/Private Key 7-27
Access an Oracle Cloud Service Using SSH 7-29
Access the Oracle Maven Repository 7-30
Generate a Dependency Vulnerability Analysis Report 7-33
Move Oracle Integration Projects, Artifacts, Packages, and Lookups Between Instances 7-38
Run Unix Shell Commands 7-59
Build Maven Applications 7-62
Build Ant Applications 7-67
Build Gradle Applications 7-68
Build Node.js Applications 7-70
Access an Oracle Database Using SQLcl 7-71
Use OCIcli to Access Oracle Cloud Infrastructure Services 7-73
Run Docker Commands 7-74
Run Fn Commands 7-76
Use SonarQube 7-78
Publish JUnit Results 7-85
Use the Xvfb Wrapper 7-88
Publish Javadoc 7-89
Archive Artifacts 7-89
Copy Artifacts from Another Job 7-90
Configure General and Advanced Job Settings 7-91
Manage Build Actions 7-93
Change a Job's Java Version 7-93
Change a Job’s Build Executor Template 7-94
Run a Build 7-95
View a Job’s Builds and Reports 7-95
View a Build’s Logs and Reports 7-95
View a Project’s Build History 7-99
View a Job’s Build History 7-100
View a Job’s History by User Name 7-101
View a Build’s Details 7-101
Download Build Artifacts 7-101
Watch a Job 7-102
Build Executor Environment Variables 7-102
Run Jobs in a Pipeline 7-107
What Is a Pipeline? 7-108
Pipeline Layout 7-109
Create and Manage Pipelines 7-110
Use the Pipeline Designer 7-113
Create a One-to-One Dependency 7-114
Create a One-to-Many Dependency 7-115

Create a Many-to-One Dependency 7-116
Edit Pipeline Dependencies 7-118
Protect Your Pipeline: Restrict Who Can Start It Manually or Edit Its Configuration 7-119
Add Manual Pipeline Approvals 7-122
Create a Pipeline Approvers Group for Your Project 7-123
Add a Manual Approval Item to Your Pipeline at Design Time 7-126
Approve the Manual Approval when the Pipeline Runs 7-132
Notify the Pipeline Approvers 7-136
Monitor Pipeline Status 7-138
View Pipeline Run Details 7-139
View Pipelines in Progress 7-140
Add or Export Parameters and Parameter Lists 7-140
Add a Parameter 7-141
Add a Parameter List 7-142
Export a Parameter 7-142
Export a Parameter List 7-143
Configure Jobs and Pipelines with YAML 7-143
What Are YAML Files Used for in VB Studio? 7-144
REST API for Accessing YAML Files 7-147
How Do I Validate a Job or Pipeline Configuration? 7-147
Create a Job or a Pipeline Without Committing the YAML File 7-147
How Do I Use YAML to Create or Configure a Job? 7-148
What Is the Format for a YAML Job Configuration? 7-149
YAML Job Configuration Examples 7-164
How Do I Use YAML to Create or Configure a Pipeline? 7-166
What Is the Format for a YAML Pipeline Configuration? 7-167
YAML Pipeline Configuration Examples 7-172
Set Dependency Conditions in Pipelines Using YAML 7-173
Define and Use Triggers in YAML 7-173

## 8 Deploy and Manage Your Applications

Package, Deploy, and Manage Extensions 8-2
Deploy an Extension to a Oracle Cloud Applications Development Instance 8-2
Deploy an Extension to an Oracle Cloud Applications Production Instance 8-2
View a Deployed Extension 8-3
Delete an Extension 8-5
Delete an Extension Manually 8-6
Configure a Job to Manage a Deployed Extension 8-7
Configure a Job to Audit and Test Your Extension 8-9
Configure a Job to Delete an Extension 8-11
Package, Deploy, and Manage Visual Applications 8-13

Deploy a Visual Application to a Development Visual Builder Instance 8-13
Deploy a Visual Application to a Test or Production Visual Builder Instance 8-13
View a Deployed Visual Application 8-14
Lock, Unlock, or Roll Back a Deployed Visual Application 8-14
Undeploy a Visual App 8-15
Undeploy a Visual App Manually 8-16
Configure a Job to Manage a Deployed Visual Application 8-16
Configure a Job to Audit and Test Your Visual Application 8-18
Configure a Job to Import Data to or Export Data from a Visual Application 8-20
Configure a Job to Lock, Unlock, or Roll Back a Deployed Visual Application 8-21
Configure a Job to Undeploy a Visual Application 8-23
Deploy Build Artifacts to Oracle Cloud Services 8-24
Deploy an Application to JCS 8-24
Access a Deployed Application 8-25
Manage Oracle Cloud Service Deployments 8-26

Part III Maximize the Power of Your Project

9 Track and Manage Tasks, Defects, and Features

Issue Types 9-1
Issue Statuses 9-2
Create Issues 9-2
Create an Issue from the Issues Page 9-2
Search Issues 9-3
Save a Custom Search 9-3
Share Custom Search Filters 9-4
View and Update Issues 9-4
Edit an Issue and Associate a Branch with a Merge Request 9-5
Resolve an Issue 9-9
Mark an Issue as Duplicate 9-10
Update Time Spent on an Issue 9-10
Associate an Issue with a Sprint 9-11
Create Parent-Child Relationships Between Issues 9-11
Link One Issue to Another One (or to Multiple Issues) 9-11
Update Multiple Issues 9-13
Archive Issues 9-14
Watch an Issue 9-15

## 10 Manage Software Releases


What Are Release States? 10-1
Create a Release 10-2
Change a Release’s Status 10-2
Specify a Release's Artifacts 10-2
Manage Releases 10-3
Download a Release's Artifacts 10-3


## 11 Use Agile Boards to Manage and Update Issues


Agile Boards Concepts and Terms 11-1
Create and Configure Agile Boards 11-2
- Create a Board 11-2
- Add and Manage Progress States of a Board 11-3
- Configure Working Days of a Board 11-5
- Configure and Manage a Board 11-5

Use Scrum Boards 11-6
- Create and Manage Sprints 11-6
- Add and Manage Issues in a Sprint 11-8
- Update Issues in an Active Sprint 11-8
  - Update an Issue’s Progress in an Active Sprint 11-9
  - Reschedule a Sprint 11-9
  - Complete a Sprint 11-10
- Review Issue Reports for a Scrum Board 11-10

Use Kanban Boards 11-10
- Add and Manage Active Issues 11-10
- Update Active Issues 11-11
  - Update the Progress State for an Active Issue 11-11
  - Complete a Sprint 11-12
- Review Issue Reports for a Kanban Board 11-12

Review Agile Reports and Charts 11-13
- Burndown Charts 11-13
- Sprint Reports 11-14
- Issues Reports 11-15
- Epic Reports 11-15
- Velocity Reports 11-16
- Cumulative Flow Charts 11-16
- Control Charts 11-17

## 12 Manage Binaries and Dependencies with Maven


Maven Concepts and Terms 12-1
Set Up and Populate Your settings.xml File 12-1
Copy Distribution Management Snippets 12-7
Upload an Artifact Manually 12-7
    Upload Artifacts Using the Maven Command-Line Interface 12-8
Download an Artifact Manually 12-9
Search Artifacts 12-10
Sort Maven Artifacts and Snapshots 12-11
Maven Repository Administration 12-12
    Configure Auto-Cleanup for Snapshots 12-12
    Configure Overwriting for Release Artifacts 12-13


## 13 Access External Docker Registries


Link an External Docker Registry to Your Project 13-1
Browse a Linked Docker Registry 13-2


## 14 Use the Project's NPM Registry


Configure Your Connection to the Project's NPM Registry 14-1
Create and Manage a Project's Remote NPM Registry Connection 14-5
Configure a Job to Connect to the NPM Registry 14-6
Use the NPM Command Line with the Project's NPM Registry 14-7
Publish JS Packages to VB Studio's NPM Registry 14-9
    Select and Publish Packages from the NPM Page 14-9
    Monitor the Upload and View the Published Package(s) 14-10
    Publish Packages with a Job or Pipeline 14-11
Check for Security Vulnerabilities in your Project's NPM Packages and Dependencies 14-12
Browse and Search Packages in Your Project's NPM Registry 14-13


## 15 Send Notifications to External Software Using Webhooks


Slack 15-1
    Get the Slack Channel’s Incoming Webhook URL 15-1
    Configure a Slack Webhook in VB Studio to Send Event Notifications 15-3
PagerDuty 15-5
    Set Up the PagerDuty Account 15-6
    Configure a PagerDuty Webhook in VB Studio to Send Event Notifications 15-8
Jenkins 15-8
    Use SCM Polling to Trigger a Jenkins Job 15-9
        Set Up Git on Jenkins 15-9

Configure the Jenkins Job to Use the VB Studio Git Repository and Enable SCM Polling 15-10
Configure a Hudson/Jenkins Git Plugin Webhook 15-13
Create a Hudson/Jenkins - Build Trigger Webhook to Trigger a Jenkins Job on an Update to a Git Repository 15-13
Install the Build Authorization Token Root Plugin on Jenkins 15-14
Get the Jenkins API Access Token 15-15
Configure the Jenkins Job to Set an Authentication Token 15-16
Configure a Hudson/Jenkins Build Trigger Webhook 15-17
Use the Jenkins - Merge Requests Webhook to Link a Jenkins Job to a Merge Request 15-20
Install the Notification Plugin on Jenkins 15-20
Install the Build Authorization Token Root Plugin on Jenkins 15-21
Get the Jenkins API Access Token 15-22
Configure the Jenkins Job to Set an Authentication Token and Accept Build Parameters 15-23
Configure a Webhook in VB Studio to Trigger a Jenkins Job on a Merge Request Update 15-24
Link the Jenkins Job with the Merge Request 15-25
Create a Jenkins - Notification Plugin Webhook So VB Studio Accepts Build Notifications from a Jenkins Job 15-26
Install the Notification Plugin on Jenkins 15-26
Configure a Webhook in VB Studio to Accept Notifications from Jenkins 15-27
Configure the Jenkins Job to Send Build Notifications 15-27
Hudson 15-28
Trigger a Hudson Job on SCM Polling 15-29
Set Up Git on Hudson 15-29
Configure the Hudson Job to Use VB Studio Git Repository and Enable SCM Polling 15-31
Configure a Webhook in VB Studio to Trigger a Hudson Job When the Git Repository Gets Updated 15-32
Trigger a Hudson Job on a Git Repository Update 15-33
Configure the Hudson Job 15-33
Configure a Webhook in VB Studio to Trigger a Hudson Job on a Git Repository Update 15-34
GitHub Apps 15-35
Send Event Notifications to Any Application 15-36
What Is a Generic Webhook's Data Structure? 15-37

## 16 Share and Use Code Snippets

Create and Manage Snippets 16-1
Add and Manage Files in a Snippet 16-2
Copy a Snippet File's Contents 16-3
Add a Comment to a Snippet 16-4
Use Git with Snippets 16-5

Download an Archive of the Snippet 16-5

## 17 Co-Author Wikis

Working with Markup Languages in VB Studio 17-1
- Textile Syntax Reference 17-3
- Confluence Syntax Reference 17-6
- Markdown Syntax Reference 17-8
- Use the Markup Toolbar 17-9

Create and Manage Wiki Pages 17-10
Add Comments to a Wiki Page 17-15
Watch a Wiki Page 17-16
View a Wiki Page's History and Compare Versions 17-16
Wiki Administration 17-17
- Configure Edit and Delete Rights for Wiki Pages 17-17
- Change a Project’s Wiki Markup Language 17-18

# Part IV Troubleshooting

## 18 Troubleshooting VB Studio Issues

Fix OCI Storage Issues 18-1
Fix “E-mail address of PersonIdent must not be null" Error 18-1

# Preface

*Using Oracle Visual Builder Studio* describes how to manage, build, merge, and deploy generic projects, projects for Oracle Cloud Application extensions, and projects for VB Studio visual applications.

**Topics:**


- Audience
- Documentation Accessibility
- Diversity and Inclusion
- Related Resources
- Conventions


## Audience

This document is intended for users who are responsible for managing, building, merging, and deploying generic projects, projects for Oracle Cloud Application extensions, and projects for VB Studio visual applications.

## Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://support.oracle.com/portal/ or visit Oracle Accessibility Learning and Support if you are hearing impaired.

### Keyboard Shortcuts

When working in the VB Studio user interface, you can use these keyboard shortcuts to help you move around quickly:

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Find and open a file in an app | Command-P | Ctrl+P |
| Find a given string in a file (or in the Code editor) | Command-Shift-F | Ctrl+F |

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Undo/redo | Command-Shift-Z | Ctrl+Z<br>Except for common Windows shortcuts such as cut, copy, and paste, VB Studio does *not* support shortcuts like the ones described here. |

# Keyboard Shortcuts for Code Editors

To help you work efficiently, VB Studio supports the following keyboard shortcuts when working with code editors.

### Tab behavior shortcut

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Change the code editor's default behavior for the Tab key; pressing Tab adds four spaces in the editor | Control-Shift-M | Ctrl+M |

### Show Command Palette and Save shortcuts

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Show Command Palette | F1 | F1 |
| Save | Command-S | Ctrl+S |

### Basic Editing shortcuts

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Cut line if there is no selection; cut selection if there is one | Command-X | Ctrl+X |
| Copy line if there is no selection; copy selection if there is one | Command-C | Ctrl+C |
| Move line up | Option-Up | Alt+Up |
| Move line down | Option-Down | Alt+Down |
| Copy line up | Shift-Option-Up | Shift+Alt+Up |
| Copy line down | Shift-Option-Down | Shift+Alt+Down |
| Delete line | Shift-Command-K | Ctrl+Shift+K |
| Insert line below | Command-Enter | Ctrl+Enter |
| Insert line above | Shift-Command-Enter | Ctrl+Shift+Enter |
| Jump to matching bracket | Shift-Command-\ | Ctrl+Shift+\ |
| Indent line | Command-] | Ctrl+] |
| Outdent line | Command-[ | Ctrl+[ |
| Go to beginning of line | Home or Fn-Left | Home |
| Go to end of line | End or Fn-Right | End |
| Go to beginning of file | Command-Home | Ctrl+Home |
| Go to end of file | Command-End | Ctrl+End |
| Scroll line up | Command-Up | Ctrl+Up |

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Scroll line down | Command-Down | Ctrl+Down |
| Scroll page up | Option-PgUp | Alt+PgUp |
| Scroll page down | Option-PgDn | Alt+PgDn |
| Fold (collapse) region | Command-Shift-[ | Ctrl+Shift+[ |
| Unfold region | Command-Shift-] | Ctrl+Shift+] |
| Toggle line comment | Command-/ | Ctrl+/ |
| Toggle block comment | Shift-Option-A | Shift+Alt+A |
| Toggle word wrap | Option-Z | Alt+Z |

### Navigation shortcuts

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Go to line ... | Command-G | Ctrl+G |
| Go to symbol ... | Shift-Command-O | Ctrl+Shift+O |
| Go to next error or warning | F8 | F8 |
| Go to previous error or warning | Shift+F8 | Shift+F8 |
| Go back | Control-- | Alt+Left |
| Go forward | Control-Shift-- | Alt+Right |

### Search and Replace shortcuts

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Find | Command-F | Ctrl+F |
| Replace | Option-Command-F | Ctrl+H |
| Find next | Command-G | F3 |
| Find previous | Shift-Command-G | Shift+F3 |
| Select all occurrences of Find match | Option-Enter | Alt+Enter |
| Add selection to find matches | Command-D | Ctrl+D |

### Multi-cursor and Selection shortcuts

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Insert cursor | Option-Click | Alt+Click |
| Insert cursor above | Option-Command-Up | Ctrl+Alt+Up |
| Insert cursor below | Option-Command-Down | Ctrl+Alt+Down |
| Undo last cursor operation | Command-U | Ctrl+U |
| Insert cursor at end of each line selected | Shift-Option-I | Shift+Alt+I |
| Select current line | Command-L | Ctrl+L |
| Select all occurrences of current selection | Shift-Command-L | Ctrl+Shift+L |
| Select all occurrences of current word | Command-F2 | Ctrl+F2 |
| Expand selection | Control-Shift-Command-Right | Shift+Alt+Right |

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Shrink selection | Control-Shift-Command-Left | Shift+Alt+Left |
| Column (box) selection | Shift-Option- (drag mouse) | Shift+Alt + (drag mouse)<br>Ctrl+Shift+Alt+(arrow key) |
| Column (box) selection page up | Shift-Option-Command-PgUp | Ctrl+Shift+Alt+PgUp |
| Column (box) selection page down | Shift-Option-Command-PgDn | Ctrl+Shift+Alt+PgDn |

**Rich Languages Editing shortcuts**

| To do this: | Use this on Mac: | Use this on Windows: |
|---|---|---|
| Trigger suggestion | Command-Space | Ctrl+Space |
| Format document | Shift-Option-F | Shift+Alt+F |
| Go to definition | F12 | F12 |
| Peek definition | Option-F12 | Alt+F12 |
| Show references | Shift+F12 | Shift+F12 |
| Rename symbol | F2 | F2 |

# Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

# Related Resources

For more information, see these Oracle resources:

- Oracle Public Cloud

  http://cloud.oracle.com
- Create Visual Applications in VB Studio in *Building Responsive Applications with Visual Builder Studio*
- What Can You Do with Oracle Visual Builder Studio? in *Extending Oracle Cloud Applications with Visual Builder Studio*
- Create an Oracle Visual Builder Studio Generation 2 Instance in *Administering Visual Builder Studio*

# Conventions

The following text conventions are used in this document:

| Convention | Meaning |
| --- | --- |
| **boldface** | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| *italic* | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| `monospace` | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

### Videos and Images

Your company can use skins and styles to customize the look of the application, dashboards, reports, and other objects. It is possible that the videos and images included in the product documentation look different than the skins and styles your company uses.

Even if your skins and styles are different than those shown in the videos and images, the product behavior and techniques shown and demonstrated are the same.

# Part I
# Overview

This part introduces you to the basic information about Oracle Visual Builder Studio and walks you through the process of getting set up to use the service.

**Topics:**


- The Basics
- Get Yourself Set Up

1

# The Basics

Learn about Oracle Visual Builder Studio, its projects, components, roles, and how to access the service.

See Oracle Cloud Terminology in *Getting Started with Oracle Cloud* for definitions of terms found in this and other documents in the Oracle Cloud library.

## What Is Oracle Visual Builder Studio?

Oracle Visual Builder Studio (VB Studio) is a robust application development platform that helps your team effectively plan and manage your work throughout all stages of the app dev lifecycle: design, build, test, and deploy. It makes it easy for your entire team to develop the artifacts they need, including:

- Oracle Cloud Applications developers, who need to create *extensions* to customize their Apps—like Human Capital Management, Supply Chain and Manufacturing, and so on—to meet their business needs.
- Low-code developers, who want to create *responsive applications* using a visual designer, then deploy them to their own version of Visual Builder.
- Experienced programmers, who want to modify the source code for applications created by others, or to develop bespoke apps using the web programming language of their choice.

Regardless of what type of developer you are—or if you're focused on managing the work that others are creating—everyone can take advantage of these key VB Studio features:

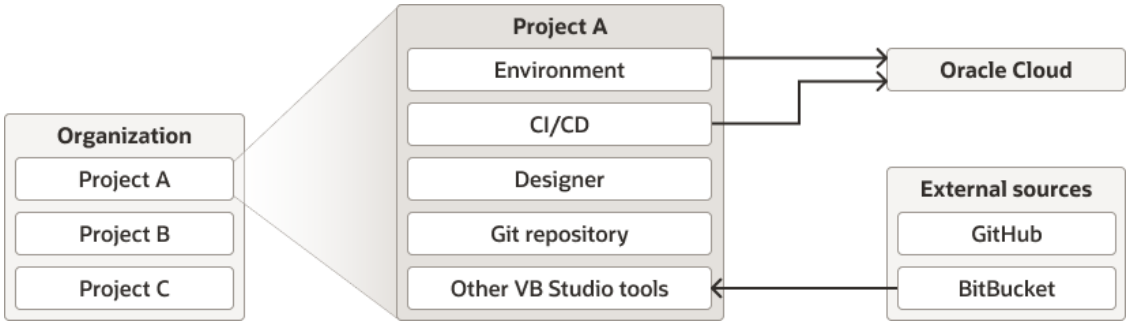
- A rich visual designer integrated with source control (Git) so that developers can manage changes, apply version control best practices, and collaborate with their teammates to develop applications
- The ability to build and display different flavors of the UI to meet the needs of discrete users of certain Oracle Cloud Applications (those built with VB Studio and Oracle JavaScript Extension Toolkit (Oracle JET)), also within a Git framework
- Built-in repositories for hosting code in Git and for hosting binaries, such as Maven dependencies
- A continuous integration service so you can automate your build and test systems
- A continuous delivery service that tightly integrates with Oracle Cloud Applications
- Agile boards and an issue tracking system for tracking sprints, tasks, defects, and features

VB Studio enables developers to easily deploy their applications to their preferred target, whether it’s a staging or production instance of Oracle Cloud Applications, an Oracle Cloud Infrastructure (OCI) service instance, or a Visual Builder instance.

## What Are the VB Studio Building Blocks?

Before you start working with VB Studio, it's important to understand how some of the key components work together.

At a high level, the key components are shown here:




Let's take a closer look at this diagram, starting at the far left:

- Within a single VB Studio instance, you and your team members who use that instance are considered an *organization*. Think of an organization as the umbrella for all the projects in a given identity domain. (From a process standpoint, you should limit the people who are designated as organizational administrators—that is, those who have the DEVELOPER_ADMINISTRATOR role—as they have the ultimate power in the VB Studio ecosystem. See Administering Visual Builder Studio for a full description of everything the org admin can do.)
- Within your organization, you will likely belong to one or more *projects*, each of which is devoted to a discrete software effort. For example, you might have a project for extending an HCM Oracle Cloud Application, and a different project for creating a bespoke vacation tracking application. A project brings together all the tools you need to create those artifacts. The middle set of boxes shows just a few of these tools in the context of Project A. See Work with Projects for more on working with projects.
- In VB Studio terms, an *environment* defines your deployment target and helps you manage it from within VB Studio. If you're creating an extension to customize an Oracle Cloud Application, you must define an environment that points to that Oracle Cloud Application instance, as well as the sign-on credentials necessary to access it. If you're creating a visual application, on the other hand, you need an environment that defines the target Visual Builder instance (which must be purchased separately from VB Studio). Create and Use Environments tells you how to set up and work with environments.
- To achieve continuous development and integration (CI/CD) in VB Studio, you set up *jobs* (also called *build jobs*) to perform various actions, such as packaging artifacts, running shell commands, running unit test scripts, and deploying application artifacts. You then can string these jobs together in a *pipeline*, to help you run continuous integration jobs. Build Your Applications explains how to set these up.
- The *Designer* is a declarative environment that you use to extend Oracle Cloud Applications or to build visual applications. You enter the Designer by creating a *workspace* within a project. See What Is the Designer? to learn more about this important tool.
- Each project can have one or more *Git repositories* to store source code and provide version control. VB Studio offers the full complement of Git commands from within the Designer, or you can operate from the command line. See Manage Source Code Files with Git for more.
- VB Studio offers a wide range of tools to help you manage your software development process and to better collaborate with your team members, including wikis, an issue tracking system, agile boards, and much more. Take a look at this book's table of contents to see what's available.
- At the far right, you can see that Environments and CI/CD have arrows pointing to Oracle Cloud, which reflects your ability to:

- Deploy extensions to your Oracle Cloud Application production instances
- Deploy responsive applications to Visual Builder
- Make calls against REST endpoints for various Oracle Cloud infrastructure services (for example, to fetch or deploy data).

Finally, the arrow in the External Sources portion of the illustration shows that you don't have to use the Git repository that comes with VB Studio to store your source code. If you prefer, you can define an external repository—like Bit Bucket or GitHub—and use it when configuring your build and deploy jobs, just as if you were using the internal repository.

Now that you have a basic idea of what VB Studio has to offer, it's a good time to establish some basic preferences and to customize your personal operating environment. Get Yourself Set Up can help guide you.

# What Are My Identity Roles?

IDCS roles (also called identity roles) define who can sign in to VB Studio, as well as what privileges they enjoy after they do so.

To access VB Studio, you must have one of two roles:

- DEVELOPER_ADMINISTRATOR - Also known as an organization administrator, users with this role can create and set up an instance of VB Studio, manage all projects, manage VM executors and executor templates, and update the organization's details. Because this role is so powerful, it should be granted sparingly.
- DEVELOPER_USER - Users with this role can create and access VB Studio projects, and create things like wikis, tracking issues, and much more.

In addition to these VB Studio roles, you may have other roles related to other facets of VB Studio. For example, if you have a separate instance of Visual Builder to which you deploy responsive applications, you may have the ServiceAdministrator, ServiceDeveloper, or ServiceUser role.

To find out what identity roles you've been assigned:

1. In a web browser, go to `https://cloud.oracle.com`, and click **Sign In**.
2. On the Sign-In page, in **Account**, enter your account or tenant name, and click **Next**.
3. On the Oracle Cloud Account sign-in page, enter your Oracle Cloud account credentials, and click **Sign In**.
4. If you land on the My Oracle Services page, click **Infrastructure Dashboard**.
5. On the OCI Console, click ☰ in the top-left corner.
6. Under **Governance and Administration**, select **Identity**, and then select **Federation**.
7. Select the identity service provider.
8. In the **IDCS Username** column, click your name.
9. Click **Manage Service Roles**.
10. In the **Service** column, find the Developer service.
11. Click the three vertical dots on the right, and select **Manage instance access**.
12. In the **Instance Role** column, note your roles.
13. Click **Cancel** to return to the last page.

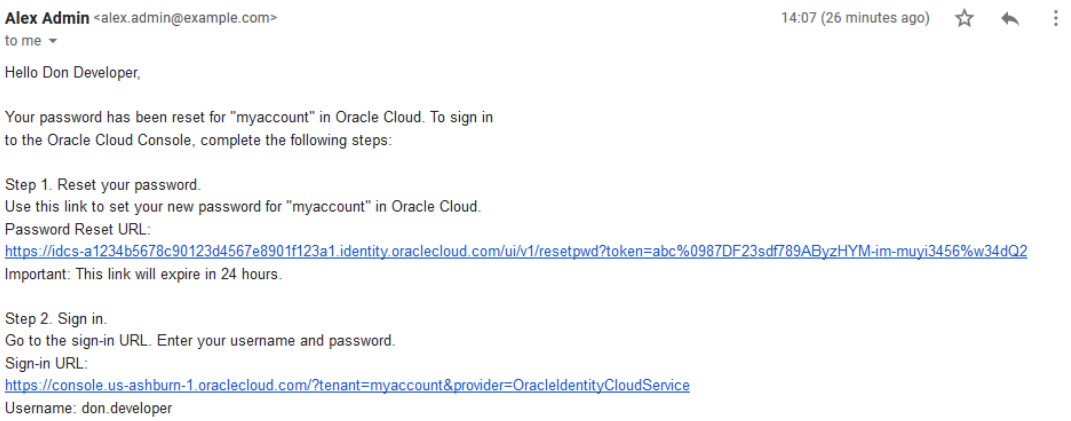
# How Do I Access VB Studio?

You can access VB Studio using the latest version of the Chrome browser running on Mac OS X and Windows. Other browsers and platforms are not supported.

To access VB Studio, you need the service URL, plus your identity domain name, username, and password. If you're a new user, you can sign in from the Oracle Cloud home page. If you're a returning user, you can find the service URL from several of the emails you received, the ones with the subject **Welcome to Oracle Visual Builder Studio** or **Verify your Oracle Visual Builder Studio**.

During the onboarding process, you'll receive a series of emails, including some optional ones:

1. After adding you as a user, the OCI administrator can choose to send an email to you:



**Alex Admin** <alex.admin@example.com> 14:07 (26 minutes ago)
to me

Hello Don Developer,

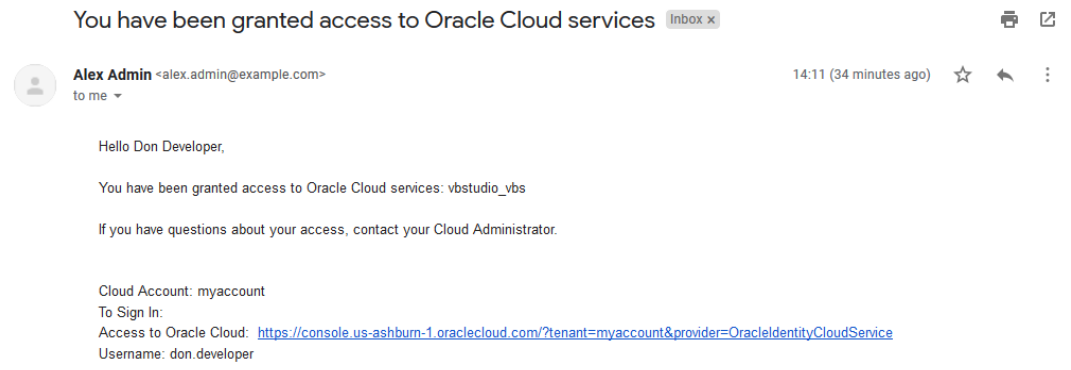
Your password has been reset for "myaccount" in Oracle Cloud. To sign in to the Oracle Cloud Console, complete the following steps:

Step 1. Reset your password.
Use this link to set your new password for "myaccount" in Oracle Cloud.
Password Reset URL:
https://idcs-a1234b5678c90123d4567e8901f123a1.identity.oraclecloud.com/ui/v1/resetpwd?token=abc%0987DF23sdf789AByzHYM-im-muyi3456%w34dQ2
Important: This link will expire in 24 hours.

Step 2. Sign in.
Go to the sign-in URL. Enter your username and password.
Sign-in URL:
https://console.us-ashburn-1.oraclecloud.com/?tenant=myaccount&provider=OracleIdentityCloudService
Username: don.developer

This email serves two purposes: to send the password reset URL and to provide the OCI sign-in url. The Oracle Cloud account name in the email is in double quotes (myaccount). If you bookmark the second URL in this email, the sign-in URL, you won't need the account name for signing in.

2. After the OCI administrator assigns you a VB Studio IDCS role, you'll receive this email:



You have been granted access to Oracle Cloud services Inbox ×

**Alex Admin** <alex.admin@example.com> 14:11 (34 minutes ago)
to me

Hello Don Developer,

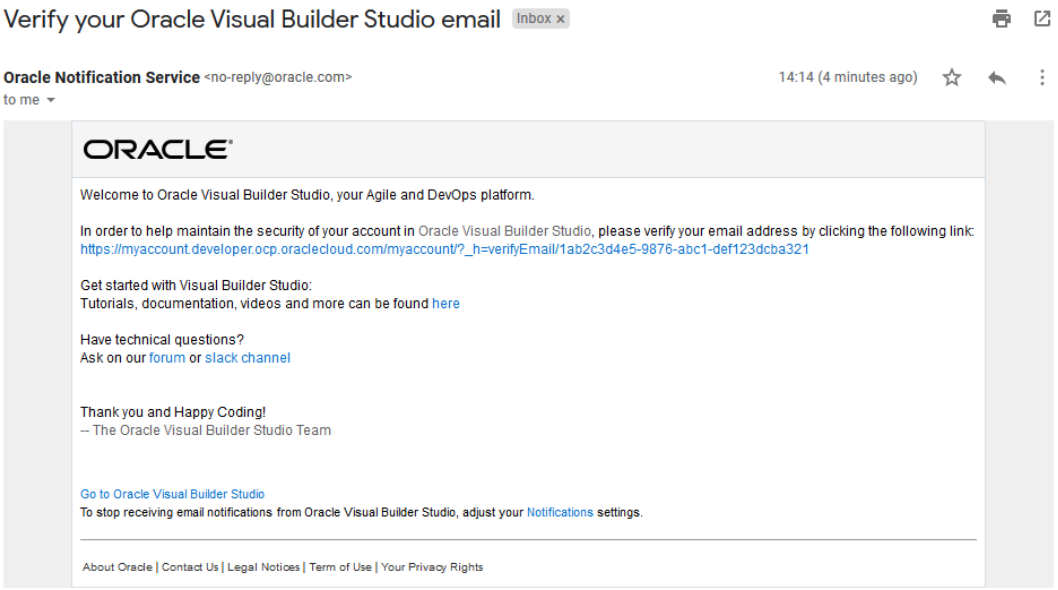
You have been granted access to Oracle Cloud services: vbstudio_vbs

If you have questions about your access, contact your Cloud Administrator.

Cloud Account: myaccount
To Sign In:
Access to Oracle Cloud: https://console.us-ashburn-1.oraclecloud.com/?tenant=myaccount&provider=OracleIdentityCloudService
Username: don.developer

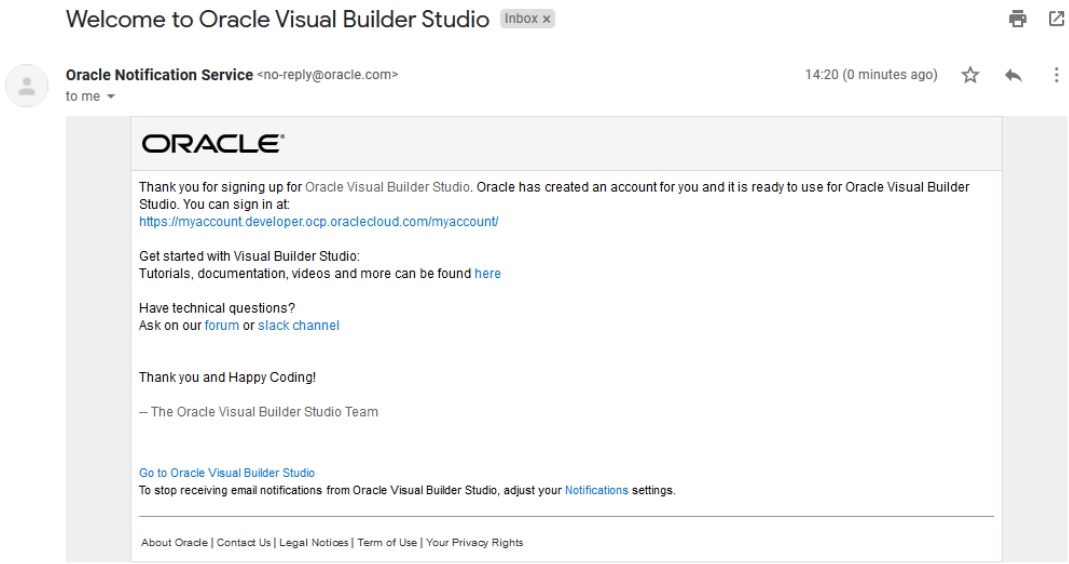
This email shows the Oracle Cloud account name (myaccount), the sign-in URL for Oracle Cloud, and the username (don.developer).

3. After the VB Studio organization administrator adds you to a VB Studio project, you may receive a verification email:

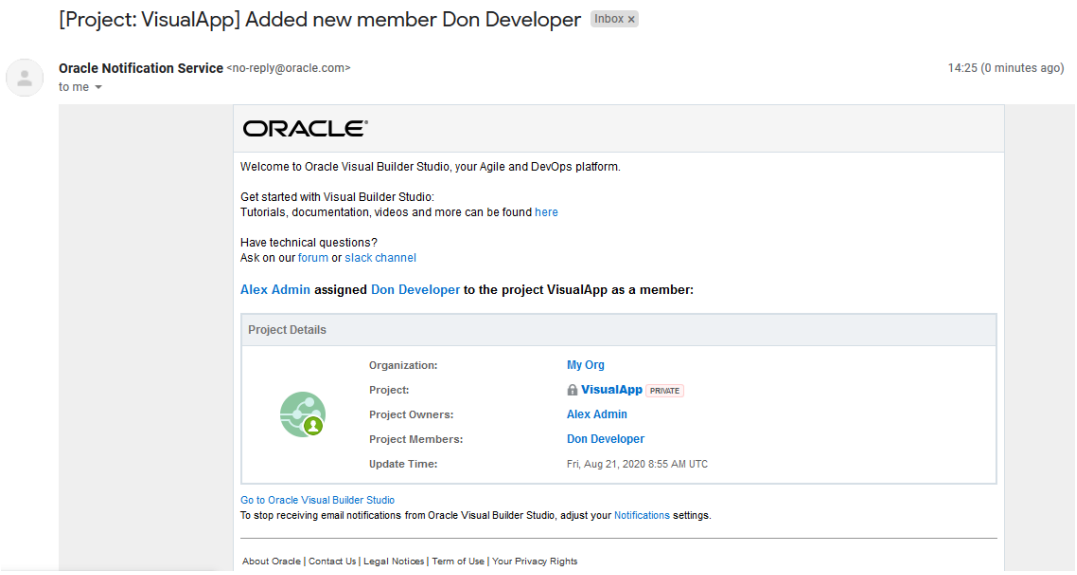



Click the verification link in this email to verify your email address to the service.

4. After signing in to VB Studio for the first time, you'll receive this Welcome email, with the VB Studio sign-in URL:




Each time you're added to a project, you'll receive this confirmation email:




The email contains project information such as the organization name (My Org), the project name (VisualApp), the project's privacy setting (private), the name of the project owner Alex Admin), a list of the project's members (on Developer), and the date and time that you were added to the project.

# Access VB Studio from the Oracle Cloud Home Page

If you've signed out of the Oracle Cloud or VB Studio, follow these steps to sign back in.

1. In a web browser, go to `https://cloud.oracle.com`.
2. On the Sign-In page, in **Cloud Account Name**, enter your Oracle Cloud account or tenancy name and click **Next**.
3. On the Single Sign-On (SSO) panel, click **Continue**.
4. Enter your Oracle Cloud account credentials and click **Sign In**.

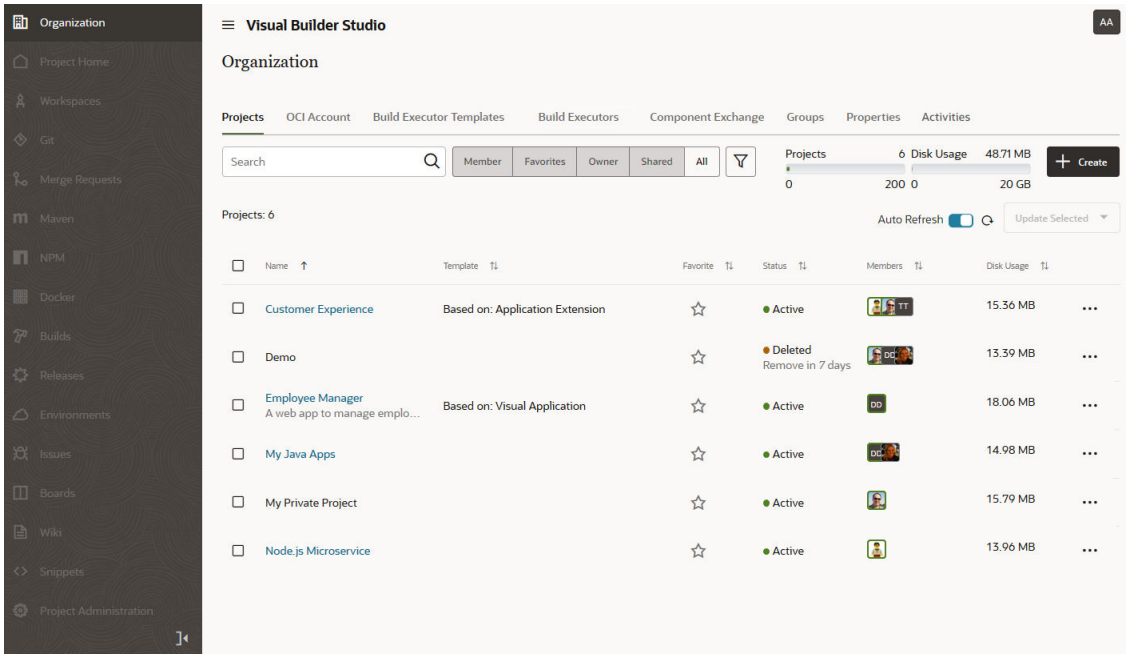
   The Oracle Cloud Console, also called the OCI console, opens.
5. In the upper-left corner, click **Navigation Menu** ☰.
6. Select **OCI Classic Services** and then select **Developer**.

   If you don't see the **Developer** option, contact your administrator and make sure that you've been assigned the required IDCS roles. See Get the Right IDCS Roles.
7. In the **Instances** tab, click **Manage this instance** ☰ and select **Access service instance**.

The VB Studio **Organization** page opens, displaying all the projects you're a member of, as well as your favorite projects, projects you own, and all your organization's shared projects.

If you're an organization administrator, you'll see additional tabs where you can set up OCI connections and build executors, and configure your organization's component exchange, groups, and properties. You'll also have access to the **Activities** tab that tracks changes made at the organization level, such as deleted projects or projects that you assigned yourself as owner.

Here's an example of the Organization page as it appears to the organization administrator:




# Who Should Use This Guide?

No matter what you want to do with Visual Builder Studio, the VB Studio library has a guide to help you along your journey.

Here's an easy way to determine which book applies to you:

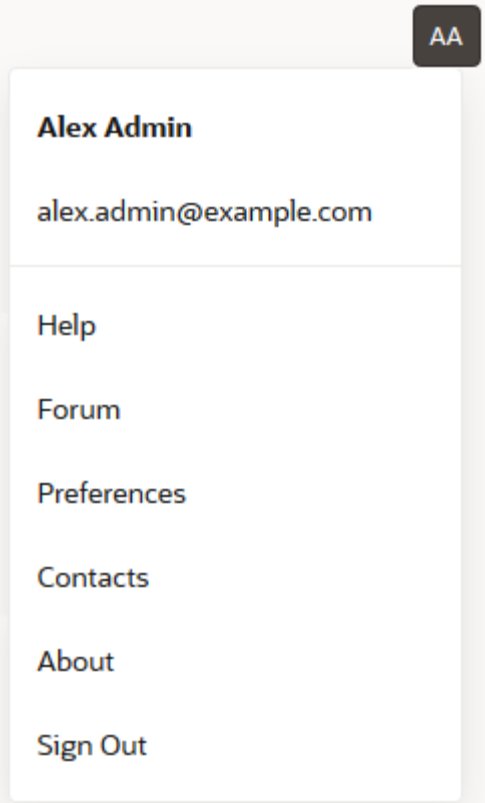
| If you want to: | Check out this guide: |
|---|---|
| Establish your user preferences across VB Studio | Using Visual Builder Studio , especially Get Yourself Set Up. |
| Set up build and deploy jobs, create and manage projects, and take advantage of collaboration tools like wikis, agile boards, and so on | Using Visual Builder Studio |
| Create an instance of VB Studio and set it up for others to use | Administering Oracle Visual Builder Studio |
| Customize an Oracle Cloud Application to suit your business needs | Extending Oracle Cloud Applications with Visual Builder Studio |
| Customize an Oracle Cloud Application in a scaled-down, easy to use environment | Extending Oracle Cloud Applications in Visual Builder Studio Express Mode |
| Build and deploy a responsive app to Visual Builder | Building Reponsive Applications with Visual Builder Studio |

2

# Get Yourself Set Up

You can set your preferences, including your display name, email address, gravatar, and email notifications, from the User Preferences page.

To get to this page, click the user avatar and select **Preferences**:




After setting your user preferences, you'll need to set up a Git client and decide whether to display the news banner or hide it.

## Update Your Display Name

By default, VB Studio displays your Oracle Cloud account name as your display name across all pages. If you want to change it, you can do so from the User Preference page's Profile tab.

1. On the User Preferences page, click the **Profile** tab.
2. In **First Name** and **Last Name**, update your name.

   The name is saved when the focus moves out of the field.
3. To the left of the **User Preferences** title, click ‹ to return to the last opened page.

# Update Your Email Address

By default, VB Studio displays your Oracle Cloud email address across all pages and sends email notifications, such as merge request notifications and issue notifications, to this email address.

If you want VB Studio email notifications sent to another email address, you can change it on the User Preference page's Profile tab:

- If you're using an email address as your Oracle Cloud login username, your original Oracle Cloud email address continues to be your login username even after you change your email address preference.
- After you provide another email address, you'll receive a verification email. It's important that you take the time to verify the new email address because, if you don't, you won't receive any VB Studio email notifications. You can, however, continue to use VB Studio.

Here's how to change the email address that email notifications will be sent to:

1. On the User Preferences page, click the **Profile** tab.
2. In the **Email Address** field, enter your new email address.

   The email address is saved when the focus moves out of the field.
3. Click the **Re-send email** button.
4. In the email that you receive, click the confirmation link to confirm the email address.

   After the verification, you're redirected to the service page.
5. Open the **Profile** tab again and verify that the **Email Address** field displays the **Verified** label.
6. To the left of the **User Preferences** title, click ‹ to return to the last opened page.

# Add Your Avatar Picture

VB Studio displays your Gravatar picture as the avatar picture. If you don't have a picture set in Gravatar or don't have a Gravatar account, VB Studio displays your initials instead of your avatar picture. To find out more about Gravatar, see `https://gravatar.com/`.

Here's how to create a Gravatar account and upload your Gravatar picture:

1. Open `http://en.gravatar.com/` in your browser.
2. Click **Create Your Own Gravatar**.
3. Follow the on-screen instructions, enter the required details, and sign up.

   Create your account with the same email address that you used to subscribe to Oracle Cloud.
4. After activating your account, sign in to Gravatar.
5. Upload the avatar picture to your Gravatar account.

The picture uploaded to your Gravatar account is automatically displayed as your avatar picture in VB Studio.

# Set Your Preferred Color Theme

VB Studio provides settings to control the color palette for your work environment, which by default uses a Light theme. You use these settings to configure a preferred color theme, which takes effect for all pages in the user interface, including the Designer.

To set your preferred color theme:

1. Click your user avatar and then click **Preferences**.
2. Select the **General** tab.
3. Select a theme under **Preferred color theme**:
   - Select **Dark theme** to switch the appearance to a dark color display, more suited for low-light conditions.
   - Select **Theme color determined by OS setup** to inherit the theme specified in your OS setting.
   - If you changed the default, select **Light theme** to switch back to a lighter background with dark text display.
4. To the left of the **User Preferences** title, click ‹ to return to the last opened page.

   The page's color theme should reflect your preferred choice.

> **Tip:**
>
> You can also control this setting from within the Designer. Click **Menu** in the upper-right corner of the Designer, then select **Theme**. See Tour the Designer.

See Customize a Web App's Appearance if you want to set a web app's color theme in the Designer. You configure this setting for a visual application in the Settings editor, which is accessed by clicking **Menu** in the upper-right corner of the Designer, then selecting **Settings**.

# Configure Your Global Email Notifications

You can configure your preferences to receive email notifications when a component, such as an issue or a Git repository's branch that you're subscribed to, is updated. Your preferences apply to all projects in which you're a member.

> **Note:**
>
> If your email address has changed, click the verification link in the email from VB Studio if you want to receive VB Studio email notifications. If you don't do this, when you go to the **Profile** tab under **User Preferences**, you'll see a **Not Verified** message next to your email address. Click the **Re-send email** button next to the error message and respond to that email when it reaches your inbox.

Configure your notifications:

1. On the User Preferences page, click the **Notifications** tab.

2. Select or deselect the **Notify Me Of** check boxes.

   Some check boxes are selected by default. For a selected component, its notifications from all projects of the organization where you're a member are enabled. You must subscribe or set up a watch on the component to get notifications about its updates:

| Select this check box ... | To receive email notifications about: |
|---|---|
| **Issue updates, attachments and comments** | Issues you're assigned to, or you're watching. |
| **Merge Request updates and comments** | Merge requests where you're a reviewer, or you're watching. |
| **New features, tips, and events** | New features, tips, and events from the VB Studio team. |
| **Service and system maintenance updates** | Service and system maintenance updates from the VB Studio team. |
| **Build activities** | Jobs you're watching and pipeline approvals that require your attention and a response. |
| **SCM/Push Activities** | Git repository branches you're watching. |
| **Wiki page updates and comments** | Wiki pages you're watching. |
| **Project Updates** | User updates when you or a user is added to or removed from a project, or the project role is changed. |
| **Include my Own Updates** | Your own changes.<br>If you don't select the check box, you won't receive email notifications for issues, merge requests, pipeline approvals, and Git updates that you initiated or created even though the **Issues updates, attachments and comments**, the **Merge Request updates and comments**, **Build activities**, and the **SCM/Push Activities** check boxes are selected. |

3. To the left of the **User Preferences** title, click ‹ to return to the last opened page.

> **Tip:**
>
> You should also make sure that notifications from these email addresses aren't blocked, filtered, or marked as spam:
>
> - `no-reply@oracle.com`
> - `no-reply@oraclecloud.com`
> - `noreply@developer.ocp.oraclecloud.com`
> - `noreply@developer.ocp.oraclecloud<realm_number>.com`, such as `noreply@developer.ocp.oraclecloud5.com`

# Set Up a Git Client

You can use use any Git client, such as the Git command-line interface (CLI), to access Git repositories from your computer. However, you cannot access projects, issues, and builds from a Git client.

# Git Command-Line Interface

Before you can use a Git client to access your project's Git repository, you must first install and configure it on your computer. The Git command line-interface (CLI) is the most popular Git client.

Here's how to download, install, and configure the Git CLI:

1. Download and install the Git CLI.

   On Windows, use the Git Bash CLI to access project Git repositories. You can download Git Bash (version 1.8.x or later) from `http://git-scm.com/downloads`.

   On Linux and Unix, install Git using the preferred package manager. You can download Git for Linux and Unix from `http://git-scm.com/download/linux`.

2. The VB Studio pages display your username and email address as the committer's name and email ID. Configure variables to set up your name and email address:

   - To configure your user name, set the `user.name` variable:

     ```
     git config --global user.name "John Doe"
     ```

   - To configure your email address, set the `user.email` variable:

     ```
     git config --global user.email "johndoe@example.com"
     ```

   - To disable SSL or configure the proxy server, set the `http.sslVerify` or `http.proxy` variables:

     ```
     git config --global http.sslVerify false
     ```

     ```
     git config --global http.proxy http://www.testproxyserver.com:80/
     ```

> **Tip:**
>
> To find out the value of a variable, use the `git config <variable>` command:
>
> ```
> git config user.name
> ```

# Upload Your Public SSH Key

Before you can connect to a Git repository using SSH, you must first generate a private-public RSA SSH key pair and upload the public key to VB Studio. If you use multiple computers to access Git repositories, you need to generate an SSH key pair from each computer and upload its public key.

VB Studio uses SSH protocol, version 2. For user authentication, VB Studio supports the ssh-ed25519 and the ssh-rsa algorithms for key types. The ed25519 algorithm is preferred. For the host verification, ssh-ed25519, ecdsa-sha2-nistp521, and ssh-rsa are supported. The RSA algorithm is the original one; the other two were added as alternatives for verifying the host.

> **Note:**
>
> In 24.04.0, there is an important change in SSH-RSA handling. VB Studio no longer supports the SSH-RSA (SHA1) signature scheme. This can be a problem for the old OpenSSH client (OpenSSH_7.4p1) and some of the older clients (OpenSSL 1.0.2k-fips). Modern clients, such as OpenSSH_8.9p1 Ubuntu-3ubuntu0.4, and OpenSSL 3.0.2, support rsa-sha2-512 and rsa-sha2-256.

## Generate an SSH Key

To generate an RSA SSH key pair, you can use any SSH client, including the Git CLI.

These steps assume you're using Git CLI to generate the SSH keys:

1. Open the Git CLI.
2. On the command prompt, enter `ssh-keygen -m PEM -t rsa`.

   > **Tip:**
   >
   > A key generated using `ssh-keygen -t rsa` would work as long as Git clone operations are performed from inside the Git command line interface. However, if you take that key out of the Git cli to use elsewhere, it won't work for several services, including Oracle Cloud services and Azure. The recommended way to generate a key is by always including the `-m PEM` switch, as in `ssh-keygen -m PEM -t rsa`. This ensures that the key will work for most services, including Azure and Oracle OCI.

   To generate a larger key, enter `ssh-keygen -m PEM -t rsa -b 4096`.
3. When prompted, enter a file name for the key and press **Enter**.

   If you don't want to specify a file name, leave the name blank and press **Enter**. By default, the key pair files are saved as `id_rsa.pub` and `id_rsa` in the `.ssh` sub-directory under the Git `HOME` directory. For example, on Windows, the files are saved in `C:\Users\<USER_PROFILE>\.ssh\`.
4. Enter a passphrase and press **Enter**. If you don't want to specify a passphrase, leave it blank and press **Enter**.

   When prompted to confirm the passphrase, enter the same passphrase. If you didn't specify a passphrase earlier, leave it blank and press **Enter**.

By default, Git CLI access the `C:\Users\<USER_PROFILE>\.ssh\` directory to locate the private key. If you are using another Git client, you may need to configure it to access the private SSH key. Check your Git client's documentation to find out how to do that.

## Add the Public SSH Key to Your VB Studio Account

After generating an SSH private-public key pair, add the public key to your User Preferences page's **Authentication** tab.

1. On the computer where you generated the SSH key pair, navigate to the directory where the public key is saved.

2. Open the public key file in a text editor, select the contents, and copy them to the clipboard.
3. In VB Studio, click the user avatar and select **Preferences**.
4. Click the **Authentication** tab.
5. Click **Add Key**.
6. In the New SSH Key dialog box, enter a unique name and paste the SSH key that you copied in Step 2.
7. Click **Create**.
8. To the left of the **User Preferences** title, click ‹ to return to the last opened page.

# Set Up Token-Based Authentication

You can create temporary OAuth access tokens to enable access to VB Studio project operations, including Git, Maven, and NPM actions, from your VB Studio account preferences.

Keep the following in mind when creating access tokens:

- Copy a token immediately after you generate it and paste it somewhere that you can access it later. This is the only opportunity you'll have to access the token.
- An access token can't be edited once it's been created. If you want to change the permissions for a token, you'll need to delete it and then create a new one.
- If the permissions for a user are changed at the project level, this can affect the token-based authentication.
- If a per-user limit on the number of personal access tokens has been set at the organization level, this can affect how many new tokens you can create and make it necessary for you to perform some maintenance by removing expired or no-longer used tokens before you can create new tokens.

> **Note:**
>
> If you're using curl to work with REST APIs and federated users in a multi-stripe Oracle Cloud Applications environment, one approach is to authenticate with a bearer token.
>
> Bearer authentication (sometimes called token authentication) is performed by sending a security token back to the server in every authorization header when requesting protected resources. To pass the bearer token in the Curl authorization header, add the following command line parameter to the Curl request:
>
> ```
> -H "Authorization: Bearer {token}"
> ```
>
> For example, the following example shows the general format for sending a bearer token with a Curl POST request. POST data is passed with the -d command-line option, and the authorization header; the bearer token is passed with the -H command-line option.
>
> ```
> curl -X POST https://[REST-endpoint]
>      -H "Authorization: Bearer {token}" -d "[post data]"
> ```

The format for a Curl GET request would be very similar, except it doesn't include -d command-line option.

**Create a Personal Access Token**

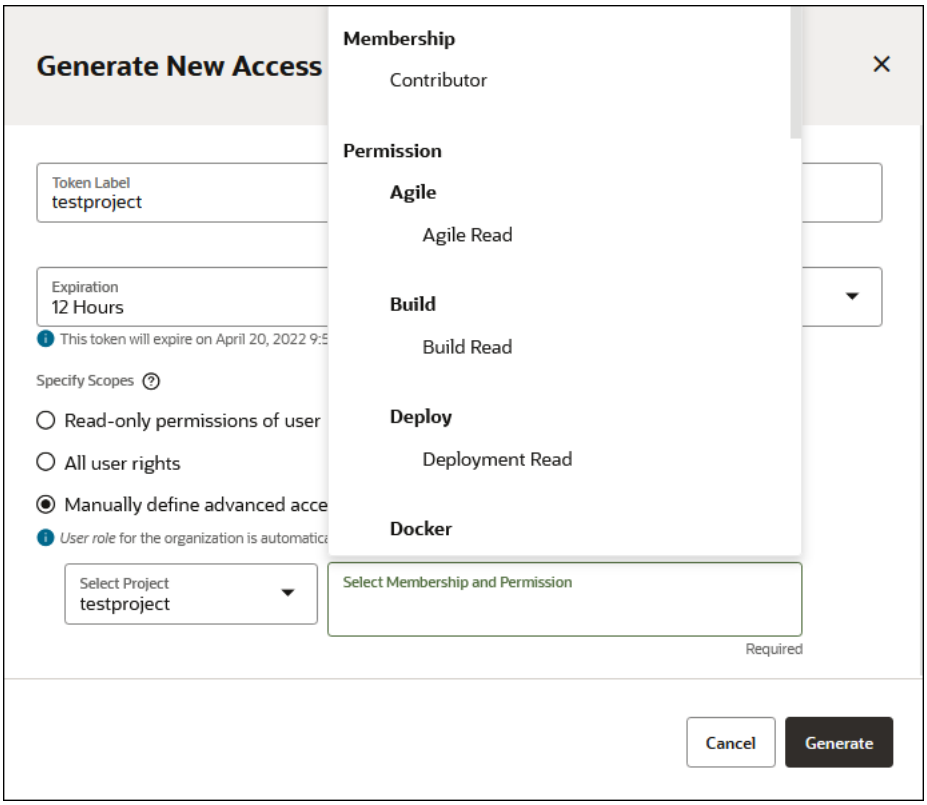
To create a VB Studio authentication token:

1. Click your VB Studio user avatar and select **Preferences**.
2. Click the **Personal Access Tokens** tab.
3. Click **+ Access Token**.

   If you are an org administrator and see a warning message that you have reached the maximum number of tokens allowed per user, to create more you can either click **Dismiss** and then click the **Delete** icon to delete some of your existing tokens or click **Open Access Token Settings** and change the maximum value for the entire organization.

   If you aren't an org administrator and see the message about having too many tokens, you can either click **Dismiss** and then click the **Delete** icon to delete some of your existing tokens or ask one of the org administrators shown in the window to change the maximum number of tokens allowed per user.
4. In **Token Label**, add a label for the token.
5. In **Expiration**, choose the expiration time period. You can choose from the list or set a custom time.
6. Specify the permission scope:
   - **Read-only permissions of user**: Gives you read-only access to all projects you are part of.
   - **All user rights**: Applies the token to all of your current project user permissions. This doesn't include ability to create or delete tokens or modify any other profile preferences.

- **Manually define advanced access level**: Select a project and click in the **Membership and Permission** field to select permissions that you want to assign to the token.




7. Click **Generate**.

   An email notification will be sent to the address of the user registered for the account. This notification is sent for both expiring and non-expiring tokens.

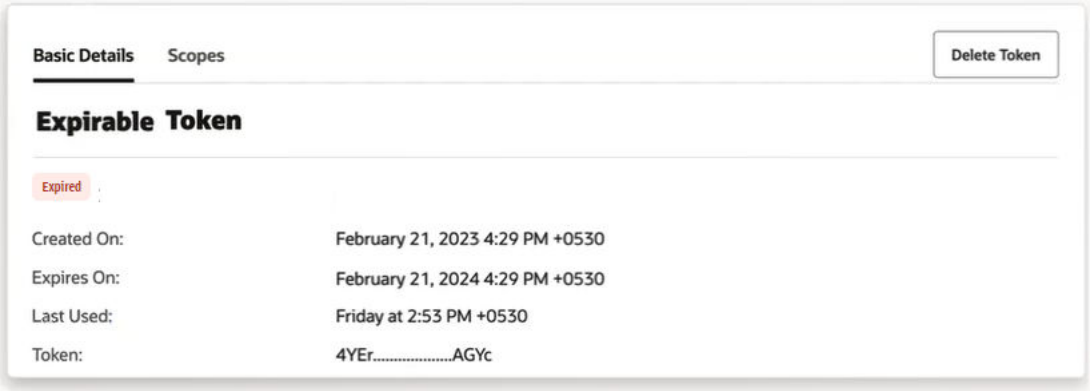
8. Copy the token and paste it in a text file that you can access later. You won't be able to access the token after the dialog is dismissed.

9. Click **Dismiss**.

You can view the details for each token you've created by selecting the token from the list shown in the Personal Access Tokens tab and viewing the information in the **Basic Details** and **Scopes** tabs.

### Delete a Token

From the **Basic Details** tab, you can press the **Delete Token** button to delete a Personal Access Token (PAT). This will remove any permissions granted by the token.

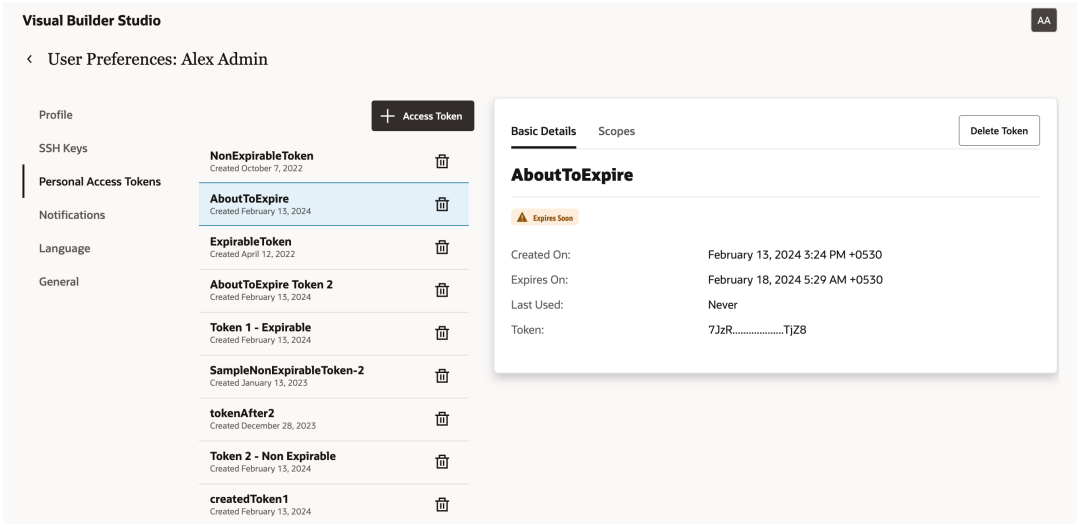



After a token has been deleted, applications or scripts that use that token will no longer have access to the VB Studio API.

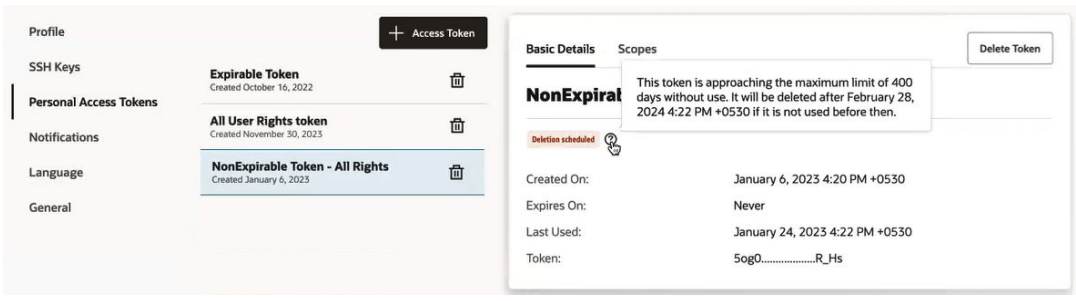
### Automatic Token Maintenance Performed by VB Studio

VB Studio tracks when tokens were last used. You can see this information in the **Basic Details** tab, next to the **Last Used** label. If an unexpired token isn't used for 400 days, VB Studio will automatically delete it. VB Studio also deletes expired tokens 30 days after they expire. In both cases, VB Studio gives token owners plenty of advance notice first:

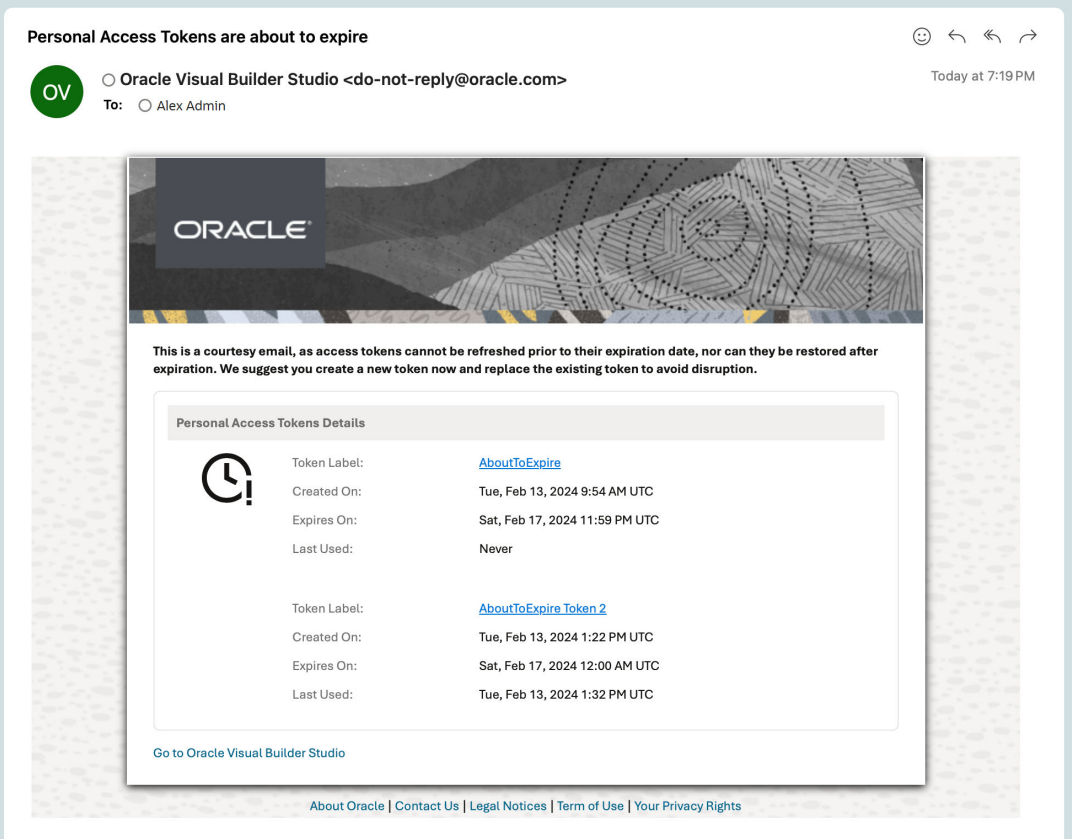
- The **Basic Details** tab for the selected token displays an "Expires Soon" status label 5 days before the actual expiration date. The actual expiration date is listed in the panel.




- The **Basic Details** tab for the selected token displays an "Deletion scheduled" status label next to a token that hasn't been used for 395 days.

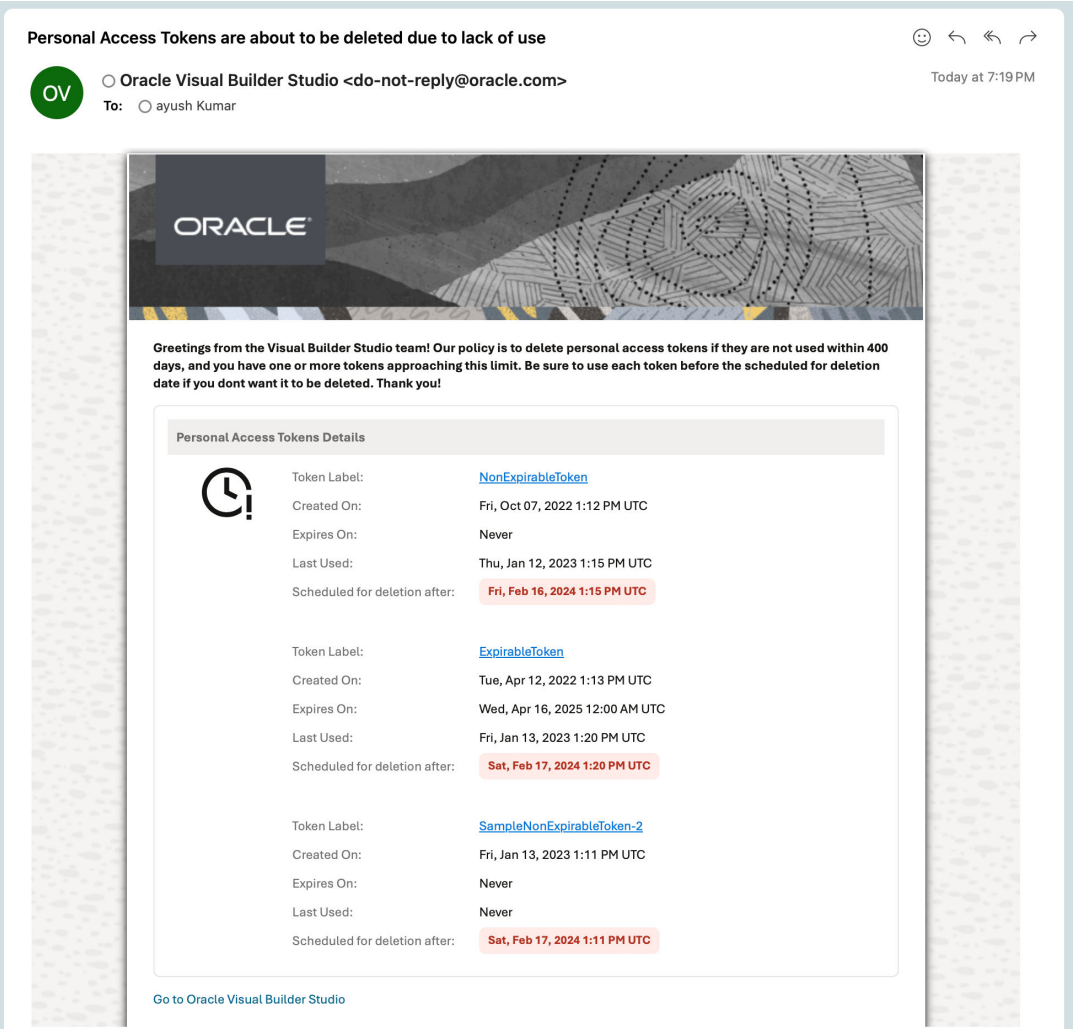


- An email notification is sent to the user 5 days before the user's personal access token is set to expire.




Then, if an unused token isn't used in the allotted time, VB Studio revokes it.

- An email notification is sent to the user 395 days after its last use (5 days before it will be automatically removed due to 400 days of non-use).




# See the News Banner

A banner with the latest news from the VB Studio team is displayed on the **Organization** and **Project Home** pages' header.

To navigate between news pages, click the navigation buttons. To expand or collapse the banner, use the **Expand** or **Collapse** icons. To close the banner, click **Close**.

If the banner isn't visible, here's how to enable it:

1. On the User Preferences page, click the **General** tab.
2. Select the **Show News Banner on Organization and Project Home** check box.
3. To the left of the **User Preferences** title, click to return to the last opened page.

# Part II
# Common Tasks

You'll perform most of these common tasks as you create and work with projects in Oracle Visual Builder Studio.

**Topics:**


- Work with Projects
- Manage Source Code Files with Git
- Create and Use Environments
- Use Workspaces and the Designer
- Build Your Applications
- Deploy and Manage Your Applications

3

# Work with Projects

After signing in to Visual Builder Studio (hereafter called VB Studio), you can create a project, open a shared project, or open a project you're a member of.

After creating a project, you can add users, set up environments, and manage the projects. Optionally, you can start creating issues to track your work, and defining Agile boards and sprints to manage your team's progress.

## What Is a Project?

A project collects all the people, tools, and processes you need to complete a unit of work in VB Studio.

A unit of work might be a software effort dedicated to a new payroll app for use internally, or perhaps a new version of your company's primary external offering. Regardless, you use a project to host source code files, track issues, collaborate on code, and build and deploy your applications. A project can host multiple Git repositories, and each Git repository can have multiple branches and hundreds of code files.

Within the project, you can require team members to create a merge request for each branch of the Git repository and ask reviewers to review and approve the code. You can create and configure multiple build jobs to generate different project artifacts that you can deploy to Oracle Cloud or your on-premise web server.

### What Are Project Memberships?

Here's what you can do in VB Studio pages, depending on your project membership status:

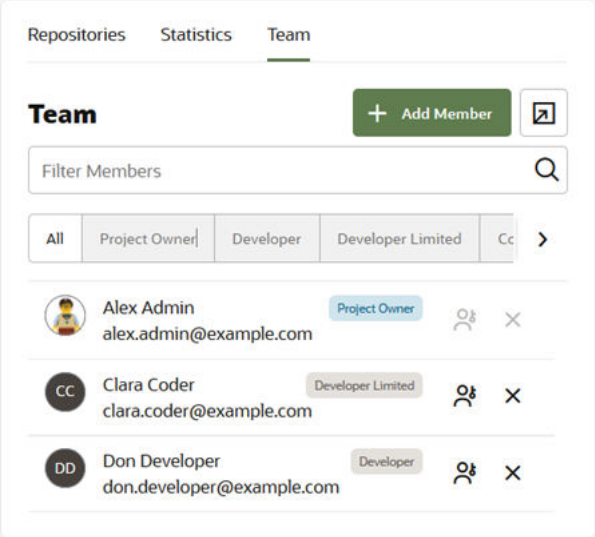
| This project membership... | Enables a user to: |
|---|---|
| Organization Administrator | Access and manage all projects of the organization, and set connections to OCI and OCI Classic.<br><br>**Note:**<br>In any project, an organization administrator can assign himself or herself the Project Owner role. This will grant them the same project permissions as any other project owner. An organization administrator can't even open a project unless they are a member and, even then, it is the project membership type that determines their project permissions.<br><br>Individuals that head up organizations and members of the IT department are assigned this membership. The individual that creates the service instance is automatically assigned this membership. |

| This project membership... | Enables a user to: |
|---|---|
| Project Owner | Access all components of the project and perform project management and administrative tasks, such as adding or removing Git repositories, managing project users, assigning default reviewers, and configuring Webhooks.<br>Project managers and team leaders are assigned this membership. The individual that creates the project is automatically assigned this membership. |
| Developer | Access most components of the project, but has restricted project management or administrative actions.<br>Senior developers are assigned this membership. |
| Developer Limited | Access some components of the project, but has restricted job configuration, environment management, board management, project management, and administrative actions.<br>Junior developers and members of the QA team are assigned this membership. |
| Contributor | Access the project's components in read-only mode but can enter comments, update issues, view wikis, and download build artifacts.<br>Usually, new developers, technical writers, and other members are assigned this membership. |
| Non-member | Access the same things as the Contributor membership, but for shared projects only. |

# Check Your Project Membership

To see what membership you have for a project, sign in to VB Studio and click the project's name:

| To discover | Do this: |
|---|---|
| If you're the organization administrator | In the branding bar, click the user avatar, then click **Contacts**. If you see your name listed under Organization Admins, you're assigned the Organization Administrator role.<br>You could also check by selecting **Organization** in the navigation menu and displaying the Organization page. If you see a series of tabs (Projects, OCI Account, Build Executor Templates, Build Executors, Component Exchange, Groups, Properties, Activities), you're an organization administrator. If you only see a list of projects, you've been assigned some other role. |
| Projects you own | On the Organization or Project Home page, click the **Owner** toggle button. |
| Projects in which you're a member | On the Organization or Project Home page, click the **Member** toggle button. |

| To discover | Do this: |
| --- | --- |
| Your project membership status | 1. In the left navigator, click **Project Home** .<br>2. On the right side of the page, click the **Team** tab.<br>If you see the **Owner** tag next to your name, you're a project owner. If you don't see the **Owner** tag next to your name, you're a project member. The Developer, Developer Limited, or Contributor tag next to your name indicates your project membership. If you can't find your name, you're a non-member.<br>For example, in this graphic, Alex Admin is a project owner, Clara Coder is a Developer Limited, and Don Developer is a Developer:<br>  |

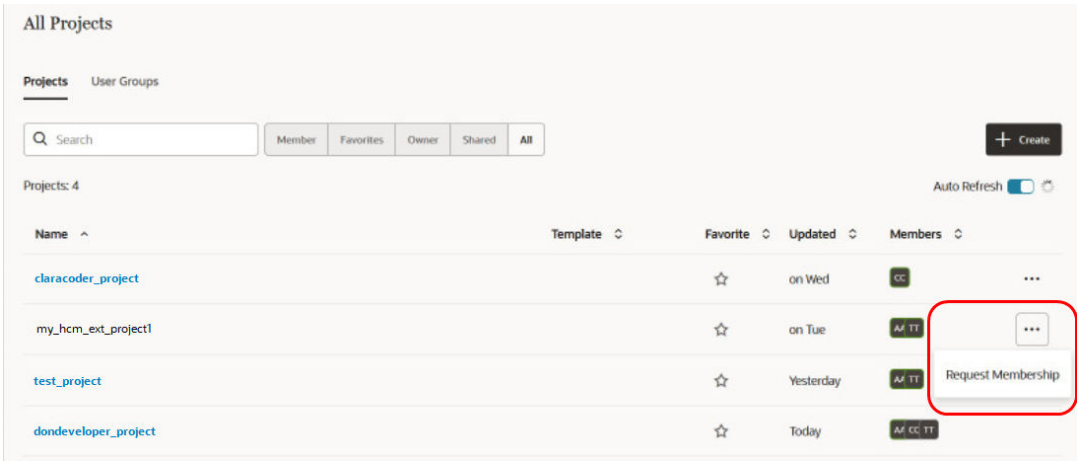
# Request Membership in a Project You Can't Access

If you're a member of a project, you can access the project. If you aren't a member of a shared project, you have read only access to that project. If you aren't a member of a discoverable private project, you can see the name of the project and the owners but you can't open the project. You can, however, request membership in the project by following the instructions in this topic. If you aren't a member of a private project that isn't discoverable, you'll never even see the project in the project list.

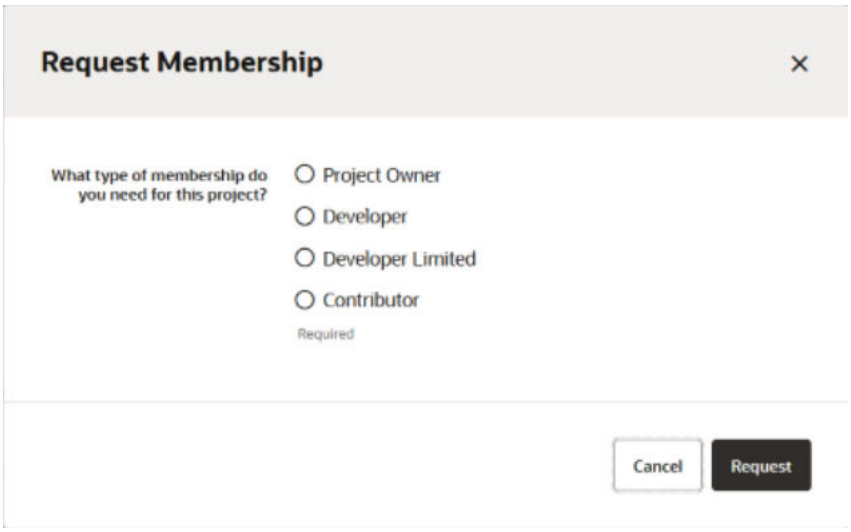
The ability to request project access and membership is provided for non-administrative users. That is, it is intended to be used by individuals who aren't project owner, who will receive the request and grant (or deny) access and membership.

If you can see the shared or discoverable private project in the project list but aren't a member:

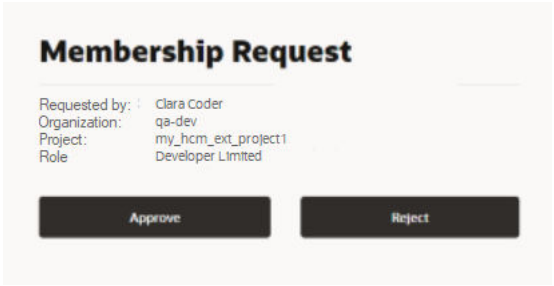
1. Click the Action menu *** and select the **Request Membership** option to request access to the project.




The **Request Membership** dialog box is displayed.

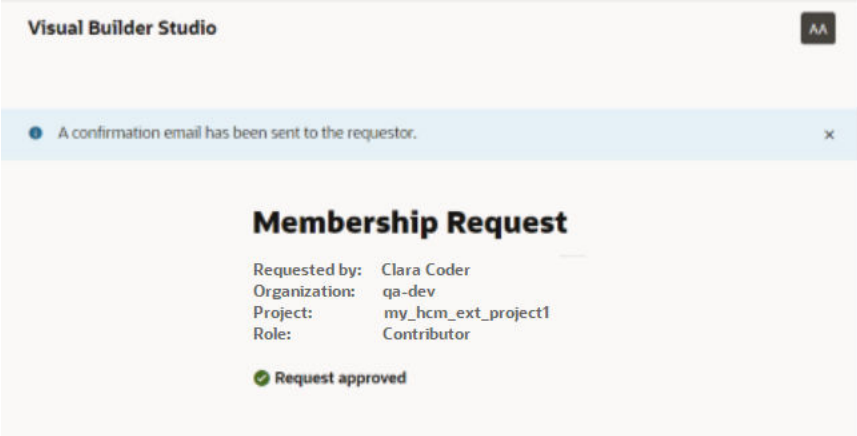



2. Select one of the four membership roles and then click **Request**.
   You're required to select one of the memberships. If you don't, the request won't be sent and you'll see a message reminding you to select a membership type.
3. VB Studio sends an email to the project owner(s). The email contains a link to a page that prompts the project owner to approve or reject the request.

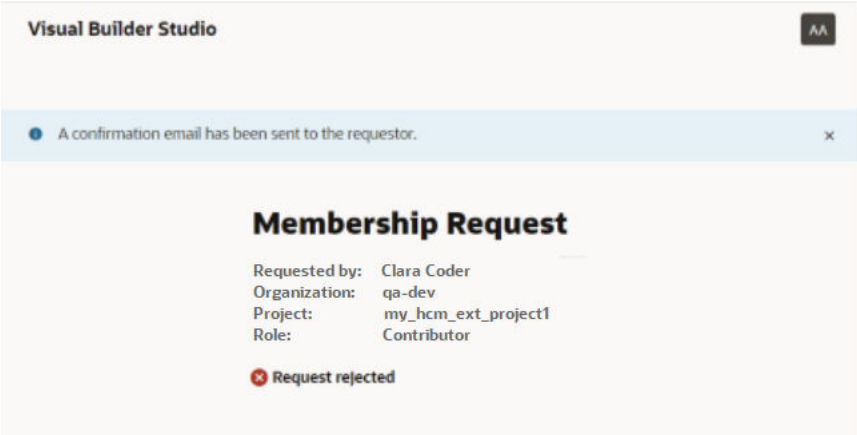



If the project has multiple project owners that get the email, after the first owner responds, subsequent owners that click on the link will see a message that the request has already been addressed and it has already been approved/rejected by another project owner.

4. If the project owner approves the membership request, this screen will be displayed.




5. If the project owner rejects the membership request, this screen will be displayed.




6. You'll receive an email that tells you if your request was approved or denied. This email looks like the confirmation mail you get when a project owner manually adds you from the **Team** tab on the Project page.
7. Begin to work on the project.

## What Can I Do at the Project Level?

The actions you can perform in VB Studio depend on your project membership status.

> **Note:**
>
> In any project, an organization administrator can assign himself or herself the Project Owner role. This will grant them the same project permissions as any other project owner. An organization administrator can't even open a project unless they are a member and, even then, it is the project membership type that determines their project permissions.

Here's what you can do in VB Studio pages, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| **Git Repository** | | | | | |
| Create, edit, or delete a Git repository | ✓ | ✓ | ✓ | | |
| Push commits to a Git repository | ✓ | ✓ | ✓ | | |
| Clone and read files from a Git repository | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Maven Repository** | | | | | |
| Write or upload files to the Maven repository | ✓ | ✓ | ✓ | | |
| Read files from the Maven repository | ✓ | ✓ | ✓ | ✓ | ✓ |
| **NPM Registry** | | | | | |
| Read packages from the project's NPM registry | ✓ | ✓ | ✓ | ✓ | ✓ |
| Publish packages to the project's NPM registry | ✓ | ✓ | ✓ | | |
| **Snippets** | | | | | |
| View snippets | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create, edit, and delete snippets | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Merge Requests** | | | | | |
| Search and view merge requests | ✓ | ✓ | ✓ | ✓ | ✓ |
| Add comments to merge requests | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create, merge, and close merge requests | ✓ | ✓ | ✓ | | |
| Add and remove reviewers | ✓ | ✓ | ✓ | | |
| Add and remove linked issues | ✓ | ✓ | ✓ | | |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Add and remove linked builds | ✓ | ✓ | ✓ | | |
| Start linked builds | ✓ | ✓ | ✓ | | |
| **Issues** | | | | | |
| Search and view issues | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create and edit issues | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Agile** | | | | | |
| View boards (backlog, active sprints and issues, but not reports) | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create, copy, and edit one's own boards | ✓ | ✓ | | | |
| Copy and edit boards owned by others | ✓ | ✓ | | | |
| Delete one's own boards | ✓ | ✓ | | | |
| Delete boards owned by others | ✓ | | | | |
| View sprint reports | ✓ | ✓ | | | |
| **Wikis** | | | | | |
| View wiki pages | ✓ | ✓ | ✓ | ✓ | ✓ |
| Add comments to wiki pages but cannot change page content | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create, edit, and delete wiki pages | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Builds** | | | | | |
| View builds | ✓ | ✓ | ✓ | ✓ | ✓ |
| Run builds and receive build notifications | ✓ | ✓ | ✓ | | |
| Create, configure, and delete jobs | ✓ | ✓ | | | |
| Create, configure, and delete pipelines | ✓ | ✓ | | | |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| **Environments** | | | | | |
| View environments | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create, configure, and delete environments | ✓ | ✓ | | | |
| Add and remove instances | ✓ | ✓ | | | |
| **Workspaces** | | | | | |
| View workspaces | ✓ | ✓ | ✓ | | |
| Add and manage workspaces | ✓ | ✓ | ✓ | | |
| **Releases** | | | | | |
| View releases | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create, edit, and delete releases | ✓ | ✓ | | | |
| **Docker** | | | | | |
| View a Docker registry | ✓ | ✓ | ✓ | ✓ | ✓ |
| Write to a Docker registry | ✓ | ✓ | ✓ | | |
| **Projects** | | | | | |
| Access to project administration pages for all features, team administration | ✓ | | | | |

## Permissions for the Project Home Page

Depending on your project membership status, you have varying privileges on the Project Home page, particularly in the areas of Recent Activities, the Repositories tab, the Graphs and Statistics tab, and the Team tab.

> **Note:**
>
> In any project, an organization administrator can assign himself or herself the Project Owner role. This will grant them the same project permissions as any other project owner. An organization administrator can't even open a project unless they are a member and, even then, it is the project membership type that determines their project permissions.

### Workspaces Panel

Here's what you can do in the Workspaces panel, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View the project's workspaces | ✓ | ✓ | ✓ | | |
| Open a workspace in the Designer | ✓ | ✓ | ✓ | | |

### Environments Panel

Here's what you can do in the Environments panel, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View the project's environments | ✓ | ✓ | ✓ | ✓ | ✓ |
| Select an environment and open it in the Environments page | ✓ | ✓ | ✓ | ✓ | ✓ |

### Recent Activities Feed

Here's what you can do in the Recent Activities feed, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View the recent activities feed | ✓ | ✓ | ✓ | ✓ | ✓ |
| Filter the recent activities feed | ✓ | ✓ | ✓ | ✓ | ✓ |
| Search activities | ✓ | ✓ | ✓ | ✓ | ✓ |

### Repositories Tab

Here's what you can do on the Repositories tab, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a Git repository | ✓ | ✓ | ✓ | | |
| View Git repositories | ✓ | ✓ | ✓ | ✓ | ✓ |
| Mark a Git repository as your favorite | ✓ | ✓ | ✓ | ✓ | ✓ |
| Copy a Git repository's URL | ✓ | ✓ | ✓ | ✓ | ✓ |
| Browse the Maven repository | ✓ | ✓ | ✓ | ✓ | ✓ |
| Copy the Maven repository's URL | ✓ | ✓ | ✓ | ✓ | ✓ |
| Browse a project's NPM registry and copy its URL | ✓ | ✓ | ✓ | ✓ | ✓ |
| Browse a Docker registry and copy its URL | ✓ | ✓ | ✓ | ✓ | ✓ |

### Statistics Tab

Here's what you can do on the Statistics tab, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View personal statistics for issues, commits, and merge requests | ✓ | ✓ | ✓ | ✓ | ✓ |

### Team Tab

Here's what you can do on the Team tab, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View project users | ✓ | ✓ | ✓ | ✓ | ✓ |
| Export the users list | ✓ | ✓ | ✓ | ✓ | ✓ |
| Add or remove a user | ✓ | | | | |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Change a user's project role | ✓ | | | | |

## What Else Can I Do Within a Project?

Depending on your project membership status, your permissions allow you varying privileges.

> **Note:**
>
> In any project, an organization administrator can assign himself or herself the Project Owner role. This will grant them the same project permissions as any other project owner. An organization administrator can't even open a project unless they are a member and, even then, it is the project membership type that determines their project permissions.

Visit the topics listed under **On This Page**, to the right, to see what privileges come with each project membership.

## Permissions for Working with Git Repositories

Here's how you can interact with Git repositories, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a hosted Git repository, add an external Git repository, or import a Git repository | ✓ | ✓ | ✓ | | |
| Clone the Git repository | ✓ | ✓ | ✓ | ✓ | ✓ |
| Push commits to the Git repository | ✓ | ✓ | ✓ | | |
| Set a Git repository's default branch | ✓ | ✓ | ✓ | | |
| Set Git repository branch restrictions | ✓ | ✓ | ✓ | | |
| View file contents and commits | ✓ | ✓ | ✓ | ✓ | ✓ |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create or delete branches and tags | ✓ | ✓ | ✓ | | |
| Compare files and revisions | ✓ | ✓ | ✓ | ✓ | ✓ |
| Lock or protect a branch | ✓ | ✓ | ✓ | | |
| Download an archived branch or a tag | ✓ | ✓ | ✓ | ✓ | ✓ |
| Add comments to commits | ✓ | ✓ | ✓ | ✓ | ✓ |
| View graphical history of commits | ✓ | ✓ | ✓ | ✓ | ✓ |
| Index a Git repository | ✓ | ✓ | ✓ | | |
| Delete a Git repository | ✓ | ✓ | ✓ | | |

Both Developers and Developer Limiteds can use the VB Studio UI to create repositories, but must use the Git command line to delete the repositories created with the UI. Project Owners can delete repositories by using the Project Administration pages.

Non-members can clone a Git repository and make commits to it, but they can't push the commits to the remote Git repository.

## Permissions for Merge Requests

Here's what you can do with merge requests, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a merge request | ✓ | ✓ | ✓ | | |
| Add comments or reply to a comment | ✓ | ✓ | ✓ | ✓ | ✓ |
| Subscribe to merge request email notifications | ✓ | ✓ | ✓ | ✓ | ✓ |

Reviewers are automatically subscribed to merge request email notifications. Non-members can subscribe to email notifications by opening the merge request and clicking the **CC Me** button.

When a merge request is created, all reviewers are assigned the Reviewer role. The individual that submits the request is assigned the Requester role. This table lists additional actions Reviewers and Requesters can perform:

| Action | Requester | Reviewer | Other Users |
|---|---|---|---|
| Add or remove reviewers | ✓ | ✓ | |
| Approve or reject a merge request | ✓ | ✓ | |
| Merge branches or close a merge request | ✓ | ✓ | |

A Project Owner can approve or reject a merge request, merge branches, or close a merge request, even if he or she isn't assigned the Reviewer role.

## Permissions for Maven

Here's how you can interact wioth a project's Maven repository, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Browse the Maven repository | ✓ | ✓ | ✓ | | |
| Download artifacts from the Maven repository | ✓ | ✓ | ✓ | | |
| Upload artifacts to the Maven repository | ✓ | ✓ | ✓ | | |
| Search artifacts in the Maven repository | ✓ | ✓ | ✓ | | |
| Configure the automatic cleanup of the Maven repository | ✓ | | | | |

## Permissions for Docker

Here's how you can interact with Docker registries, repositories, and images, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Link an external Docker registry | ✓ | | | | |
| View external Docker registries, their repositories, and images | ✓ | ✓ | ✓ | ✓ | ✓ |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Download an external Docker registry repository's image manifest | ✓ | ✓ | ✓ | ✓ | ✓ |
| Delete an external Docker registry repository's image manifest | ✓ | ✓ | ✓ | ✓ | ✓ |

## Permissions for Jobs, Builds, and Pipelines

Here's how you can interact with a project's jobs, builds, and pipelines, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Set up connections to OCI Compute and OCI Object Storage | | | | | |
| Create, configure, and manage build executor templates | | | | | |
| Add and manage VM build executors | | | | | |
| Create a job | ✓ | ✓ | | | |
| View job details | ✓ | ✓ | ✓ | ✓ | ✓ |
| View job configurations | ✓ | ✓ | ✓ | ✓ | ✓ |
| Edit job configurations | ✓ | ✓ | | | |
| Run a build | ✓ | ✓ | ✓ | | |
| Download artifacts | ✓ | ✓ | ✓ | ✓ | ✓ |
| View logs, including build console, audit, and Git polling logs | ✓ | ✓ | ✓ | ✓ | ✓ |
| Disable or delete a job | ✓ | ✓ | | | |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a pipeline | ✓ | ✓ | | | |
| Configure a pipeline | ✓ | ✓ | | | |
| View a pipeline's instances | ✓ | ✓ | ✓ | ✓ | ✓ |
| Delete a pipeline | ✓ | ✓ | | | |

## Permissions for Releases

Here's how you can interact with releases, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a release | ✓ | ✓ | | | |
| Clone or edit a release | ✓ | ✓ | | | |
| Delete a release | ✓ | ✓ | | | |

## Permissions for Environments

Here's how you can interact with a project's environments, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View an environment | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create/delete an environment | ✓ | ✓ | | | |
| Add an instance to/remove an instance from an environment | ✓ | ✓ | | | |

Project owners are the only project team members who can delete environments that were created in a different identity stripe. If a project member with the Developer role attempts this operation, they'll see an error message informing them that they can't delete an environment that has resources from another identity stripe.

An IDCS resource cannot be removed from an environment if the instance is in another identity stripe. Nobody can remove an IDCS resource from a different identity stripe from an environment, but the project owner can still delete the entire environment. In addition, an IDCS resource from a second identity stripe cannot be added into the same environment.

## Permissions for Workspaces

Here's how you can interact with a project's workspaces, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View a workspace | ✓ | ✓ | ✓ | | |
| Create and manage a workspace | ✓ | ✓ | ✓ | | |

Managing a workspace includes such tasks as renaming, assigning a new owner to, and deleting a workspace.

## Permissions for Issues

Here's how you can interact with a project's issue tracking system, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create an issue | ✓ | ✓ | ✓ | ✓ | ✓ |
| Update an issue | ✓ | ✓ | ✓ | ✓ | ✓ |
| Create and configure issue products | ✓ | | | | |
| Create and configure issue tags | ✓ | | | | |
| Create and configure issue custom fields | ✓ | | | | |

## Permissions for Agile Boards and Sprints

Project memberships Here's how you can interact with a project's Agile boards and sprints, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a board | ✓ | ✓ | | | |
| Use Scrum board | ✓ | ✓ | ✓ | ✓ | ✓ |
| Use Kanban board | ✓ | ✓ | ✓ | ✓ | ✓ |

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| View burndown charts and sprint reports | ✓ | ✓ | ✓ | ✓ | ✓ |

When you create a board, you become the owner of the board. As the board owner, you have special privileges over the board and sprint-related actions:

| Action | Board Owner | Other Project Members | Non-Member |
|---|---|---|---|
| Add issues to a sprint | ✓ | | |
| Start a sprint | ✓ | | |
| Delete a sprint | ✓ | | |
| Configure the board | ✓ | | |

## Permissions for Wikis

Here's how you can interact with wiki pages, depending on your project membership status:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Set the organization's default wiki markup language | | | | | |
| Set the project's wiki markup language | ✓ | | | | |
| Create a wiki | ✓ | ✓ | ✓ | ✓ | ✓ |
| View a wiki page | ✓ | ✓ | ✓ | ✓ | ✓ |
| Edit a wiki page | ✓ | ✓ By default | ✓ By default | | |
| Delete a wiki page | ✓ | ✓ By default | ✓ By default | | |

The Project Owner (or member, if allowed) can grant edit and delete rights over a wiki page to all users, or restrict edit access to members or Project Owners only.

## Permissions for Snippets

Here's how you can interact with snippets, depending on your project membership:

| Action | Project Owner | Developer | Developer Limited | Contributor | Non-member |
|---|---|---|---|---|---|
| Create a snippet | ✓ | ✓ | ✓ | | |
| View snippet files | ✓ | ✓ | ✓ | ✓ | ✓ |
| Insert a snippet file or copy a snippet file's text | ✓ | ✓ | ✓ | ✓ | ✓ |
| Clone the snippet Git repository | ✓ | ✓ | ✓ | ✓ | ✓ |
| Push the commits to the snippet's Git repository | ✓ | ✓ | ✓ | | |
| Download the archive of the snippet's Git repository | ✓ | ✓ | ✓ | ✓ | ✓ |
| Like a snippet | ✓ | ✓ | ✓ | ✓ | ✓ |
| Add comments | ✓ | ✓ | ✓ | ✓ | ✓ |

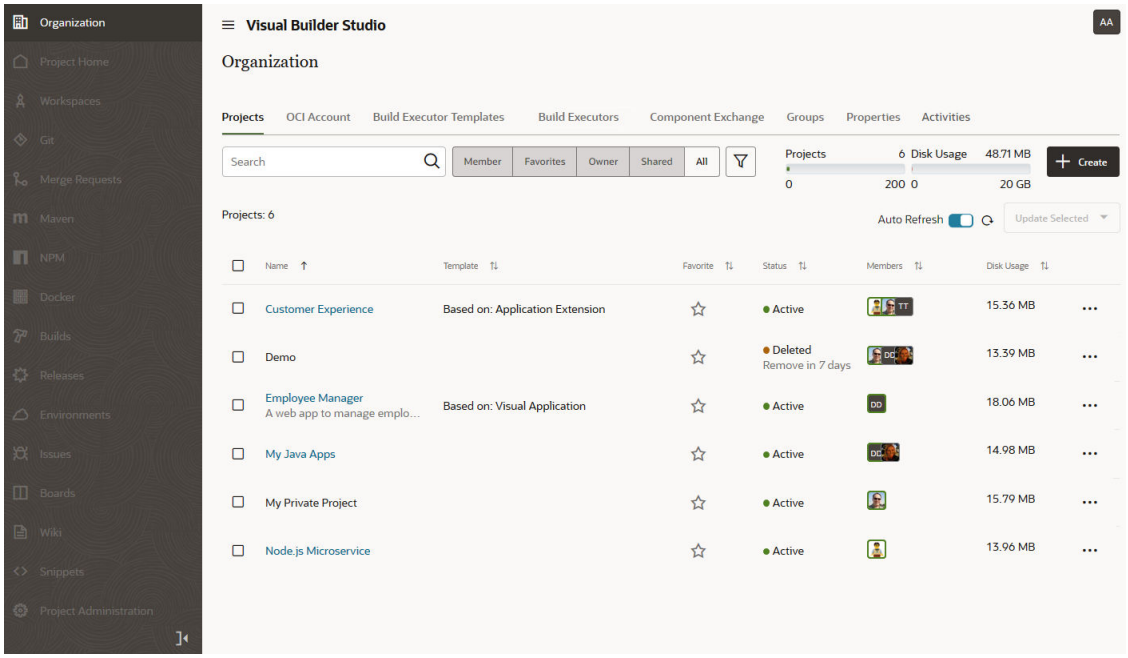
A non-member can clone the snippet's repository and make commits, but can't push the commits to the snippet's Git repository.

If you're a snippet owner, you can also perform these actions:

| Action | Snippet Owner | Other Project Members | Non-Member |
|---|---|---|---|
| Add, update, or remove snippet files | ✓ | | |
| Create a snippet from the selection | ✓ | | |
| Delete a snippet | ✓ | | |

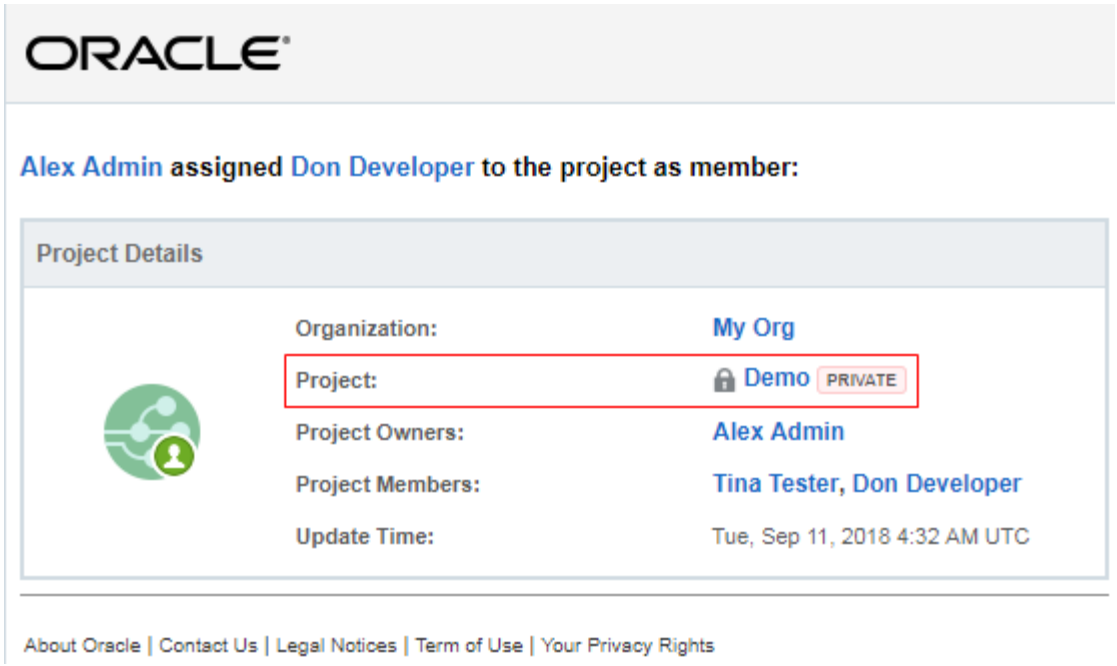
# Open a Project

You can open a project only if you're a member or an owner, or if the project is shared. To open a project, click its name as it appears on the Organization page. To search for a project, use the filter toggle buttons or the search box:

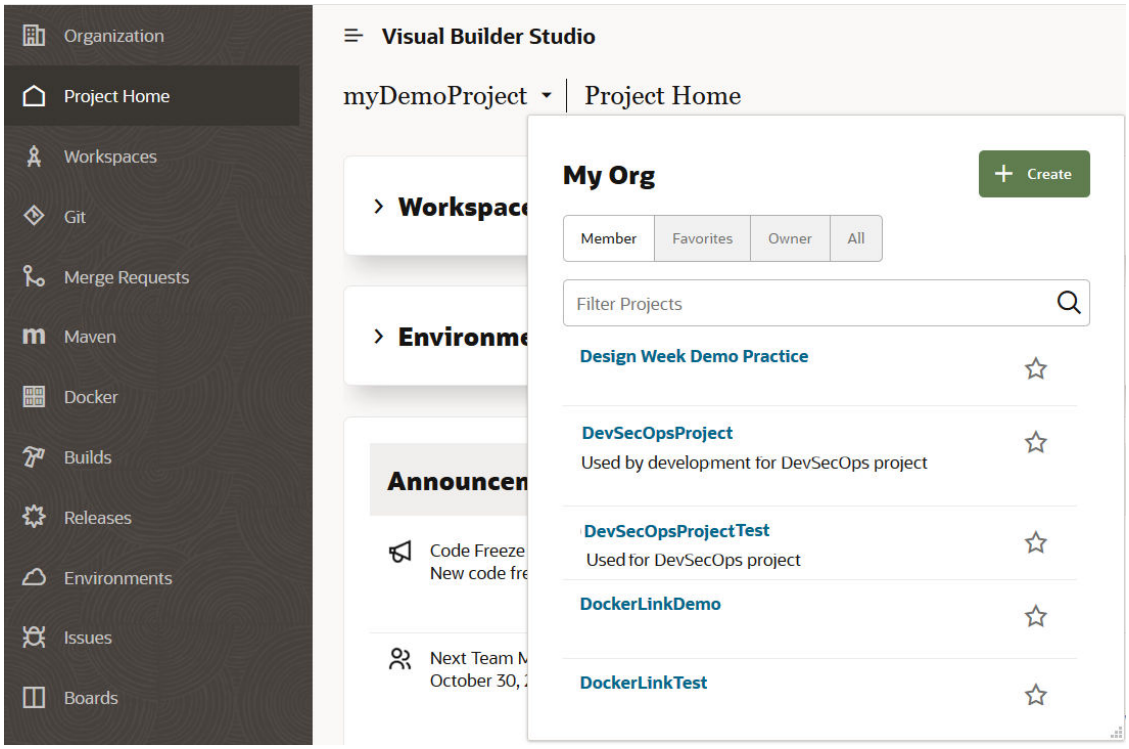



To quickly access a project, click **Favorite** ☆and add it to your favorites list. To see your favorite projects, click the **Favorites** toggle button.

If you're invited to join a project, you'll find the project link in the email you received when you were added to the project.




To switch to another project from an open project, click ⌄ next to the project name. From the menu, click the project name to open it.




After opening the project, you land on the **Project Home** page.

# Create a Project

From the Organization page, you can create different types of projects:

- Empty Project
- With an Initial Git Repository
- From an Imported Project
- From a Project Template
- Without an Environment

> **Tip:**
>
> Newly created projects are listed in the Recent Activities stream on the Organization page's Activities tab. This tab is available to organization administrators only.

## Empty Project

If you haven't decided which applications you want to upload, or want to start from scratch, create an empty project that has no pre-configured Git repository or any other artifact:

1. On the Organization page, click **+ Create**.
2. On the Project Details page of the New Project wizard, in **Name** and **Description**, enter a unique project name and a project description.

3. In **Security**, select the project's privacy:
   a. Select **Private** to restrict access to project members only.

      Select the **Discoverable** checkbox to allow organization members that aren't org admins or project members to see basic information, such as name and owner contact information, about your private project. Private projects that aren't discoverable won't be exposed to non-members.

   b. Select **Shared** to make the project code, wiki docs, tasks, and builds available to anyone inside your organization.
4. Click **Next**.
5. On the Template page, select **Empty Project**, and click **Next**.
6. On the Project Properties page, from **Wiki Markup**, select the project's wiki markup language.

   Project team members use the markup language to format wiki pages and comments.
7. Click **Next**.
8. On the **Team** page:
   a. Click **Add Members** and select users or groups to add to the project, from the list displayed, if you know they may work in this project.

   b. Select the membership (Project Owner, Developer Full Access, Developer Limited Access, or Contributor) that the members you're adding will have in the project:

      See What Are Project Memberships? for more information about each membership.

   c. Click **Add**.

   d. Repeat substeps a, b, and c for different users and groups with various membership types, if needed.
9. Click **Finish**.

## With an Initial Git Repository

If you plan to upload application files soon after you create a project, you should create a project with an initial Git repository. You can choose the Git repository to be empty, populated with a readme file, or populated with data imported from another Git repository:

1. On the Organization page, click **+ Create**.
2. On the Project Details page of the New Project wizard, in **Name** and **Description**, enter a unique project name and a project description.
3. In **Security**, select the project's privacy:
   a. Select **Private** to restrict access to project members only.

      Select the **Discoverable** checkbox to allow organization members that aren't org admins or project members to see basic information, such as name and owner contact information, about your private project. Private projects that aren't discoverable won't be exposed to non-members.

   b. Select **Shared** to make the project code, wiki docs, tasks, and builds available to anyone inside your organization.
4. Click **Next**.
5. On the Template page, select **Initial Repository**, and click **Next**.

6. On the Project Properties page, from **Wiki Markup**, select the project's wiki markup language.

   Project team members use the markup language to format wiki pages and comments.

7. Click **Next**.

8. On the **Team** page:

   a. Click **Add Members** and select users or groups to add to the project, from the list displayed, if you know they may work in this project.

   b. Select the membership (Project Owner, Developer Full Access, Developer Limited Access, or Contributor) that the members you're adding will have in the project:

      See What Are Project Memberships? for more information about each membership.

   c. Click **Add**.

   d. Repeat substeps a, b, and c for different users and groups with various membership types, if needed.

9. In **Initial Repository**, specify how to initialize the Git repository.

   - If you prefer a blank repository or want to push a local Git repository to the project, select **Empty Repository**.
   - Some Git clients can't clone an empty Git repository. Select **Initialize repository with README file** if you're using such a client. VB Studio creates a `readme.md` file in the Git repository.

     You can edit the contents of the `readme.md` file after creating the project, or delete the file if you don't want to use it.
   - To import a Git repository from another platform such as GitHub or Bitbucket, or from another project, select **Import existing repository**.

     In the text box, enter the external Git repository's URL. If the repository is password protected, enter the credentials in **Username** and **Password**. Note that VB Studio doesn't store your credentials.

10. Click **Finish**.

## From an Imported Project

If you've created a project before and backed up its data to an OCI Object Storage bucket, you can create a project and import the data directly from the bucket.

To import project data from an OCI Object Storage bucket, you need this information:

- Target bucket's name
- Exported archive file's name
- Private key and fingerprint of a user who has the `BUCKET_INSPECT` (or `BUCKET_READ`) and `OBJECT_READ` bucket permissions
- Details of the compartment that hosts the bucket

After you have all the required input values, import the project:

1. On the Organization page, click **+ Create**.
2. On the New Project wizard's Project Details page, in **Name** and **Description**, enter a unique project name and a project description.

3. In **Security**, select the project's privacy setting:
   a. Select **Private** to restrict access to project members only.

      Select the **Discoverable** checkbox to allow organization members that aren't org admins or project members to see basic information, such as name and owner contact information, about your private project. Private projects that aren't discoverable won't be exposed to non-members.

   b. Select **Shared** to make the project code, wiki docs, tasks, and builds available to anyone inside your organization.
4. Click **Next**.
5. On the Template page, select **Import Project**, and click **Next**.
6. In the Project Properties page's Storage Connection section, in **Account Type**, select **OCI** and enter the required details:
   a. In **Tenancy OCID**, enter the tenancy's OCID copied from the Tenancy Details page.
   b. In **User OCID**, enter the user's OCID value (for a user that can access the bucket).
   c. In **Home Region**, select the OCI account's home region.
   d. In **Private Key**, enter the user's private key (for a user who can access the bucket).
   e. In **Passphrase**, enter the passphrase used to encrypt the private key. If a passphrase wasn't used, leave the field empty.
   f. In **Fingerprint**, enter the private-public key pair's fingerprint value.
   g. In **Compartment OCID**, enter the compartment's OCID copied from the Compartments page.
   h. In **Storage Namespace**, enter the storage namespace copied from the Tenancy Details page.

   > **Note:**
   >
   > If the project administrator selected the **Remember** checkbox when the OCI storage connection was established during a previous export operation, the credential fields in this step will automatically be populated with the saved data. See step 14 in Export Project Data to an OCI Object Storage Bucket Using OCI Credentials.
7. Click **Next**.
8. On the Project Properties page, from **Wiki Markup**, select the project's wiki markup language.

   Project team members use the markup language to format wiki pages and comments.
9. In **Container**, select the storage bucket where the data was exported.
10. In **File**, select the exported file.
11. Click **Next**.
12. On the **Team** page:
    a. Click **Add Members** and select users or groups to add to the project, from the list displayed, if you know they may work in this project.

    b. Select the membership (Project Owner, Developer Full Access, Developer Limited Access, or Contributor) that the members you're adding will have in the project:

See What Are Project Memberships? for more information about each membership.

c. Click **Add**.

d. Repeat substeps a, b, and c for different users and groups with various membership types, if needed.

13. Click **Finish**.

If the import fails, an empty project will be created. You can try to import the data again without creating a project. For details about the import, check the import log under **Project Administration**, in the **Data Export/Import** page's **History** tab.

## From a Project Template

Using a project template, you can quickly create a project with predefined and populated artifacts, such as Git repositories and build jobs. When you create a project from a project template, the defined artifacts of the project template are copied to the new project. If you don't want to use a copied artifact, you can delete it. Note that after you create a project from a template, updates made to the project template won't be reflected in the project you created.

These types of project templates are available:

| Project Template | Description |
|---|---|
| Public templates | The VB Studio team creates and manages the public templates. They are available to all users across all identity domains and are marked by a **Public Template** label. |
| Shared templates | Your organization users create and manage shared templates. They are listed by name and are available to all users of the organization. |
| Private templates | Not listed by name to general users, but accessible through their private keys. To create a project from a private project template, you must have its private key.<br>Private templates are visible by name only to the members of the project template. |

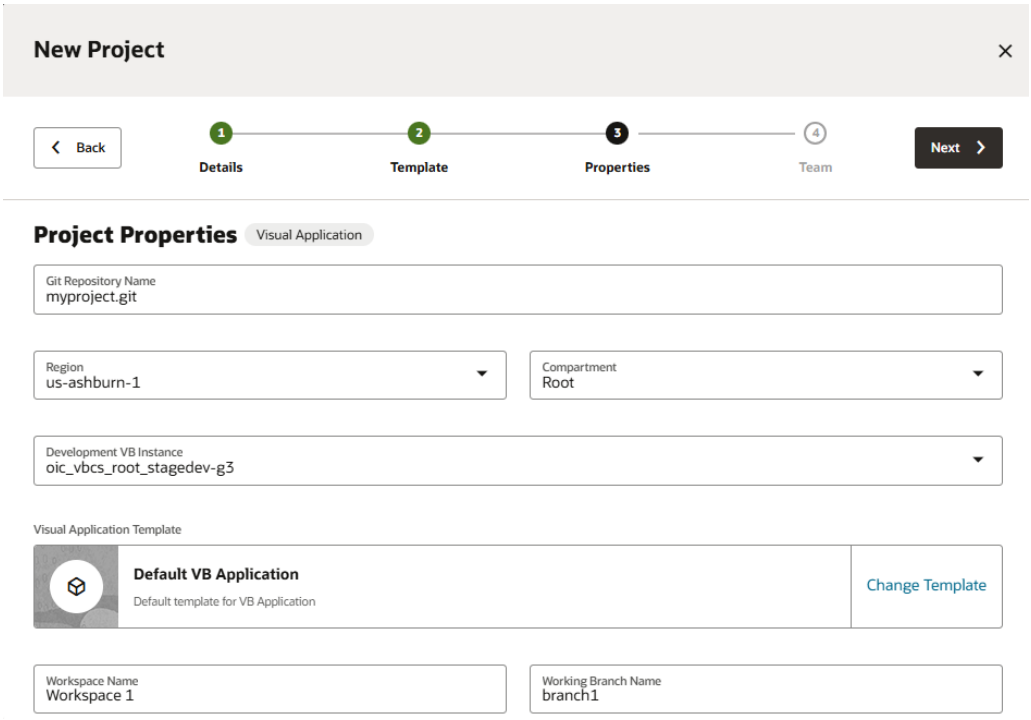
1. On the Organization page, click **+ Create**.
2. In the New Project wizard's **Project Details** dialog:
   a. In **Name** and **Description**, enter a unique project name and a project description.
   b. In **Security**, select the project's privacy setting:
      - Select **Private** to restrict access to project members only.
        Select the **Discoverable** checkbox to allow organization members that aren't org admins or project members to see basic information, such as name and owner contact information, about your private project. Private projects that aren't discoverable won't be exposed to non-members.
      - Select **Shared** to make the project code, wiki docs, tasks, and builds available to anyone inside your organization.
   c. In **Preferred Language**, specify the language for email notifications that project users would receive.

      To change the language of the user interface, update the language in your account preferences.
3. Click **Next**.
4. In the **Template** dialog, select the project template and then click **Next**:

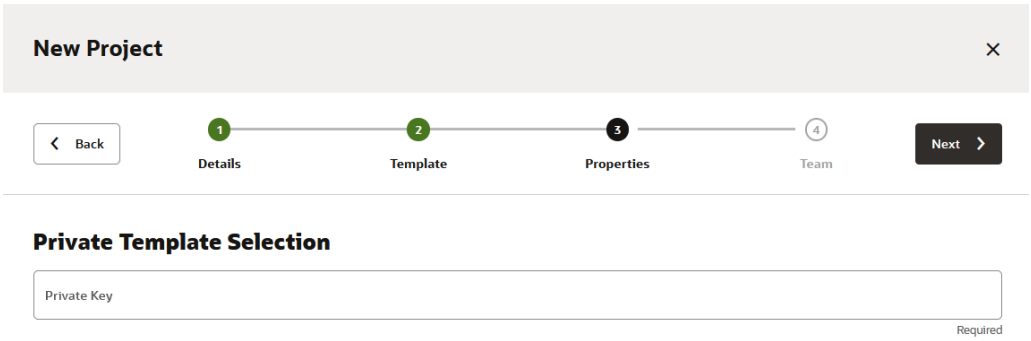
- To create a project for a visual application, select the **Visual Application** template and follow the instructions in the next step.
- To create a project from a private template, select **Private Template**, and follow the instructions in the next step.

> **Note:**
>
> To create a project for an extension, the recommended approach is to click the **Edit Page in Visual Builder Studio** link in the Oracle Cloud Application you are extending and follow the steps in the New App Extension wizard. See Create a Simple Extension. But if you don't want to configure an existing Oracle Cloud Application page, and instead would like to create your own application that lives as a peer alongside Oracle applications, you should create an empty project and manually set up the required artifacts. See Manually Create a Project for Extensions.

5. In the **Project Properties** dialog:
   - If you previously selected the **Visual Application** template, enter the Git Repository Name, select the Region and Compartment where your Visual Builder instance is (prepopulated based on your location), select the Development VB Instance and Visual Application template, and enter the Workspace Name and Working Branch Name.




   - If you previously selected **Private Template**, in the **Private Template Selection** dialog, enter the private key in **Private Key**.




6. Click **Next**.
7. In the **Team** dialog:
   a. Click **Add Members** and select users or groups to add to the project, from the list displayed, if you know they may work in this project.
   b. Select the membership (Project Owner, Developer Full Access, Developer Limited Access, or Contributor) that the members you're adding will have in the project:

      See What Are Project Memberships? for more information about each membership.
   c. Click **Add**.
   d. Repeat substeps a, b, and c for different users and groups with various membership types, if needed.
8. Click **Finish**.

In the new project, these artifacts are copied from the project template:

| Artifacts | Description |
|---|---|
| Git repositories | The project template's defined Git repositories are copied to the new project. You can use the copied Git repositories and modify their files, or delete them.<br><br>In the left navigator, click **Git** to view the copied Git repositories. |
| Build jobs and pipelines | All the project template's build jobs and pipelines are copied to the new project. You can change these jobs, create their copies, or delete them.<br><br>In the left navigator, click **Builds** to view the copied jobs and pipelines. |
| Wiki pages | All the project template's wiki pages are copied to the new project. You can change the wiki pages or delete them.<br><br>In the left navigator, click **Wiki** to see the copied wiki pages. |

| Artifacts | Description |
|---|---|
| Announcements | All the project template's active project announcements are copied to the new project. You can't edit the copied announcements since they are read-only, but you can activate or deactivate them.<br><br>In the left navigator, click **Project Administration** , and then click **Announcements** to activate or deactivate them. |
| Links | All the project template's link rules are copied to the new project. Link rules enable you to convert plain text to links when the text is entered in the commit and merge request comments.<br><br>In the left navigator, click **Project Administration** and then click **Links** to see the copied link rules. |

## Without an Environment

Projects almost always require an environment against which you develop and deploy your visual application—but you can create a project without an environment simply to explore VB Studio's visual development capabilities within the Designer.

> **Note:**
>
> When your project doesn't have an environment, you won't be able to create business objects or deploy (share and publish) your app—although you can click **Publish** in the header to merge your changes to the default branch (`main`) in the project's Git repository. To explore VB Studio without any restrictions, you need to add a Visual Builder instance to your VB Studio instance's environment.

To create a project without an environment:

1. On the Organization page, click **+ Create**.
2. In the New Project wizard's Project Details dialog, in **Name** and **Description**, enter a unique project name and a project description.
3. In **Security**, select the project's privacy:
   a. Select **Private** to restrict access to project members only.

      Select the **Discoverable** checkbox to allow organization members that aren't org admins or project members to see basic information, such as name and owner contact information, about your private project. Private projects that aren't discoverable won't be exposed to non-members.

   b. Select **Shared** to make the project code, wiki docs, tasks, and builds available to anyone inside your organization.
4. Click **Next**.
5. In the Template dialog, select the **Visual Application** project template, and click **Next**:
6. In the Project Properties dialog, fill out or select the following:
   a. In **Git Repository Name**, enter the name of the repository where your application will be stored or accept the default name.
   b. In **Region** and **Compartment**, accept the defaults.

c. In **Development VB Instance**, select **No Environment** to create a Visual Application workspace without a specific environment.

> **Note:**
>
> If this option isn't available to you, it means that this functionality has not been enabled in your environment. File a service request with Oracle Cloud Support.

d. In **Visual Application Template**, accept the **Default VB Application** template or click **Change Template** and select another template from the list displayed.

   The default application template does not create any artifacts, apps, or other resources. Other templates might create resources or apps that already include artifacts.

e. In **Workspace Name**. enter a name for your workspace or accept the default name.

f. In **Working Branch Name**, enter a name for your working branch or accept the default name.

7. Click **Next**.
8. In the **Project Team members** dialog:
   a. Click **Add Members** and select users or groups to add to the project, from the list displayed, if you know they may work in this project.
   b. Select the membership (Project Owner, Developer Full Access, Developer Limited Access, or Contributor) that the members you're adding will have in the project.
   c. Click **Add**.
   d. Repeat substeps a, b, and c for different users and groups with various membership types, if needed.
9. Click **Finish**.

# Set Project Protection

A project can be private or shared; it cannot be public. You can define a project's protection when you create it, or from its properties page later.

**Shared projects** can be accessed by all the organization's users. Any user can view the source code, create or update issues, edit wiki pages, and interact with project builds. However, only invited users can make updates to the source code in Git repositories, create and run build jobs, and perform deployment operations.

**Private projects** are accessible to invited users only but they can also be marked as discoverable. A **discoverable private project** allows organization members that aren't org admins or project members to see basic information about the private project. This information includes name and owner contact information. Private projects that aren't discoverable won't be exposed to non-members.

See What Are Project Memberships? to learn more about VB Studio project memberships.

# Manage Project Users and Groups

After creating a project, you'll probably want to add team members to collaborate with. You may also want to allow or limit their access to project data or actions they can perform on the project.

You must have the Project Owner project membership to add and manage project users (team members) and groups from the Project Home page's **Team** tab. If you're a team member with read/write permissions (a Project Owner, Developer, or Developer Limited member), you can:

- Export (download or copy to the clipboard) a CSV-, JSON-, and plain text-formatted users list that includes all the project's members, even those in groups.
  See Export the Users List.
- Create a local VB Studio group from the Organization page. See Manage Local VB Studio Groups for more information.

You can add users who are assigned the `DEVELOPER_ADMINISTRATOR` (Developer Service Administrator) or the `DEVELOPER_USER` (Developer Service User) identity domain role to your project. If your organization administrator has already created VB Studio user groups or imported Oracle Identity Cloud Service (IDCS) groups to VB Studio, you can add them too.

Note that you can't add or remove users from the imported IDCS group. Only your organization administrator can, from the IDCS Console. After users have been added to or removed from that IDCS group, you should be able to see the changes after the imported IDCS group syncs with VB Studio, which happens about every five minutes.

You'll also have to contact your organization administrator to add a user who doesn't have the required identity domain role. To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will be displayed.

When you add a user or a group, you assign one of these membership types:

- Project Owner
- Developer
- Developer Limited
- Contributor

Project users and groups are managed from the Project page's **Team** tab by performing these administrative tasks:

| If you want to: | Do this: |
|---|---|
| Add a user to the project | 1. In the left navigator, click **Project Home**.<br>2. Click the **Team** tab.<br>3. Click **+ Add Member**.<br>4. Click the **Username** drop-down list.<br>5. Under **Users**, select the user.<br>If you can't find a particular user, enter the user's name or username in the search box. As you begin typing, users matching the search term are displayed.<br>6. From the membership option types, select the user's membership.<br>7. Click **Add**. |

| If you want to: | Do this: |
| --- | --- |
| Add a group to a project | 1. In the left navigator, click **Project Home**.<br>2. Click the **Team** tab.<br>3. Click **+ Add Member**.<br>4. Click the **Username** drop-down list.<br>5. Under **Groups**, select the group.<br>6. From the membership option types, select the membership you want to assign to the group's members.<br>7. Click **Add**. |
| Add multiple users or groups to the project | 1. In the left navigator, click **Project Home**.<br>2. Click the **Team** tab.<br>3. Click **+ Add Member**.<br>4. Click the **Username** drop-down list.<br>5. From the drop-down list, select a user or a group. Click **Username** again to select another user or group.<br>If you can't find a particular user, enter the user's name or username in the search box. As you begin typing, users matching the search term are displayed.<br>6. From the membership option types, select the user's membership.<br>7. Click **Add**. |
| Change a user's or a group's project membership | To change a user's or a group's project membership, click the **Change Membership** icon . From the dropdown, select a new project membership (Contributor, Developer, Developer Limited, or Project Owner). |
| Remove a user or a group from the project | Before removing a user, change the ownership of any assigned issues and merge requests to another user.<br>Select the user or group you want to remove, then click **Remove**.<br>Note that you can't remove a user from a group. To do that, you need to contact your organization administrator. |

# Manage Local VB Studio Groups

Any user in the organization, even one who doesn't have organization administrator rights, can create a local VB Studio group. However, only an organization administrator with the DEVELOPER_ADMINISTRATOR VB Studio role can create an IDCS group and import it into VB Studio. So, any user with the DEVELOPER_USER VB Studio role (any user with a Project Owner, Developer, Developer Limited, or Contributor project membership) can create a local VB Studio group.

After you create a local VB Studio local group, you can add more members to or remove members from it, add the local group to projects, and delete local groups you've created. You can't modify or delete a local group you didn't create. When you add a local VB Studio group to a project, you can assign a common membership type to all of that group's users.

Email notifications are sent to users in a particular user group when that group is added to or removed from a project, or when that group's membership privileges are changed.

You can use the groups you define to:

- Restrict build access (see Configure Job Protection Settings)
- Restrict pipeline access (see Protect Your Pipeline: Restrict Who Can Start It Manually or Edit Its Configuration)
- Protect Git branches (see Protect a Branch)
- Select reviewers to review source code changes in merge requests (see Create a Merge Request and Add or Remove Reviewers)

This table describes the actions you can perform to create and manage local VB Studio groups.

| Action | How To |
|---|---|
| Create a VB Studio local group | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Click **+ Create Group**.<br>4. In **Type**, if not already selected, select the **VB Studio** tile.<br>5. In **Name** and **Description** (optional), enter the group's name and description.<br>6. To see all members you can add, click the members list. Users who are assigned the DEVELOPER_ADMINISTRATOR or the DEVELOPER_USER IDCS role are displayed.<br>7. From the users drop-down list, select users to add to the group.<br>If you can't find a particular user, enter the user's name or username in the search box. As you type, the drop-down list displays users matching the search term.<br>8. Click **Create**. |
| See a local group's members | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate the VB Studio group and click the user gravatars in the **Members** column. |
| Add members to an existing VB Studio local group you've created | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate the VB Studio group and click **Add Member to the Group**.<br>4. Click the members drop-down list.<br>5. Select the user from the drop-down list.<br>If you can't find the user, enter the user's name or username in the search box. As you type, users matching the search term are displayed.<br>6. Click **Add**.<br>7. In the **Members** tab, verify the added members.<br>8. Click **Close**. |

| Action | How To |
| --- | --- |
| Remove members from a VB Studio local group | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate and double-click the VB Studio local group you created.<br>4. In the **Members** tab, select the members to remove.<br>5. Click **Remove Members**.<br>6. Click **Remove Members** to confirm. |
| Add a local group to a project | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate the VB Studio group and click **Add Group to a Project**<br>4. From the project drop-down list, select the project.<br>5. From the roles list, select the role you want to assign to the group's members.<br>6. Click **Add**.<br>7. Click **Close**. |
| Remove a local group from a project | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate and double-click the VB Studio group.<br>4. In the **Projects** tab, select the projects to remove.<br>5. Click **Remove Group from Projects**.<br>6. Click **Remove Group from Projects** to confirm. |
| See the projects a local group is added to | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate the VB Studio group and click the group's **Projects** column to see the list of projects.<br>4. Click **Close**. |
| Edit a local group's name or description | 1. In the left navigator, click **Organization**.<br>2. Click the **Groups** tab.<br>3. Locate and double-click the VB Studio group.<br>4. Click **Actions** and select **Edit**.<br>5. Update the group's name and description<br>6. Click **Save**.<br>7. Click **Close**. |

| Action | How To |
| --- | --- |
| Delete a local group you've created | 1. In the left navigator, click **Organization** .<br>2. Click the **Groups** tab.<br>3. Locate the VB Studio group.<br>4. Click **Delete**.<br>5. Click **Delete** to confirm. |

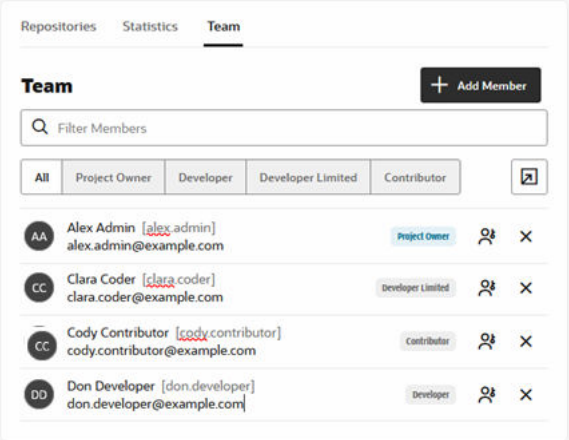
# Export the Users List

All users can export the project's users list, but project owners and administrators will use it the most. These lists can be used in compliance reports to show what the users' roles are (who has access to source code), their email addresses are, and so forth. This feature also comes in handy when customers want to migrate to another VB Studio instance in a different data center. Administrators can use generate these lists to and use them to easily set up the users in the new instance.

To export the project's list of users:

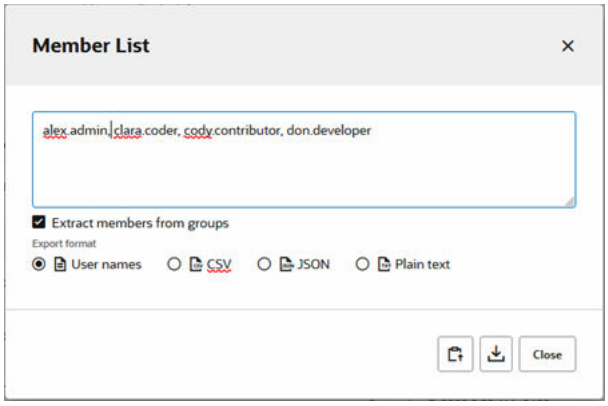
1. From the Project Home page, choose a project and then select the **Team** tab.

   The Team panel displays.




2. Click **Generate a list of current project members** .

   The Member List panel displays.



3. Select the **Extract members from groups** checkbox (default) to extract individual members from groups or show the groups only.
4. Under Export format, select one of the following:
   - **User names** shows a list of users by "Username".
   - **CSV** (Comma-Separated Values) shows a comma-separated list of users using the "Username","Name","Email","Gravatar hash","Project membership","Member type" format, including the header row as the first line:

     ```
     "Username","Name","Email","Gravatar hash","Project membership","Member
     type"
     "alex.admin","Alex
     Admin","alex.admin@example.com","f16503adcb57babf7af69512d650748e","Proj
     ect Owner","user"
     "mary.jane","Mary
     Jane","mary.jane@example.com","834a227063b4c50c5b0af3564ba9b2d5","Projec
     t Owner","user"
     "clara.coder","Clara
     Coder","clara.coder@example.com","a8b5f1d293f46f48d7011c7612072917","Dev
     eloper Limited Access","user"
     "cody.contributor","Cody
     Contributor","cody.contributor@example.com","277b45bbea3341d3de14dfeb19f
     6b8d7","Contributor","user"
     "don.developer","Don
     Developer","don.developer@example.com","5e99c123bab8c0b3e8a529a76ff71d3a
     ","Developer Full Access","user"
     ```

     The Gravatar hashes have been anonymized, by replacing a single random alphanumeric character in the hash with a random alphanumeric character (0-9 or a-f) so the hashes presented here don't represent hashes that can be reconstructed.
   - **JSON** (JavaScript Object Notation) shows the users list in the following format (line breaks added for readability):

     ```
     [{"username":"alex.admin","fullname":"Alex
     Admin","email":"alex.admin@example.com",
     "gravatar":"f16503adcb57bab87af69512d650748e","membership":["Project
     Owner"],"type":"user"},
     {"username":"mary.jane","fullname":"Mary
     Jane","email":"mary.jane@example.com",
     "gravatar":"834a227063b4c50c5b0af3564b99b2d5","membership":["Project
     Owner"],"type":"user"},
     {"username":"clara.coder","fullname":"Clara
     Coder","email":"clara.coder@example.com",
     "gravatar":"a8b5f1d293f46f48d7011c76120a2917","membership":["Developer
     Limited Access"],"type":"user"},
     {"username":"cody.contributor","fullname":"Cody
     Contributor","email":"cody.contributor@example.com",
     "gravatar":"277b45bbeaa341d3de14dfeb19f6b8d7","membership":
     ["Contributor"],"type":"user"},
     {"username":"don.developer","fullname":"Don
     Developer","email":"don.developer@example.com",
     "gravatar":"5e99c123ba18c0b3e8a529a76ff71d3a","membership":["Developer
     Full Access"],"type":"user"}]
     ```

- **Plain text** shows the list of users in the "Name","Username","Email"format.

5. Click **Copy to clipboard** to copy the list of users to the clipboard.
6. Click **Download file** to download the file locally.
7. Click **Close** when you're done.

# What Else Can You Do with Your Project?

After you create your project, add users, set the privacy level, and define environments, there are several other things you can add, depending on how you will use and manage your project:

- You may want to use issues to track and manage tasks, defects, and features (see Track and Manage Tasks, Defects, and Features ) or use Agile boards to manage and update issues and help plan your team's workflow (see Use Agile Boards to Manage and Update Issues).
- You may want to encourage collaboration between team members by using merge requests to review, critique, and approve source code changes (see Review Source Code with Merge Requests), by using wikis to document project information (see Co-Author Wikis ), or by using snippets to share reusable pieces of source code (see Share and Use Code Snippets).
- You may want to use the project's Maven repository for managing binary files and dependencies (see Manage Binaries and Dependencies with Maven) or use the project's NPM registry to download and/or publish private Javascript packages using Node.js/NPM command line tools (see Use the Project's NPM Registry).
- You may want to use webhooks to integrate use event notifications with Jenkins, GitHub applications, Slack, and PagerDuty (see Send Notifications to External Software Using Webhooks) or link external Docker registries to your project (see Access External Docker Registries).

Or, you may simply want to continue on your current path and see how to set up Git repositories, if needed (see Manage Source Code Files with Git).

# Manage Projects

As an Oracle Visual Builder Studio (VB Studio) project owner, you can perform various project-wide actions, such as edit a project's name and description, configure the project as a project template, create announcements, set project tags, manage repositories, and configure link rules.

## Delete a Project

It's a good practice to delete inactive projects when they're no longer needed. Doing this frees up storage space and cuts down the overall number of projects in the organization, thereby saving resources.

There are two places in the VB Studio interface where project owners (and organization administrators) can delete projects:

- Delete a Project from the Project Administration Properties Page
- Delete a Project from the Organization Page

> **Tip:**
>
> Deleted projects are listed in the Recent Activities stream on the Organization page's Activities tab. This tab is available to organization administrators only.

## Delete a Project from the Project Administration Properties Page

A project owner can delete a project from the **Project Administration : Properties** page.

To delete a project:

1. Open the project.
2. In the left navigator, click **Project Administration** .
3. Click **Properties**.
4. In the Properties page, click **Delete Project**.
5. In the Delete Project dialog, select the **I understand that my project will be permanently deleted** check box.
6. Click **Delete**.

## Delete a Project from the Organization Page

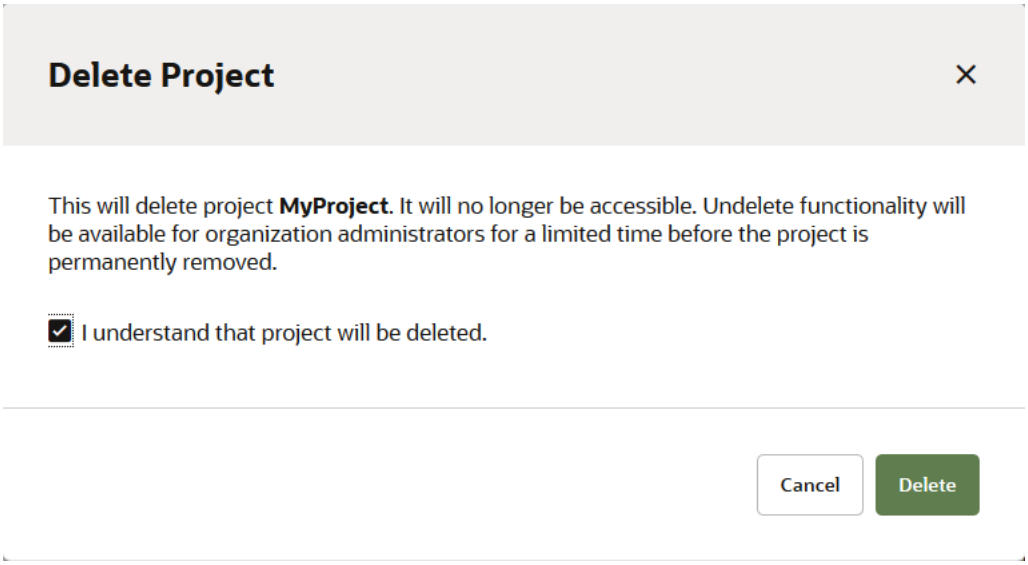
A project owner (or organization administrator) can delete a project from the **Projects** tab on the **Organization** page:

1. In the left navigator, click **Organization** .
2. Click **Projects**.
3. (Optional) In the **Projects** tab, select **Owner** to show a project list that contains all the projects that you own.

   If you skip this step, you'll notice that some projects don't display an action menu. You don't own those projects.
4. Click the three dots *** at the end of the row, and select **Delete** to delete the selected project.

   The Delete Project dialog is displayed. **Cancel** is available initially but **Delete** is only available after you select the checkbox to acknowledge the pending deletion.

   You'll only see this checkbox if you delete a project as the project owner (not as an organization administrator) or, if you are *permanently* deleting the project as an organization administrator. When an organization administrator simply deletes a project, the checkbox isn't shown.
5. Select the checkbox, and then click **Delete**.




Organization administrators have additional deletion options. From the project list, they can select a single project and then click the three dots *** at the end of the row, or select one or more projects and then click **Update # Selected Projects**, to access these options:

- Delete an active project
- Postpone the permanent deletion of a deleted project
- Remove a deleted project (or projects) forever
- Undelete a deleted project

The Recent Activities feed on the Organization page's Activities tab records the above actions so that organization administrators can monitor how and when projects are deleted and undeleted.

## Export and Import Project Data

To perform a backup, a project owner can export the project's data to an OCI Object Storage bucket in any data center. Then later, the exported project data can be imported into the same project (or a different one) in the same data center (or a different one).

> **Note:**
>
> OCI Classic Storage will be decommissioned in March of 2025, at which time you will lose access to this storage. Exporting data to OCI Classic is no longer available, although you can still import data until March 2025. Please consider moving your data to OCI Storage as soon as possible.

To export or import data, you'll need to set up a connection to OCI Object Storage. Contact your OCI administrator to get the required input values to set up the connection to the OCI Object Storage bucket. To successfully connect, you must provide either OCI credentials or a pre-authenticated request URL.

The values you need are explained in the import and export sections that follow.

# What Project Data Does VB Studio Automatically Export?

Before exporting your project's data, you should be aware that VB Studio doesn't export all the artifacts automatically. You'll still have to manually export any artifacts and data that weren't exported automatically.

Here's a list of the artifacts that are exported by VB Studio and the ones you'll have to export manually:

| Artifact | Exported? | Notes |
|---|---|---|
| Project users | No | When you export a project's data, its users are not exported. However, all the data associated with the usernames (issue ownership and reviewers of a merge request, for example) will be preserved.<br>After you import the project's data to another project, all the data associated with the username will automatically be restored after you add a user with the same username to the project. |
| User's favorite settings or personal preferences | No | |
| Hosted Git repositories | Yes | |
| Mirrored public external Git repositories | Yes | |
| Mirrored private external Git repositories | No | Password-protected external Git repositories aren't exported.<br>After you import the project's data to another project, you must add each external private Git repository. |
| Branch restrictions | Yes | |
| Merge Requests | Yes | |
| Default reviewers of a branch | Yes | After you import the project's data to another project, default reviewers will be automatically added after you add the same users to the target project. |
| Workspaces | No | Repositories that are associated with your user's workspaces will be exported, but the workspaces themselves will not be. |
| Maven artifacts | No | |
| Linked Docker registries | No | |
| Build jobs | Yes | All builds of jobs are exported, along with their logs and artifacts. If a job retains an excessive number of builds, it will adversely affect the export process and will require a large amount of storage space in your bucket. Configure a job to retain a *reasonable* number of builds before you export the project. |
| Build job's history and artifacts | Yes | |
| Name of the build job's build executor template | No | |
| Pipelines | Yes | |
| Releases | Yes | |
| Deployment configurations | No | |

| Artifact | Exported? | Notes |
|---|---|---|
| Environments | No | |
| Issues | Yes | |
| Agile boards | Yes | |
| Wiki pages | Yes | |
| Snippets | Yes | |
| Project template definition | No | |
| Announcements | No | |
| Webhooks | No | |
| RSS/ATOM feeds | No | |
| Link rules | No | |
| Project tags | Yes | |
| Issue products and components | Yes | |
| Default owners of issue components | Yes | After you import your project's data to another project, owners will be automatically activated after the same users are added to the target project. |
| Issue custom fields | Yes | |
| Named passwords/private keys | Yes | |

**Note:**

The passwords are securely exported and imported as keystore binary data.

## Export to and Import from an OCI Object Storage Bucket

If you're a project owner in an organization that runs VB Studio on OCI, you'll export project data to and import exported project data from an OCI Object Storage bucket.

Exported project data isn't encrypted and can be downloaded from the bucket. Your OCI administrator will already have set up an OCI Object Storage bucket for the project, and which users can read from or write to it. It may be a common bucket that's used for all the organization's projects, but more likely it's separate buckets for each project, which allows archive files to be organized without being grouped with archive files from other projects.

You can connect to the OCI Object Storage bucket using OCI credentials or a pre-authenticated request (PAR) URL:

- The preferred method is to use OCI credentials since they're permanent and don't typically expire.
- Using a PAR URL is simpler, although that convenience comes at a cost, since PAR URLs can fail due to various reasons and can be difficult to troubleshoot. For example, if the OCI administrator who gives you the PAR URL ever leaves the company, you'll need a new PAR URL.
  PAR URLs are also considered less safe than OCI credentials because you must store PAR URLs someplace *outside* Object Storage, and anyone with access to the URL has

access to the bucket. For this reason, always ensure you store a PAR URL in a safe, recoverable place.

For more information about PAR URLs, see Object Storage Pre-Authenticated Requests.

## Export Project Data to an OCI Object Storage Bucket Using OCI Credentials

You must be a Project Owner to export project data. When you export project data, VB Studio will export data to an archive file in the specified OCI Object Storage bucket, connected using OCI credentials.

To export project data to an OCI Object Storage bucket, you need the following:

- Name of the target bucket
- Private key and fingerprint of a user who can write objects to the bucket
- Details of the compartment that hosts the bucket

> **Note:**
>
> If you are an Oracle Cloud Applications customer, contact Oracle Support for the details you'll need to complete steps 6-13.

Here's how to export your project's data:

1. Open the VB Studio project.
2. In the left navigator, click **Project Administration** .
3. Click **Data Export/Import**.
4. Click the **Job** tab.
5. In **Account Type**, select **OCI**.

   Users that are extending Oracle Cloud Applications are strongly encouraged to contact Oracle Support to get the values they will need to complete steps 6-13.
6. In **Tenancy OCID**, enter the tenancy's OCID copied from the Tenancy Details page.
7. In **User OCID**, enter the user's OCID value who can access the bucket.
8. In **Home Region**, select the home region of the OCI account.
9. In **Private Key**, enter the private key of the user who can access the bucket.
10. In **Passphrase**, enter the passphrase used to encrypt the private key. If no passphrase was used, leave the field empty.
11. In **Fingerprint**, enter the fingerprint value of the private-public key pair.
12. In **Compartment OCID**, enter the compartment's OCID copied from the Compartments page.
13. In **Storage Namespace**, enter the storage namespace copied from the Tenancy Details page.
14. Click **Connect**.

> **Tip:**
>
> Select the **Remember** check box before you press the **Connect** button if you want VB Studio to remember the OCI credential input values you entered.
>
> After the export operation starts, the fields in the OCI credentials window that were saved are cleared. The data from those fields has been saved, but it is cleared to protect the administrator's OCI information while the export operation is in progress. After the operation concludes, the saved OCI information will be filled in and visible once again. Credentials are saved only after successfully connecting to the storage. This prevents saving bad credentials.
>
> Credentials are also saved separately for each user. so even if the same storage account is being connected to, each user has a distinct set of credentials that enable access by that user only.
>
> To clear the saved OCI credential information, you need to explicitly uncheck the **Remember** check box and reconnect. This will clear all the fields.

15. In the Create Job section, in **Type**, select **Export**.
16. In **Name**, enter a name for the export job.
17. In **Description**, enter the job's description.
18. In **Storage Container**, select the bucket to export the project data.
19. In **Storage Object**, if required, update the default `.zip` file name.
20. Click **Export**.
21. In the Confirm Project Export dialog box, select the **Export project data** check box, and click **Yes**.
22. In the Exporting Project page, expand **Steps** to see the status of each module.

After the export is complete, the Recent Activities feed on the **Project Home** page (and the History tab of the **Data Export/Import** page) display messages about the export action.

## Import Project Data from an OCI Object Storage Bucket Using OCI Credentials

You must be a Project Owner to import project data. When you import project data, it will overwrite all the project data in the target project. All of the project's artifacts will be replaced with the components from the project imported from the specified OCI Object Storage bucket, connected using OCI credentials.

To import project data from an OCI Object Storage bucket, you need the following information:

- Name of the target bucket
- Name of the archive file with the project data
- Private key and fingerprint of a user who can read objects from the bucket
- Details of the compartment that hosts the bucket

Contact Oracle Support or your OCI administrator for the details and get the required input values that you'll be entering in steps 6-13. These values are explained in Get the Required OCI Input Values in *Administering Visual Builder Studio*.

Here's how to import data to your target project:

1. Open the VB Studio project.

2. In the left navigator, click **Project Administration** .
3. Click **Data Export/Import**.
4. Click the **Job** tab.
5. In **Account Type**, select **OCI**.

   Users that are extending Oracle Cloud Applications are strongly encouraged to contact Oracle Support to get the values they will need to complete steps 6-13.
6. In **Tenancy OCID**, enter the tenancy's OCID copied from the Tenancy Details page.
7. In **User OCID**, enter the user's OCID value who can access the bucket.
8. In **Home Region**, select the home region of the OCI account.
9. In **Private Key**, enter the private key of the user who can access the bucket.
10. In **Passphrase**, enter the passphrase used to encrypt the private key. If no passphrase was used, leave the field empty.
11. In **Fingerprint**, enter the fingerprint value of the private-public key pair.
12. In **Compartment OCID**, enter the compartment's OCID copied from the Compartments page.
13. In **Storage Namespace**, enter the storage namespace copied from the Tenancy Details page.
14. Click **Connect**.

> **Tip:**
>
> Select the **Remember** check box before you press the **Connect** button if you want VB Studio to remember the OCI credential input values you entered.
>
> After the import operation starts, the fields in the OCI credentials window that were saved are cleared. The data from those fields has been saved, but it is cleared to protect the administrator's OCI information while the import operation is in progress. After the operation concludes, the saved OCI information will be filled in and visible once again. Credentials are saved only after successfully connecting to the storage. This prevents saving bad credentials.
>
> Credentials are also saved separately for each user. so even if the same storage account is being connected to, each user has a distinct set of credentials that enable access by that user only.
>
> To clear the saved OCI credential information, you need to explicitly uncheck the **Remember** check box and reconnect. This will clear all the fields.

15. In the Create Job section, in **Type**, select **Import**.
16. In **Name**, enter a name for the import job.
17. In **Description**, enter the job's description.
18. In **Storage Container**, select the bucket to import the project data from.
19. In **Storage Object**, select the `.zip` file name of the exported data.
20. Click **Import**.
21. In the Confirm Project Import dialog box, read and verify the bucket and object details, select the **Import project data** checkbox, and click **Yes**.

22. In the Importing Project page, expand **Steps** to see the status of each module.

    If you want to cancel the import process, click **Cancel**.

When an import job is in progress, the project will be in the locked state. You can't access other pages of the project until the import job has finished. After the import is complete, you'll be redirected to the **Project Home** page. The Recent Activities feed on the **Project Home** page (and the History tab in the **Data Export/Import** page) display messages about the import action.

## Export Project Data to an OCI Object Storage Bucket Using a PAR URL

You must be a project owner to export project data. When you export project data, VB Studio will export data to an archive file in the specified OCI Object Storage bucket, connected using a pre-authenticated request (PAR) URL for that bucket.

To do this, you need the following:

- Pre-authenticated request URL for the bucket, which you can obtain from your OCI administrator

Here's how to export your project's data:

1. Open the VB Studio project.
2. In the left navigator, click **Project Administration** .
3. Click **Data Export/Import**.
4. Click the **Job** tab.
5. For the **Object Storage authentication method**, select **Pre-Authenticated Request (PAR)**.
6. In the Create Job section, in **Type**, select **Export**.
7. In **Name**, enter a name for the export job.
8. In **Description**, enter the job's description.
9. In **Storage Container PAR URL**, enter the pre-authenticated request URL that you obtained from your OCI administrator. This URL is your access to the OCI bucket.

   Click **Test PAR URL** to confirm that the URL is valid.
10. In **Storage Object**, if required, update the default `.zip` file name.
11. Click **Export**.
12. In the Confirm Project Export dialog box, select the **Export project data** checkbox, and click **Yes**.
13. In the Exporting Project page, expand **Steps** to see the status of each module.

After the export is complete, the Recent Activities feed on the **Project Home** page (and the History tab of the **Data Export/Import** page) display messages about the export action.

## Import Project Data from an OCI Object Storage Bucket Using a PAR URL

You must be a project owner to import project data. When you import project data, it will overwrite all the project data in the target project. All of the project's artifacts will be replaced with the components from the project imported from the specified OCI Object Storage bucket, connected using a pre-authenticated request (PAR) URL for that bucket.

To do this, you need the following:

- Pre-authenticated request URL for the bucket, which you can obtain from your OCI administrator
- Name of the archive file with the project data

Here's how to import data to your target project:

1. Open the VB Studio project.
2. In the left navigator, click **Project Administration** .
3. Click **Data Export/Import**.
4. Click the **Job** tab.
5. For the **Object Storage authentication method**, select **Pre-Authenticated Request (PAR)**.
6. In the Create Job section, in **Type**, select **Import**.
7. In **Name**, enter a name for the import job.
8. In **Description**, enter the job's description.
9. In **Storage Container PAR URL**, enter the pre-authenticated request URL that you obtained from your OCI administrator. This URL is your access to the OCI bucket.

   Click **Test PAR URL** to confirm that the URL is valid.
10. In **Storage Object**, select the `.zip` file name of the exported data.
11. Click **Import**.
12. In the Confirm Project Import dialog box, read and verify the container and object details, select the **Import project data** check box, and click **Yes**.
13. In the Importing Project page, expand **Steps** to see the status of each module.

    If you want to cancel the import process, click **Cancel**.

When an import job is in progress, the project will be in the locked state. You can't access other pages of the project until the import job has finished. After the import is complete, you'll be redirected to the **Project Home** page. The Recent Activities feed on the **Project Home** page (and the History tab in the **Data Export/Import** page) display messages about the import action.

## View Export and Import History of the Project

You can examine a project's export and import history from the **Data Export/Import** page's **History** tab:

1. In the left navigator, click **Project Administration** .
2. Click **Data Export/Import**.
3. Click the **History** tab.

The history of all export and import jobs is displayed. Select a job to view its details. In the case of a failed job, expand **Steps** to view the modules that passed and those which failed.

# Edit a Project's Name, Description, or Visibility

After creating a project, you can edit its properties from the **Project Administration** page.

1. In the left navigator, click **Project Administration** .

2. Click the **Properties** tile.
3. In **Name** and **Description**, update the project name and description.
4. In **Security**, update the project's share status:
   a. Select **Private** to restrict non-members from seeing basic information about the project.

      Select the **Discoverable** checkbox to allow organization members that aren't org admins or project members to see basic information, such as name and owner contact information, about your private project. Private projects that aren't discoverable won't be exposed to non-members.
   b. Select **Shared** to make the project code, wiki docs, tasks, and builds available to anyone inside your organization.

When you're finished, use the left navigator to switch to another page.

## See When a Project Was Created and Last Updated

Administrators can tell how old a project is by checking its creation date in the Project Administration page's **Properties** tile.

To see when a project was created:

1. In the left navigator, click **Project Administration** .
2. Click **Properties**.

   The **Created** box displays the project's creation date.

> **Tip:**
>
> The Recent Activities feed on the Organization page's Activities tab also lists the creation dates of all projects in the organization.

To ensure sufficient storage, administrators often need to perform housekeeping duties that include going through the project list and removing older projects that haven't been worked on or updated in a long time. VB Studio provides some help in this area, in the Organization page's **Projects** tab. The Last Updated column can be sorted with the toggle to show which projects haven't been updated recently (are no longer being worked on actively). As administrator, you can notify the project owners that the projects are in this category and ask if they can be removed to free up storage space.

## Get a Project's Unique Identifier

Each project has a unique ID that you can copy from the **Project Administration** page:

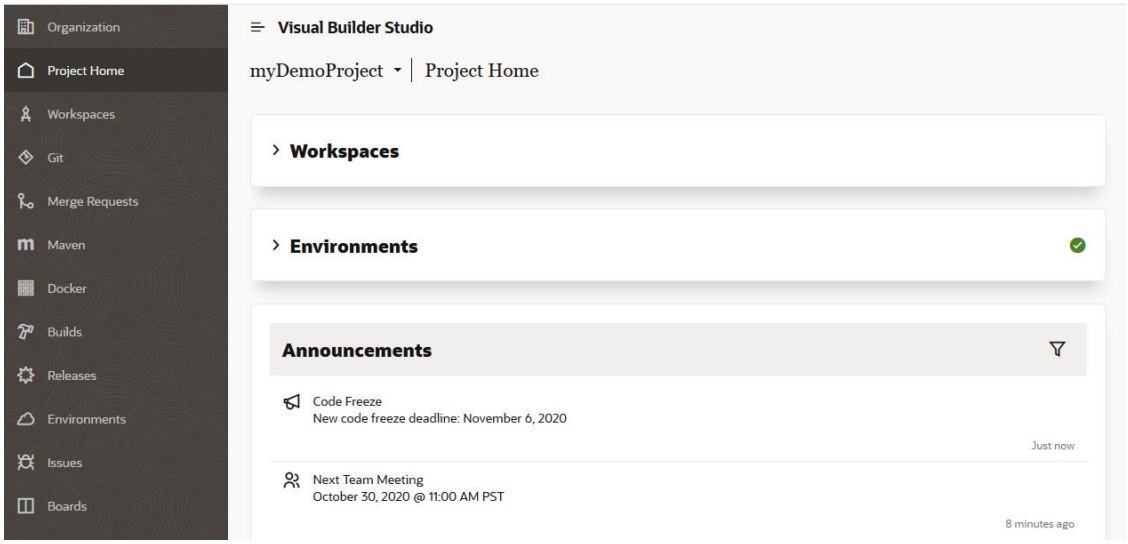
1. In the left navigator, click **Project Administration** .
2. Click **Properties**.

   **Identifier** displays the project's unique identifier.
3. Click to copy the ID to the clipboard.

The identifier you copied can now be pasted into a cURL command, for example, that can be used with the service's REST APIs for Issues and Builds. This is much more expedient than trying to construct the URL by hand.

# Manage Announcements

Project announcements are messages displayed on the **Project Home** page, between the Environments panel and the Recent Activity feed:




To create an announcement, follow these steps:

1. In the left navigator, click **Project Administration** .
2. Click **Announcements**.

| Action | How To |
|---|---|
| Create an announcement | 1. On the Announcements page, click **+ New Announcement**.<br>2. In **Name** and **Contents**, enter name and announcement's text. You can use the project's wiki markup to format the text.<br>3. Upload an icon, if necessary. The icon's size must be 48x48 pixels.<br>4. Click **Done**. |
| Copy an announcement | Instead of creating an announcement, you can copy the contents and icon of an existing announcement, and edit it.<br>1. In the Announcements list, select the announcement to copy.<br>2. Click **Copy Announcement**.<br>3. Edit the details in the Create Announcement page.<br>4. Click **Done**. |
| View or edit an announcement | 1. In the Announcements list, select the announcement.<br>2. In the Announcement section to the right of the list, view or edit the announcement's details.<br>Any changes made to the fields are saved immediately when the focus moves out of the field. |

| Action | How To |
| --- | --- |
| Deactivate or activate an announcement | If you don't want to display an announcement and don't want to delete it either, you can deactivate it. Deactivated announcements aren't visible on the **Project Home** page. Later, if you want, you can activate it and make it visible.<br>1. In the Announcements list, select the announcement.<br>2. Click **Deactivate** or **Activate**.<br>The deactivated announcement will be greyed-out in the Announcements list. |
| Delete an announcement | 1. In the Announcements list, select the announcement.<br>2. Click **Delete**.<br>3. In the Delete Announcement dialog box, click **Yes** to confirm. |

## Manage Project Tags

A project tag is a keyword that you can use to categorize an artifact, such as an issue or a merge request. You can use the tags to search for artifacts. By default, three tags (Plan, Release, and Epic) are available in a project.

1. In the left navigator, click **Project Administration** .
2. Click **Tags**.

| Action | How To |
| --- | --- |
| Create a project tag | 1. In the Tags page, click **New Tag**.<br>2. In **Tag Name**, enter a unique name and press the Enter key.<br>The tag name must contain only letters and numbers. |
| Rename a tag | 1. In the Tags page, select the tag.<br>2. Type a new name and press **Enter**. |
| Delete a tag | You can't delete a tag if artifacts refer to it. First, remove all artifacts that refer to the tag or remove the tag from those artifacts, and then remove the tag.<br>In the Tags page, to the right of the tag name, click **Delete** . |

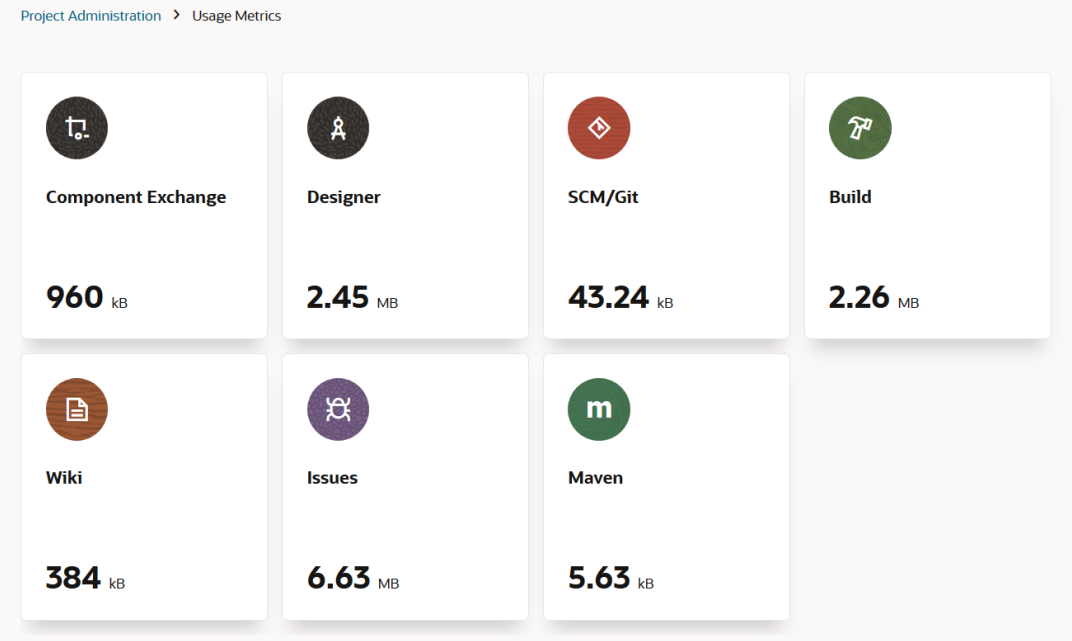
## View Your Project's Usage Metrics

When you subscribe to VB Studio, you're entitled to storage space on Oracle Cloud.

You can find out the disk usage metrics of your project's components, such as Git repositories, wikis, and issues.

1. In the left navigator, click **Project Administration** .
2. Click **Usage Metrics**.

Usage statistics for your project's components are displayed in an easy-to-read page.




# Display RSS/ATOM Feeds

Some websites use ATOM and RSS feeds to publish news feeds. You can subscribe to the RSS/ATOM feeds and configure your project to display them in the Recent Activities feed of the **Project Home** page. All project members can see the feed. You can add any RSS/ATOM feed, including Oracle-approved RSS feeds, news, site monitors, and Jenkins or Hudson servers.

To configure RSS/ATOM feeds, follow these steps:

1. In the left navigator, click **Project Administration** .
2. Click **RSS/ATOM Feeds**.

| Action | How To |
|---|---|
| Create an RSS/ATOM Feeds handler | 1. On the RSS/ATOM Feeds page, click **+ New Handler**.<br>2. In **Name**, enter the name of the handler.<br>3. In **URL**, enter the URL of the feed.<br>4. From **Display Type**, select the feed's display type.<br>5. In **Fetch Interval**, enter the feed's fetch interval. By default, the interval is set to 1 day.<br>For the fetch interval period, the feed results are cached. All requests during the interval period retrieve the cached results. When the time expires the cache is cleared. The next request would check for the cached results and not find them and proceed to fetch a new copy to be cached.<br>6. Click **Done**. |

| Action | How To |
|---|---|
| Test a feed handler | 1. On the RSS/ATOM Feeds page, select the feed.<br>2. Click **Test** .<br>3. Click **Done**.<br>If the test is successful, the status icon changes from **Untested** to **Tested** . If the test fails, the status icon changes to **Failed** . |
| View logs | 1. On the RSS/ATOM Feeds page, select the feed.<br>2. Click **Logs**.<br>3. Click **Done**.<br>In the Logs page, all Request and Response logs of each test are available. Select the date-time stamp in the left list of the test to view its logs. |
| Edit a feed's handler | 1. On the RSS/ATOM Feeds page, select the feed.<br>2. Edit the fields on the left.<br>3. Click **Done**. |
| Deactivate or activate a handler | 1. On the RSS/ATOM Feeds page, select the feed.<br>2. Click **Deactivate**. The icon of the feeds handler is greyed out and the **Active** check box is deselected.<br>To activate the feeds handler, click **Activate** or select the **Active** check box.<br>3. Click **Done**. |
| Delete a handler | 1. On the RSS/ATOM Feeds page, select the feed.<br>2. Click **Remove** or .<br>3. In the Remove ATOM/RSS Handler dialog box, click **Yes** to confirm.<br>4. Click **Done**. |

# Configure Link Rules

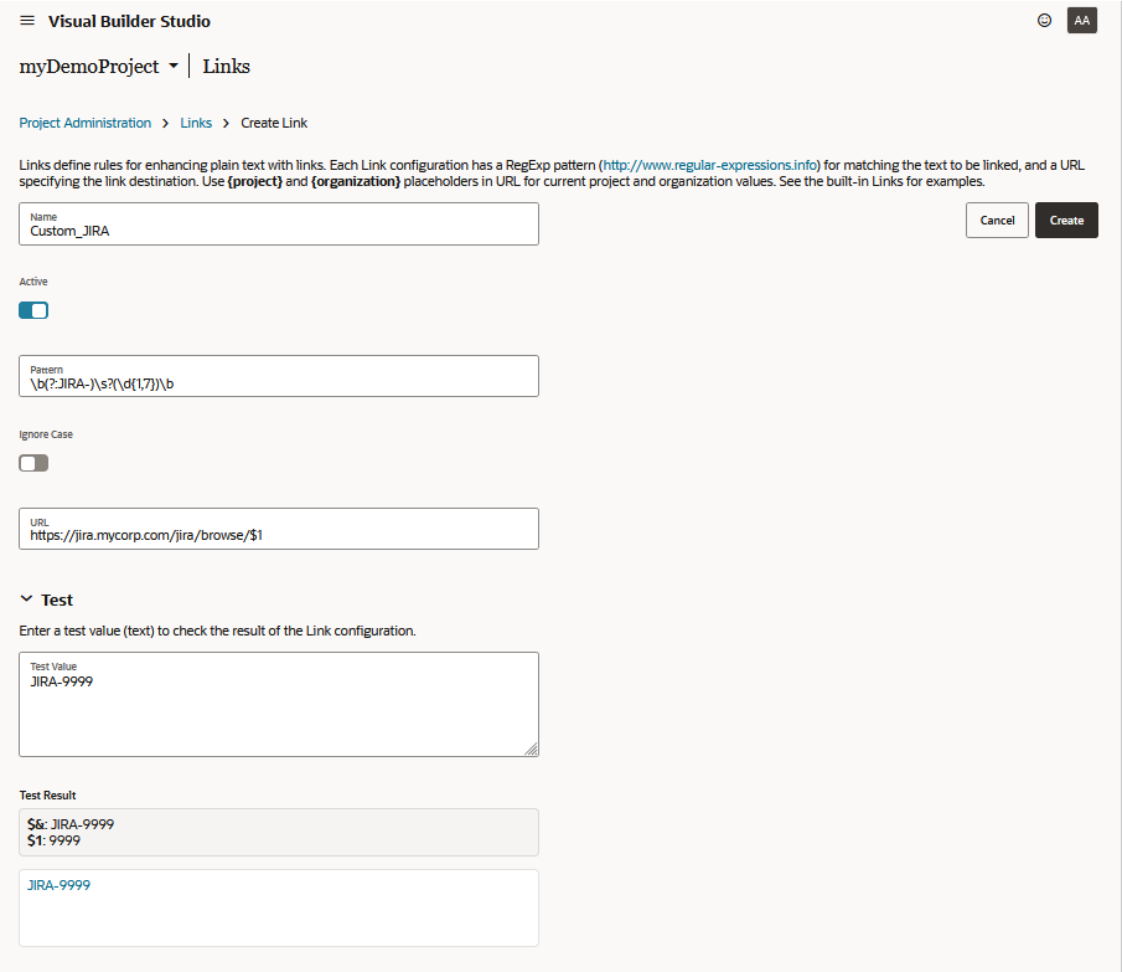
In a project, you can define rules to convert plain text to URL links automatically when the text is entered in commit comments and merge request comments. For example, when you enter an email address or a URL in a merge request comment, it's automatically converted to a link.

To configure link rules, follow these steps:

1. In the left navigator, click **Project Administration** .
2. Click **Links**.

From the Links page, you can create and manage link rules that convert plain text to URL links automatically. You can use Regular Expressions, also called as RegExp, to define the link rules. Some pre-defined built-in link rules are available on the **Links** page. To create a custom rule, you can either copy an existing link rule or create a blank rule. To find more about RegExp, see `http://www.regular-expressions.info`.

This illustration shows an example of a custom link rule.




| Action | How To |
|---|---|
| Create a link rule | 1. On the Links page, click **+ Create Link**.<br>2. In **Name**, enter a name.<br>3. In **Pattern**, enter the RegExp link rule pattern.<br>4. In **URL**, enter the link URL.<br>You can also use placeholders:<br>• Use `{project}` to insert the project ID.<br>• Use `{organization}` to insert the organization ID.<br>• Use `$&` to insert the entire matching text, or use `$1`, `$2`, `$3`, and so on to insert text of matched groups. For more information, see `http://www.regular-expressions.info/brackets.html`.<br>5. To test the rule, expand **Test** and **Test Value**, enter a test value. Verify the result link in **Test Result**.<br>6. Click **Done**. |

| Action | How To |
|---|---|
| Copy a link rule | 1. On the Links page, click **+ Copy Link**.<br>2. On the Create Link page, edit the name, RegExp link rule pattern, and the URL of the link with parameters.<br>You can also use placeholders:<br>• Use `{project}` to insert the project ID.<br>• Use `{organization}` to insert the organization ID.<br>• Use `$&` to insert the entire matching text, or use `$1`, `$2`, `$3`, and so on to insert text of matched groups. For more information, see `http://www.regular-expressions.info/brackets.html`.<br>3. To test the rule, expand **Test** and **Test Value**, enter a test value. Verify the result link in **Test Result**.<br>4. To test the rule, in the Test section, click **Test Value**. Enter a value and verify its result in **Test Result**.<br>5. Click **Done**. |
| Edit a link rule | You can't edit a built-in link rule. You can create a copy of built-in rule and edit it, and if required, disable the original pre-configured rule.<br>On the Links page, in the link rule list on the left, select the rule to edit its details on the right. |
| Activate or deactivate a link rule | If deactivated, the text that matches the rule is not converted to a link.<br>On the Links page, in the link rule list on the left, select the rule. Click **Activate** or **Deactivate**. You can also select or deselect the **Active** check box. |
| Delete a link rule | You can't delete a built-in rule.<br>On the Links page, in the link rule list on the left, select the rule. On the right side of the page, click **Delete**. |

# Set Up Issue Products and Custom Fields

Before creating and assigning issues to project members, you can define products, components, default owners of components, and releases for your project.

You can:

- Create multiple product categories, components, and sub-components
- Customize the releases
- Add custom fields for your project

You must be a project owner to add and manage issue products, components, and custom fields.

## Create and Configure Issue Products

When you define a product, you also define its releases and components. A product is a category that represents an entity. A component is a product subsection. A release is a release name or product number.

You can create multiple products for a project and select them from the **Products** drop-down list on the create or edit issue page. Each product must have at least one component and one release. For example, you can create a **Report** product with **1.0**, **2.0**, **3.0**, and **PS1** as its releases, and **Sales**, **Marketing**, and **Demographics** as its components.

You can define products, components, and releases from the Administration: Issue Tracking page's Products tab.

Open the Issue Tracking page's Products tab:

1. In the left navigator, click **Project Administration** ⚙.
2. Click **Issue Tracking**.
3. Click the **Products** tab.

Here's a list of the product management tasks you can do from the Product's tab:

| Action | How To |
|---|---|
| Create a product | 1. Click **+ New Product**.<br>2. On the Create Product page, in **Name**, specify a unique product name.<br>3. To create a release, click **+ New Release**, and enter a release name.<br>To make a release the product's default release, click **Mark as Default** ✓.<br>4. To create a component, click **+ New Component**, enter a component name, and select its default owner (optional).<br>To make a component the product's default component, click **Mark as Default** ✓.<br>5. To create a Found In tag, click **+ New Found In Tag**, and enter a tag name.<br>6. Click **Done**.<br>To reorder a release or component, mouse over the name and drag-and-drop it to move it up or down in the list. |
| View or edit a product | From the products list, select the product. On the right side of the page, view or edit its details. |
| Delete a product | You can't delete a product if any issues or merge requests refer to it. First, remove all issues and merge requests that refer to the product, and then remove the product.<br>1. In the products list, click **Delete** ⊗.<br>2. In the Delete Product dialog box, click **Yes** to confirm. |

## Create and Configure Custom Fields in an Issue

If an issue's default fields don't meet your needs, you can create custom fields for your project's issues. You can create and manage the fields from the Issue Tracking page's Custom Fields tab. When you create or update an issue, you can see the custom fields in the New or Edit Issue page's Details section.

1. In the left navigator, click **Project Administration** ⚙.
2. Click **Issue Tracking**.
3. Click the **Custom Fields** tab.

You can create these kinds of custom fields:

- Single line input text

- Single selection
- Multi selection
- Long text input
- Time and Date
- Check box

Here's how to manage custom fields:

| Action | How To |
|---|---|
| Create a custom field | 1. Click **+ New Custom Field**.<br>2. On the Create Custom Field page, in **Name**, specify a unique name.<br>3. In **Label**, enter the field's display label.<br>4. If you don't want the custom field to appear as a parameter when new issues are created, deselect the **available for New Issues** check box.<br>5. From the **Type** drop-down list, select the field type.<br>If you select **Single Selection** or **Multi Selection**, click **+ New Value** to specify the field's options.<br>6. Click **Done**. |
| View or edit a custom field | From the custom fields list, select the field. View or edit its details, located on the right side of the page.<br>You can't change a custom field's **Name** or **Type**. To edit the value of **Name** or **Type**, remove and then recreate the custom field. |
| Hide a custom field | From the custom fields list, select the field. Select the **Obsolete (hidden)** check box, located on the right side of the page. |
| Delete a custom field | 1. In the custom fields list, click **Delete** ⊗ located to the right of the field name.<br>2. In the Delete Custom Field dialog box, click **Yes** to confirm.<br>All existing issues will be automatically updated to remove the custom field. |

# Configure Project Templates

You can define an existing project as a template for new projects that users can use as a starting point. When creating a project, if a user selects a project template, the project template's data is copied to the new project, which the user can modify.

Project template data may include its Git repositories, build job configurations, deployment configurations, links, wikis, and announcements. For example, if a project template hosts an application in its Git repositories; information on how to use the application in its wiki pages; build configuration in its jobs; and Oracle Cloud deployment target details in its deployment configurations, then the data of the project template is copied to the new project. Members of the new project can use the application, run builds of pre-configured jobs, and deploy build artifacts to Oracle Cloud using pre-configured deployment configurations without making any changes to the code or any of its configurations.

When you define a project template, you define its visibility (who can use the project template), configure rules (what data can be copied from the project template), and use variables (customize actions based on user input when the data is copied).

After a project is created using a template, any updates made to the template project aren't reflected in the created project.

## Visibility, Rules, and Variables

While defining a project template, you define its visibility, rules, and variables.

### Visibility

Visibility defines who can access the template.

| Type | Description |
|---|---|
| **Draft** | The project template is under design and isn't available in the Templates page of the Create Project wizard. No user can copy data from a draft template. |
| **Private** | The project template is published and available to organization's users, but isn't visible by name to all. It's visible by name to members of the project template.<br>A non-member user can access the template using the project template's private key and can copy data. |
| **Shared** | The project template is published and is available to all users of the organization.<br>Any user can copy data from this project. |

### Rules

Rules define data to be copied to the new project.

When you define a project as a project template, all rules are enabled but some cannot be edited or added more than once.

| Set this rule ... | To copy this data to the new project: | Can this rule be edited? | Can this rule be added more than once? |
|---|---|---|---|
| Build Jobs | The project template's build jobs and pipelines | No | No |
| Wiki Content | The project template's wiki pages | No | No |
| Links | The project template's link rules | No | No |
| Git Repository | The specified Git repositories and its branches | Yes | Yes |
| External Git Repository | The specified external Git repository and its branches | Yes | Yes |
| Announcements | The project template's announcements | No | No |

### Variables

Variables define user input. Based on the input, you can configure the template to change the action or properties of data that's copied to the new project.

| Use this variable type: | To accept a: |
|---|---|
| **Boolean** | Boolean string value (such as True-False, Yes-No, or any string values). |
| **Choice** | Value from a list of values configured by project owners. |
| **String** | String value. |
| **URL** | URL value. |

## Define and Manage a Project Template

You can define a project template from the **Project Administration** page.

1. In the left navigator, click **Project Administration** .
2. Click **Properties**.
3. On the Properties page, in the **Template** section, click **Define Template**.

   The project is now marked as a template with default rules and visibility state.
4. To edit the template, click **Edit**.
5. In **Visibility**, select the template's visibility.

| Visibility Type | Description |
|---|---|
| **Draft** | The project template is under design and isn't available in the Templates page of the Create Project wizard. No user can copy data from a draft template. |
| **Private** | The project template is published and available to organization's users, but isn't visible by name to all. It's visible by name to members of the project template.<br>A non-member user can access the template using the project template's private key and can copy data. |
| **Shared** | The project template is published and is available to all users of the organization.<br>Any user can copy data from this project. |

6. To change the name and description of the project's template, enter a new name in **Name** and description in **Description**.
7. To specify an icon for the template, in **Icon**, click **Change**, browse and upload an image of size 48x48 pixels.
8. In the **Variables** and **Rules** sections, specify the variables and rules. Click **Save** when you're done.

## Define a Private Project Template

You can define a private project template if you don't want all users of the organization to copy data from the project. Private project templates aren't listed by name in the Templates page of the New Project wizard unless you're a member of the project template. Non-members can use a private project template only if they have the private key of the project template.

1. In the left navigator, click **Project Administration** .
2. Click **Properties**.
3. In the **Template** section, click **Define Template**.

The project is now defined to be used as a template with the default rules and properties.

4. Click **Edit**.
5. On the Template page, in the **Visibility** section, select **Private**.
6. For the **Private Key** field, click **Show**. Note down the key value.
7. Update settings, add variables, and define rules, as desired.
8. Click **Save**.

Share the private key with users whom you want to use the project template and copy the project data.
To generate a new private key, edit the project template, click **Show**, and then click **Regenerate**. You may want to do this if you don't want users who already have the old key value to copy the project data from the template.

## Define Project Template Rules

Rules define which artifacts are copied from the template project to the new project.

When you define a project as a project template, all rules will be enabled by default. However, some rules can't be edited or added more than once.

Here are the available rules and information about the artifacts that are copied from the project template to the new project:

| Use this rule ... | To copy: | Can this rule be edited? | Can this rule be added more than once? |
|---|---|---|---|
| Build Jobs | Build jobs and pipelines of the project template to the new project. | No | No |
| Wiki Content | Wiki pages of the project template to the new project. | No | No |
| Links | Link rules of the project template to the new project. | No | No |
| Git Repository | Specified Git repositories along with its branches to the new project. | Yes | Yes |
| External Git Repository | Specified external Git repository along with its branches to the new project. | Yes | Yes |
| Announcements | Announcements of the project template to the new project. | No | No |

To add or remove a rule, follow these steps.

1. Open the Template Settings page.
2. To add a rule, scroll to the **Rules** section. From the **Add Rule** drop-down list, select the **Git Repository** rule.

   To remove a rule, click **Remove** ⊗.

## Add or Edit a Git Repository Rule

By default, all hosted and external Git repositories rules are enabled.

You can edit the default rules and add new Git Repository rules if you added Git repositories to the project after the template was defined. You can also add a rule to make a copy of an existing Git repository on the new project that uses the template. If you don't want a Git repository to be included in the template, remove its rule.

Here's how to add or edit hosted and external Git repository rules:

| Action | How To |
|---|---|
| Add or edit a hosted Git repository rule | 1. On the **Properties** page, in the **Template** section, click **Edit**.<br>2. In the **Rules** section, to add the rule, click **Add Rule**. From the menu, select **Git Repository**.<br>To edit an existing rule, to the right side of the Git repository rule, click **Edit** .<br>3. In **Source Repository**, specify the name of the Git repository to be copied.<br>4. In **Repository Name**, specify the new name of the Git repository. A `.git` extension is automatically added if you missed it.<br>To use the new project name as the name of the Git repository, select the **Use target project name** check box.<br>To allow a user creating a project from the template to change the default Git repository name, select the **Override name on create** check box.<br>5. If necessary, in **Replacements**, define file name replacements of the Git repository matching the specified criterion. This is useful if you want to make a copy of an existing Git repository.<br>To add a new replacement rule, click **Add new replacement**.<br>a. From the **In** drop-down list, select the files where the replacements apply.<br>You can specify all files, files matching an Ant pattern, or a specific file.<br>b. In **Replace**, specify the search term.<br>c. In **With**, specify the replacement term.<br>You can select a pre-defined variable such as **Project Id**, **Project Name**, **Project URL Name**, or **Repository Name**. To use a variable defined in the **Variables** section, select **Variable** and then select the variable name.<br>d. Click **Save** .<br>When you create a project using this template, the project creation wizard searches through the specified files in the Git repository and replaces the term with the specified value of the selected variables.<br>6. Click **Save** to save the Git repository rule.<br>7. Click **Save** to save the project template. |

| Action | How To |
|---|---|
| Add or edit an external Git repository rule | 1. On the **Properties** page, in the **Template** section, click **Edit**.<br>2. In the **Rules** section, to add the rule, click **Add Rule**. From the menu, select **External Repository**.<br>To edit an existing rule, to the right side of the Git repository rule, click **Edit** .<br>3. In **Repository URL**, enter the external Git repository URL. To update a rule, enter a new URL.<br>4. In **Username** and **Password**, enter credentials to access the external Git repository.<br>For public Git repositories, don't fill these fields.<br>5. In **Repository Name**, specify the new name of the Git repository. A `.git` extension is automatically added if you missed it.<br>To use the new project name as the name of the Git repository, select the **Use target project name** check box.<br>To allow the user that creates the project from the template to change the default Git repository name, select the **Override name on create** check box.<br>6. Click **Save** to save the Git repository rule.<br>7. Click **Save** to save the project template. |

## Add and Manage Variables

Variables define user input. Based on the input, you can configure the template to change the action or properties of data that's copied to the new project.

Here's how to add or edit variables from the Variables section on the Template Settings page:

1. Open the Template Settings page.
2. In the Variable section, from the **Add Variable** menu, select the variable type.

| Use this variable type: | To accept a: |
|---|---|
| **Boolean** | Boolean string value (such as True-False, Yes-No, or any string values). |
| **Choice** | Value from a list of values configured by project owners. |
| **String** | String value. |
| **URL** | URL value. |

3. In **Name**, **Display Name**, **Description**, enter the variable's unique name, the display name, and its description. Fill in the other fields of the variable and click **Save** .

   Later, if you want to edit the variable, click **Edit** . Update the fields of the variable and click **Save** . To delete the variable, click **Remove** .

## Perform Build Administration Tasks

The project owner can use the Build Administration page to configure a project to access the Oracle Maven Repository and NPM registry, set up a SonarQube Server to analyze build reports, create named passwords/private keys, and set job protections.

See these sections for more information about what a project owner can do to make the VB Studio build system as efficient as possible for the project members who use it:

- Configure Job Protection Settings
- Create a Pipeline Approvers Group for Your Project
- Protect Your Pipeline: Restrict Who Can Start It Manually or Edit Its Configuration
- Create and Manage Oracle Maven Repository Connections
- Create and Manage a Project's Remote NPM Registry Connection
- Create and Manage the Pre-Defined SonarQube Server Connection
- Create and Manage Named Passwords/Private Keys

## Manage Repositories

The project owner can manage the Git, Maven, and linked Docker registries from the **Project Administration**: **Repositories** page.

See these links to find out how to manage repositories:

- Create a Git Repository
- Maven Repository Administration
- Link an External Docker Registry to Your Project

4

# Manage Source Code Files with Git

A Visual Builder Studio project uses hosted Git repositories to store source code files and to provide version control. A project can have just one repository or it can have multiple ones. If you're developing visual applications or extensions using the Designer, this Git repository is associated with your project as well as with your **workspace**, which is a private work area within the Designer. Regardless of whether you're developing applications inside or outside the Designer, you can choose whether to use a Git repository provided by VB Studio, hosted on Oracle Cloud, or to associate your project with an external Git repository (like GitHub).

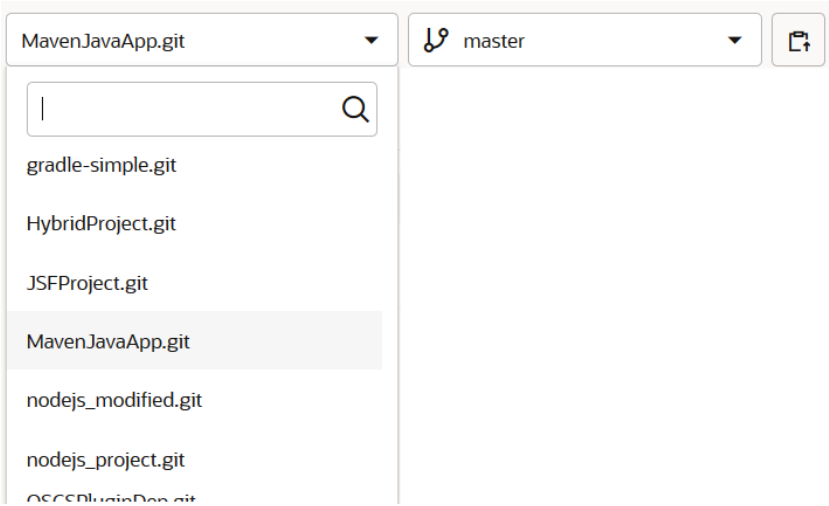
This chapter describes how to manage your Git repository for non-Designer applications. Interacting with a Git repository in the context of the Designer is described in What Is the Designer?.

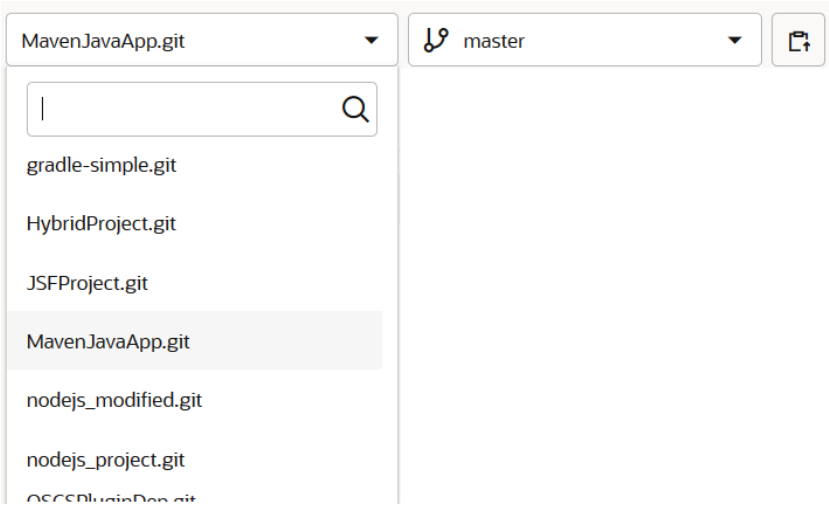
## Git Concepts and Terms

Git is a distributed version control in which you clone the entire remote (or central) repository, including its history, to your computer. You add and commit the files on your computer and, when you're done, push the commits to the remote repository.

If you are new to Git, read the Git documentation at `https://git-scm.com/book/` and `http://git-scm.com/doc` to learn more about Git repositories and Git basics, such as remote repositories, cloning, commits, pushes, SHA-1 checksum hashes, branches, and tags.

This documentation uses these terms to describe a project's Git components:

| Term | Description |
| --- | --- |
| Project Git repository | A remote or hosted Git repository of your project.<br>A project can host multiple Git repositories. You can view all Git repositories from the **Repositories** drop-down list on the Git page.<br>  |
| Local Git repository | A cloned Git repository on your computer. If you're creating an extension, your workspace orients you to the right Git repository. It also reflects the current branch in the header. |
| External Git repository | A Git repository that's hosted outside the project. It could be a Git repository of another project, or a Git repository available on another platform, such as GitHub or Bitbucket. |

| Term | Description |
|---|---|
| Revision | A snapshot of the Git repository at a given time. The revision could be a branch, tag, or commit. The **Revisions** menu displays the selected Git repository's revisions (branches, tags, and commits).<br>  <br>When entering a search criteria, add a space at the end of the search term to search for an exact match.<br>To search for a commit, enter the first three characters of the SHA-1 checksum hash of the commit in the search box at the top of the menu. The commit that matches the search term appears next to **Commit**<br> <br>, at the bottom of the menu.<br>To copy the revision name to the clipboard, click **Copy** . For example, while viewing files of a particular commit, you can copy the SHA-1 checksum hash of the commit. |
| **Files** view | Displays the Git repository's files and lets you manage them. |
| **Logs** view | Lists and graphically displays the Git repository's commit history. |
| **Refs** view | Displays the Git repository's branches and tags and lets you manage them. |
| **Compare** view | Compares and displays the differences between revisions in a Git repository. |

# Migrate to Git

If you've been using a version control system such as CVS or Subversion and want to migrate to Git, you can use the Git commands in the Git command-line interface to do so:

| To migrate from ... | Use this command: |
|---|---|
| CVS | `git-cvsimport`<br>For more information, see `http://git-scm.com/docs/git-cvsimport`. |
| Subversion (SVN) | `git svn`<br>For more information, see `http://git-scm.com/docs/git-svn`. |

| To migrate from ... | Use this command: |
|---|---|
| Other version control systems | See the Git Book at `http://git-scm.com/book/es/v2/Git-and-Other-Systems-Migrating-to-Git`. |

# Set Up a Git Repository

To set up a Git repository for your project, you first create a repository, and then upload application files to it. After you've set up the repository, all project users can access its files.

## Create a Git Repository

As a project owner, you can add multiple Git repositories to a project. VB Studio gives you the option to create an empty Git repository, a Git repository with a readme file, or import files from another Git repository.

You may want to create an empty repository if you plan to upload your application files to it from your computer or import files from another Git repository. Some Git clients can't clone an empty Git repository. If this is the case with the Git client you use, you may want to create a Git repository initialized with a file.

You can create a Git repository from these pages:

- The **Project Home** page's **Repositories** tab
- The **Git** page, accessed from the left navigator
- The **Project Administration** : **Repositories** page

Before creating a Git repository, especially if it will contain large files and binaries, you should review the best practices section that examines how best to deal with these types of files. See Best Practices for Storing Large Files and Binaries.

### Best Practices for Storing Large Files and Binaries

If one or more of your project's repositories contain large files and/or many binaries, it may make sense to use Cloud storage, Maven, or Git LFS instead of versioning these kinds of files with Git.

Here are a few reasons why you shouldn't store binary files in a Git repository:

- You can't use Git to diff binary files.
- Large files grow your repository's history every time they're updated, because Git stores the full size of every version of every binary file. If these binaries are large, they'll quickly become the largest item(s) in the repository.
- As the repository size grows, Git operations, such as cloning, fetching, and pulling will become extremely slow.

In general, the way you store binaries should be based on the type and use of the binary file:

- If the binary file is a build artifact, you should store it directly, when the artifact is built, in a Maven repository.
- If the binary file is a binary document that is saved by an application, such as Microsoft Excel or Word, it should be stored in either:

- Oracle Cloud storage
- Git LFS, if the binary documents are somehow associated with source files stored in the same Git repository

Git LFS uses pointers, instead of the physical files, when the files or file types are marked as LFS files. When you pull a Git LFS file to your local repository, the file is sent through a filter that replaces the pointer with the file. The physical files are located on the remote server and files that are pulled are located *in a cache* on your local machine. This means that your local repository will be smaller in size than the remote repository, where all the files and all the differences are still physically stored. See Enable and Use Git LFS to Version Large Files for more information about using Git LFS.

There are a few potential drawbacks you should consider as you decide whether or not to use Git LFS. Here are a few:

- The local Git LFS cache won't be cleaned up automatically. Just as you have to prune remote branches on a regular basis, you also have to prune your Git LFS content with the `git lfs prune` command.
- You need to make sure that all the developers have Git LFS installed. When someone who doesn't have Git LFS installed commits a file that should be associated with Git LFS, you'll see some strange errors. These problems can be fixed, but it's better to prevent this from happening.

To summarize, the best approach is to store large files and binaries in Cloud storage or Maven whenever possible. If you absolutely need to version these types of files, consider using Git LFS, but recognize that it isn't a panacea. It has its own associated costs, especially for maintenance.

## Default Repository Size Limits in VB Studio

By default, VB Studio doesn't limit the maximum number of repositories for a project. However, VB Studio does set default limits for the maximum repository size and the maximum object size for code repositories, as well as for snippets.

Here are the default limits set for code repositories:

- Maximum object size: 50MB
- Maximum repository size: 10GB

See Share and Use Code Snippets to learn about the default size limits for snippets.

## Enable and Use Git LFS to Version Large Files

If your projects contain large files, especially ones that are modified often, an initial clone can take a long time because your Git client needs to download every version of every file. Git LFS (Large File Storage) minimizes the time needed for downloading large files in your repository by downloading the relevant versions during the checkout process instead of during clone or fetch operations. Git LFS does this by replacing large files in your repository with small pointer files that you'll probably never even see. Git LFS automatically handles the replacements and stores the large file contents on a local file system or a remote server.

In VB Studio, by default, Git LFS is enabled in all Git repositories for the entire organization.

If you need to store large or binary files in your project, this is how you enable and use Git LFS for a repository in the project:

1. Clone the project's Git repository.

You can find the URL for the remote repository from the Project Home page's **Repositories** tab. From the **Clone** drop-down list, click **Copy to clipboard** to copy the URL, then paste it to the command line:

```
git clone <SSH/HTTPS repository URL>
```

You must be connected to the Internet and to the remote repository to check out a branch locally. This is required.

2. Enable LFS for the local Git repository:
```
git lfs install
```
This is required because the **+ File** button on the Git page can't be used to upload or add a binary file directly in a remote Git repository. The only way to add a binary file is through a cloned repository.

3. Specify the file(s) to track:
```
git lfs track *.bin
```

4. Stage the files to be committed:
```
git add .
```
```
git add .gitattributes
```

5. Commit the files to the local repository:
```
git commit -A -M "Initial commit using LFS"
```

6. Push the commits to the remote repository:
```
git push origin main
```
When a binary file is pushed to a remote Git repository, it's stored on the VB Studio-connected OCI Object Storage bucket. A reference is added to the Git repository, not to the file.

If a connection to the OCI Object Storage bucket can't be made, you'll see an error message that indicates storage hasn't been enabled.

When a Git repository that uses Git LFS is deleted, its binary files on OCI Object Storage will also be deleted.

## Create an Empty Git Repository

Create an empty Git repository if you plan to upload your application files to it from your computer or import files from another Git repository.

1. Click **+ Create Repository**.
2. In the New Repository dialog box, in **Name** and **Description**, enter a unique name and a description.

   Once you create a repository, you can’t change its name.
3. In **Initial Content**, select the **Empty Repository** option.

   To initialize the repository with a file, select the **Initialize repository with README file** option.
   You can add to and format the contents of the readme file using the Markdown markup language. Use the markup toolbar for frequently used formatting actions and to customize the content editor itself.

   If you don’t want to keep the file after VB Studio creates the repository, delete it.
4. Click **Create**.

## Import an External Git Repository

If you've been using a Git repository on another platform such as GitHub or Bitbucket, you can import files from the external Git repository to your project's Git repository.

When you import an external Git repository, VB Studio creates a Git repository in the project and copies the branches, tags, and commit history to it from the external Git repository. No changes made to the external Git repository after the import succeeds are reflected in the imported Git repository.

1. Click **+ Create Repository**.
2. In the New Repository dialog box, in **Initial content**, select **Import existing repository**.
3. In the text box, enter the URL of in the external Git repository.

   If the imported Git repository is password protected, enter the repository credentials in **Username** and **Password**. Note that VB Studio doesn't store the credentials.
4. Click **Create**.

You can also import an existing Git repository to an empty project Git repository from the Git page. If the added hosted Git repository is empty, enter the Git repository's URL in the Git page's **Import existing repository** section. Enter any repository credentials, if required, and click **Import**.

## Mirror an External Git Repository

If you've been using a Git repository on another platform, such as GitHub or Bitbucket, and don't want to import it to a project's Git repository, you can mirror it in VB Studio. Mirroring copies the repository to VB Studio and then VB Studio automatically synchronizes its files. In an active VB Studio project, repositories are synchronized approximately every five minutes, but the duration may vary according to the number of external Git repositories in all projects across the organization.

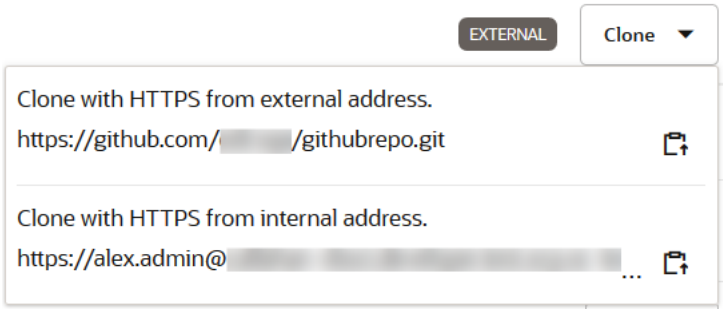
Note that you can't add or update files or manage branches of a mirrored Git repository from the project's Git page.
If the external Git repository is a private repository or is password protected, you'll need to create an authentication token, such as GitHub's personal access token or BitBucket's App Password, and use it to provide access to the external Git repository. Never provide your account's password.

Here's how to mirror an external repository:

1. In the left navigator, click **Project Administration** .
2. Click **Repositories**.
3. Under **External Repositories**, click **+ Link External Repository**.
4. In the New Repository dialog, enter the URL of the external Git repository in **URL** and the repository description in **Description**.
5. Expand the **Credentials for non-public repos** section and provide the credentials to access the external Git repository.

   In **Username**, enter the username of the external repository account. In **Token**, enter the authentication token.
6. Click **Create**.

The external repository is now available on the Git page and in the Project Home page's **Repositories** tab.
When you add an external Git repository, VB Studio shows two URLs in the repository's **Clone** drop-down menu:



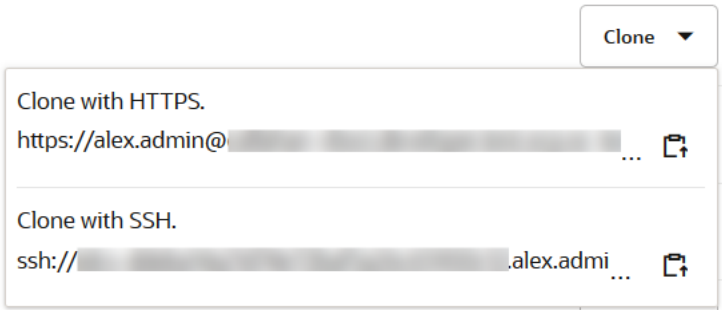

Use the URL with the external address to access the repository directly. You may want to use this URL to access the repository's updates immediately, but you'll need to enter credentials to access a private repository. Use the URL with the internal address to access the mirrored repository. You'll want to use this URL to access a private repository because it doesn't require credentials.

# Upload Files From Your Computer to the Project's Git Repository

After using a Git client to clone the project's Git repository to your computer, adding files, committing the changes to the cloned Git repository, you can then push the commit to the project's Git repository:

1. Copy the Git repository URL.

   On the Git page, from the **Repositories** drop-down list, select the Git repository. From the **Clone** drop-down list, click **Copy to clipboard** to copy the HTTPS or the SSH URL:




2. Open the Git client - perhaps the Git CLI.
3. Navigate to the directory where you want to clone the remote Git repository.

   If the directory into which you want to clone the repository isn't empty, you'll need to create a new subdirectory and clone the repository into it. You can only perform a cloning operation into an empty directory.
4. Using the Git client, clone the project's Git repository.

   For example, if you're using the Git CLI, use the `git clone <repository-url>` command. Use the Git repository's URL copied from step 1.

Here's an example that uses HTTPS:

```
git clone https://john.doe%40example.com@developer.oraclecloud.com/
developer1111-usoracle22222/s/developer1111-usoracle22222_myproject/scm/
developer1111-usoracle22222_myproject.git
```

Here's an example that uses SSH:

```
git clone ssh://
usoracle22222.john.doe%40example.com@developer.oraclecloud.com/developer1111-
usoracle22222_myproject/developer1111-usoracle22222_myproject.git
```

5. Open the directory to access files.

   You'll notice a `.git` subdirectory in the repository directory. Don't add, delete, or modify the files of the `.git` subdirectory.

6. Copy your application files to the cloned Git repository directory.

7. To add new files to the repository, use the Git client to add them to the repository index.

   For example, if you're using the Git CLI, use the `git add` command:

   ```
   git add readme.txt
   ```

   To add a directory and its files, navigate to the directory and use `git add .`

8. Commit the updated files to the cloned Git repository.

   For example, if you're using the Git CLI, use the `git commit` command to save the changes:

   ```
   git commit -am "Sample comment"
   ```

9. Push the commit from the cloned Git repository to the hosted Git repository.

   For example, if you're using the Git CLI, use the `git push` command:

   ```
   git push origin main
   ```

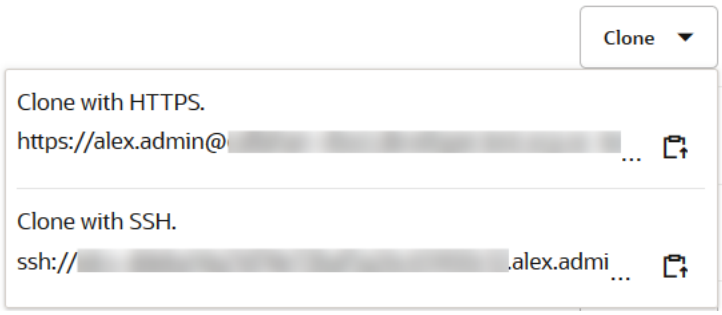
## Push a Local Git Repository to the Project's Git Repository

If your application source code files are available in a local Git repository, you can push them to a project's empty Git repository.

You can use any Git client to push the local Git repository to the remote Git repository:

1. Copy the URL for the project's Git repository.

   On the Git page, from the **Repositories** drop-down list, select the Git repository. From the **Clone** drop-down list, click **Copy to clipboard** to copy the HTTPS or the SSH URL, as shown:



2. Open the Git client - perhaps the Git CLI.
3. Navigate to the local Git repository directory.
4. Add the project's Git repository as the remote repository of the local repository. Use the Git repository's URL copied from step 1.

   For example, if you're using the Git CLI, use the `git remote add <remote-repository-name> <repository-url>` command.

   Here's an example that uses HTTPS:

   ```
   git remote add origin https://
   john.doe%40example.com@developer.oraclecloud.com/developer1111-
   usoracle22222/s/developer1111-usoracle22222_myproject/scm/developer1111-
   usoracle22222_myproject.git
   ```

   Here's an example that uses SSH:

   ```
   git remote add origin ssh://
   usoracle22222.john.doe%40example.com@developer.oraclecloud.com/developer1111-
   usoracle22222_myproject/developer1111-usoracle22222_myproject.git
   ```

   Both examples add a remote repository `origin` for the repository at `developer.oraclecloud.com/developer1111-usoracle22222_myproject/developer1111-usoracle22222_myproject.git`.
5. Push the local Git repository to the project's Git repository.

   For example, if you're using the Git CLI, use the `git push` command:

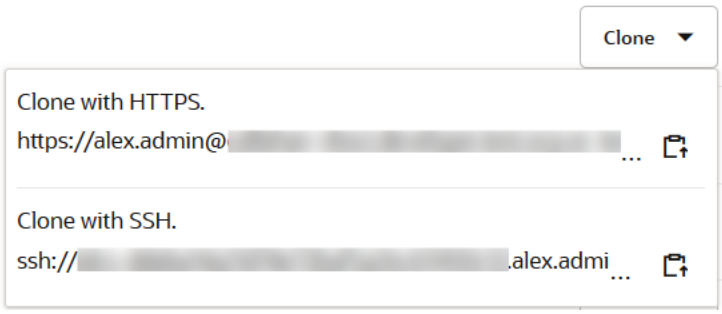
   ```
   git push –u origin main
   ```
6. In your project, open the Git page and check the files in the project's Git repository.

## Access a Git Repository Using SSH

1. On the computer that you'll use to access the Git repository, generate a SSH key pair and upload its private key to VB Studio. See Upload Your Public SSH Key for instructions. Make sure that the Git client can access the private key on your computer.

   Ignore this step if you've already uploaded the SSH public key.
2. Copy the Git repository's SSH URL.

   On the Git page, from the **Repositories** drop-down list, select the Git repository. From the **Clone** drop-down list, click **Copy to clipboard** to copy the SSH URL:




3. Open the Git client - perhaps the Git CLI.
4. Navigate to the directory where you want to clone the remote Git repository.

If the directory into which you want to clone the repository isn't empty, you'll need to create a new subdirectory and clone the repository into it. You can only perform a cloning operation into an empty directory.

5. Using the Git client, clone the project's Git repository.

   For example, if you're using the Git CLI, use the `git clone <repository-ssh-url>` command:

   ```
   git clone ssh://
   usoracle22222.john.doe%40example.com@developer.oraclecloud.com/developer1111-
   usoracle22222_myproject/developer1111-usoracle22222_myproject.git
   ```

   If you've already cloned the Git repository to your computer using HTTPS, use the `git add remote` command to add the SSH URL of the Git repository:

   ```
   git remote add ssh-origin ssh://
   usoracle22222.john.doe%40example.com@developer.oraclecloud.com/developer1111-
   usoracle22222_myproject/developer1111-usoracle22222_myproject.git
   ```

6. Commit the updated files to the cloned Git repository.
7. Push the commit from the cloned Git repository to the hosted Git repository:

   ```
   git push ssh-origin main
   ```

## Access a Git Repository Using Token-Based Authentication

You can use an authentication token that has been generated in VB Studio instead of your password to access a Git repository.

> **Note:**
>
> If you have user rights to read and write to Git for all projects you are a member of, you can create a token using the **All User Rights** option; however, be aware that any token with this scope won't provide a user with the ability to create/delete tokens or modify any other profile preferences settings.

1. From your VB Studio preferences, generate an authentication token for your project with both Read and Write Git permissions. See Set Up Token-Based Authentication.
2. Open the Git client.
3. Navigate to the directory where you want to clone the remote Git repository.

   If the directory into which you want to clone the repository isn't empty, you'll need to create a new subdirectory and clone the repository into it. You can only perform a cloning operation into an empty directory.

4. Using the Git client, clone the project's Git repository.

   For example, if you're using the Git CLI, enter the following `git clone` command. You'll be prompted to enter your username and password. When prompted for your password, enter your personal access token instead of a password.

   ```
   git clone https://myinstance.com/organization_name>/s/<project_name>/scm/
   <repo_name>.git
   ```

```
Username: YOUR_USERNAME
Password: YOUR_PERSONAL_ACCESS_TOKEN
```

You could also enter a command like this, using the username and token in a `git clone` command with this format:

```
git clone https://<username>:<personal_access_token>@https://
myinstance.com/organization_name>/s/<project_name>/scm/<repo_name>.git
```

You could even enter a command like this, using just the token in a `git clone` command with this format:

```
git clone https://<personal_access_token>@https://myinstance.com/
organization_name>/s/<project_name>/scm/<repo_name>.git
```

> **Tip:**
>
> Use `<username>:<personal_access_token>` if the specified user owns the token. If the user doesn't own the token, an Auth error will be encountered.
>
> Use just `<personal_access_token>` when the user is the one who owns the token.

5. Commit the updated files to the cloned Git repository.
6. Push the commit from the cloned Git repository to the hosted Git repository, using a command like this example shows:

```
git push origin main
```

The username and token from the git clone command may be cached, so you may not be prompted for them. If you are, enter the token for the password, as you did previously.

# Add and Manage a Git Repository's Files

You can add and update a Git repository's files online from the Git page or clone the Git repository to your computer and update the files locally.

## Manage Files from the Git Page

If you're a project member, you can browse, add, edit, rename, and delete a Git repository's files. You can also view commit histories for the files but you can't add or update files in a linked external Git repository.

You must be a project member to add or update a Git repository's files:

1. In the left navigator, click **Git** .

2. From the **Repositories** drop-down list, select the Git repository. From the **Revisions** drop-down list, select the branch.
3. On the right side of the page, click **Files**, if necessary.
4. Browse and click a directory name to open it.
   To go back to a file's or a sub-directory's parent directory, click **/** and select the file or directory from the menu. To go to the root directory, click . To copy a file's or a directory's path, click **Copy to clipboard** .

You can perform these file management tasks from the Git page:

| Action | How To |
|---|---|
| Add a file | 1. Click **+ File**.<br>2. In **File Name**, enter the file's name and extension.<br>3. In the code editor, add or enter the file contents.<br>4. Click **Commit**.<br>5. In the Commit Changes dialog, enter a commit summary in the first text box, any details in the **Details** text box, and then click **Commit**.<br>To save the file to a new directory or a directory structure, include the file path in **File Name**. The path can be a relative path or an absolute path. To specify an absolute path, add a `/` in the beginning.<br>For example:<br>• Enter `test/text_file.txt` to save the `text_file.txt` file in the `test` directory on the current path. If the `test` directory doesn't exist, it's created.<br>• Enter `/test/text_file.txt` to save the `text_file.txt` file in the `test` directory on the root. If the `test` directory doesn't exist, it's created. |
| View a file's contents | In the Git page, browse to and click the file name link to see its contents. The file opens in **File** tab:<br>• If you open a text file, its contents are displayed in a read-only editor.<br>• File and Blame views clearly indicate that a file is a symlink or a submodule (a gitlink) and display information about the link's target.<br>• The contents of empty files, large files, and binaries aren't displayed but you can use the browser URL to download it.<br>• If the file is a text file or an image (such as `.png`, `.jpg`, `.bmp`, and `.gif`), it's displayed in the browser.<br>If the text content exceeds the width of the editor, use the arrow keys to scroll left, right, up, and down. You can also use the scroll buttons to scroll horizontally. Move the cursor to the editor's left or right edge and click **Right Scroll** or **Left Scroll** to scroll a character at a time.<br>To view the file in raw (unformatted) format in the web browser, click **Edit** , and select **Raw**. The contents of the opened file are displayed in a new tab or a window in the web browser. |

| Action | How To |
|---|---|
| View a file's details | The headers in the Files and Blame pages display the file size and number of lines of code (LOC) for each file. The tree view to the left indicates the file mode (whether a file is an executable, a symlink, a submodule, or a regular file) but only displays the file size for regular files.<br>Icons show the file mode in the Files and Blame page's headers as well in the tree listing, so you can quickly see that a file is an executable, a symlink, or a submodule.<br>Empty files, large files, and binaries are shown in a similar manner, where you see information about them, but not the contents of the files themselves. |
| View a file's annotations and commits | Open the file and click **Blame**.<br>The Blame view displays the open file's annotations for each updated code line (or group of code lines) along with commit information. The annotation includes commits that affected code lines, the author, the commit's date-time stamp, and the commit message. |
| Show annotated tag messages | The Git page's **Refs** tab displays annotated tag messages. |
| Show a file's mode | The diff view in the Git page's **Compare** tab displays the file mode.<br>Icons indicate the file mode in the File and Blame page's headers as well in the tree listing to the left so you can quickly see that a file is a regular file or an executable, a symlink, or a submodule (a gitlink) and, if it is a link, display information about the link's target.<br>Empty files, large files, and binaries are shown in a similar manner, where you see information about them, but not necessarily the contents of the files themselves. |
| Change a file's mode | If you change a file's mode, in the **Compare** tab, the comparison screen shows details about any mode changes or displays a "No diff" message if the mode wasn't changed. If you do change the mode, say from regular to executable, you're notified of the change with a message that says, `Mode change: Regular file->Executable file`. |
| Edit, rename, or move a file | Open the file and click **Edit** . Edit the file's contents in the code editor. To rename the file or move it to another directory, in the file name text box, enter the new name or path. Click **Commit** to save.<br>With the Git diff editor active, if you make a change in the file shown in the diff view, the view refreshes immediately. If you have another **Files** tab open (the editor in the **Compare** tab with the change isn't active), the diff info gets refreshed when you return to the tab it's in. The refresh happens as soon as you select the **Compare** tab. |
| Delete a file | To delete a file, click **Actions** next to **Edit** , and select **Delete**. In the **Commit Changes** dialog box, enter the commit summary in the first text box, the details in the **Details** text box, and then click **Commit**.<br>When a file is deleted (or moved), any open diff editors associated with that file are closed. |

# Use Git Commands to Manage Files

To access and manage your project's Git repository files from your computer, use a Git client - perhaps the Git CLI.

Here are some of the most common Git commands you can run in the Git CLI to work on files in your local Git repository:

| Run this command ... | To: |
|---|---|
| `git clone <repository-url>` | Clone a project's Git repository to your computer:<br>`git clone https://john.doe%40example.com@developer.us.oraclecloud.com/developer1111-usoracle22222/s/developer1111-usoracle22222_myproject/scm/developer1111-usoracle22222_myproject.git`<br><br>**Note:**<br>Git over HTTPS works if your Cloud account uses federation with Oracle Identity Cloud Service. If you are federating with other identity providers, such as Microsoft Azure Active Directory or Microsoft Active Directory, Git over HTTPS won't work. We recommend using Git over SSH instead, when you use federation with identity providers other than Oracle Identity Cloud Service. |
| `git add <filename>` | Add a file that you've added to the repository's directory to the repository's index:<br>`git add readme.txt`<br>Add all new files to the index:<br>`git add --all`<br>To add a directory and all its contents to the index, navigate to the directory and use this command:<br>`git add .` |
| `git rm <filename>` | Remove a file from the repository:<br>`git rm readme.txt` |
| `git status` | Check the status of added and edited files:<br>`git status` |
| `git branch` | Create a branch:<br>`git branch new_branch`<br>List all branches in the repository:<br>`git branch`<br>Delete a *local* branch:<br>`git branch -d local_branch`<br>If the branch contains commits that haven't been merged into any other local branches or pushed to a remote repository, Git may not perform the deletion. This protects you from inadvertently losing commit data. To force the deletion regardless, use the `-D` option instead.<br>To delete a *remote* branch, you must use the `git push` command:<br>`git push origin --delete remote_branch` |

| Run this command ... | To: |
|---|---|
| `git checkout` | Checkout and switch to a branch:<br>`git checkout new_branch`<br>Pass the `-b` option in to the `git checkout` command to create a branch and switch to it immediately:<br>`git checkout -b new_branch` |
| `git merge` | Merge a branch with the checked out branch:<br>`git merge new_branch` |
| `git commit` | Commit changes to the local Git repository:<br>`git commit -m "Initial commit"` |
| `git pull` | Incorporate changes from the project's Git repository to the local Git repository:<br>`git pull origin main` |
| `git push` | Push commits to the project's Git repository:<br>`git push -u origin main` |

To display the Git help index, use the `git help` command. Use the `git help git` command to open the help index in a web browser. To display help for a particular command, use the `git help <command>`.

## Associate a VB Studio Issue with a Commit

When you save changes to a Git repository, you might want to link a VB Studio issue that's assigned to you with the commit.

To associate an issue with a commit, add `Task-URL: <issue-url>` in the commit message:

```
git commit -AM "Update for Issue 4 Task-URL:https://developer.ourcompany.com/qa-dev/#projects/mydevproject/task/4"
```

If the commit is successful, the SHA-1 checksum hash of the commit will be added to the issue. If you want, you can open the issue in the Issues page, locate the commit link in the **Commits** section under **Associations**, open the commit, and verify the SHA-1 checksum hash.

Alternatively, you could use an abbreviated form, such as "Update for Issue4" in the commit message and you'd see a link to both the commit and the issue in the Activities feed. There wouldn't, however, be a link in the **Commits** section under **Associations** in the Issues page.

## Work with Git from the Designer

Several common Git commands can be used directly from the Git menu in the Designer header or from the Options menu in the Git panel:

- Switch Branch/Switch Sandbox
  Commit your changes and then switch branches, or create a new branch from a parent branch that you select.
- Add file(s)
  Select an untracked file and mark it for inclusion (add) in your next commit to the current branch. Once the file has been added, it'll move from the Untracked category to the Changed category. You can select and add multiple untracked files in one operation.

- Save uncommitted files locally
  Temporarily **stash** changes you've made to your working copy when you're mid-way through a code change and aren't quite ready to commit. When you're ready to work on something you previously stashed, you can apply that stash to bring back your saved changes and pick up right where you left off.

  You can either **apply** a stash or **pop** it to your working branch:

  – When you apply a stash, its changes are restored and the stash remains in the stash list, so you can apply the same stash to multiple branches.

  – When you pop a stash, its changes are restored and it is removed from the stash list.

  When you no longer need a stash, you should delete it and keep your stash list clean. However, keep in mind that once you've deleted a stash, you can't recover it.

- Rollback file
  Select a changed file and revert modifications made to the file since its last commit to the current branch. This option is available only for files with changes that have been committed to the branch. The option isn't available when your workspace is in an interrupted state, like when conflicts occur during Merge or Pull operations.

- Commit
  Group changes that you've made and save them to the local repository. Can be used in tandem with the Push option, where you can commit your changes before pushing them to the remote repository as part of the push operation. You can also deselect any changed files that you do not want to commit or push.

- Status
  Display the state of the working directory and the staging area. This option tells you which changes have been staged, which haven’t, and which files aren’t being tracked by Git.

- Diff
  Compare two input data sets and output the changes (differences) between them. This option is often used with the Status option to analyze the current state of your Git repository.

- Pull
  Download and integrate remote changes. The branch that the data is integrated into is always the currently checked out HEAD branch.

- Push
  Publish new local commits on a remote server. The branch that the data is uploaded from is always the currently checked out HEAD branch.

  If you're working in a scratch repository, you can push your scratch repository's content to a new remote Git repository that VB Studio can create for you so other team members can work with your visual app.

- Reset to HEAD
  Undo changes. Moves the HEAD ref pointer as well as the current branch ref pointer.

- Merge
  Integrates changes from another branch. The branch that receives changes is always the currently checked out HEAD branch.

- Rename a Branch
  Change your current branch's name, retain its history, and push the renamed branch to the remote repository. The current remote branch won't be renamed or deleted. If the renamed branch is local only, it will be renamed, but won't be automatically pushed to the remote repository.

- Delete a Local Branch

Delete local Git branches you no longer need, after you finish working on a branch and you've pushed your changes to a remote repository. If your local branch has uncommitted changes, you'll lose them after the branch is deleted so, before deleting the local branch, commit any changes you want to keep.

- View Git history
  VB Studio keeps track of all the Git actions you perform in your workspace by logging them as action details. You can use the Git History panel to view your Git actions, see the results of each action, and keep track of what you've done in the workspace. Accessing this panel is useful for checking the sequence of recent actions and their details, especially when troubleshooting issues with version control. You can also filter the actions by various criteria (commit message, revision ID, branch name, action type, and so on) to quickly locate events and check details to understand its history.

# Use Branches

Branching lets you work on different features and updates at any time without affecting the original source code.

Before you start working on a new feature or update major portions of the source code, it's considered a good practice to create a branch and commit your changes to the new branch. This way your changes don't affect the original source code and are safe to test and review. To learn more about the Git branch workflow, read the *Git Branching - Branching Workflows* topic in the Git book at `https://git-scm.com/book/en/v2/`.

By default, all Git repositories have one default `main` branch. However, you can add more branches to the repository. You can also subscribe to email notifications for commits made to the repository's branches, and you can compare, rename, and delete branches.

## Create a Branch

You can't create a branch in an empty Git repository. First, you have to clone the repository to your computer, add and commit files to the default `main` branch that's automatically created, and then push the branch to the project's Git repository. Only after the `main` branch has been pushed to the repository, can you create additional branches.

A branch can be marked as a private branch. Only branch owners can push commits to a private branch. You must be a project member to be able to create a branch.

You can create a branch from the VB Studio Designer as well. We'll describe that here too.

From the **Git** page's **Refs** view , you can create a branch from the base branch, from the head (tip) of an existing branch, or from a tag:

| Action | How To |
|---|---|
| Create a branch from a base branch | 1. In the **Git** page's **Refs** view, click **Branches** .<br>2. From the **Repositories** drop-down list, select the repository.<br>3. Click **+ Create Branch**.<br>4. In the New Branch dialog box, in **Name**, enter the branch name. From the **Base** drop-down list, select the base revision name.<br>5. To mark the branch as a private branch, select the **Private Branch** check box.<br>6. Click **Create**. |

| Action | How To |
|---|---|
| Create a branch from the head (tip) of another branch | 1. In the **Git** page's **Refs** view, click **Branches**.<br>2. From the **Repositories** drop-down list, select the repository.<br>3. Click **+ New Branch**.<br>4. In the branch list, next to the source branch name, click **Actions** ***, and select **Branch**.<br>5. In the New Branch dialog box, enter a name for the new branch.<br>6. To mark the branch as a private branch, select the **Private Branch** check box.<br>7. Click **Create**. |
| Create a branch from a tag | 1. In the **Git** page's **Refs** view, , click **Tags**.<br>2. From the **Repositories** drop-down list, select the repository.<br>3. Click **+ New Branch**.<br>4. In the tags list, next to the tag name, click **Actions** *** and select **Branch**.<br>5. In the New Branch dialog box, enter a name for the new branch.<br>6. To mark the branch as a private branch, select the **Private Branch** check box.<br>7. Click **Create**. |

## Protect a Branch

By default, any project member can rename or delete a repository branch, and push or merge another branch into it. The project owner can protect a branch from these actions by setting restrictions on the branch:

1. In the left navigator, click **Project Administration**.
2. Select the **Branch Protection** tile.
3. Click in the search repository field and select the Git repository that has the branch you want to protect.

   All rules protecting the branches in that repository are displayed. To filter the list, type a full or partial rule name. Select a rule to display its details in the right-hand pane.
4. Select the **Branch name** radio button, click in the search field below it, and select the branch.

   If no rules have been associated with the branch, proceed to the next step.

   If there any rules already associated with the branch, those will be displayed. If you select a rule, its details will be displayed in the right-hand pane.
5. Click **+Rule** to create a new rule for the repository and associate it with a branch or multiple branches that match a defined pattern.

   The **New Protection** dialog is displayed.

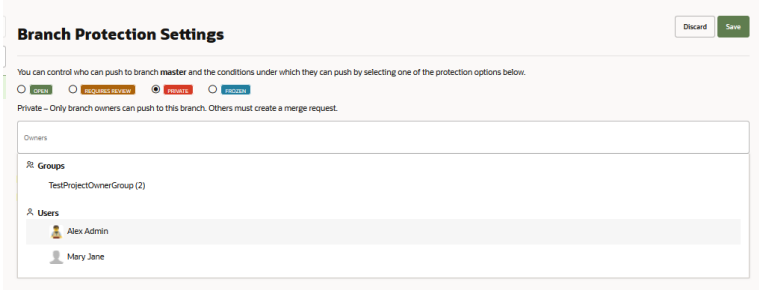
a. Click in the search repositories field and select the repository that you want to define the rule for.

b. Click in the Provide glob pattern or select branch field.

   Select one of the displayed branches to associate the new rule with a specific branch or add an expression using wildcards to associate the rule with branches that match the glob pattern you provide. See Glob Pattern Reference for Matching Branch, Job, and Pipeline Names.

c. Set the protection level (**Open**, **Requires Review**, **Private**, **Frozen**) and corresponding options.

   Here are the branch protection actions you can define:

| Action | How To |
|---|---|
| Require review and restrict merge actions | Select the **Requires Review** option and configure the review options. See Set Review and Merge Restrictions on a Repository Branch. |
| Restrict push actions to project owners and branch owners | Select the **Private** option, as shown:<br>  <br>To define branch owners, click **Owners** and select the user(s) and/or group(s). You can select multiple users (or groups).<br>To push commits to a private branch from your computer, you must always use SSH. In addition, to run a build of job that uses a private branch, you must configure the job to use SSH. |
| Lock a branch | Select the **Frozen** option. No changes are allowed to a locked branch by any user. |
| Prevent forced pushes to the branch | Select the **Do not allow forced pushes** check box. The check box isn't available when the **Requires Review** or the **Frozen** option is selected as force push aren't allowed on a review or a frozen branch. |
| Prevent renaming and deleting the branch | Select the **Do not allow renaming and deleting branch** check box. The branch can be renamed or deleted after you deselect the check box. The check box isn't available when the **Requires Review** or the **Frozen** option is selected. |

> **Tip:**
>
> On the **Refs** page, you can also click the **Open**, **Private**, **Requires Review**, or the **Frozen** branch label to edit its protection settings.

d. Click **Create**.

6. To copy a branch protection rule, click the Actions icon (**...**) for the rule, then click **Copy Rule**. On the Copy Rule dialog box, select repositories, then provide a glob pattern or select a branch. Click **Copy** to copy the pattern to the specified repositories.

   If a pattern already exists in the target repository, click **Replace** to replace the pattern, or click **Skip** to leave the existing pattern as-is.

You can also click **Cancel** to return to the Copy Rule dialog box and adjust your settings.

7. To delete a branch protection rule, click the Actions icon (**...**) for the rule, then click **Delete**.

The Activities stream on the Project Home page will report that the branch protection settings were modified.

## Glob Pattern Reference for Matching Branch, Job, and Pipeline Names

Glob syntax can be used to specify pattern-matching behavior. A glob pattern is specified as a string and is matched against a branch, job, or pipeline name. These wildcard characters can be used in glob patterns:

| Wildcard | Description |
|---|---|
| * | Matches zero or more characters of a name without crossing directory boundaries. |
| ** | Matches zero or more characters of a name crossing directory boundaries. |
| ? | Matches exactly one character. |
| [] | A bracket expression that matches a single character out of a set of characters or, when the hyphen character is used, a range of characters.<br>For example, [abc] matches "a", "b", or "c". [a-z] specifies a range that matches from "a" to "z", inclusive. Forms can be mixed, so [abce-g] matches "a", "b", "c", "e", "f" or "g". If the character after the left bracket is an exclamation mark (!), it indicates negation, so the expression [!a-c] matches any character except "a", "b", or "c".<br>Within a bracket expression, the *, ? and \ characters match themselves. The (-) character matches itself if it is either the first character within the brackets or the first character after the !, if negating. |
| {} | Represents a group of subpatterns. The group matches if any subpattern in the group matches. Uses a comma (",") to separate subpatterns. Groups can't be nested. |
| \ | Escapes characters that would otherwise be interpreted as special characters.<br>For example, the expression "\\" matches a single backslash and "\{" matches a left brace. |

The forward slash (/) represents the directory separator on all platforms.

# Manage a Branch

After you create a branch, you can rename it, compare it with another branch of the Git repository, or delete it.

You must be a project owner or member to edit and update a branch. You can perform the branch management actions from the **Refs** view of the **Git** page.

| Action | How To |
|---|---|
| Rename a branch | You can't rename a restricted branch or the `main` branch.<br>After renaming a branch, update all related merge requests, build jobs, and deployment configurations to use the new branch name. When you rename a branch, Git creates a branch with the new name and transfers all content from the old branch to the new branch. After the transfer is complete, the old branch is removed.<br>1. In the branch list, to the right of branch name, click **Actions** ***, and select **Rename**.<br>2. In the Rename Branch dialog box, in **Name**, enter the new branch name.<br>3. Select the **I want to rename the branch** check box and click **Rename**. |
| Compare branches | In the branch list, to the right of branch name, click **Actions** ***, and select **Compare**. By default, the branch is compared with the `main` branch. |
| Protect a branch or set branch restrictions | In the branch list, to the right of branch name, click **Actions** ***, and select **Protection Settings**. See Protect a Branch. |
| Delete a branch | You can't delete a restricted branch or the `main` branch.<br>1. In the branch list, to the right of branch name, click **Actions** ***, and select **Delete**.<br>2. In the Delete Branch dialog box, select the **I want to delete the branch** check box, and then select **Delete**.<br>After you delete a branch, update, close, or remove all related merge requests and build jobs. |

# Manage a Git Repository

After you've created a Git repository, you can edit its description, set its default branch, index it, and delete it but you cannot change its name:

| Action | How To |
|---|---|
| Edit a Git repository's description | On the Git page, from the **Repositories** drop-down list, select the Git repository. In the **Files** or **Logs** view, click the repository description to edit it.<br>Alternatively, on the **Project Settings**: **Repositories** page, mouse over the Git repository name, click **Menu** ☰, and select **Edit** ✎. In the Edit Repository dialog box's **Description** field, enter or edit the repository description, and click **Update**. |
| Set the default branch | When you open a Git repository on the Git page, the contents of the default branch are displayed. By default, the `main` branch of a Git repository is set as the default branch. However, you can set any branch to be the default branch of a Git repository.<br>On the **Project Settings**: **Repositories** page, mouse over the Git repository name, click **Menu** ☰, and select **Edit** ✎. From the Edit Repository dialog box's **Default Branch** drop-down list, select the branch, and click **Update** ✓. |

| Action | How To |
|---|---|
| Index a Git repository | Indexing a Git repository creates or updates the Git repository index file with the latest changes. A Git index file is a binary file that serves as a virtual staging area for the next commit. This file contains a sorted list of object path names, each with a blob object's permissions and the SHA-1.<br>To index a repository, on the **Project Settings**: **Repositories** page, mouse over the Git repository name, click **Menu**, and select **Index**. |
| Delete a Git repository | On the **Project Settings**: **Repositories** page, mouse over the Git repository name, click **Menu**, and select **Delete**. In the Remove Repository dialog box, click **Yes**. |

# Copy a Git File/Repository's URL

From the Git page, you can copy and share the URL of a Git repository, a file in the Git repository, or a line in a file in the Git repository.

Before you share the URL, remember that only project members can use the URL to access the file or clone the repository. If the project is shared, organization members can also access files in the project's repository or clone the repository, but they can't update them.

These are the copy URL actions you can perform from the Git page:

| Action | How To |
|---|---|
| Copy a Git repository's URL | To clone a Git repository or to access it using a Git client, you use the URL of the repository. You can copy the URL from the Project Home page's **Repositories** tab, the Git page, or from the Project Settings : Repositories page.<br>In the Project Home page's **Repositories** tab or the Project Settings : Repositories page, search for the Git repository, and click the **Clone** drop-down list to see the HTTPS and SSH URLs of the repository. To the right of the URL, click **Copy** (or select the URL and press `Ctrl + C` or use the mouse context menu) to copy the URL to clipboard.<br>**Note:**<br>Git over HTTPS works if your cloud account uses federation with Oracle Identity Cloud Service. If you are federating with other identity providers, such as Microsoft Azure Active Directory or Microsoft Active Directory, Git over HTTPS won't work. We recommend using Git over SSH instead, when you use federation with identity providers other than Oracle Identity Cloud Service.<br>The SSH URL of an external Git repository isn't available. |
| Copy the URL of a file in the Git repository | In the **Files** view of the Git page, open the file. From the address bar of the browser, copy the URL. |

| Action | How To |
|---|---|
| Copy the URL of a line in a file in the Git repository | In the Git page's **Files** view, open the file. On the left side of the line, in the number column, click the line number. The entire line is selected. From the address bar of the browser, copy the URL.<br>Example: To copy the URL for line number `2` in `myfile.txt`, click the line number 2. Clicking the line number updates the URL in the browser's address bar to `http://developer.us2.oraclecloud.com/my-org/#projects/demo/scm/demo.git/blob/myfile.txt?revision=main&sl=2`. You can copy and use this URL to open `myfile.txt` in the `demo.git` repository – `main` branch with line number `2` selected. |
| Copy the URL of a group of lines in a file in the Git repository | In the **Files** view of the Git page, open the file. On the left side of the line, in the number column, click the line numbers with the `Shift` key pressed to select them. From the address bar of the browser, copy the URL.<br>Example: With the `Shift` key pressed, clicking line numbers `2` through `5` of `myfile.txt` selects those lines. The URL in the browser's address bar changes to `http://developer.us2.oraclecloud.com/my-org/#projects/demo/scm/demo.git/blob/myfile.txt?revision=main&sl=2-5`. Copy the URL and share it with project members. When the URL is entered, the `myfile.txt` file of the `demo.git` repository – `main` branch opens with line numbers `2` through `5` selected. |

# View File/Repository History

Use the Git page's **Logs** view to see the history of commits, branches, and merges of a file or Git repository and its revisions.

1. From the **Repositories** drop-down list, select the Git repository. From the **Revisions** menu, select the branch.
2. To view the commit history of a file, browse to and open the file.

   Skip this step to view the commit history of the selected Git repository.
3. On the right side of the page, click **Logs**.

| Action | How To |
|---|---|
| View the commit history in a list format | In the **Logs** view, click the **History List** toggle button.<br>To view the history of another branch or tag, in the text box to the right of the **History** toggle button, enter branch or tag names. You may also click the text box and select the revisions from the drop-down list. You can add multiple branches or tags. To view the history of all revisions of the selected Git repository, remove all revision names from the text box. |
| View the commit history as a graph | In the **Logs** view, click the **History Graph** toggle button.<br>In the graph:<br>• Each dot represents a commit.<br>To see the details of the commit, click the dot.<br>• A splitting line represents a branch.<br>• Joining lines represent a merge.<br>• Latest commits appear at the top of the graph. |

# Use Tags

Tagging lets you mark a specific point of time in the history of the repository. For example, you can create a tag to mark the Git repository state of an application's stable state, before a release.

## Create and Manage Tags

From the **Git** page's **Refs** view, you can create and manage a Git repository's tags.

You must be a project owner or member to create and manage tags:

| Action | How To |
|---|---|
| Create a tag | 1. In the **Git** page's **Refs** view, select the repository from the **Repositories** drop-down list.<br>2. Click **Tags** .<br>3. Click **+ Create Tag**.<br>The New Tags dialog box is displayed.<br>4. Enter the tag name in the **Name** field.<br>If you choose a name that's the same as a branch name in the Git repo, you'll see a warning. Although you're not strictly prohibited from using this name, doing so can cause confusion in places where the branch is referred to, like in build jobs. It's best to choose an original name.<br>5. Click the **Base** selector and, from the list, choose the base revision.<br>6. Click **Create**. |
| Rename a tag | 1. In the **Tags** view, select the tag name to modify from the list.<br>2. Click the Actions menu ⋮, to the right, and select **Rename**.<br>The Rename Tag dialog box is displayed.<br>3. Enter the new tag name in the **New Name** field.<br>4. Select the **I want to rename the tag** check box.<br>5. Click **Rename**. |
| Compare a tag | 1. In the **Tags** view, select the tag name to modify from the list.<br>2. Click the Actions menu ⋮, to the right, and select **Compare**.<br>The Compare screen opens, comparing the selected tag with the default branch.<br>3. |
| Delete a tag | 1. In the **Tags** view, select the tag name to modify from the list.<br>2. Click the Actions menu ⋮, to the right, and select **Delete**.<br>The Delete Tag dialog box is displayed<br>3. Click the **I want to delete the tag** check box.<br>4. Click **Delete**. |

# Compare Revisions

You can compare any two revisions of a Git repository. The base revision indicates the starting point of the comparison and the compare revision indicates the end point. The revision could be a branch, a tag, or a commit SHA-1 checksum hash.

Here's how to compare two revisions of a Git repository:

1. On the right side of the **Git** page, click **Compare**.
2. From the **Base Revision** drop-down list on the left, select the base revision.

   By default, the **Git** page selects the last commit of the repository as the base revision and the selected revision as the compare revision.
3. From the **Compare Revision** drop-down list on the right, select the compare revision.

You can compare these revisions:

- Branch versus branch
- Tag versus tag
- Commit versus commit
- Branch versus tag
- Commit versus branch
- Tag versus commit

On the Compare Result page, the **Changed Files** tab and the **Commits** tab. The **Changed Files** tab lists files that have changed between the base revision and the compare revision. The **Commits** tab lists all commits that have happened between the base revision and the compare revision since their common commit. The **Commits** tab is enabled if **From Merge Base** is selected in **From Merge Base or Revisions** .

| Action | How To |
|---|---|
| Compare with a parent of the base revision | From the **Base Revision** drop-down list, click the **Parents** tab, and then click the SHA-1 checksum hash of the parent commit. |
| View differences between the base revision and the compare revision since the last common commit to both revisions | From the **From Merge Base or Revisions** drop-down list, select **From Merge Base (...)** . Select **Revisions (..)** to show the differences between the heads (or tips) of the base revision and the compare revision. |
| Switch the base revision and the compare revision | Click **Switch Base** . |
| Create or open a merge request | If a merge request exists with the **Compare Revision** as the review branch, click the merge request button to open the merge request review page.<br>If a merge request doesn't exist, click **+ Merge Request** to create a merge request with **Base Revision** as the target branch and the **Compare Revision** as the review branch. |
| View the compare options | Click **Diff Preferences** to view various compare options. |

## Add Comments to a File

When you're comparing files, you can add inline comments, which will be visible to all project users, to the source code changes made to a file:

1. Browse to and open the file.
2. On the right side of the page, click **Logs**.
3. For the commit that changed the file and added the changes you want to comment on, click the button with the first seven characters of the commit's SHA-1 checksum hash as the label.
4. In the **Changed Files** tab of the Compare view, mouse-over the line number of the file and click **Add Comment**

   If you selected the **Unified** view, click the line number in the second column. If you selected the **Side by Side** view, click the line number of the file on the right.
5. In the **Leave a comment** box, enter the comment, and click **Comment**.

The comment is added as an inline comment to the file and is visible to all project members. To reply to a comment, click **Reply**, enter the comment in the **Leave a reply** box, and click **Comment**.

# Watch a Git Repository

You can watch a Git repository branch and receive email notifications when any file of the branch is updated in the project's Git repository.

To get email notifications, enable them in your user preferences, and then set up a watch on the branch from the Git page's **Refs** view.

Here's how to subscribe to email notifications and get them when updates happen in branches you are watching:

| Action | How To |
|---|---|
| Enable email notifications | In your user preferences page, select the **SCM/Push Activities** check box. |
| Watch a branch | 1. Open the project.<br>2. In the left navigator, click **Git**.<br>3. On the right side of the page, click **Refs**.<br>4. If necessary, click **Branches**.<br>5. In the branch list, to the right of the branch name, click **cc**.<br>Alternatively, click **Actions**, and select **Subscribe**.<br>A **Subscribed** icon appears indicating that you are subscribed to email notifications of the branch updates.<br>To unsubscribe, click **cc** again. |

# Search a Git Repository

You can search the project's Git repositories for a file name, directory name, or a term in the source code files, file paths, and file revisions.

The search field supports common programming languages, such as HTML, JavaScript, CSS, and Java. You can use these features while searching terms:

- Language-aware
- Auto-suggest
- Symbols (class and function names) and file names
- CamelCase

Here's how to search for a term in Git repositories:

| Action | How To |
|---|---|
| Search for a term in a Git repository and a revision | 1. From the **Repositories** drop-down list, select the Git repository. From the **Revisions** drop-down list, select the revision.<br>2. In the top-right corner of the page, in the **Search Code** box, enter the search term or select it from the suggestion list.<br>3. Click **Search**. |
| Search for a term in all revisions of a Git repository | 1. From the **Repositories** drop-down list, select the Git repository. From the **Revisions** drop-down list, select the revision.<br>2. In the top-right corner of the page, in the **Search Code** box, enter the search term or select it from the suggestion list.<br>3. Click **Search**.<br>4. In the **Revisions** drop-down list, click **Reset**.<br>The **Revisions** drop-down list now shows **All Revisions**. |
| Search for a term in all Git repositories | 1. From the **Repositories** drop-down list, select the Git repository. From the **Revisions** drop-down list, select the revision.<br>2. In the top-right corner of the page, in the **Search Code** box, enter the search term or select it from the suggestion list.<br>3. Click **Search**.<br>4. From the **Repositories** drop-down list, select the **All Repositories** option, or click **Reset**. |

The search result page displays all files, file paths, and file revisions that contain or match the search term (or symbol). To reset the search, to the left of the **Search Code** box, click **Back**.

> **Note:**
>
> If you search for multiple words, grouped instances of the words will appear highlighted together in the results even if they are separated by a space, a hyphen, or a dot.

## Download a Git Repository's Archive

If a Git repository's branch or tag isn't required, and if you plan to delete it, it's considered good practice to create and back up an archive of the branch or tag before you delete it. From the **Refs** view of the Git page, you can download an archive file (zip or tgz) for a Git repository's branch or tag:

| Action | How To |
|---|---|
| Download a branch's archive | 1. In the Git page's **Refs** view, click **Branches**.<br>2. From the **Repositories** drop-down list, select the repository.<br>3. In the branches list, to the right of the branch name, click ***, select **Download**, and then select **zip** or **tgz**. |
| Download a tag's archive | 1. In the Git page's **Refs** view, click **Tags**.<br>2. From the **Repositories** drop-down list, select the repository.<br>3. In the tags list, to the right of the tag name, click ***, select **Download**, and then select **zip** or **tgz**. |

# Review Source Code with Merge Requests

Reviewing source code can help you avoid bugs, identify design issues, and catch design and implementation problems that might affect application performance. To get the source code reviewed, you need to create a merge request.

## Merge Requests Concepts and Terms

As the name suggests, a merge request is a request to merge a branch into another. Before merging the branch, you may want your team members to review updates made to the branch and share their feedback. A merge request combines the review and merge processes into one easy collaborative process.

You can also link related issues and builds to the merge request that are automatically updated or triggered when you merge branches.

Here are the terms that this documentation uses to describe the merge request features and components:

| Term | Description |
|---|---|
| Review branch | Branch to be reviewed and merged. |
| Target branch | Branch that the review branch will merge into. |

| Term | Description |
|---|---|
| Reviewer | Project user invited to review the changed files of the review branch. |
| Requester | Project user who created the merge request. |
| Subscriber | Project user who isn't a reviewer, but is watching the merge request. |
| Default reviewer | Project user who's automatically added as a reviewer if a branch is selected as the review branch. Only a project owner can create default reviewers of a branch. |
| Approved | Reviewer's feedback with no objection to the changes made to the source code in the review branch. |
| Rejected | Reviewer's feedback with objections to changes made to the source code in the review branch and a recommendation not to merge the review branch into the target branch. |
| General comment | A comment in the Conversations tab of the merge request. |
| Inline comment | A comment added to a line of a file under review. |
| Pending (or unpublished) comment | An inline comment that you didn't publish when you added it. |

To understand the workflow of a merge request, let's consider you're a software developer assigned a new feature to implement. These steps summarize the action you'd perform to set up a merge request and merge branches:

1. Create a branch from a stable branch (say `main`) of the source code Git repository. You'd add or update the files of the new branch to implement the new feature.

   You can do this in the cloned Git repository on your computer or on the VB Studio **Git** page.
2. On your computer, pull the latest content from the project's Git repository, checkout the new branch, update the required files, and commit and push the checked out branch to the project's Git repository.
3. If required, create a build job to generate artifacts from the new branch to verify the stability of the application.
4. Create a merge request with the new branch as the review branch and the stable branch as the target branch.
5. Add your manager and other team members as reviewers.
6. To resolve the feature related issues when you close the merge request, link the issues to the merge request.
7. Depending on the review feedback, you may need to update some files and check the stability of the branch. To trigger a build of the job automatically when you update the files of the review branch, link the job to the merge request.
8. Again, based on the feedback and build status of the linked jobs, you may want to merge the branch with the stable branch or abandon it. If you merge the branches, the linked issues are automatically resolved.

If you're invited to a merge request, you can add comments to the updated files, and share your feedback whether you've any objection to merge branches:

1. Open the merge request.
2. Check the commits made to the review branch and compare the changed files.
3. Add general or inline comments, if necessary.

4. Submit your feedback as **Approved** if you find the code updates acceptable, or **Rejected** if you have objections.

If you're a project member but aren't invited to a merge request, you can add comments but you can't share your feedback.

It isn't necessary to add reviewers to a merge request. If you're sure that the changes made to the review branch don't require a review, you can merge both branches without a review.
If you're comfortable using Git, you can merge branches from a Git client without creating a merge request.

## Merge Request States

A merge request can be in one of these states:

| State | Description |
|---|---|
| **Open** | Code review is in progress.<br>A merge request's status remains **Open** or **Draft** until the branches are merged or the request is closed. |
| **Draft** | Merge request is not ready for code review.<br>Draft status blocks the merge action, mutes notifications to reviewers, and prevents linked builds from running. |
| **Merged** | Code review is complete and the review branch has been merged with the target branch.<br>The review is closed for inline comments, but can accept general comments. |
| **Closed** | Code review is closed without merging the review branch with the target branch.<br>The review is closed for inline comments, but can accept general comments. |

# Create and Update a Merge Request

After creating a merge request, you can update it by adding reviewers and linking related issues and jobs to it.

> **Note:**
>
> If you see the "E-mail address of PersonIdent must not be null" error message when you create or update a merge request, make sure that your user email address has been verified. Access your profile, verify your user email address, and retry the merge request.
> See Configure Your Global Email Notifications to see how to access your user profile and configure these notifications.

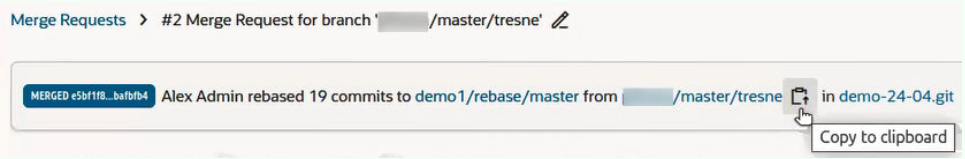
## Create a Merge Request

You must be a project member to create a merge request from the Merge Request page. You can't create a merge request if the branch that you want to be reviewed has any merge restrictions set or is already under review in another merge request.

Here's how to create a merge request:

1. In the left navigator, click **Merge Requests** .
2. Click **+ Create Merge Request**.
3. On the Branch page of the New Merge Request wizard, in **Repository**, specify the Git repository.
4. In **Target Branch**, select or enter the branch that the review branch will be merged into.
5. In **Review Branch**, select or enter the name of the branch to be reviewed. If the branch doesn't exist, it'll be created.

   If the review branch is already under review in another merge request, the branch name won't appear in the **Review Branch** list.

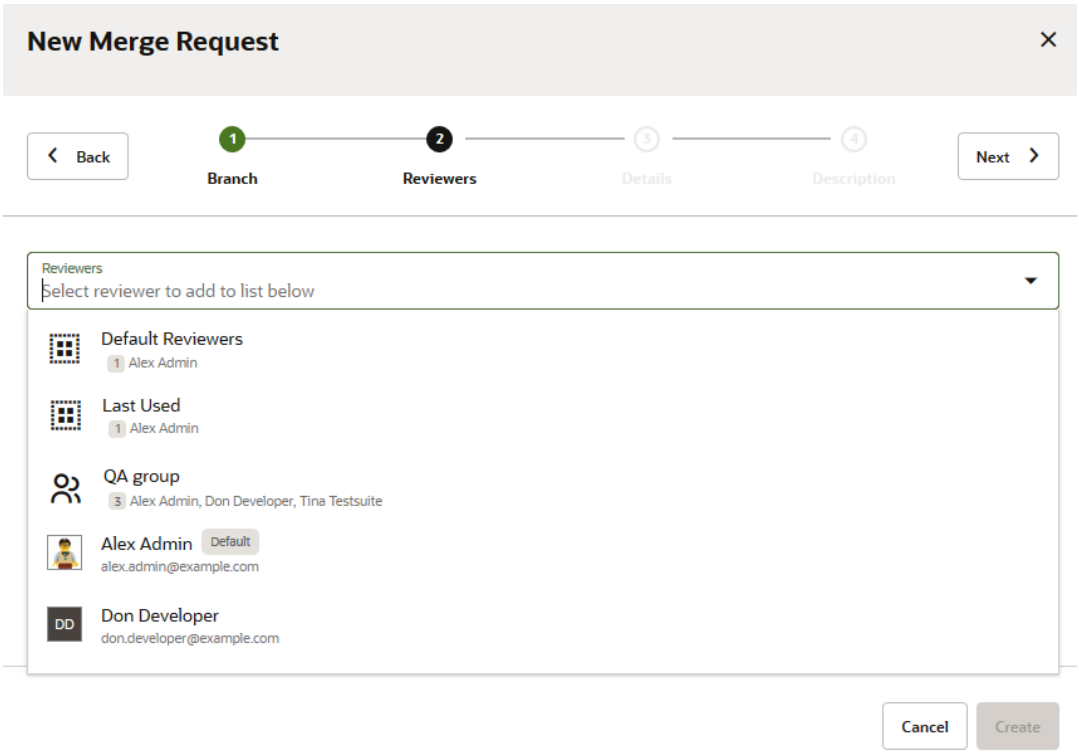
> **Tip:**
>
> You can use the **Copy** icon in the Merge Request page's top header to copy the review branch name to the clipboard.
>
> 
> 
>
> This can be useful if your repository has many branches or if the branches have long names.

6. Click **Next**.
7. On the Details page of the New Merge Request wizard, in **Linked Issues**, add issues related to the merge request.

   When you're merging branches or closing the merge request, you can mark the linked issues as resolved.
8. In **Linked Builds**, add jobs related to the merge request.

   Builds of the linked jobs will be run automatically whenever the review branch is updated.
9. In **Tags**, add project tags to associate them with the merge request.

   You can use these tags to search merge requests.
10. In **Summary**, enter a summary (or title) of the merge request. If one isn't specified, the default summary `Merge Request for branch <review_branch_name>` will be set.
11. In **Reviewers**, select team members and groups who'll review the updates.




If the branch is protected, at least one default reviewer must approve the merge request. Here's what you can select:

- To add all reviewers of the last merge request you created, select **Last Used** .
- To add all the default reviewers for the protected branch, select **Default Reviewers** .
- To add default reviewers individually, select each one from the list displayed. Default reviewers are identified by the Default Reviewer tag next to their name. Reviewers that aren't default reviewers can also be selected from the list and can be added individually.
- To add a group of team members, select a group that has been previously defined by your organization administrator, such as the QA group in the image. This can be quite a timesaver, especially if the group you're adding has many members.

To remove a selection, click the **X** icon to the right of the selection.

12. Click **Next**.
13. On the the New Merge Request wizard's Description page, enter a description, and click **+ Create**.

    You can use the project's wiki markup to format the description.

After the merge request has been created, you're assigned the **Requester** role and all reviewers are assigned the **Reviewer** role. Email notifications are sent to reviewers informing them that they've been added as reviewers.

> **Note:**
>
> If you see the "E-mail address of PersonIdent must not be null" error message when you create or update a merge request, make sure that your user email address has been verified. Access your profile, verify your user email address, and retry the merge request. See Configure Your Global Email Notifications to see how to access your user profile and configure these notifications.

## Create a Merge Request from the Command Line

If you use Git commands to manage source files from your computer, you can create merge requests from the command line when you're publishing changes to a project's repository. You can also add reviewers to merge requests made from the command line.

Use `git push` options to create a merge request that publishes changes from your local branch to a remote branch:

`git push -o mr.target=<target-branch> origin <feature-branch>`

where:

- *<target-branch>* is the branch where your changes will be merged.
- *<feature-branch>* is the branch to be reviewed. If the *feature-branch* you specify is already under review, the merge request won't be created.

For example, this command creates a merge request for the branch `myfeature` before merging to main:

```
git push -o mr.target=main origin myfeature
```

> **Note:**
>
> The Git push `-o` option is available only with Git version 2.10 or higher. With these versions, you can use the `--push-options` option or the shorter `-o` option.

If your merge request option includes text with spaces, enclose the text in double quotes (" "). If there are no spaces, the quotes are optional.

To add reviewers to the merge request, include the `mr.add.reviewer` option (or use the `mr.add.defaultReviewers` option to set default reviewers for the target branch). For example, this command identifies two reviewers (`clara.coder` and `tina.testsuite`) by their user names:

```
git push -o mr.target=main -o
mr.add.reviewer=clara.coder,tina.testsuite origin myfeature
```

Both users will be added as reviewers to the merge request for the `myfeature` branch.

If you're adding reviewers from an Oracle Cloud Application environment that has multiple VB Studio instances in different IDCS identity stripes (or IDM identity domains), use the domain username format that includes the stripe or domain where that user has been defined. Since a unique user may have been defined for multiple domains or stripes, this format ensures that one identity can be distinguished from that user's other unique identities. This should ensure that the user you select is the correct one.

To add a reviewer from a multi-stripe environment to the example above, you need to pass the full domain username, which would be something like : "<idcs-idm_identifier_for_environment_user_is_associated_with>/username" ". This identifier indicates that the username belongs to a unique user in a specific instance in the multi-stripe environment.

The command for adding user tina.testsuite in a specific identity stripe as a reviewer is as follows:

```
git push -o mr.target=main -o mr.add.reviewer="idcs-e13d5ccf9...7684a6b283ce/
tina.testsuite" origin myfeature
```

**Tip:**

If you frequently create merge requests, it may be helpful to add an alias for the create merge request option. This needs to be added to the `.git/config` file at the local level for each repository (refer to Git documentation for details). For example:

```
cat .git/config
[alias]
   review = push -o mr.target=main
```

After doing that, you can use the `review` alias to create a merge request for the `myfeature` branch and add `clara.coder` as the reviewer, for example:

```
git review -o mr.add.reviewer=clara.coder origin myfeature
```

**Note:**

You cannot use the options to create a merge request and add reviewers with the `git push --all` command or for references other than the HEAD branch.

You can also add a group as reviewers. After you clone a repository and make some changes in a branch (for example, feature123), add and commit your changes. Create a group (for example, reviewers123) and add it as a member of the project that contains the repository you cloned. Then, from your clone, issue a command similar to this one:

```
git push -o mr.target=main -o mr.add.reviewer="reviewers123" origin feature123
```

Here are the additional options for a merge request, including adding a summary, description, and linking issues to the request:

| Option | Description |
|---|---|
| `mr.reviewers \| mr.reviewer \| mr.add.reviewer` | Add reviewers to a merge request. Separate multiple reviewers with commas. For example:<br><br>`-o mr.reviewers=clara.coder,tina.testsuite` |

| Option | Description |
|---|---|
| `mr.default-reviewers` \| `mr.add.defaultReviewers` | Add the default reviewers to a merge request. For example:<br><br>`-o mr.add.defaultReviewers` |
| `mr.summary` | Add a summary of the merge request. For example:<br><br>`-o mr.summary="Summary of the MR"` |
| `mr.description` | Add a description of the merge request. For example:<br><br>`-o mr.description="Description of the MR."` |
| `mr.issues` \| `mr.issue` | Link issues to a merge request. Separate multiple issues with commas. For example:<br><br>`-o mr.issues=VBS-33997,VBS-33998` |

> **Note:**
>
> If a merge request already exists, you don't need to specify the target branch when adding an issue or a reviewer (`-o mr.target` can be omitted).

After your changes have been successfully pushed and the merge request has been created, click the `See merge request` link included in the command output to view the merge request that was created for you in VB Studio, for example:

```
user123@rmt123 /tmp/code2cloud.example (myfeature) $ git push -o
mr.target=main origin myfeature

Enumerating objects: ...

...

remote: [Push Options] See merge request: http://192.0.2.1:8888/test/?
_h=projects/test_example/review/39
```

If you omit push options when pushing updates to a branch without a merge request, the command output includes the command format to create a merge request, along with a link to the New Merge Request UI wizard for the branch. The link opens the New Merge Request wizard with the Repository and Review Branch fields already populated. For example:

```
C:\Users\user123\git\tutorial-hr-project>git push origin hrbranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 22 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1)
```

```
remote: [Merge Request] Do you want to create a merge request? Use git push -
o mr.target=<target-branch> origin <feature-branch>
remote:  or click on https://url123.com/?_h=projects/dev-test_user123-
test_40689/reviews&new=true&repo=tutorial-hr-project.git&branch=hrbranch
remote: Updating references: 100% (1/1)
To https://url123.com/s/dev-test_user123-test_40689/scm/tutorial-hr-
project.git
   249fcd5..76f7aaa  hrbranch -> hrbranch
```

If you omit push options when pushing updates to a branch that already has a merge request, the command output includes a link to the updated Merge Request UI page for the branch. For example:

```
C:\Users\user123\git\tutorial-hr-project>git push origin hrbranch
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 22 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 323 bytes | 323.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1)
remote: [Merge Request] https://url123.com/s/dev-test_user123-test_40689/
review/1 (updated)
remote: Updating references: 100% (1/1)
To https://url123.com/dev-test/s/dev-test_user123-test_40689/scm/tutorial-hr-
project.git
   76f7aaa..7a7c0cd  hrbranch -> hrbranch
```

## Set a Merge Request to Draft State

Apply **Draft** state to a merge request to indicate that it is not ready to be merged and to apply specific behavior changes to the merge request.

When a merge request is in **Draft** state, its merge action is blocked, notifications to reviewers are paused, and linked builds are prevented from automatically running.

You can create a merge request in **Draft** state by either toggling the **Draft** button in the New Merge Request form's Description field or by prepending the keyword "[Draft]" to the Description field's summary.

In existing merge requests, you can switch between **Open** and **Draft** states by clicking the **Draft** button next to the merge request. The merge request's badge changes from **Open** to **Draft** to indicate that the merge request is not ready for review.

## Use Templates to Improve MR Descriptions

You can use MR description templates to improve and simplify MR decriptions by guiding and motivating developers to add more information about what changed in the codebase. This, in turn, will help code reviewers better understand the subject of the review, ultimately helping them to provide better feedback.

The MR template definition is stored in the Git repository so it can be tracked as it evolves. Different templates can be used for different target branches, so a specific template can be mapped to a specific branch. Individual teams will define the template content, which follows standard markup syntax (Markdown, Confluence, Textile).

Only `.txt` and `.md` (Markdown) files will be recognized as MR description template files. The templates must be in your repository's default branch.

There are two types of MR description templates:

- Default MR description templates, where there is one template for all branches.
- Branch-specific MR description templates, where there are different templates for different branches.

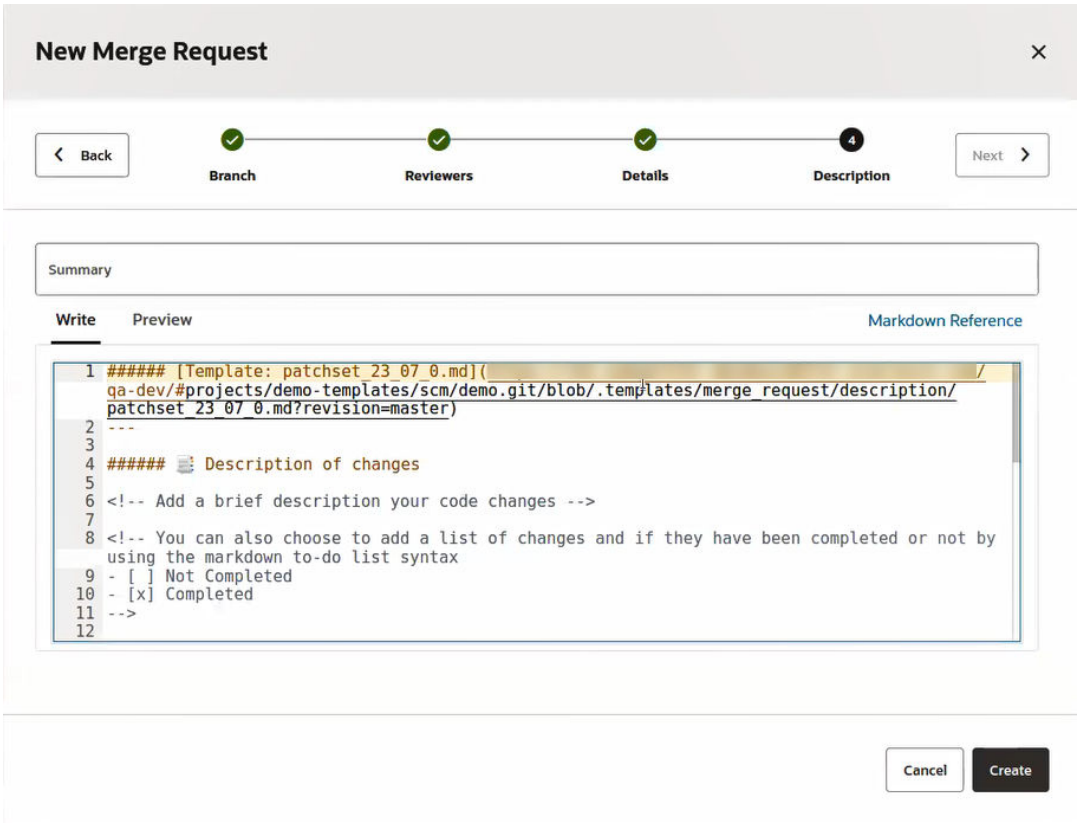
1. Create your template definition file(s) for the MR description(s) and save them to your Git repository's default branch's `.templates/merge_request/description` folder.

   Make sure you save your template to files that have a `.txt` or `.md` (Markdown) extension. If you use any other extension, your files won't be recognized as MR description template files.

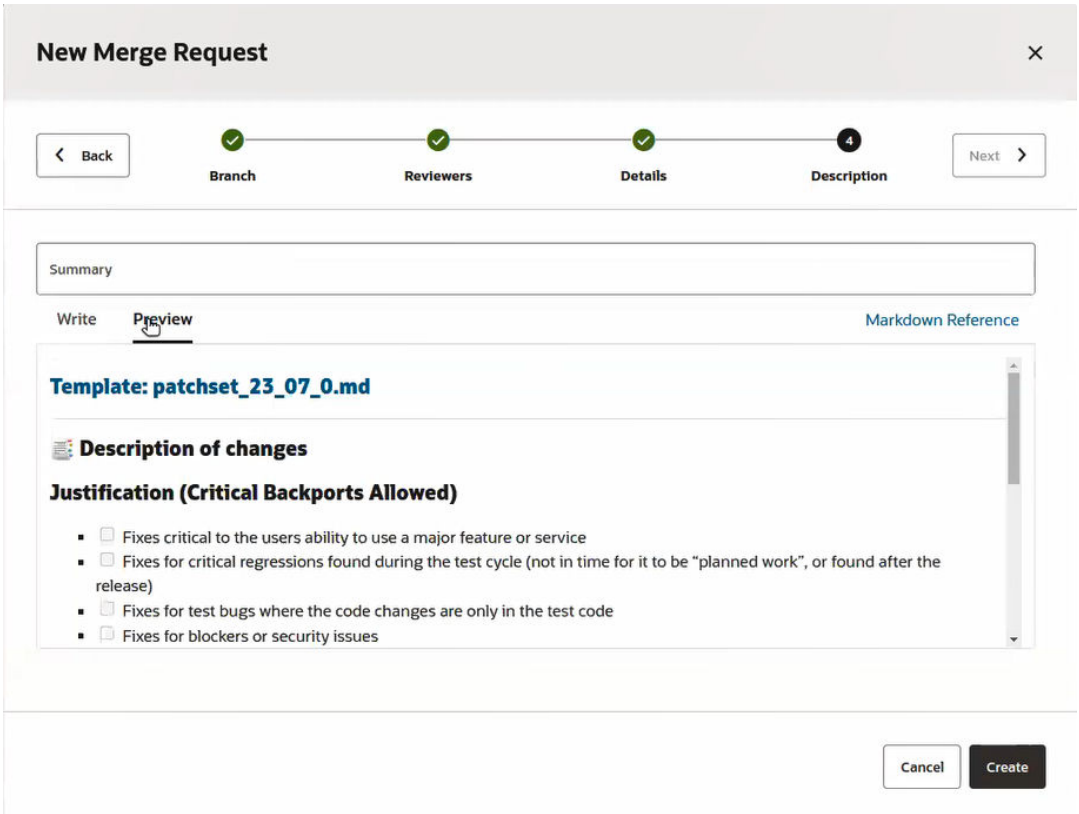
   > **Tip:**
   >
   > Open the terminal in the root of the folder and run the `cat` command (`cat .git/HEAD`) to read the contents of HEAD.

2. Name your template file according to how you want it to be used. The type of merge request definition template is determined by its filename:
   - The name for default MR description templates is `default_template.md`.
   - The name for branch-specific templates must exactly match the branch name or match the first level of the branch name, such as main, patchset, dev, release, feature, and so on.
     The template named `patchset.md` would be acceptable for version-based branch naming. The template would be applied to patchset or patchset/* (for example, patchset/24/01/0). Other separators (slash, underscore, hyphen, and dot) are supported, so it would also work for patchset/*/*.* or patchset_*_*_* or patchset-*-*-* or patchset.*.*.*.
   - You can also use issue.md and feature.md for ticket-based branch naming, such as issue/VBS-1234 or feature/29706999.
3. In the left navigation menu, select **Merge Requests** and display the **Merge Request** page.
4. Click **+ Create Merge Request** to bring up the MR wizard.
5. In the Branch dialog, select the repository, target and review branches, and click **Next**.
6. In the Reviewers dialog, select your reviewers and click **Next**.
7. In the Details dialog, search and link issues and build jobs, select tags, and click **Next**.
8. In the Description dialog, fill in the summary.

   Notice that if your branch name matches your MR description template name, the **Description** is filled out for you.




9. Click Preview to see how it looks rendered.



10. Make sure that the templated description matches the code changes and other modifications you want reviewers to look at.
11. Click **Create** to create the MR.

Once the merge request has been created, the template's contents are available in the **Conversation** tab, where any developer can edit the description and add more information or insert comments.

## Add or Remove Reviewers

You can add reviewers when you create a merge request or while the review is open. You must be the requester or a reviewer to add or remove reviewers.

1. In the left navigator, click **Merge Requests** .
2. Click the merge request summary to open it.

You can manage the reviewers from the Review Status section available on the right side of the page.

| Action | How To |
|---|---|
| Add a reviewer | 1. Above the **Review Status** section, click **Click to add a reviewer**.<br>2. From the **Reviewers** drop-down list, enter the project member name or select the member.<br>3. Click **OK**. |
| Add a group | 1. Above the **Review Status** section, click **Click to add a reviewer**.<br>2. From the **Reviewers** drop-down list, enter the group's name or select it from the dropdown list's Groups category.<br>3. Click **OK**. |
| Add a code owner | 1. Above the **Review Status** section, click **Click to add a reviewer**.<br>2. From the **Reviewers** drop-down list, enter code owner or select it from the dropdown list's Code Owners category.<br>3. Click **OK**.<br>See Assign Owners to Files in the Project's Git Repository for more information about code owners. |
| Add yourself (project member) as a reviewer | If you're a project member but not a reviewer, you can submit a request to add yourself as a reviewer to a merge request.<br>Above the **Review Status** section, click **Add me** . If you're a project owner, you'll be added automatically to the merge request. If you're a project member, enter a justification for why you want to be added in the **Request to be added as a reviewer** dialog box, and click **OK**. |

| Action | How To |
|---|---|
| Add yourself (not a member of the project) as a reviewer | Non-members can search and view MRs in shared projects. If you're *not* a member of a shared project, you can still submit a request to add yourself as a reviewer to a merge request:<br>1. Above the **Review Status** section, click **Add me**.<br>2. Enter a justification for why you want to be added in the **Request to be added as a reviewer** dialog box.<br>3. Click **OK**.<br>A reviewer can approve your request to join the merge request as a reviewer. After becoming a reviewer, you can approve or reject the review and add comments to the merge request.<br>**Note:**<br>You won't be able to create/merge/close MRs; add/remove reviewers, linked issues, or linked builds; or start linked builds. Because you aren't a project member, you won't be able to do what a project member can. |
| Approve a reviewer request | If you're the requester or a reviewer, you can approve requests of project users to join the merge request as reviewers. In the Conversation tab, click **Add User** in the **requested to be a reviewer** request. |
| Remove a reviewer or a group | 1. Above the **Review Status** section, click **Click to add a reviewer**.<br>2. Click **Remove Reviewer** next to the reviewer or group you want to remove.<br>3. Click **OK**. |

## Link an Issue to a Merge Request

Linking issues to a merge request enables you to resolve them automatically when you merge or close a merge request:

1. Open the merge request.
2. Click the **Linked Issues** tab.

   The tab displays issues linked to the merge request.
3. To link an issue to the merge request, enter the issue summary text or the issue ID in the **Search and Link Issues** search box, select the issue from the drop-down list, and click **Save**.

## Link a Build Job to a Merge Request

Linking build jobs to a merge request enables you to monitor them from the merge request and trigger them when a commit is pushed to the review branch. Depending on the build's status, reviewers can determine whether the merge request is ready to be merged with the target branch.

1. Configure the job to accept merge request parameters.

   See Use Build Parameters.

2. Open the merge request.
3. Click the **Linked Builds** tab.

   The tab displays linked jobs, if any.

4. In **Search and Link Build Jobs**, enter the job name and select it from the list.

> **Note:**
>
> When you create a job and select the **For Merge Request** option in the Create Job dialog, VB Studio automatically adds the Merge Request parameters and parameterized Git settings to the job configuration. These jobs appear in the list of jobs that can be linked to a merge request.
>
> To convert an existing job (a job that wasn't created using the **For Merge Request** option) into a Merge Request job, add the Merge Request parameters (MERGE_REQ_ID, GIT_REPO_BRANCH, GIT_REPO_URL) manually in the **Parameters** tab and parameterize the Git setting (with ${GIT_REPO_URL}) in the **Git** tab. Only after you do that, will the job will appear in the list of jobs that can be linked to a merge request.

5. Click **Save** ✓.

After a job is linked to a merge request, a build automatically runs when the review branch is updated with a commit.

> **Note:**
>
> Linked builds will not automatically run while a merge request is in **Draft** state.

When a build of a linked job runs, a comment is automatically added to the **Conversation** tab. If the build succeeds, it will auto-approve the merge request and add itself to the **Approve** section of the Review Status list. If the build fails, it will auto-reject the merge request and add itself to the **Reject** section of the Review Status list.

## Add an Attachment (Image or File) to a Merge Request Comment

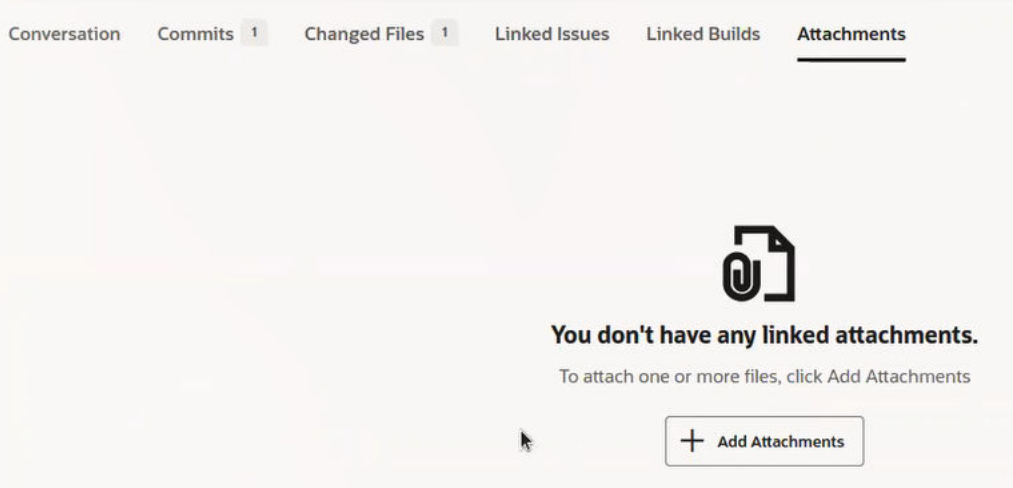
You can use the **Attachments** tab on the **Merge Request** page to add an uploaded screen shot, image, or file to a merge request comment:

1. Open the merge request.
2. Go to the **Conversation** tab.
3. In the **Write** tab, use the toolbar and add some text and images to a comment in the **Conversation** tab.

   The toolbar provides several frequently-used features (add headings, boldface, italics, or strikethrough text; add quotes, code, links, or lists; or add a file or image) that reduce or eliminate the need to use markdown to implement the same features. See Use the Toolbar to Work with Comments.

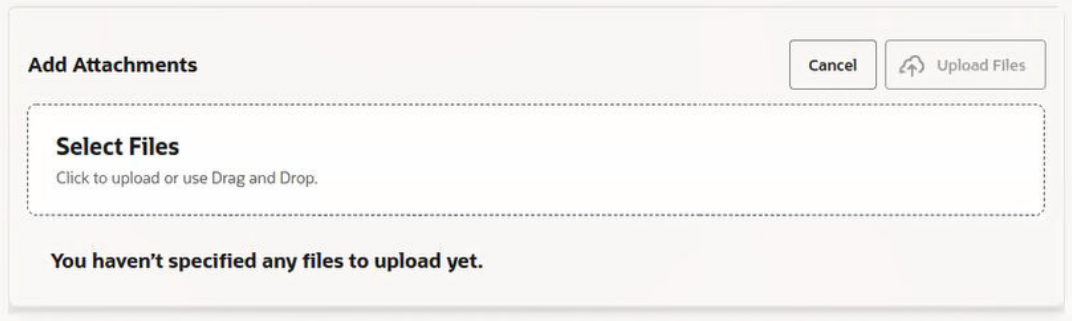
4. Click the **Attach a file or image** icon in the toolbar or go directly to the **Attachments** tab.

   If you have no attachments, this is what you'll see:



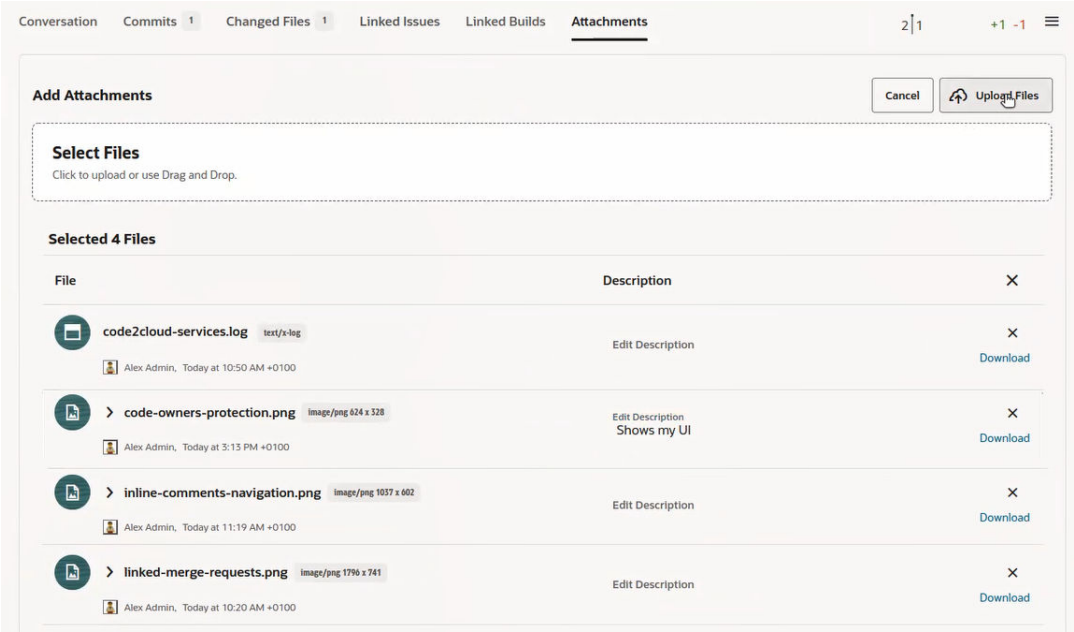

5. To attach one or more files, click **+ Add Attachments**.

   The Add Attachments page is displayed.




6. Click in the **Select Files** area and use the file selector or drag and drop the files to the area.

   If you're using the file selector, choose the files to upload and click <uicontrol>Open</uicontrol>. You can attach files that are smaller than 100MB. The page displays a list of the selected files below the Select Files area.



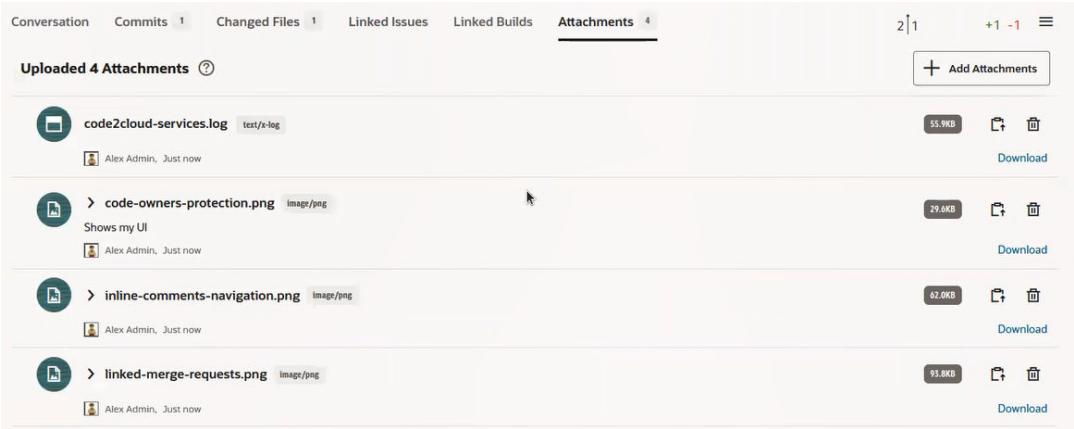

**Tip:**

From this list, you can:

- View statistics about the file (date created, owner, file type, and the DPI if it is an image file).
- Click > to preview the file, add or edit the description, which can be up to 80 characters long.
- Download the file.
- Delete the file from the list.

7. Click **Upload Files** .

The uploaded attachments are listed in the tab.



> **Tip:**
>
> From this list, you can:
>
> - View statistics about the uploaded files, including owner, upload date/time, file size, and file type.
> - Click › to preview the file.
> - Click **Delete** to remove it from the list of uploaded attachments.
> - Click **Copy URL** to copy its URL to use later if you plan to add the attachment to a comment using markdown directly. You don't need to copy the URL if you are using the toolbar. It will insert the URL in the markdown it generates for you.
> - Download the attachment.

8. Add the screen shots, images or files you just uploaded in the **Attachments** tab to the comment.
   - (Recommended) With the toolbar:
     a. If you are creating a new comment, click the **Attach a file or image** icon in the **Write** tab's toolbar.
     The Insert Attachments dialog box is displayed.
     b. Select the image or file to add to the comment.
     c. Add the screen shots, images or files you just uploaded in the **Attachments** tab to the comment.
     d. Click **Insert Attachments**.
     The **Write** tab shows the markdown that was inserted for you.
     e. (Optional) Select the **Preview** tab to see what the image or file looks like in the comment.
     f. Click **Add Comment**.
   - To work directly with wiki markup in comments, instead of using the toolbar, for adding screen shots, images, and files:
     a. Click the **Help** icon and open the link for the project's wiki markup reference, where you'll find the syntax you need to use for adding the attachment.
     If, for example, you're using markdown, the default wiki markup language for VB Studio projects, according to the reference, you'll need to use this syntax to add an attachment to the comment:

     ```
     ![alt text](image_url)
     ```

     b. Go to the **Attachments** tab and click **Copy URL** to get the URL you'll use for `image_url`.
     c. Append `images/myImage.png` to it, replacing `images` with the folder name where the image file is located and `myImage.png` with the name of your image file.
9. If you are editing an existing comment, go to the **Conversation** tab, locate your comment in which you want to add the image as an attachment, click **Edit** [icon], and either use

the toolbar to add the image to the comment or drop in the markup with the URL you copied from the **Attachments** tab.

10. (Optional) Click the **Preview** tab and make sure that what you added looks correct.
11. Click **Save**.

## Use the Toolbar to Work with Comments

You can use the markup toolbar to work with comments in the Conversations tab. The toolbar provides commonly used formatting actions so that you don't have to remember or search for the right markup language syntax you need (whether Textile, Confluence, or Markdown). The toolbar also provides options to customize the content editor itself, so you can toggle line numbers on and off, change line wrapping, and show (or hide) the content editor's minimap.

1. Open the merge request.
2. Go to the **Conversation** tab.
3. In the **Write** tab, add some text to a comment then use the toolbar as needed.

   For more details about how the toolbar adds phrase or block modifiers when quoting text or adding code snippets, see Use the Markup Toolbar.
4. (Optional) Go to the **Preview** tab and make sure that what you entered looks correct.
5. Click **Add Comment** if you're creating a new comment or **Update** if you're modifying an existing comment.

# Watch a Merge Request

You can set up a watch on a merge request and get email notifications when a reviewer adds a comment, a user updates files of the review branch, or a reviewer shares a feedback.

Here's how to subscribe to email notifications for merge request updates when you are a reviewer and when you are not:

| Action | How To |
|---|---|
| Merge requests where you're a reviewer | By default, you get email notifications of merge requests where you're a reviewer. If you aren't getting the email notifications, select the **Merge Request updates and comments** check box in your user preferences page.<br>1. In the branding bar, click the user avatar, and select **Preferences**.<br>2. Click the **Notifications** tab.<br>3. Select the **Merge Request updates and comments** check box, if not selected.<br>4. To the left of the **User Preferences** title, click **Close** ‹ to return to the last opened page. |
| Merge requests where you're not a reviewer | 1. Open the merge request.<br>2. Click **+ Add me**. You'll see this button for an open merge request if you're a project member, but neither the submitter nor a reviewer.<br>To stop watching, remove your name from the **Watchers** list. |

> **Note:**
>
> A merge request in **Draft** state has muted update notifications until it is set to the **Open** state.

## Merge Request Email Notifications

If you're a reviewer, requester, or watcher, you'll receive email notifications when the merge request is created or updated. A notification of the event will also appear in the Recent Activity feed on the **Project Home** page.

Email notifications for a single merge request share the same subject line for easier reading when your inbox is grouped by subject.

Some events that send notifications are:

- Merge request is created
- Additional source code changes are committed to the review branch and pushed to the upstream
- A general comment is added
- An attachment is added
- An issue is linked
- A build job is linked
- An inline comment is published
- Reviewers are added or removed
- Merge request is approved or rejected
- Merge request is closed or merged

> **Note:**
>
> If you see the "E-mail address of PersonIdent must not be null" error message when you create or update a merge request, make sure that your user email address has been verified. Access your profile, verify your user email address, and retry the merge request. See Configure Your Global Email Notifications to see how to access your user profile and configure these notifications.

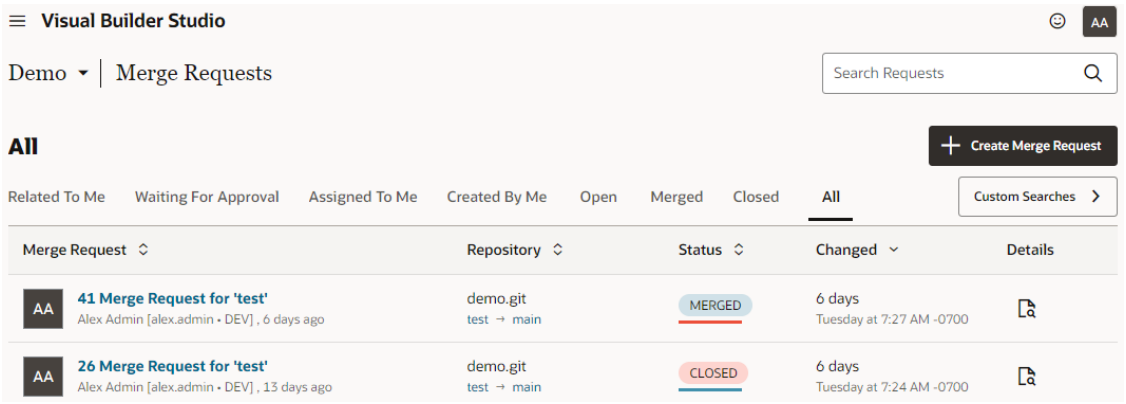
# Review a Merge Request

To review a merge request, on the Merge Request page, click its summary. On the Review page, you can view the commits of the review branch, review changed files, add inline and general comments, and submit your feedback.

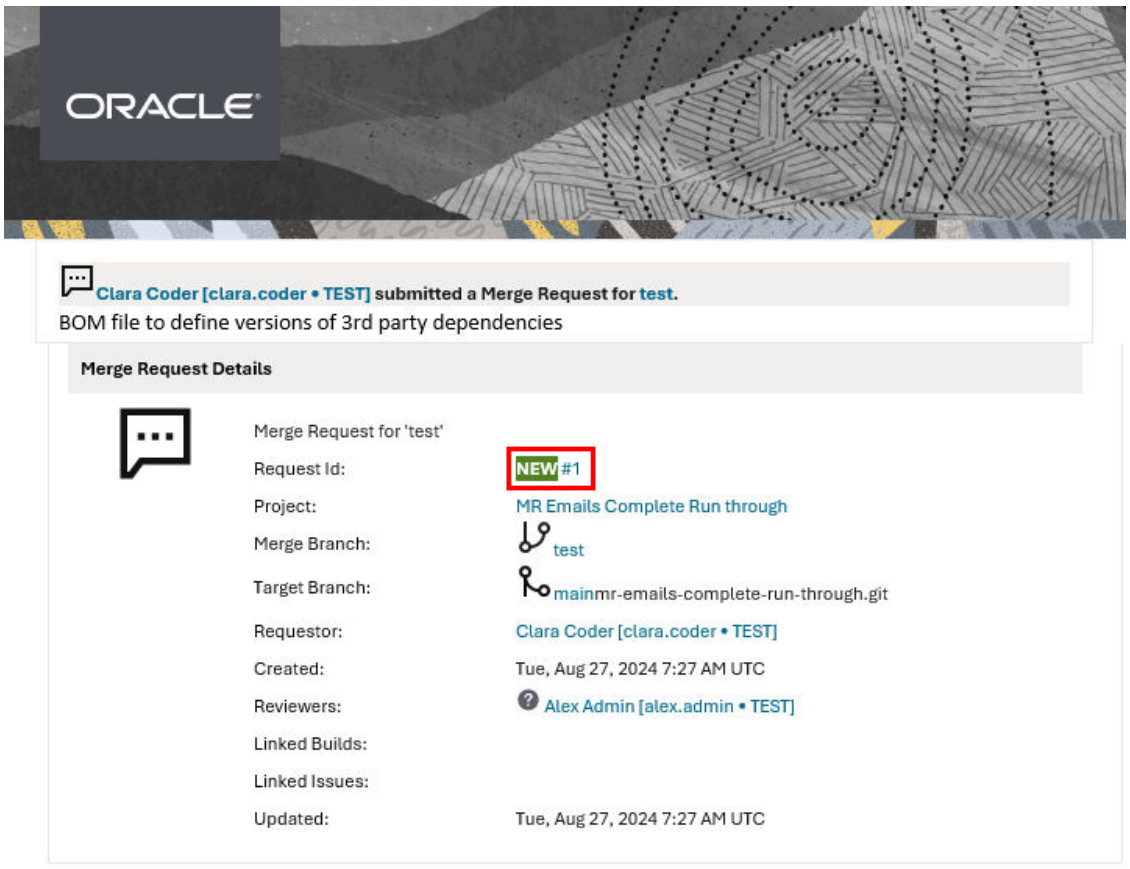
## Open a Merge Request

To open a merge request, on the Merge Requests page, click its name.

Use the filter tabs to search for the merge request. **Related To Me**, **Waiting For Approval**, **Assigned To Me**, **Created By Me**, **Open**, **Merged**, **Closed**, and **All** filters are available.




If you're invited to a merge request, you can also click the request ID from the email notification.




If you still can't find the merge request through the available filters, use the search box at the top of the page or click **Custom Searches >** to run an advanced search.

You can also save the advanced search for future use. From the Settings icon, click **Save As** and then, in **Search Name**, enter a name and click **Save As**.

## View Commits and Changed Files

You can view commits and changed files from the **Commits** and the **Changed Files** tabs.

The **Commits** tab shows all commits made to the review branch. Here are several common actions you can perform from the **Commits** tab:

| Action | How To |
|---|---|
| Compare the files of one commit with another | Click the button with the first seven characters of the commit's SHA-1 checksum hash. By default, the page compares the commit with the previous commit. |
| View all files of the repository when the commit was pushed to the branch | Click **Code**. |
| View files that were updated in the commit | Click **Show Details**. To compare a file with its parent commit, click the file name to compare the file changes with its previous commit. |

The **Changed Files** tab shows the files in the compare mode. Here are some common actions you can perform from the **Changed Files** tab:

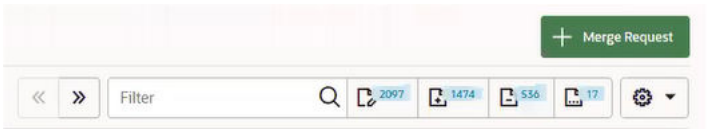
| Action | How To |
|---|---|
| Select the **Changed Files** tab to open the tree view and show changed files | Click **Changed Files Tree** . |
| View the compare options | Click **Diff Preferences** . |
| Add a comment to a code line or reply to one | Mouse over the line number of the file and click **Add Comment** . |

### Tips for Working with Merge Requests that Have Large Numbers of Changed Files

If you're working with merge requests that have a large number of changed files, you have several ways to narrow your view and focus in on just the files or commits that you are interested in examining:

- On the upper left side of the Merge Request window, above the tree view, VB Studio prominently displays the number of changed files shown in the tree and, if the number is very large, the number of files that aren't shown in the tree. The maximum number of files that can be displayed is 4,000.

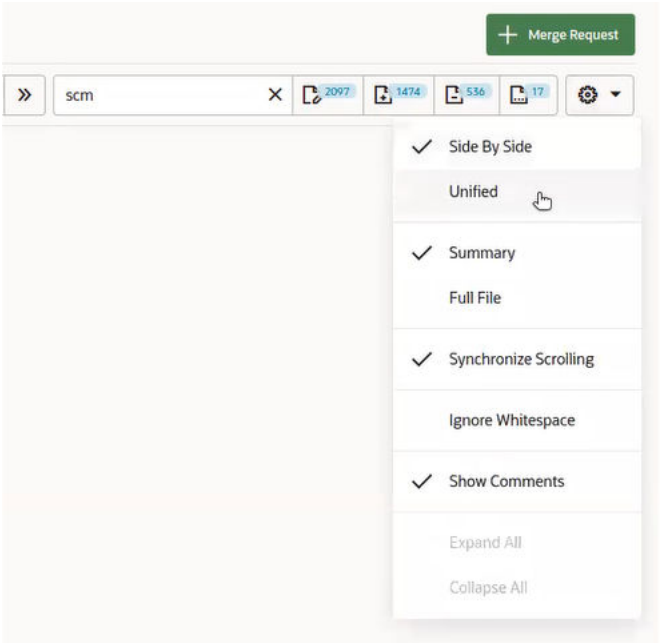
- You can use the directional arrows to traverse the tree. Use the ‹ or « left arrows to ascend the tree and the right arrows › or » to descend it.




- You can enter a search term (perhaps a file type, such as .css or .js) in the **Filter** field, to limit your view to a particular file type.
- You can use the quick filter buttons on the upper right side of the screen to focus in on the files that you want to examine. At the touch of a button, you can:
  - **Show modified files**

  - **Show new files**
  - **Show removed files**
  - **Show renamed files**

- From the **Settings** menu, you can select a single option or multiple ones that adjust what you see in the tree.




  You can also expand all or collapse all files in the tree with one click.

You'll have access to many similar options in the **Conversations** tab, so you can see the history of commits, comments, approvals and rejections and focus in on the items that you're interested in examining when the merge request is a large one.

## Add a General Comment

In the **Conversation** tab, you can view ongoing conversations and add comments. Your comments can be generic, questions you want to ask reviewers, or comments about an event, such as a commit. You can also edit your comments after entering them or filter them to make it easier for you to find a specific comment in a long thread.

Here's how to add a comment to an ongoing conversation:

1. Open the merge request.
2. In the **Conversation** tab's **Write** tab, enter your comment.

> **Tip:**
>
> Use the project's wiki markup to format the comment and then click the **Preview** tab to see what it looks like. To remove all of the comment text, click the **Clear comment** icon.

3. Click **Add Comment**.

The **Conversation** tab shows the comment you added, along with icons you can use to **Reply**, **Edit**, and **Delete** your comment. Note that you can only edit or delete your own

comments, not those added by other users. You can, however, reply to any comment in any open merge request.

## Add an Inline Comment to a File

When a code review is in progress, you can add inline comments directly to the lines of code in a file. You can't, however, add an inline comment after a merge request has been merged or closed.

Here's how to add inline comments directly to the source code being reviewed:

1. Open the merge request.
2. Click the **Changed Files** tab.
3. Mouse-over the line number of the file where you want to add your comment and click **Add Comment** .
4. Add your comment in the comment box.
   - Use the project's wiki markup language to format the comment.
   - Click **Comment** to publish it and make it visible to all reviewers.

     You can't edit or delete a published comment.
   - Click **Save** to save the comment and publish it later. The comment isn't published so it isn't visible to reviewers.
   - Click **Cancel** to discard the comment. It won't be added.

Click the **Pending Comments** tab to see your pending or unpublished comments.

To reply to a published comment, click **Reply** , enter your comment, and click **Reply**. Replies to published comments will be published immediately but you can only edit or delete them if you added them. In addition, you can't edit or delete comments added by other users. You can only reply to those.

To filter published comments, select **Filter** and enter a term or partial string to filter by.

To edit one of your comments, select the comment and then click **Edit** . Make your changes and then click **Save** to post your edited comment.

## Manage Unpublished Comments

The **Pending Comments** tab displays all pending comments with the code where these comments were added. The comments appear inline in the code.

Here are several things you can do with unpublished comments:

- To edit a comment, click **Edit** .
- To publish a pending comment, click **Publish** to the right side of the comment header.
- To publish all pending comments, click **Publish All**.
- To discard all pending comments, click **Discard All**.
- To delete a comment, click **Delete** .

## Manage Inline Comments

You can use the **Changed Files** tab to work with inline comments. Click the **Line comment navigator** icon to open the comments list and navigate to a comment.

Each inline comment includes icons you can use to update the comment.

- Use the **Expand** and **Collapse** icons to expand and collapse the comment thread. Outdated comments (that applied to subsequently deleted code) are collapsed by default.
- To edit a comment, click the **Edit comment** icon.
- To delete a comment, click the **Delete comment** icon.
- To publish a pending comment, click the **Publish comment** icon.
- To reply to a comment, click the **Reply** icon, then select **Reply**. You can also select **Need Attention**, **Addressed**, or **Resolve**. This supports the Need Attention > Addressed > Resolved workflow. **Resolve** is always available unless a comment is marked Need Attention – a comment marked Need Attention must be addressed before it can be resolved.

# Filter a Merge Request Conversation

The Conversation tab on the Merge Request page displays commits, comments, and approvals related to the selected merge request. If there are many entries, you may want to filter these items to more easily locate a particular item you are looking for, say to reply to another reviewer's comment.

To filter merge request conversations:

1. In the Merge Request page, select the **Conversations** tab.
2. On the right side of the page, select **Filter Options** .
   The **Filter conversation** dialog box is displayed.

3. Use the dialog box to select the criteria by which you want to filter the comments:
   a. In By User, select **All**, **Only mine**, or **Select any** then select one of the users who is a reviewer.
      The **Select any** option displays only the names of team members who entered comments in the selected MR, not all users who are reviewers. You're prompted to

select one or more of the team members whose comments you want to see or remove the team member or members whose comments you want to filter out.

b. In By Date, select **Anytime**, **Today**, **Yesterday**, **Last 3 days**, **Last 7 days**, or **Last 30 days**.

c. In By Text, enter any text string to search for.

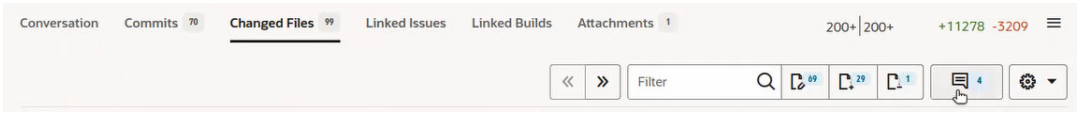
d. In By Type, select **Commits**, **Comments**, and/or **Approvals**. Shaded choices are active.

You can also sort the entries by clicking the **Sort Options** dropdown and selecting Oldest First, Newest First, and/or Keep Order. You can combine choices, such as Oldest First and Keep Order, too.

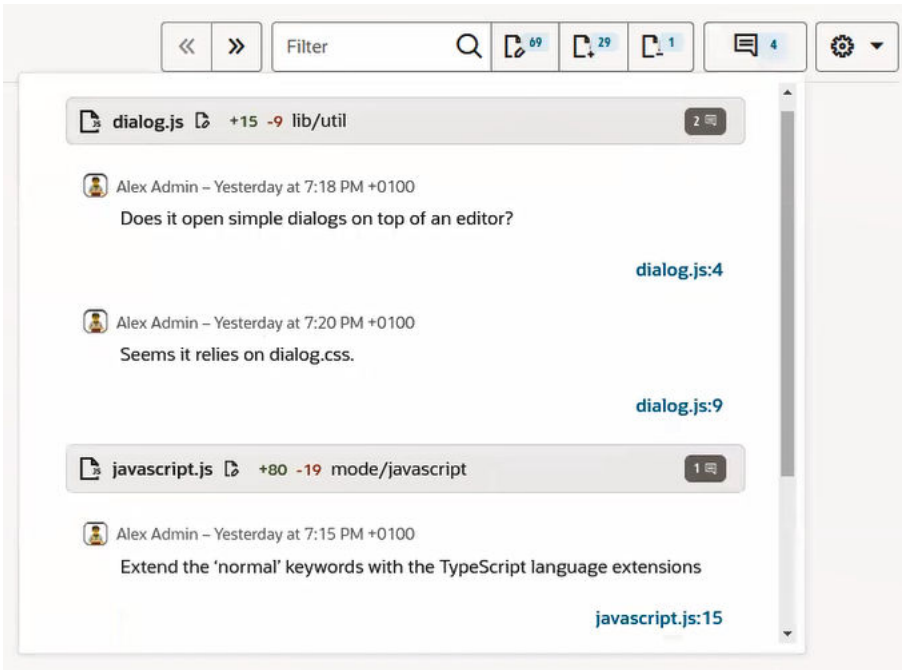
## Navigate Multiple Comments

By using the **Line comment navigator** button on the floating tool bar in the Merge Request page's **Changed Files** tab, you can quickly locate files that have comments and easily navigate from comment to comment within a file. This can be quite a timesaver, especially when there are many changed files and the files themselves are large or contain many comments.

1. Open the merge request.
2. Go to the **Changed Files** tab, click the **Line comment navigator** button on the floating toolbar.

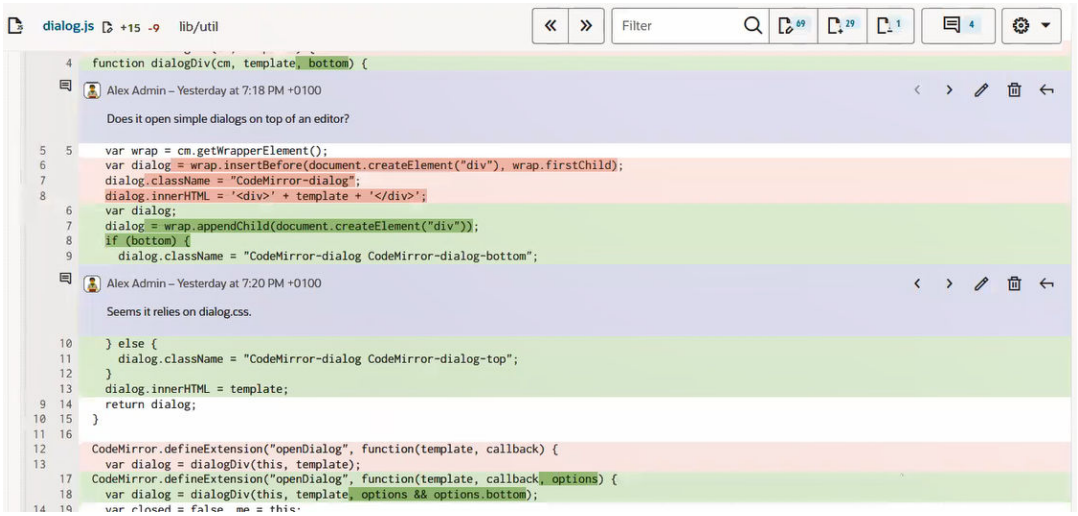



The popup displays a list of comments.




In this MR, there are two comments in the first file and one in the one shown further down the list.

3. Click the first file name link (`dialog.js`) for the comment you want to go to and the file will open in the code editor.

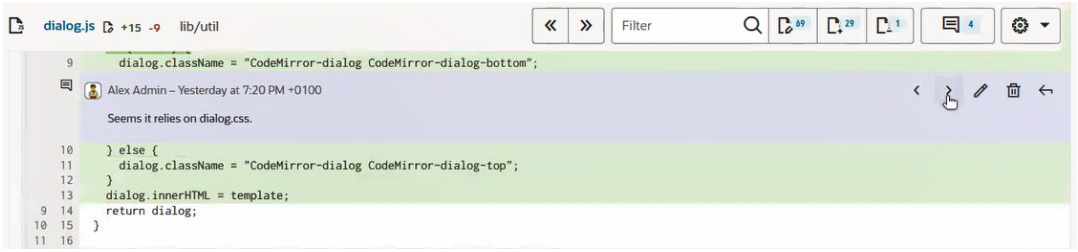



Comments are displayed inline and are identified by the icon to the left.

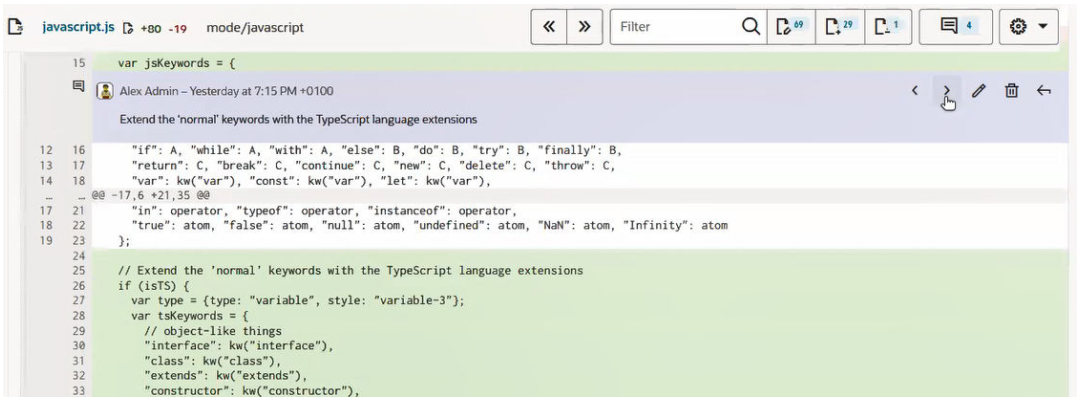
4. On the right side of the comment, you can use the **Previous comment** ‹ and **Next comment** › icons to navigate to the previous or next comment.

   The `dialog.js` file has two comments. Notice that the **Previous Comment** ‹ icon in the first comment is grayed out. It is the first comment found in the changed files, so there is no previous comment to go to.

5. Click the **Next comment** › icon to go to the second comment in the `dialog.js` file.




6. Click **Next Comment** › icon again and you'll go to the comment in the next changed file, `javascript.js`.



Note that the floating navigation menu is always visible so, if you need to, you can always return to the list of comments by clicking the **Line comment navigator** button.

## Approve or Reject a Merge Request

As a reviewer, after you review the source code, you can add a special comment that indicates whether you approve the code changes or reject them. Approving a merge request implies that you don't have any objections to changes made to the source code. Similarly, rejecting a merge request implies that you've an objection and don't recommend merging branches.

> **Note:**
>
> If the **Merges are only allowed when there are no comments with a Needs Attention status** setting on the Branch Protection page was selected and a reviewer has added a comment with a Needs Attention status, the issue the reviewer raised must be addressed and the status cleared before the merge can happen.

Note that if you created the merge request, but didn't add yourself as a reviewer, you can't approve or reject the merge request. However, you can still close it or merge it with the target branch.

Here's how to approve or reject a merge request:

1. Open the merge request.
2. Click the **Approve** or **Reject** button at the right side of the page.
3. In the dialog box that appears, add your comment, and click **OK**.

   Use the project's wiki markup to format the comment.

You can see your feedback (approval or rejection) in the **Reviewers** list.

# Merge Branches and Close the Merge Request

After addressing reviewers' comments, you can decide whether to merge the branches or cancel the request.

Before doing that, go to the Review Status section and check the status for the reviewers and the status for any linked build jobs. Here, in addition to approvals and rejections, you can find out who is actively reviewing the merge request. When a reviewer adds a change or a comment, their icon is moved to the Review Status section and a badge representing progress is visible. This badge also displays when a reviewer rejects or approves the merge request.

Depending on the number of approvals, rejections, or non-responses, you can decide whether you want to merge the review branch, wait for more approvals, or cancel the request.

## Merge Branches

There are several different ways to merge a review branch into the target branch. You can merge commits, squash and merge, rebase and merge, or merge the branches manually. You don't need to get approvals from all reviewers before merging the review branch. If the target

branch is locked, you won't be able to merge the review branch without first contacting the project owner to unlock the target branch.

> **Note:**
>
> In a merge request, when you merge a review branch with the target branch, you merge all of the commits in the review branch. If you want to merge a particular commit or just some commits in the review branch, you should use the `git cherry-pick` command on the Git command line to apply the commit changes to the target branch. For more information, see https://git-scm.com/docs/git-cherry-pick.

To merge branches, you must be assigned either the reviewer or requester role for the merge request:

1. Open the merge request.
2. On the right side of the page, click **Merge**.
3. In the Merge dialog box, click **Merge Options**, and select the merge type:

| Use this merge type ... | To: |
|---|---|
| **Create a merge commit** | Merge all the review branch's commits to the target branch. The merge commits continue to show two parents. |
| **Squash and Merge** | Add the review branch's commit history to the target branch as a single commit. |
| **Rebase and Merge** | Reapply the review branch's commits and add them to the top of the target branch. |
| **Manual Merge** | Follow the on-screen commands to merge the branches using the Git CLI. |

> **Note:**
>
> A merging message will be displayed on the **Merge Request** page (and on the **Conversation** tab), based on the merge type used to complete the merge operation. The message is shown as Merged, Rebased (shown below), or Squashed.
>
>
> 
>
> The enhanced messaging distinguishes the type of operation that was used to complete the MR and helps the user understand the expected result shown in the Git commits tree. In addition, the Merged badge shows the range of commits/ commit IDs.

At the top of the dialog box, select the **Remember My Choice** check box to use the current option as the default setting the next time you open the Merge dialog box.

4. If necessary, update the **Merge Summary** and **Merge Description**.

The fields aren't available if you select **Rebase and Merge** or **Manual Merge**.

5. Select **Delete Branch** to delete the review branch after the commits have been merged with the target branch.

   This option is enabled by default when the **Delete the review branch after Merge Request is merged into target branch** option is set in the Merge Options tab in the Branch Protection Settings screen. See Set Review and Merge Restrictions on a Repository Branch.
   If the option was enabled, a confirmation dialog is displayed when the user deletes the review branch after the Merge Request is merged. The dialog asks if the user wants to delete the review branch. Click **No** if you want to retain the branch or **Yes** to delete it after the merge. This helps prevent an unwanted branch deletion after the **Delete the review branch after Merge Request is merged into target branch** option has been enabled.

6. If there are any linked issues, deselect the check boxes for the issues that you don't want to mark as resolved after the commits are merged with the target branch. By default, the check boxes for all linked issues are selected.

7. Submit the dialog box.

After the review branch has been merged, the merge request will be closed automatically. No other action is allowed.
If you didn't select the **Delete branch** check box when you merged the review branch, note that the review branch wasn't removed from the Git repository. You can continue to make commits to the branch and create another merge request to review the new source code.

## Resolve a Merge Conflict

Git can automatically resolve code conflicts when the review branch is merged with the target branch. In some cases, however, the conflicts must be resolved manually.

On the Merge Request page, if the **Merge** button is replaced by the **Merge Conflicts** button, it indicates a merge conflict.

Git automatically resolves conflicts if different files of the target and review branches are updated before both branches are merged. Merge conflicts are reported when the same lines of the same files are updated in the review branch and the target branch before both branches are merged. Most people will use the browser-based conflict resolution tool:

1. Open the merge request that has conflicts.
2. Click **Merge Conflicts**.

   The Merge Conflicts dialog opens.

3. Click **Resolve Conflicts** to open the conflict resolution editor.

   The pane on the left indicates the number of files with conflicts and lists them. It also shows the number of conflicts in each file. The pane on the right, the code editor, displays the highlighted file with conflicts shown in the left pane.

4. Use the right arrow to go to the first (or next) conflict.

   You can use the left arrow to go to the previous conflict, if there is one. To see the differences that cause the conflict, click **View Diff**.

5. To resolve the conflict, click the conflict marker, the circle next to the line numbers, to select the change to use.

   One marker will select **Use Their Change**, the other will select **Use Our Change**. The marker turns red to indicate your choice.

6. Go through the file and resolve each conflict independently or use **Resolve # Conflicts**, where # indicates the number of conflicts in the file, to resolve all conflicts in a file the same way.

   Continue until there are no more conflicts. If you're not satisfied with the resolution you chose last, click **Discard Resolution**. To discard all your selections and start over, click **Discard All**.

7. Click **Update Review** to commit the files to the review branch.

8. Click **Merge** and push the commit to the target branch.

Files in the review branch that no longer had conflicts were merged with the target branch. No additional action is required on the Merge Request page. If you want to delete the review branch, open or refresh the Merge Request page, and click **Delete Branch**.
If there are too many files with conflicts or if the files with conflicts are too large, you need to manually review each conflicting file in the review branch with the code of the same file in the target branch in a text editor and resolve the conflicting lines of code. Then, you need to follow the Git commands displayed in the dialog to resolve the conflicts with the Git command line:

1. On your computer, open the Git CLI.
2. If you've already cloned the project Git repository, navigate to its directory.

   If you haven't cloned the Git repository, clone it.

3. Run the commands shown on the Merge Conflicts dialog.

   The commands help you resolve the conflict and mark the conflicting code lines in files.

4. Open each file that contains conflicts in a text editor.

   Content with conflicts is marked with `<<<<<<<`, `=======`, and `>>>>>>>`. The lines between `<<<<<<<` and `=======` show the code from the target branch. The lines between `=======` and `>>>>>>>` show the code from the review branch.

5. Review the content and update it. Remember to remove the `<<<<<<<`, `=======`, and `>>>>>>>` from each conflicting file before saving it.

6. Save all files and commit them.

   Run the `git status` command to view the status of conflicting files.

7. Push the commit to the target branch.

> **Note:**
>
> If you see the message "Update requires 'Merge-Request' line present in the commit-message" when you are trying to push a commit to the remote repository, it doesn't indicate a merge conflict at all. It simply points to the fact that the remote repository you are pushing changes to is protected. The development branch in your remote repository has a merge restriction, possibly for Code review approval, before the MR can be merged. It was set up to only accept tags that point to commits with a specific commit message that starts with the text, "Merge-Request: ...". The remote repository was set up this way (Requires Review) to make sure that all changes (commits) have gone through the intended merge request workflow, where every change is reviewed and approved before being pushed.

Conflicting files in the review branch are now merged with the target branch. No additional action is required on the Merge Request page.

## Close a Merge Request

You must close a merge request after the review branch has been merged. However, it isn't necessary to merge the review branch to the target branch before closing a merge request. You can close a merge request if it was created by mistake or if you just don't want to merge the review branch to the target branch.

Make sure that you perform any needed merge actions before you close the request because once a merge request is closed, you can't merge the review branch, add comments, or review the source code:

1. Open the merge request.
2. Click **Close**.
3. Complete the elements of the Close Merge Request dialog box:
   - To change the review status to **Merged** and close the review, select the **Close as Merged** check box. You may choose the **Close as Merged** option if the review branch was merged through some other means (such as the Git CLI or though the `git cherry-pick` command).
   - If you don't select the **Close as Merged** check box, the merge request is closed without changing the review status to Merged. You may want to do this if the merge request was created on the wrong branch or created by mistake.
4. Click **OK**.

## Reopen a Closed Merge Request

You can reopen a closed merge request as long as there isn't another open merge request for the same branch, as only one merge request can be open per branch. To reopen a closed merge request, open it and click the Reopen button on the right side of the page.

# Cherry-pick a Commit

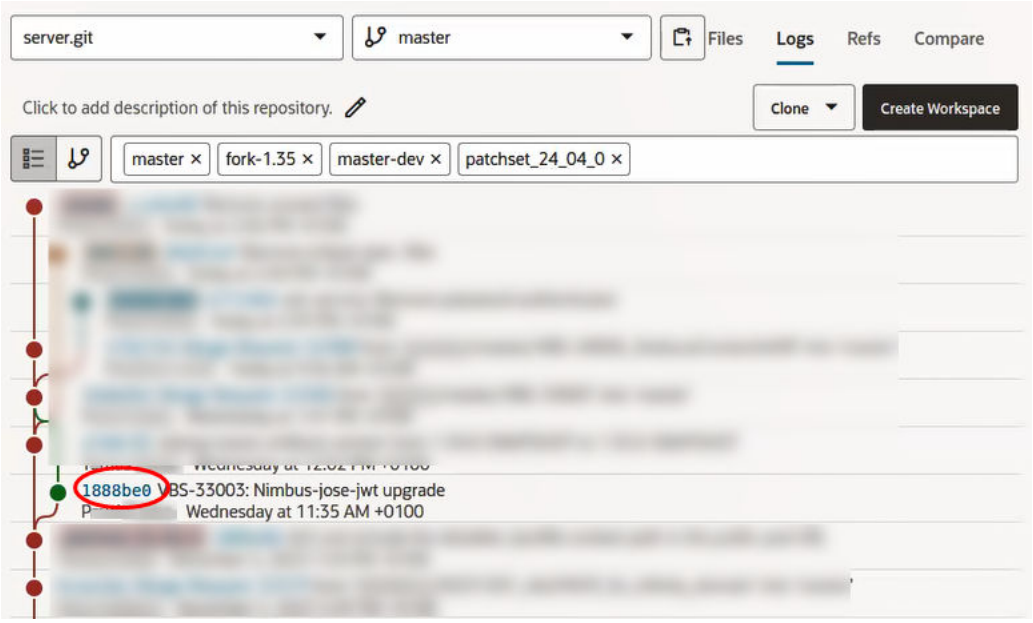
Cherry picking is the act of picking a commit from one branch and applying it to another. It is mainly used when you don't want to merge the whole branch and you just want some of the commits that were made to the branch. This operation can be useful for undoing changes, like when a commit has been accidentally made to the wrong branch. You can switch to the correct branch and cherry-pick the commit to where it should belong.

To cherry-pick a commit in a merged MR:

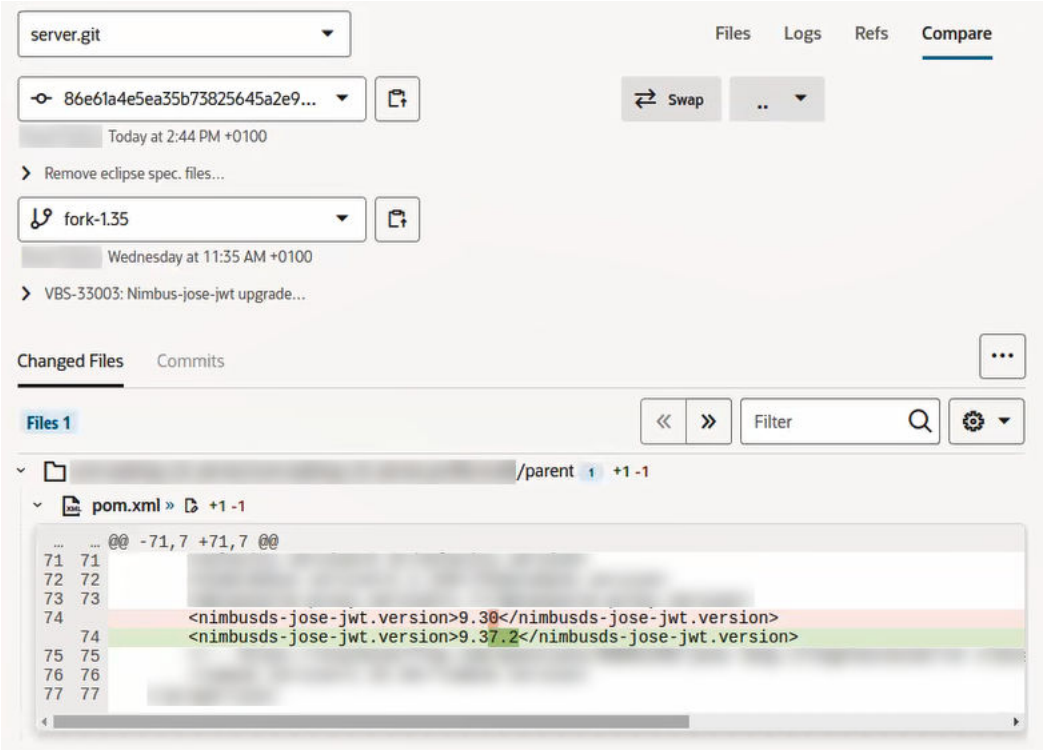
1. From the **Repositories** drop-down list, select the Git repository.
2. On the right side of the page, click the **Logs** tab.
3. Click the **History Graph** toggle button and then enter the branch name in the text box to the right of the toggle button, or select from the drop-down list.

   You can add multiple branches.

4. In the history graph, find the commit you want to pick and click its link.

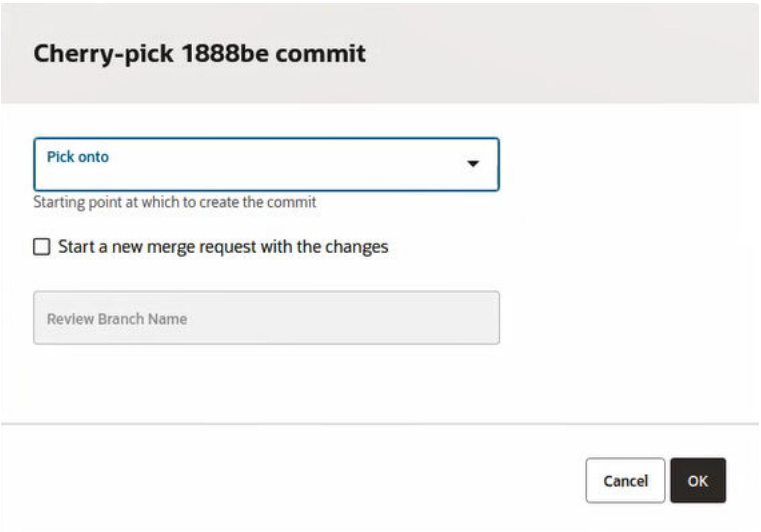



The Compare view is displayed, showing the commit that you selected and the change itself, at the file level.



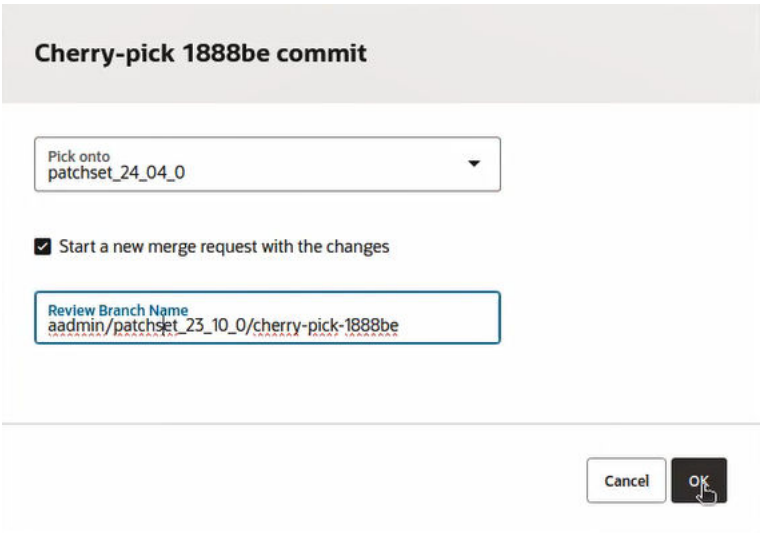

5. Click the **Actions** *** menu and select **Cherry Pick...**.

   The Cherry-pick commit dialog box is displayed.



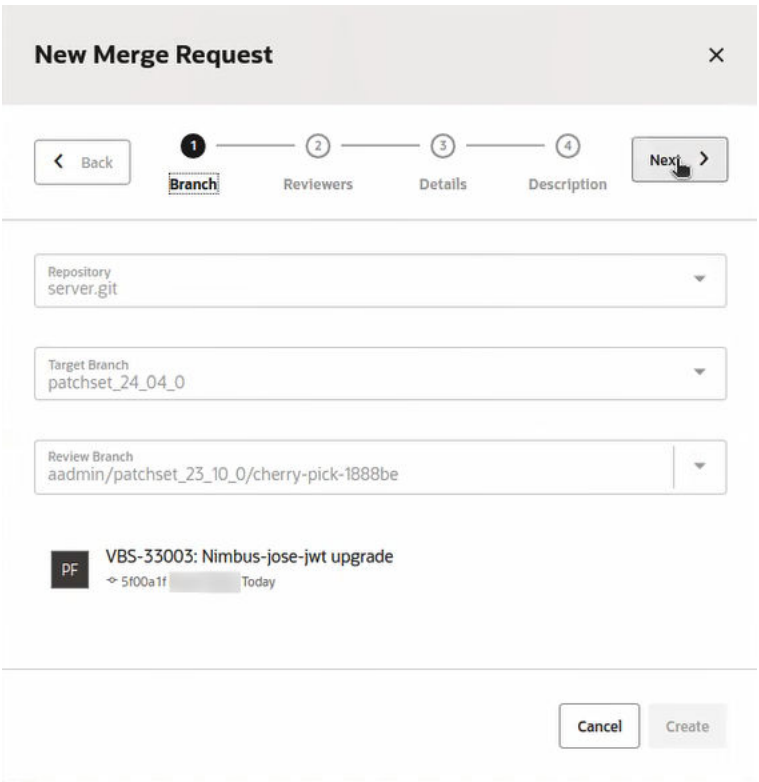

6. In the **Pick onto** field, use the selector to choose a starting point (the target branch) in which the commit will be created.
7. Use this field to check whether the change you want to make is possible or not:
   - If you see the message, "The target branch already contains changes", select a different target branch. The branch you selected already has the change.

- If you see the message, "There are conflicts and thus the commit can't be cherry-picked", you can't apply the commit because the interface doesn't provide a way to resolve the conflict. Click **Cancel**.
- If you see the message, "Target branch requires review",
  a. Click the **Start a new merge request with the changes** checkbox.




The display field right below the checkbox displays the name of the new review branch that is automatically created for you.

  b. Click **OK** and the New Merge Request wizard displays.

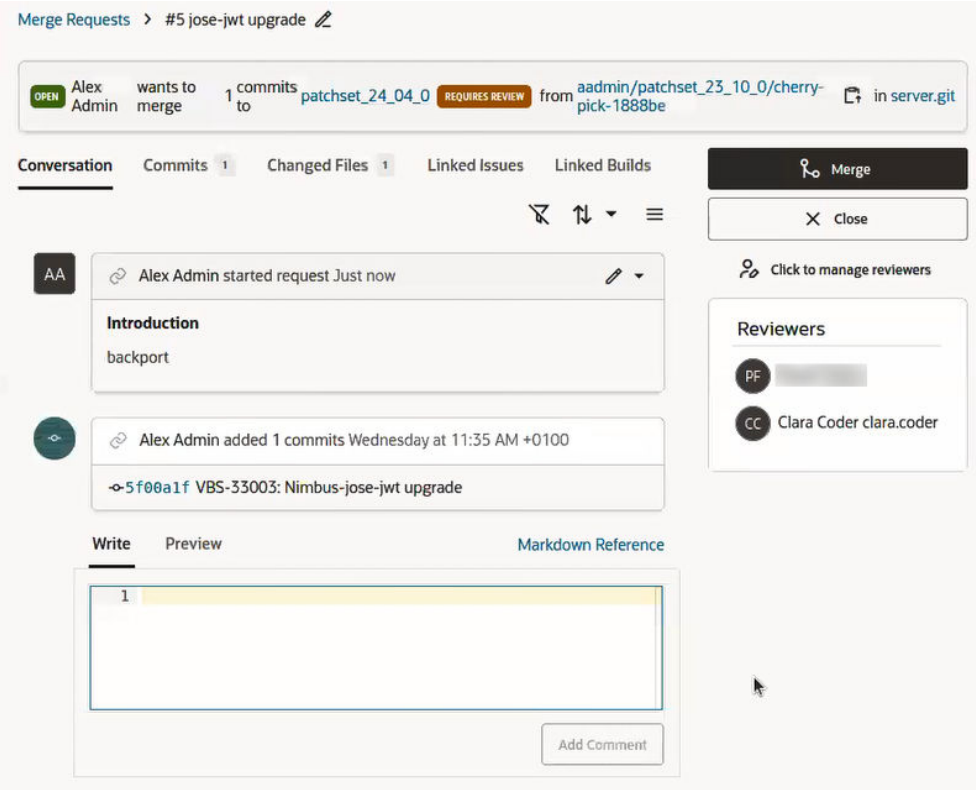


c. The fields on the Branch page are already populated, so you just need to click **Next** to go to the Reviewers page.

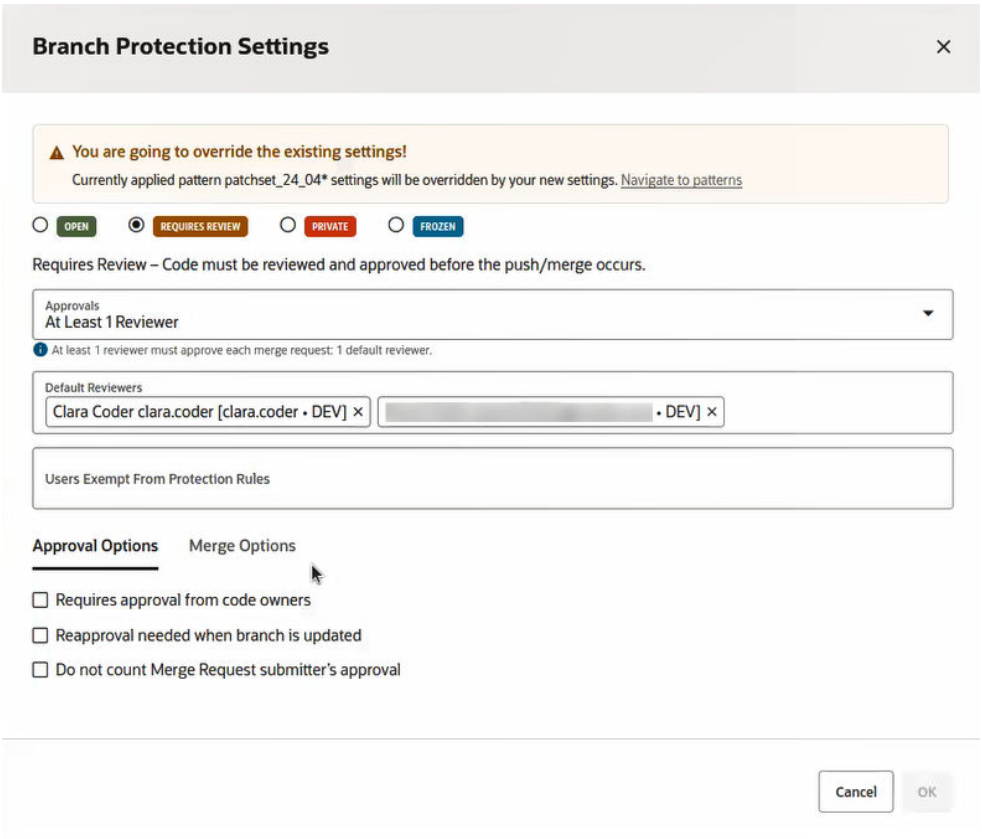
d. Select your reviewers.
Skip to the next substep or, optionally, click **Next** twice, bypassing the Details page and displaying the Description page, where you can enter a summary (and a description).

e. Click **Create** to finish creating the merge request.
See Create a Merge Request.

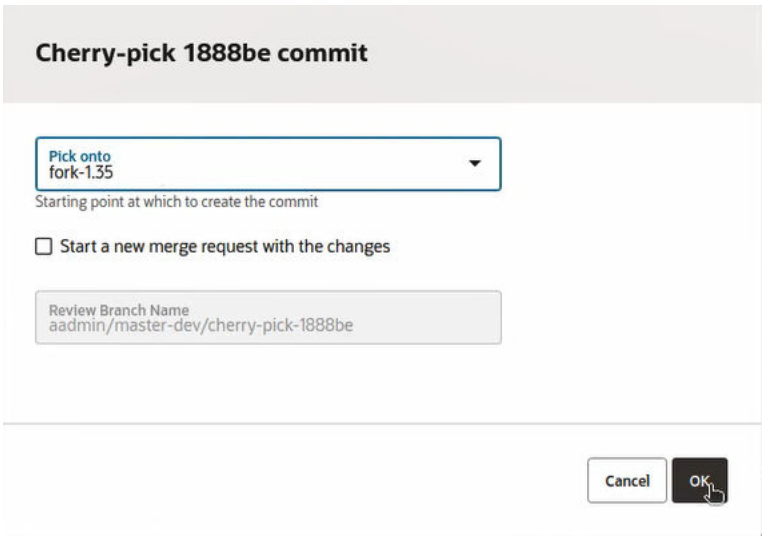
The Merge Request page displays.



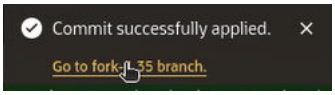

Click **REQUIRES REVIEW** to display the Branch Protections Settings page.




- If the dialog doesn't complain, click **OK** to add the commit to the target branch.




A message, "The commit was successfully applied", is displayed after it succeeds.



Click the link under the message and open the **Compare** view, where you can examine the changed files.

Return to the **Git** page, select the **Logs** view and, in the history graph, examine the results.

**Use the Git Command Line as an Alternative to the VB Studio Git User Interface**

You can use the Git command line instead of using VB Studio's Git user interface to cherry-pick a commit in one branch and add it to another.

The `git cherry-pick` command is relatively straightforward:

```
git cherry-pick <commit-ID>
```

where `<commit-ID>` is a commit reference.

You need to use the `git log` command to find a command reference. There are several options you can use with git log to format and filter the results. For example, you could use the `--graph` option to draws an ASCII graph that represents the commit history's branch structure. This option is often used with the `--oneline` and `--decorate` commands so you can see which commit belongs to which branch. The `--decorate` flag makes the `git log` command display all of the references, such as branches, tags, and so on, that point to each commit.

You can also use the Git command line to create a merge request, when one is needed. See Create a Merge Request from the Command Line.

# Reapply a Merge Request's Commits to a New Branch

VB Studio enables you to reapply commits from a merged or closed merge request to a new merge request and review branch.

Use Reapply Changes to backport your changes from merge requests that were already merged or closed into a new merge request with another target branch. It cherry-picks the original merge request's commits to create a new merge request with the original code changes inside a new review branch.

> **Note:**
>
> - Linked issues from the original merge request are retained when a merge request is reapplied.
> - Comments from the original merge request are not retained.

Reapply Changes requires that the original merge request is closed or merged. If you have open merge requests, you won't see the Reapply Changes button; open merge requests must be closed before you can use the button.

There are instances where you cannot use Reapply Changes, such as when a branch is protected or if the changes you are trying to reapply were already merged into the branch. Files with conflicts can also cause problems and must be manually merged before proceeding.

When you click **Reapply Changes**, you can select **Reapply All Commits** or **Reapply Merged Changes**.

- **Reapply All Commits** – Reapplies all of the commits listed on the merge request **Commits** tab. In other words, it reapplies all of the commits from the review branch.
- **Reapply Merged Changes** – Reapplies the actual result of the merge, rather than the individual commits from the review branch. In other words, it reapplies the commits from the target branch that were created when the merge request was merged.

### Reapply All Commits

Here's how to reapply all commits from a closed or merged merge request:

1. Click **Reapply Changes**, then select **Reapply All Commits**.
2. On the **Reapply All Commits** dialog, select a target branch from the **Target Branch** dropdown. The target branch is where the commits from your new merge request are applied after reviewing and merging your new review branch.
3. The **Review Branch** field is automatically filled out for you when you select the target branch, but you can edit the name of the branch.
   The icon indicates that this branch will be created for you.
4. Review the information in the Reapply All Commits box for a summary of the actions that will take place, as well as information about the commits you are reapplying.
5. Click **Reapply All Commits** to create a new review branch with a merge request to the target branch you selected.
   In the merge request notification that appears, you'll find a link to the original merge request from which this new one was generated.

### Reapply Merged Changes

Here's how to reapply merged changes from a closed or merged merge request:

1. Click **Reapply Changes**, then select **Reapply Merged Changes**.
2. On the **Reapply Merged Changes** dialog, select a target branch from the **Target Branch** dropdown. The target branch is where the commits from your new merge request are applied after reviewing and merging your new review branch.
3. The **Review Branch** field is automatically filled out for you when you select the target branch, but you can edit the name of the branch.
   The icon indicates that this branch will be created for you.
4. Review the information in the Reapply Merged Changes box for a summary of the actions that will take place, as well as information about the merged changes you are reapplying.
5. Click **Reapply Merged Changes** to create a new review branch with a merge request to the target branch you selected.
   In the merge request notification that appears, you'll find a link to the original merge request from which this new one was generated.

> **Note:**
>
> Reapply Merged Changes is disabled for:
>
> - Open or closed merge requests.
> - Merge requests merged from the command line.
> - Merge requests merged before VB Studio version 24.04.0.

# Retarget a Merge Request's Commits to Another Branch

VB Studio enables you to retarget your open merge request to a new target branch, which is useful for open merge requests whose target branches have been moved, deleted, frozen, or locked down.

Retargeting does not affect the Git repository. Instead, you choose a different target branch for the merge request, and the merge-base shifts. The merge request will show different commits and code changes depending on the chosen target branch. Note that because the Git repository is not changed until after merging, you can retarget a merge request multiple times.

Here's how to retarget a merge request to a new branch:

1. Click **Retarget** to display the Retarget dialog.
2. Select a target branch from the **Target Branch** dropdown.
3. The **Are you sure you want to retarget?** information box appears, informing you that some commits may be added or removed from the merge request, depending on your new target branch, and some review comments may become outdated.
4. Click **Retarget**.

   The merge request will display the new target branch.

Sometimes you might not be able to perform this operation, such as for a protected branch. Files with conflicts can also cause problems and must be manually merged before proceeding.

# Revert a Merge Request

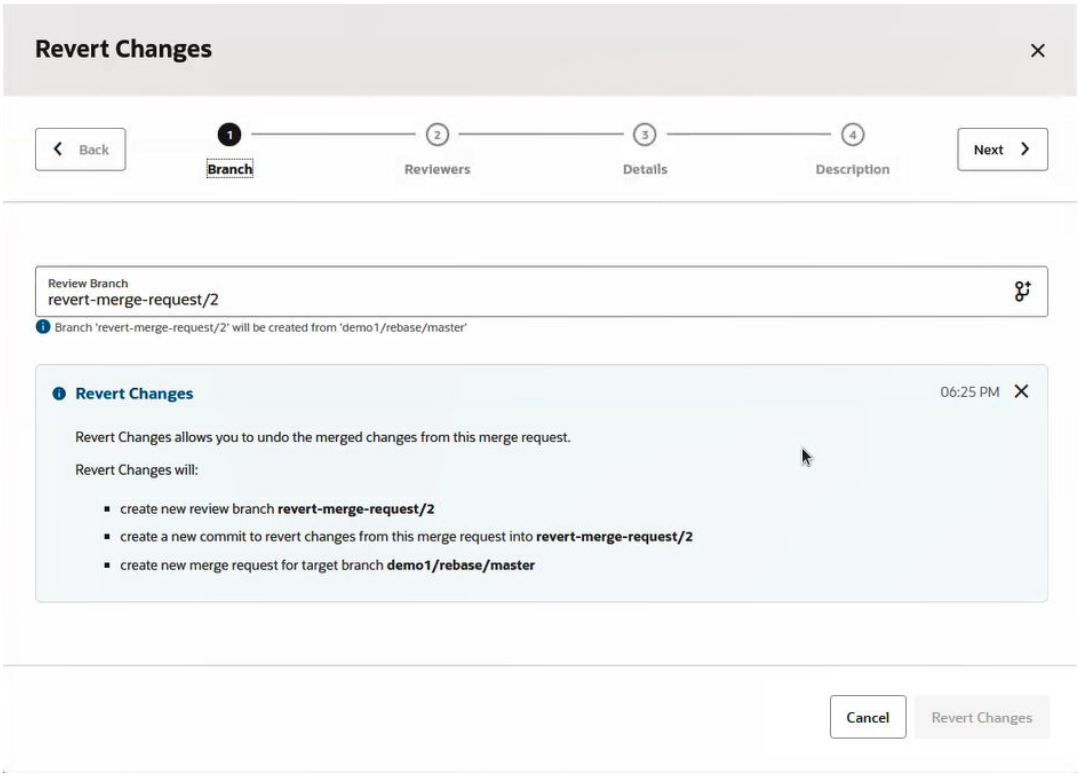
With a single click, you can revert an MR, create a new branch with the revert, and start a new review. The ability to do this is useful when it is necessary to revert a problematic MR. The revert operation undoes a commit in a forward-moving fashion. This means that a new revert commit will be created instead of rewriting the history. All original merged commits will be preserved. The revert commit will have inverse merge changes. The revert operation create a new commit that undoes the changes, effectively reversing the effects of the unwanted merge.

After a merge request has been merged, you can use the **Revert Changes** button. The changes that were merged into the target branch will be reverted. The Revert Changes operation will:

1. Create a new branch forked from the target branch.
2. Create a new commit with inverse changes on top of the newly-created branch.
3. Create a new MR.

Here's how to do it:

1. Click **Revert Changes** to display the Revert Changes dialog.




[revert-changes-dialog-page-1.jpg : desc-revert-changes-dialog-page-1]

2. The **Review Branch** field is automatically filled out for you.

   The icon to the right indicates that this branch will be created for you. Notice the informational message under the review Branch field: `Branch branch-name will be created from target-branch-name` . The target branch is the branch where the review changes are going to be reapplied (and added to the top of) after the review.

3. The Revert Changes information box summarizes the actions that will take place.

   Notice that the **Revert Changes** button at the bottom of the page is grayed out. That means you are required to fill in some information on one of the pages that follow this one.

   Click **Next** to display the second page (Reviewers) of the dialog.

4. The **Reviewers** field is required, so you must select a reviewer from the list.

   Notice that the **Revert Changes** button becomes active after you choose a reviewer. You can click this button now, unless you want to click Next and go to the (optional) **Details** page where you can specify any linked issues, linked builds, or tags or continue to the (optional) **Description** page where you can add information about the revert operation you are about to perform.

5. Click **Revert Changes**.

   A new merge request is created for you, as was explained in the third bullet in the previous page's informational box.

6. Click **Merge** to revert the changes you made in the previous merge.

Sometimes you might not be able to perform this operation, like when files have conflicts. These conflicts must be manually merged before proceeding. If the merge you are trying to revert was done through the user interface (not the Git command line), you should be able to revert the changes.

# Merge Request and Branch Administration

A project owner can assign default reviewers to a branch, or set push and merge restrictions on it. A branch owner can also change a **Private** branch's restrictions. As a project owner, you can set some restrictions on a Git repository branch you can and assign some project users as default reviewers of the branch.

For a branch, you can set rename, delete, push, and merge restrictions. You can also lock a branch if you don't want anyone to push commits to it or merge another branch with it. When a merge request is created with the branch as the target branch, the default reviewers of the branch are automatically added to the Reviewers list.

## Manage Merge Requests

Status icons on the Merge Request page provide information that helps you manage merge requests. Also, you can use the Custom Search page to list merge requests based on custom search criteria, and to close multiple merge requests.

Icons in the Merge Request list provide visual clues about the status of each merge request:

- The Status column indicates Open, Merged, and Closed merge requests.
- Merge requests with issues that require attention – such as those not updated in the past 3 months (Stale), or missing a repository – are flagged with a red exclamation point icon next to the merge request name.
- Repositories that no longer exist are indicated with a strike-through of the repository name in the Repository column.
- Merge requests that have not been updated in the past 3 months are flagged with a Stale icon in the Changed column.

You can use the Custom Search page to close multiple merge requests:

1. In the left navigator, click **Merge Requests** .
2. Click **Custom Searches**, then use the boxes at the top of the Custom Search page to fine-tune your search. To save the search for future use, click the Settings icon, then click **Save As**. On the Save As dialog, enter a name in the **Search Name** box, then click **Save As**.
3. Use the check boxes at the left of the Merge Request column to select one or more merge requests. Next, click the Actions icon (**...**) and select **Close Selected**.
4. On the Close Selected Merge Requests dialog, type in a reason for closing the merge requests. Optionally, select **Resolve all linked issues** to resolve all linked issues for the merge requests. Select the confirmation check box (**I understand that the <n> merge requests will be closed**), then click **Close**.

## Assign Owners to Files in the Project's Git Repository

If you have administrative or project owner permissions, you can define individual users or groups that are responsible for code in a repository. You designate file ownership by creating a new file called `CODEOWNERS` in the root directory of the repository, in the branch where you'd like to add the code owners. Each CODEOWNERS file assigns the code owners for a single branch in the repository. You can assign different code owners for different branches, as needed.

There are several reasons why you would want to use a CODEOWNERS file. This file is meant to be a centralized place to give credit and ownership for chunks of the codebase. Having this

paper trail can be helpful when new employees are hired and brought onto the project, and can even streamline the resolution of questions that come up during the course of development since you know who to ask. If you define file owners, you can alert those owners when the code they are responsible for has been changed or automatically include the file owner in code reviews and/or merge requests.

Code owners must have write permissions for the repository. When the code owner is a group, that group must have write permissions, even if all the individual members of the group already have write permissions directly, through organization membership, or through another group membership.

Repository owners can add branch protection rules to ensure that changed code is reviewed by the owners of the changed files. In other words, when someone enables required reviews, they can also require approval from a code owner before the author can merge a request in the repository. See (Set Review and Merge Restrictions on a Repository Branch). If a file has a code owner, you can see who the code owner is before you open a merge request. In the repository, you can browse to the file and hover over the icon, which will display its owner. You can also click the icon and go to the line in the CODEOWNERS file that assigns ownership for that file.

You can reduce the size of your CODEOWNERS file by using wildcard patterns to consolidate multiple entries into a single entry.

**Syntax:**

A CODEOWNERS file uses a pattern that follows most of the rules used in gitignore files. See Git's gitignore Documentation.

> **Note:**
>
> These syntax rules for gitignore files *do not work* in CODEOWNERS files:
>
> - Escaping a pattern starting with `#` using `\` so it is treated as a pattern, not a comment
> - Using `!` to negate a pattern
> - Using `[ ]` to define a range of characters

The pattern is followed by one or more usernames or group names using the standard `@username` or `@group-name` format.

CODEOWNERS paths are case sensitive, so even case insensitive systems, like macOS, must use paths and files that are cased correctly in the CODEOWNERS file.

If any line in your CODEOWNERS file contains invalid syntax, that line will be skipped.

**Example**

Here's a sample CODEOWNERS file:

```
# Each line is a file pattern followed by one or more owners.

/a/ @alex.admin

*.txt @alex.admin @don.developer
```

```
pom.xml @Group1 @users1 @testers2 @developers1

cc/ clara.coder@example.com

/docs/f[0-9]*.txt @alex.admin
```

In this example:

- alex.admin owns the "a" subfolder and everything in it
- alex.admin and don.developer own all the .txt files in the project's Git repository
- Groups (Group1, users1, testers2, and developers1) can own files, like the `pom.xml` file listed here
- clara.coder@example.com owns all the files under the "cc" directory
- alex.admin owns all files under the docs directory and its subfolders that start with "f", contain some number in the folder name, and have a "txt" file extension

See Example of a CODEOWNERS file for another more detailed example.

## Set Review and Merge Restrictions on a Repository Branch

You must be a project owner to configure a branch so that it allows another branch to merge into it only through a merge request after the merge request reaches the required number of approvals.

The number of approvals ensures that specified reviewers of the merge request have reviewed the changes of the review branch. You can't merge a branch outside VB Studio, such as using a Git client, without meeting the number of approvals requirement of the merge request. You can set other review restrictions on a branch, such as whether the last build of the branch must be successful to merge it.

To set review restrictions on a branch:

1. In the **Project Administration** page, click the **Branch Protection** tile.
2. Click in the **Search repositories** field and select the Git repository.
3. Select the **Branch name** radio button, click in the search field below it, and select the branch.

   If there aren't any rules associated with the branch, proceed to the next step.

   If there are any rules already associated with the branch, those rules will be displayed. You can select a rule to see its protections, which are displayed in the right-hand panel. If the rule has the **Requires Review** option already selected and the other options (default reviewers, approvals, exempt users, etc.) are acceptable, you're all set.
4. In the right-hand panel, select the **Requires Review** option.

When you select the **Requires Review** option for branch, you can merge a branch after the branch's approvals requirement is met.

> **Note:**
>
> Users in Oracle Cloud Application environments that have multiple VB Studio instances in different identity stripes have username strings that include the environment name, where that user has been defined. Since a unique user may have been defined for multiple environments, this format ensures that one identity can be distinguished from that user's other unique identities. This should help you select the correct user to add.

These are the review restrictions you can set from the Branch Protection page:

| Action | How To |
|---|---|
| Set the minimum number of approvals before a branch can be merged to the selected branch | From the **Approvals** drop-down list, select the minimum number of reviewers who must approve the review branch of a merge request, where the selected branch is the target branch. |
| Assign default reviewers to a branch | Default reviewers, who have the power to approve merge requests when a branch requires review, are automatically added to the branch's Reviewers list. If there are two default reviewers, both must approve all merge requests for this branch.<br>To specify default reviewers of the selected branch, click **Default Reviewers** and select the member, group, or code owner. |
| Specify users or groups who can bypass the branch restrictions and merge the review branch of a merge request outside VB Studio or without required approvals | In **Merge Request Exempt Users**, specify the users or groups.<br>This is useful if you want to allow some users or groups to merge the review branch irrespective of review conditions being met. |
| Require code owner approvals | When you create an MR, neither default reviewers nor code owners are automatically added.<br>Under Approval Options, select **Requires approval from code owners** to include file owners as reviewers when any part of the merge request includes files that have been assigned code owners. When selected, code owners must approve merge requests in addition to the number of approvals selected in the **Approvals** dropdown list:<br>• If a file has two or more owners, only one needs to approve the MR.<br>• If one or more files have different code owners, at least one code owner from each owned file must approve the MR.<br>See Assign Owners to Files in the Project's Git Repository for more information about code owners. |
| If a change is pushed to a branch after some reviewers have approved the merge request, merge only when they reapprove the merge request | Under Approval Options, select the **Reapproval needed when branch is updated** check box. |
| Do not count Merge Request submitter's approval | Under Approval Options, select **Do not count Merge Request submitter's approval** to not have it count toward the minimum number of approvals. |

| Action | How To |
|---|---|
| Configure the default value for the project's rebase and merge/ squash/both operations | Under Merge Options, click the **Allowed Merge Options** selection menu in the **Merge Options** tab on the Branch Protection Settings page, and configure the default value for the branch's merge/squash/rebase operations. You can choose some combination of the following values:<br>• **Squash and Merge**<br>• **Rebase and Merge**<br>• **Create a Merge Commit** |
| Allow a review branch to be merged to the selected branch only if the MR includes a linked issue (or multiple issues) | Under Merge Options, select the **Requires linked issue** check box. |
| Allow a review branch to be merged to the selected branch only if the last build of the linked job in Merge Request is successful | Under Merge Options, select the **Require successful build** check box. |
| Ensure changes pushed to the target branch match the contents of the review branch | Under Merge Options, select the **Changes pushed to target branch must match review content** check box. |
| Prevent merging any merge requests with "Needs Attention" comments | Under Merge Options, select the **Allow merge only if there are no comments with Needs Attention status** checkbox. |
| Delete the review branch after merging the Merge Request into the target branch | Under Merge Options, select the **Delete the review branch after Merge Request is merged into target branch** check box to enable deleting the review branch after the MR has been merged into the target branch. When enabled, the **Delete branch** option under Post Merge Actions in the Merge dialog will also be enabled by default. Branch deletion can still be bypassed by selecting **No** in the confirmation dialog. Select **Yes** to delete the branch after the merge. See step 5 in Merge Branches. |

5

# Create and Use Environments

An environment defines your target Oracle Cloud Applications, Oracle Visual Builder, Oracle Integration, and Oracle Cloud Infrastructure service instances as a single entity. You'll define an environment to deploy an application to or to get information from a service instance.

## Define Your Environments

An environment lets you define and manage Oracle Cloud PaaS and Oracle Cloud SaaS service instances as a single entity.

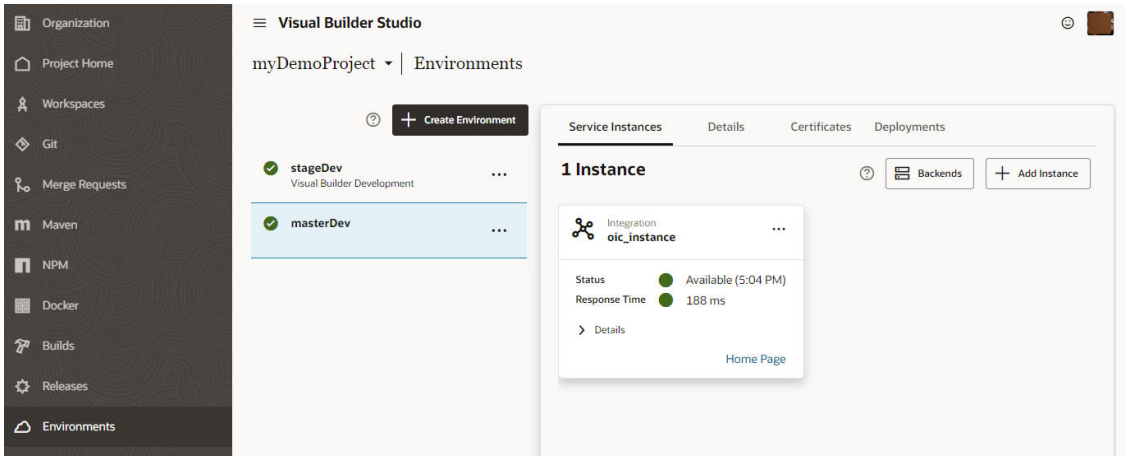
If you're working with Oracle Cloud Application extensions, you'll have a VB Studio Development environment that points to your Oracle Cloud Applications DEV instance. If every DEV and TEST instance in your Oracle Cloud Application environment family receives its own instance of VB Studio, you can create a Test environment for your extension and add your TEST Oracle Cloud Applications instance to it. Typically, your DEV (and TEST) instances are in the same identity domain as your VB Studio instance, while your PROD instance is almost always in a different identity domain. So you might want to create a separate production environment and add your PROD instance to it. Note that you can only add one Oracle Cloud Applications instance to an environment. See Add the Oracle Cloud Application's Production Instance to an Environment in *Administering Visual Builder Studio* for more information.

If you're working with visual applications, you'll have a VB Studio Development environment that points to your Visual Builder development instance (this environment is created automatically if the Visual Application template was used to create the project). You can create additional environments for visual apps and add the production instance of your current identity domain, or an external Visual Builder instance from another identity domain. Note that you can add only one Visual Builder instance to an environment. See Set Up Additional Deployment Instances in *Administering Visual Builder Studio* for more information.

If you plan to share code between different Oracle Integration instances, you'll need to add Integration instances to your environment to promote changes from a development environment to a test, and finally to a production environment.

It's worth noting that Visual Builder and Integration instances that become available to add to an environment are those connected to an OCI account. These instances must be authorized for use with your VB Studio instance. Without this authorization, you won't see any Visual Builder or Integration instances in the Environments page. See Authorize VB Studio to Access Visual Builder and Integration Instances.

You can access and manage the project’s environments from the Environments page:




From the Environments page, you can:

- Create and delete environments
- Add or remove service instances from existing environments
- Update the details of the environment
  The **Details** tab displays details, such as name and description, for the selected environment. You can also instantly see the health of all service instances comprising each environment right on the Project Home page or on the Environments page.
- View the details of an environment's service instances
  The **Service Instance** tab captures information, such as the health of and response times for service instances and their URLs, for each environment in a single place, so you won't have to hunt for it later.
- Upload and remove an environment's trust certificates
  The **Certificates** tab shows trust certificates that you upload to connect with external services in an environment, allowing all applications that use that environment to communicate with those services.
- View deployments
  The **Deployments** tab shows deployments for visual apps, for the current project only. To view deployments for application extensions, use the **Application Extensions** toggle, then click the Extension Lifecycle link.
- Manage the lifecycle of deployed extensions
  Click the **Extension Lifecycle** button to centrally manage application extensions deployed across environments (and across projects). On the Manage Extension Lifecycle page, you can view deployed extensions, deploy a published extension to a new instance, delete extensions when no longer needed, and even add new environments.

  Note that you won't see the **Extension Lifecycle** button if you haven't yet added an Oracle Cloud Applications instance to your environment.

## Authorize VB Studio to Access Visual Builder and Integration Instances

Your VB Studio instance must be authorized to access Oracle Visual Builder and Oracle Integration instances that are connected to an OCI account. This authorization allows you to

view Visual Builder and Integration instances linked to the OCI account and create environments with these instances.

Authorizing your VB Studio instance requires access to the OCI console. If you're the OCI account subscriber, you already have the required roles to do this. If you're not, talk to an administrator to authorize the VB Studio instance. See Authorize VB Studio to Access Visual Builder and Integration Instances for the authorization steps.

## Set Up an Environment

You can create an environment and add one or more service instances to it. For example, you can add an Oracle Visual Builder instance as well as an Oracle Cloud Applications instance to an environment.

You can also create multiple environments, each containing one or more service instances. For example, you can add a Visual Builder instance to a Development environment used for developing and testing your visual applications, then create another environment with a production instance of Visual Builder when your visual app is ready to go live.

1. In the left navigator, click **Environments** .
2. Click **Create** (or **+ Create Environment** if the page is empty).
3. In **Environment Name**, enter a unique name and, in **Description**, enter a description.
4. Click **Create**.
5. In the **Service Instances** tab, click **+ Add Instance**.
6. In the Add Service Instance dialog, select the type of service instance: **Visual Builder and Oracle Integration**, **Oracle Cloud Applications**, or **Infrastructure Services**. Depending on the selected instance type, you'll see a list of service instances available to add to the environment:
   - The Visual Builder and Oracle Integration instance list shows resources in a selected compartment that are connected to an OCI account and authorized for use by VB Studio. Additionally, these instances can be from the same identity domain as your VB Studio instance or a different one. If you don't see any instances even after selecting the correct region and compartment, make sure VB Studio has the authorization required to access these instances. See Authorize VB Studio to Access Visual Builder and Integration Instances.

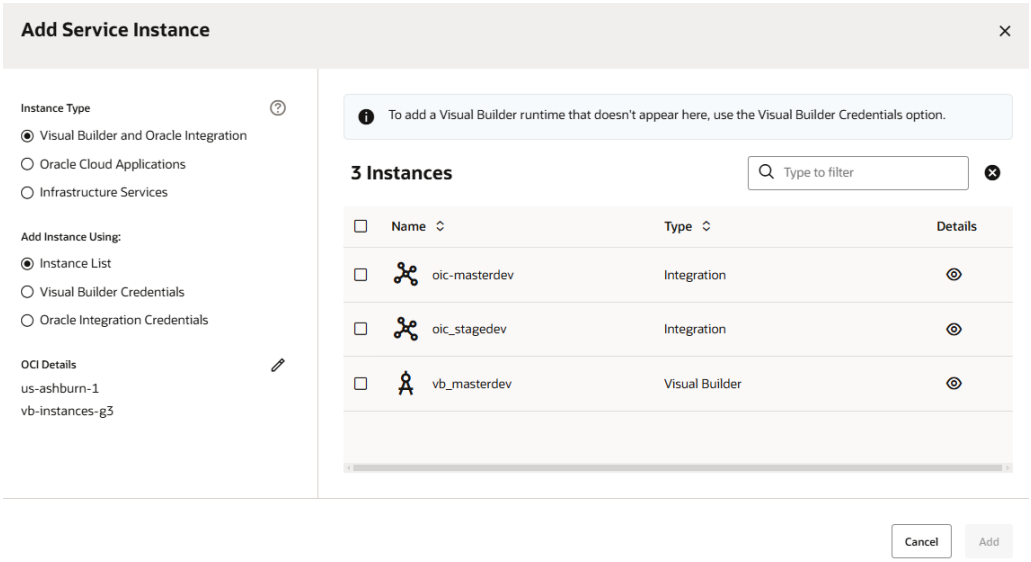
     > **Note:**
     >
     > If you have existing environments with IDCS resources, you can continue to use these environments, but if you're creating a new environment, only the Visual Builder instances that are linked to your OCI account will be available to add to the environment.

   - The Oracle Cloud Applications list only includes IDCS resources, labeled as such.
   - The Infrastructure Services list only includes legacy PaaS instances, or instances that are not Visual Builder, Integration, or Oracle Cloud Applications.

   Here's how you add a new instance to an environment:

   - To add a Visual Builder instance connected to an OCI account, select **Visual Builder and Oracle Integration** under Instance Type and **Instance List** under Add Instance Using, click  next to OCI Details and choose the region and compartment where

your Visual Builder is, then select an instance from the list. (If your environment already contains a Visual Builder instance, you'll need to remove it before adding a different one.)




To add an instance that's doesn't show in the Instance list (because it's connected to a different OCI account), select **Visual Builder and Oracle Integration** under Instance Type and **Visual Builder Credentials** under Add Instance Using. You'll need the instance's base URL, instance name, and user credentials to add an instance this way. (Note though that using this approach won't allow you to set up OAuth for this environment in the deployment build job, used by the Publish action in the Designer.)

Typically, the Visual Builder instance added to your visual application's environment uses the same identity domain as your VB Studio instance. If you choose a Visual Builder instance from a different identity domain as your deployment environment, you may see a warning about setting up the Allowed Origins configuration. Talk to your administrator to make sure your instance's domain is added to its list of allowed origins, as described in Allow Other Domains Access to Services.

When your Visual Builder instance uses a different identity domain than your VB Studio instance, OAuth tokens (via three-legged OAuth flows) are used to secure programmatic access to the target instance. Before any OAuth tokens can be created, you must provide authorization to acquire OAuth tokens. Click **Authorize** when prompted, then sign in with credentials to connect and deploy to the target instance.

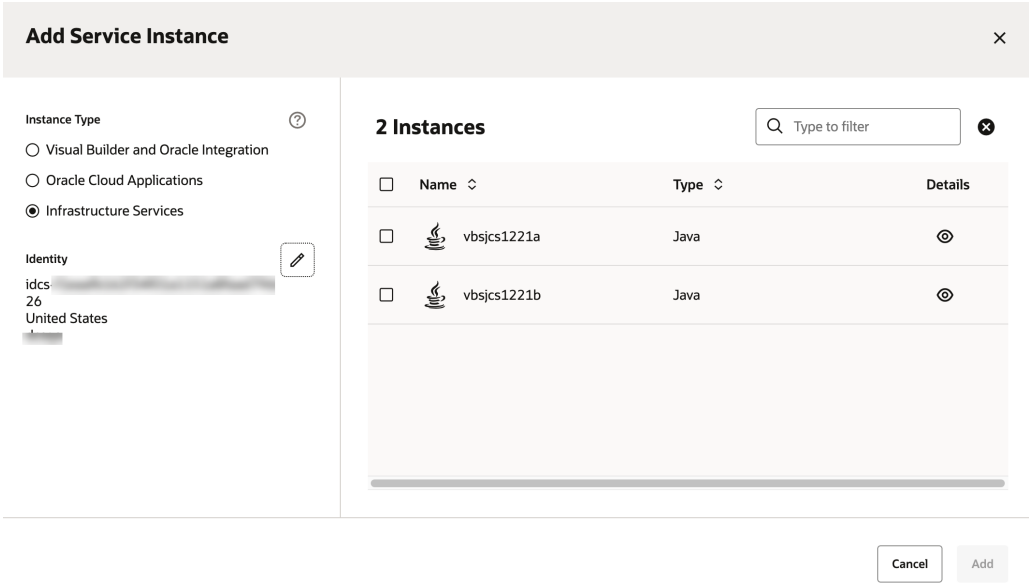
> **Note:**
>
> If you plan to use VB Studio's Oracle Integration build steps (see Move Oracle Integration Projects, Artifacts, Packages, and Lookups Between Instances), you can add Oracle Integration instances to your Environment definition. Follow the instructions in Add an Oracle Integration Instance to an Environment for adding an Integration instance to an environment. You can add a Visual Builder as well as an Integration instance to an environment.

- To add an Oracle Cloud Applications instance in the same identity domain to an environment, select **Oracle Cloud Applications** under Instance Type and **Identity**

**Domain** under Add Instance Using, then select an instance. (You can't change the identity domain for Oracle Cloud Applications.)

If you're viewing an environment that already contains services from another stripe, you'll see a message that the service instance is in a different identity domain and no details about it will be displayed. This is expected behavior. To add an Oracle Cloud Application from a different identity domain, select **Oracle Cloud Applications** under Instance Type and **Application Credentials** under Add Instance Using. You'll need the base Oracle Cloud Application's URL and the credentials of a user who can access the instance. See Add the Oracle Cloud Application's Production Instance to an Environment.

- To add an Infrastructure Services instance—such as Java Cloud Service (JCS)—to an environment, you'll need the instance's region, identity domain ID, and the credentials of a user who can access the instance to display a list of the resources you can add to the environment:




> **Note:**
>
> With Oracle Java Cloud Service reaching End of Life (EOL) in May 2025, support for JCS deployments in VB Studio is now deprecated. See Deprecated Features in *What's New in Visual Builder Studio?*.

7. If necessary, repeat steps 5 and 6 to find and add additional service instances.

The Recent Activities feed on the **Project Home** page displays messages when environments are created or deleted, and when new service instances are added to or removed from environments.

# Add an Oracle Integration Instance to an Environment

Before you can use Oracle Integration build steps in a build job, you need to add Oracle Integration instances to your VB Studio Environment definition.

See Move Oracle Integration Projects, Artifacts, Packages, and Lookups Between Instances to learn about using Integration build steps to move integrations, packages, and lookups between instances.

To add an Integration instance to an environment:

1. In the left navigator, click **Environments**.
2. Select an existing environment to add an Integration instance to, or create a new one.
3. In the **Service Instances** tab, click **+ Add Instance**.
4. In the **Add Service Instance** dialog, locate and add your Integration instance.

   Most of the time, you'll simply select an instance shown in the Instance List. The instance list shows resources in a selected compartment that are connected to an OCI account and authorized for use by VB Studio. Additionally, they can be from the same identity domain as your VB Studio instance or a different one.

   If you don't see any instances even after selecting the correct region and compartment, make sure VB Studio has the authorization required to access these instances. See Authorize VB Studio to Access Visual Builder and Integration Instances.

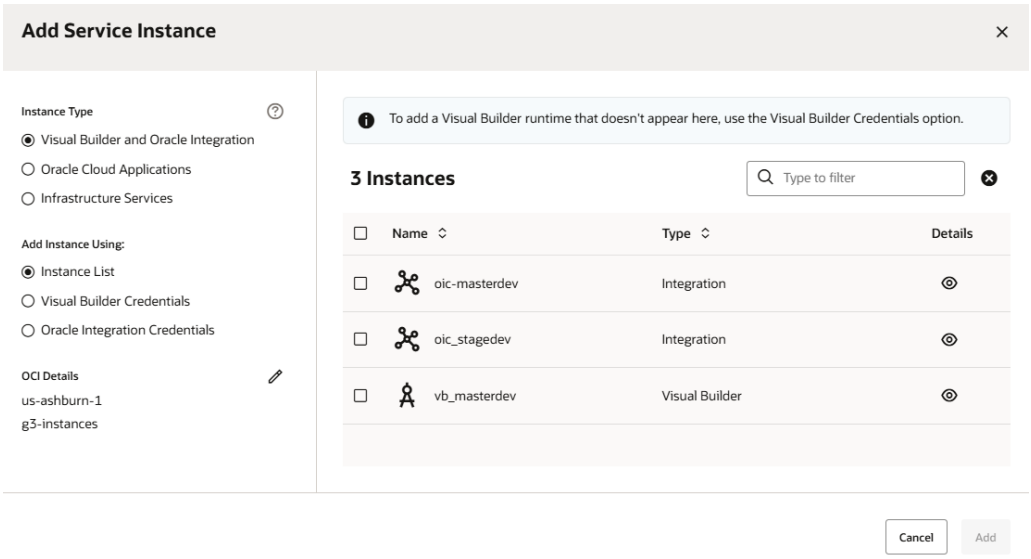
   > **Note:**
   >
   > If you have existing environments with IDCS resources, you can continue to use these environments, but if you're creating a new environment, only the Integration instances that are linked to your OCI account will be available to add to the environment.

   To select an Integration instance connected to an OCI account:

   a. Make sure Instance Type is set to **Visual Builder and Oracle Integration** and Add Instance Using is set to **Instance List**. This is the preferred approach.

   b. Click next to **OCI Details**, select the region and compartment where your Integration instance is, and click **Update**.

   c. Select your Integration instance from the instance list.



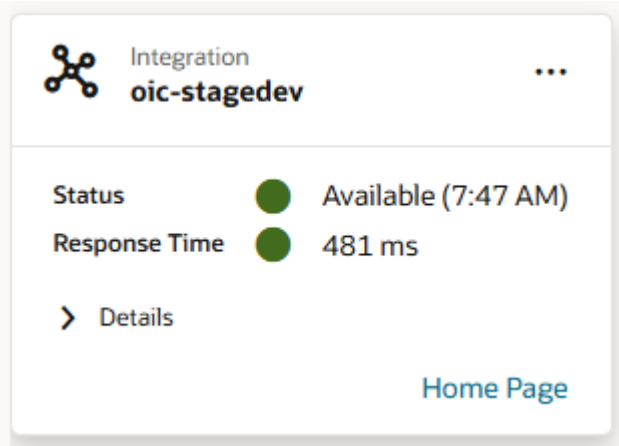

If you don't see the instance you want to add (because it's connected to a different OCI account), select **Oracle Integration Credentials** under Add Instance Using. Note that you cannot use this option to add an Oracle Integration 3 instance.

In **Base URL**, enter the Integration instance's URL; in **Instance Name**, enter the display name that will be used for the service instance within the environment; then, enter the credentials of a user who can access the Integration instance in **Username** and **Password**.

5. Click **Add**.

   When your Integration instance uses a different identity domain than your VB Studio instance, OAuth tokens (via three-legged OAuth flows) are used to secure programmatic access to the target instance. Before any OAuth tokens can be created, you must provide authorization to acquire OAuth tokens. Click **Authorize** when prompted, then sign in with credentials to connect and deploy to the instance.

The **Service Instances** tab now displays the Integration instance you just added:




If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), click **Actions** ••• and select **Renew OAuth Access** to renew the token.

In addition, the Recent Activities feed on the **Project Home** page lists the Integration instance and the environment you added it to.

## Manage Your Instance in Another Identity Domain Using OAuth

When the Visual Builder, Integration, or Oracle Cloud Applications instance in your environment is not in the same identity domain as your VB Studio instance, you can use OAuth tokens to securely access the instance.

1. OAuth tokens for your instance can be set up while adding the instance to your environment, or when setting up builds.

   Follow the steps to set up your OAuth configuration for **Visual Builder** instances:

   - To create an environment with a Visual Builder instance, see Create and Set Up a Project for Development.
   - To configure the deployment build job used by the Publish action in the Designer, see Configure the Deployment Job. OAuth is also supported in other build jobs used to import and export data, lock, unlock, roll back, or undeploy a visual application.

   Follow the steps to set up your OAuth configuration for **Oracle Cloud Application** instances:

   - To create an environment with an Oracle Cloud Applications instance, see Add an Oracle Cloud Applications instance to your environment.
   - To configure the deployment build job used by the Publish action in the Designer, see Create a Deployment Build Job. OAuth is also supported in build jobs used to delete an extension.

   Follow the steps to set up your OAuth configuration for **Oracle Integration** instances:

   - To create an environment with an Oracle Integration instance, see Add an Oracle Integration Instance to an Environment.
   - To configure build jobs used to import and export Integration artifacts from and to an Oracle Integration instance (as well as to activate or delete an Integration), see Manage Integrations . OAuth is also supported in build jobs used to manage Integration projects, packages, lookups, and connections.

   Before any OAuth tokens can be created, authorization must be provided to handle OAuth requests for your environment's instance. VB Studio can automatically detect the instance being added from a different identity domain and will prompt you for authorization. Click **Authorize**, when prompted, then sign in with credentials that can connect and deploy to the target instance.

   > **Note:**
   >
   > It is recommended that you authorize your OAuth connection during initial configuration. If you skip this step, developers won't be able to publish changes from the Designer until the required authorization is provided.

2. OAuth tokens (access and refresh) are typically cycled during regular use. If a token expires during extended periods of inactivity (say, when you're away on vacation), you'll need to renew the token:

   - To renew OAuth tokens on the Environments page, click **Actions** *** and select **Renew OAuth Access**.

- To renew OAuth tokens on the Builds page, locate your build job, then click **Renew Authorization** on the Steps tab. You can also run the job manually, so you'll be prompted to authorize any expired OAuth tokens.

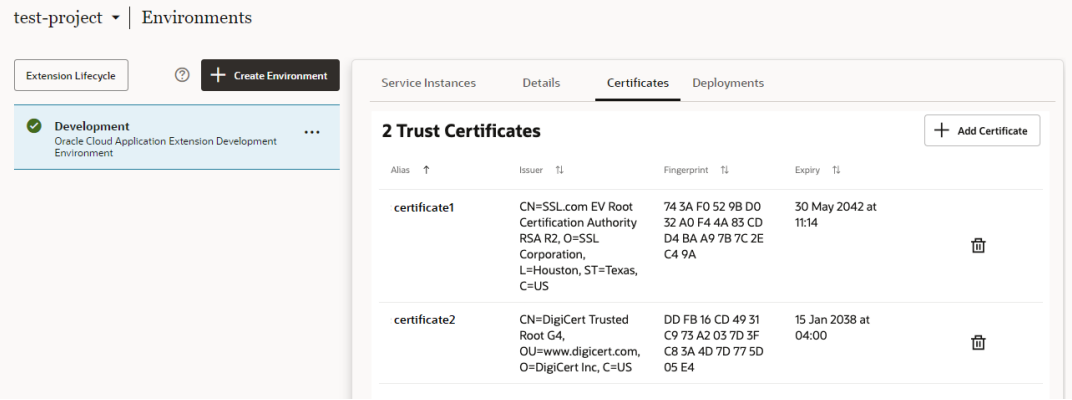
## Manage Trust Certificates in an Environment Definition

Visual Builder Studio uses trust certificates to connect with external services. If a service/endpoint needs a special certificate, you can get it from the service you're trying to connect with, and then upload it from the **Certificates** tab on the Environments page. After you do that, any time you use that environment, you'll get all the certificates that were added to it.

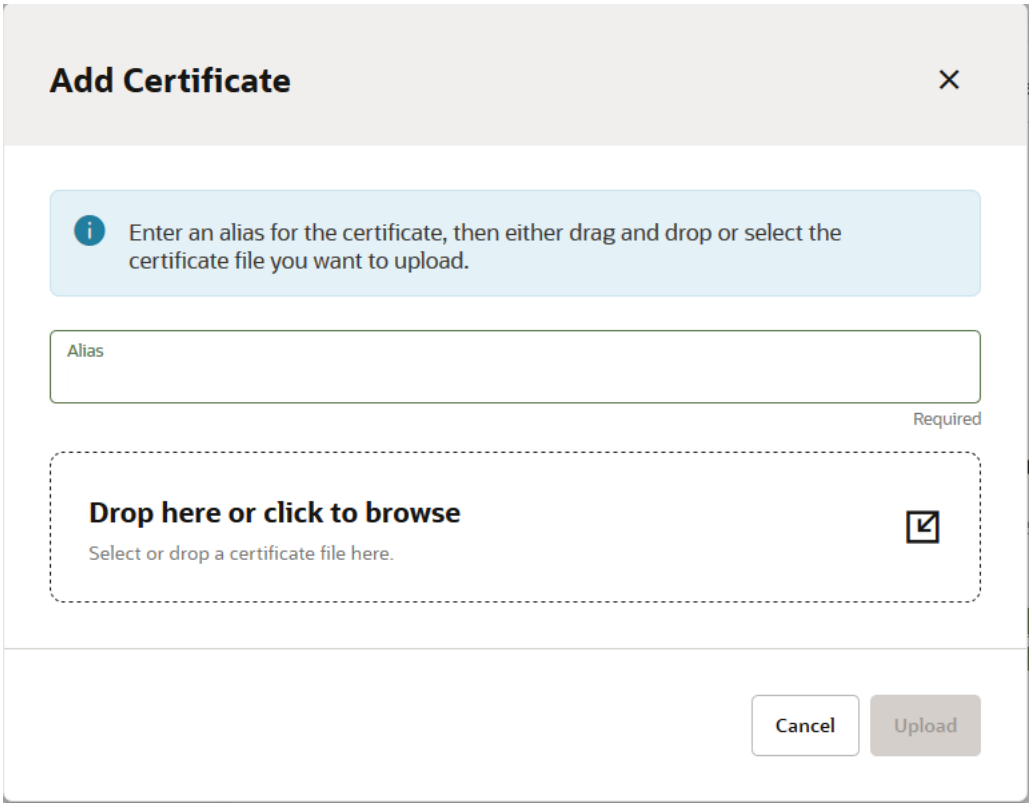
You can use the Certificates tab to upload and remove certificate files for services. Uploading a service's certificate file to the truststore will allow all applications that use that environment to communicate with that service. The Certificates tab displays a list of certificates that have been added. You can click **Delete** in a row to remove the certificate.

To upload a certificate:

1. In the left navigator, click **Environments** .
2. Click the **Certificates** tab.
   The Certificates page displays a list of the trust certificates that have already been uploaded for the environment, similar to this:




3. Click **+ Add Certificate** to open the **Add Certificate** dialog.
   You use the **Add Certificate** dialog shown here to create an alias for the certificate and upload the service's certificate file from your local system:

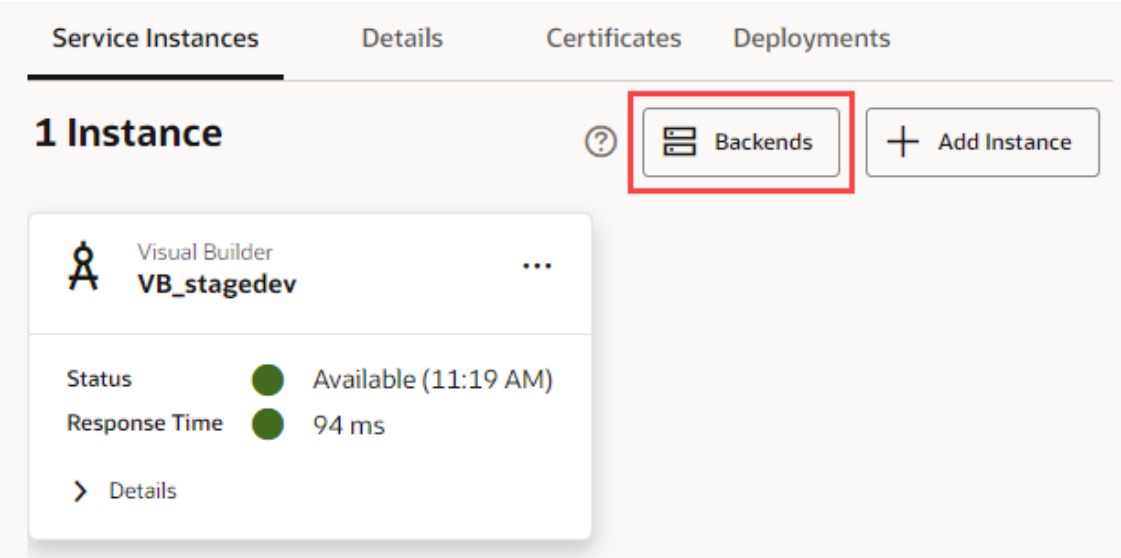



4. Type the alias in the **Alias** field.
   The alias is used to identify the certificate in the table on the Certificates page.
5. Drag the certificate file from your local system to the upload target area, or click the upload target area to browse your local system to locate, select, and add the file.
6. Click **Upload** to add the certificate to the environment's truststore.

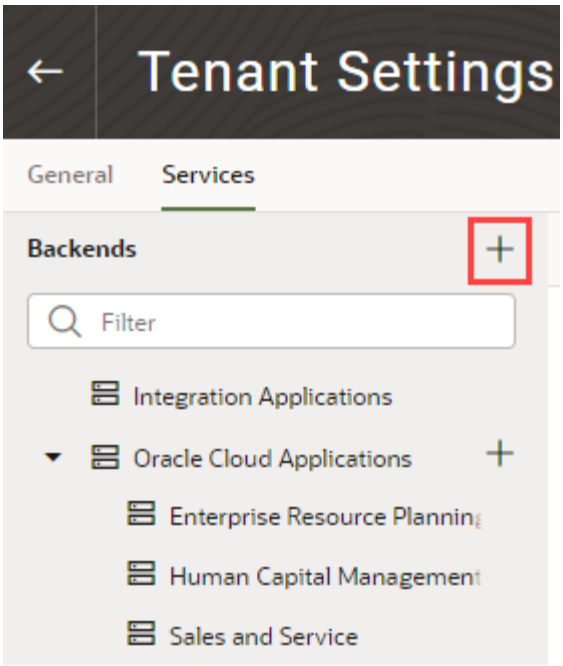
## Create and Edit VB Studio Custom Backends for Visual Applications

*Backends* define the servers that your visual applications and extensions can access, allowing you to keep information about commonly used servers in one place. In the context of visual applications, the VB Studio catalog of predefined services can include several backends, such as Oracle Cloud Applications and Integration Applications, depending on the tenant-level settings of your environment's Visual Builder instance. If these backends don't meet your needs, you can use the Environments page to access the environment's catalog and create a custom backend.

After you create an environment and add a Visual Builder instance to it, click **Backends** on the Environments page's **Service Instances** tab to load the *catalog editor* for that specific environment:




An environment's catalog editor performs essentially the same function as the Tenant Settings->Services tab does for a Visual Builder instance:




For example, using the Services tab, suppose you've configured a backend called `ORDS` on the VB instance where you plan to deploy your visual applications. This backend immediately becomes accessible to all visual applications deployed to or created on this instance, *without* having to define it explicitly within the application itself. A VB Studio environment's catalog editor allows you to achieve the exact same thing. That is, after you've defined the `ORDS` backend for a given environment, it automatically becomes available to all visual application workspaces that are based on that environment.

> **Note:**
>
> Backends that you add using VB Studio's catalog editor are **not** propagated to the Visual Builder instance named in the environment. You'll need to use the Tenant Settings Services tab for each target Visual Builder instance to configure the backends explicitly.

You can use an environment's catalog editor to modify existing custom backends or create a new one to access services that aren't predefined for you in the default catalog. For each backend, you can use these tabs to view and edit the backend's details:

| Tab | Description |
|---|---|
| Overview | Displays the name and type of the backend.<br>Use the topmost **+** button to create a new backend or use the lower **+** button to create a child backend. |
| Servers | Displays the servers associated with the backend and includes the instance URL associated with the instance. From here you can add, edit, or remove backend servers.<br>You can have only one server for a backend in the environment's catalog editor. (This mirrors the same restriction for servers and backends in the Visual Builder's runtime instance's tenant settings: you can have only one server per backend.)<br>For each server, you can define:<br>• Headers<br>• Security and connection details |
| Headers | Displays the static headers defined for the backend at the server level. You can add and edit headers in the tab. |
| Source | Displays the description of the backend stored in the environment-level `catalog.json` file. If you override the environment-level definition, this file shows the contents of the application-level catalog from the `services/catalog.json` file. You can edit the entries in the Source tab, if you want. |

> **Note:**
>
> If your visual applications consume Integration Applications or Oracle Cloud Applications backends, you don't need to configure or edit them using the environment's catalog editor. These particular backends are autocreated for you when you select them from Add Service Instance, which you access by clicking **+ Add Instance** on the Environments page.

See What Are Backends? to learn more about backends.

To learn more about adding an Oracle Cloud Application instance to a visual application, or creating a backend, custom backend, or child backend, see Manage Backends in Your Visual Application.

To learn more about selecting authentication and connection types for backends, see Configure Authentication and Connection Types for Service Connections and Backends.

# Manage an Environment

After creating an environment and adding instances to it, you can manage its instances, as shown here:

| Action | How To |
|---|---|
| Edit an environment's name and description | In the **Details** tab, click **Edit**. Edit the details and click **Save**. |
| Refresh a service instance's status | In the **Service Instances** tab, mouse-over the instance, click **Actions** ***, and select **Refresh Status** to refresh and check the current status of the service. |
| Check a service instance's response time | The **Response Time** provides information about the instance:<br>• A green dot indicates that the response time is less than 5 seconds.<br>• A yellow dot indicates that the response time was between 5 and 30 seconds.<br>• A red dot indicates that the response time was greater than 30 seconds or that the connection timed out. |
| Edit a service instance's credentials | When your instance is set up for Basic authentication, mouse-over the Oracle Cloud Applications, Visual Builder, or Integration instance in the **Service Instances** tab, click **Actions** ***, and select **Edit Credentials**. The Edit Credentials dialog is displayed. Edit the **Username** and **Password** fields to change the credentials you use to log in to the application, then click the **Update** button. |
| Renew a service instance's OAuth access | When your instance is set up for OAuth authentication and the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), mouse-over the Oracle Cloud Applications, Visual Builder, or Integration instance in the **Service Instances** tab, click **Actions** ***, and select **Renew OAuth Access**. The Renew OAuth Access dialog is displayed. Click **Renew** to renew the token using the credentials to access the instance. |
| Remove an instance from an environment | In the **Service Instances** tab, mouse-over the instance, click **Actions** ***, and select **Remove**.<br>An instance can't be removed if it's associated with any workspace. If you try, an error message displays, listing the impacted users and the number of workspaces they have. You'll be instructed to ask those users to delete their workspaces. Once this has been done, you can go back and retry the Remove operation.<br>**Note:**<br>An IDCS resource cannot be removed from an environment if the instance is in another identity stripe. Nobody can remove an IDCS resource from a different identity stripe from an environment, but the project owner can still delete the entire environment. In addition, an IDCS resource from a second identity stripe cannot be added into the same environment.<br>This action is recorded in the Recent Activities feed on the **Project Home** page. |

| Action | How To |
|---|---|
| Delete an environment | In the environments list, mouse-over the environment, click **Actions** ***, and select **Delete**.<br>Remember that the environment's instances won't be deleted.<br>An environment can't be deleted if it's associated with any workspaces. If you try, you'll see an error message listing the impacted users and the number of workspaces they have. You'll be instructed to ask those users to delete their workspaces. After they do that, you can go back and try to delete the environment again.<br>**Note:**<br>Project owners are the only project team members who can delete environments that were created in a different identity stripe. If a project member with the Developer role attempts this operation, they'll see an error message informing them that they can't delete an environment that has resources from another identity stripe.<br>This action is recorded in the Recent Activities feed on the **Project Home** page. |

# Use Service Instance Statuses to Troubleshoot Problems

You can use the service instance statuses and troubleshooting/support information in this section to understand and correct problems indicated by error statuses. Understanding what these error conditions result from can shed light on what could be causing the problems and point to what you need to do to fix them.

Here are the current Service Instance statuses and some of their causes:

| Status | Description |
|---|---|
| **Unauthorized** | Usually occurs when the user can't be validated with the target instance:<br>• For an IDCS resource, ensure that the logged-in user is a valid user in the target instance.<br>• For an application endpoint connection, ensure that the user and credentials specified are valid.<br>• For an Oracle Cloud Applications target instance, ensure that the VB Studio instance has been allowlisted (formerly called whitelisted).<br>If all the causes have been ruled out, the problem may be due to an infrastructure issue. Contact Oracle Support. |
| **No OAuth** | The OAuth flow triggered to securely access the target instance was not completed. Click **Actions** *** and select **Enable OAuth Access**. |
| **OAuth Expired** | The OAuth token created to authenticate access to the target instance has expired.<br>Click **Actions** *** and select **Renew OAuth Access**. |
| **Timeout** | The target instance didn't return a response in an acceptable length of time. The request to return status may eventually complete, at which time the status will change, but this status indicates a problem with the target instance's health and should be investigated if it persists. |

| Status | Description |
|---|---|
| **Available** | The target instance is available and responding to API requests. |
| **Not Available** | The target instance reported an Unavailable status (HTTP 503). |
| **Error** | An unexpected error occurred while contacting the target instance for its status. If this condition persists, contact Oracle Support. |
| **Not Found** | An HTTP 404 error was returned when contacting the target instance:<br>• For an IDCS resource, ensure that the instance is up and running and there is a network route between the VB Studio and target instances.<br>• For an application endpoint connection, ensure that the URL is specified correctly and points to a Visual Builder, Integration, or Oracle Cloud Applications instance.<br>Could also be caused by proxy timeouts, network connection issues, or load balancer problems. |
| **Deactivated** | IDCS resources were deactivated.<br>Returned for an IDCS resource only, indicating that the resource has been deactivated in IDCS, preventing access to that resource. Contact your IDCS administrator to reactivate it or remove the IDCS resource from the environment. |
| **Unknown** | This is the default status before the target instance is contacted and verified. It should change to one of the other statuses listed in this table.<br>If the status persists in this state, contact Oracle Support. |

6

# Use Workspaces and the Designer

Create or edit extensions and visual applications in a workspace. (The changes you make in the Designer are stored as source code in the Git repository associated with your workspace.)

## What Is the Designer?

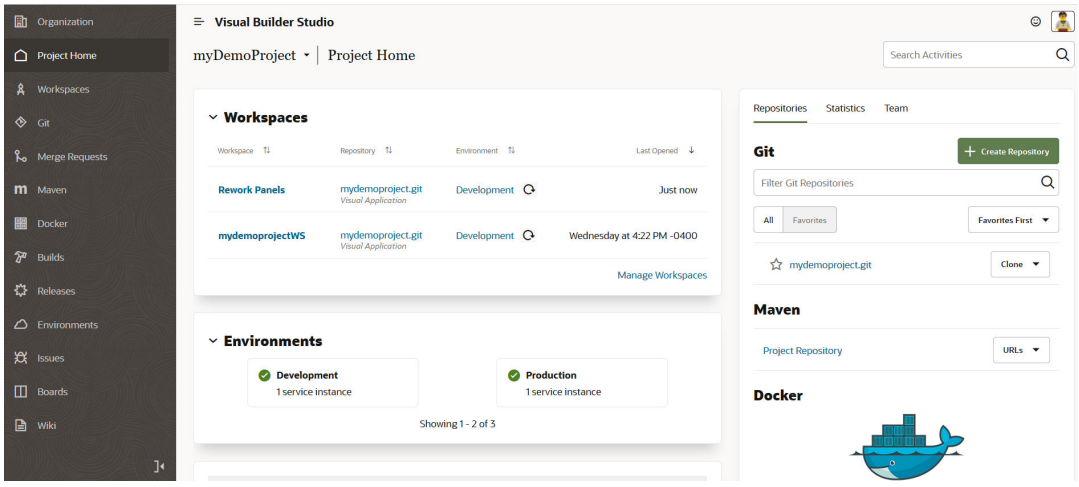
VB Studio includes the **Designer**, a declarative development environment that you use to create **visual applications** or to customize your Oracle Cloud Applications through **extensions**. For example, you may want to extend an Oracle Cloud Application so that certain fields are displayed only for managers, while the same fields are hidden from your users that aren't managers.

## What Is a Workspace?

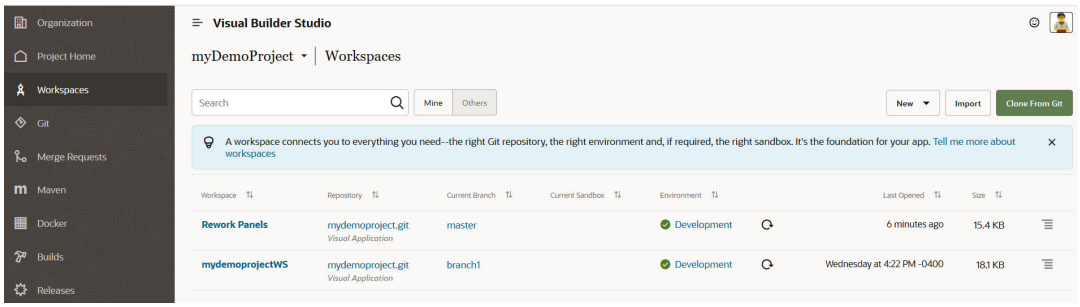
A workspace defines the resources that are available to you when you open the Designer. These resources include the Git repository—and the branch—containing the source files you want to use, the extension's or visual application's development environment, and, in certain cases for extensions, a sandbox. You can think of a workspace as your private editing environment while you're working with the Designer. If you're not using VB Studio to create or update an extension or a visual application, you won't need a workspace.

There are three ways to access a workspace (and thus, the Designer):

- From the Project Home page:

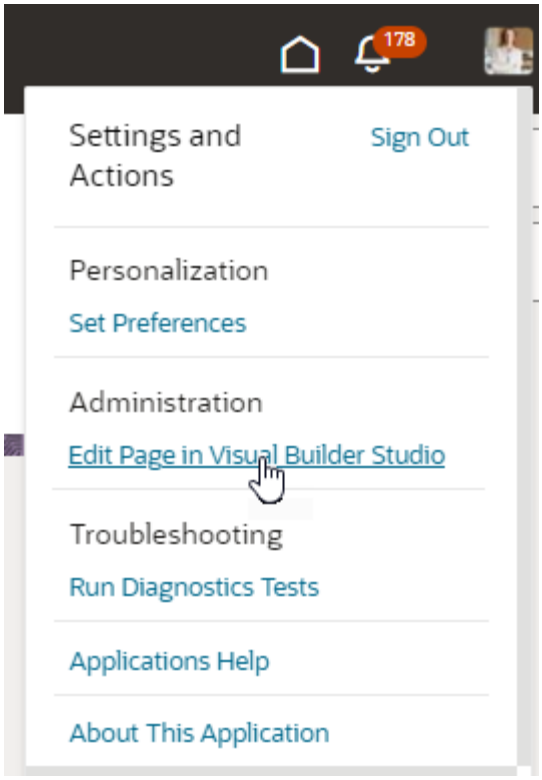



- From the Workspaces option in the left navigator:




In either case, click the workspace name to launch the Designer or right-click and select **Open in New Tab** to open the workspace in a new tab. (To see all the workspaces in the project while on the Workspaces page, click the **Others** toggle button.) Once you're in the Designer, you can change the Git repository branch, if you wish, but you can't change the Git repository where your work is stored—that's set at the project level.

- From an Oracle Cloud Application editing session:




When you click **Edit Page in Visual Builder Studio** while in Oracle Cloud Applications, you are automatically sent over to VB Studio. If you have a workspace already set up for this app, that's where you'll land in the Designer. If you don't, VB Studio will create a workspace for you.

In some cases, you may want to create a workspace explicitly, rather than allowing VB Studio to create one for you. Create a Workspace explains these options.

You're the only one on your team who can access your workspace. Changes to files you make in your workspace aren't visible to other team members until you save them to a branch (or unless you use the Share action). You can have multiple workspaces, each with a different branch and sandbox, or you can use one workspace and switch to a different branch and sandbox while you're in the Designer.

See What Is a Workspace? in *Extending Oracle Cloud Applications with Oracle Visual Builder Studio* for additional information about using workspaces with extensions. See Create Visual Applications in VB Studio for information about using workspaces with visual applications.

# Create a Workspace

A workspace may have automatically been created for you when you opened VB Studio from a page. If not, or if you have other requirements, you can create a workspace explicitly.

You can create workspaces in projects where you've been added as a team member. A workspace requires an environment against which you develop your application and deploy it to. A project could have one or more environments available for different purposes, with workspaces mapped to each of these environments:

- An app extension project will often have one development environment that one or more users will use to extend/configure the Oracle Cloud Application app. There will, however, be additional environments that users will publish/deploy their changes to and, of course, the production environment. Many projects with extensions have three environments - the first one is the development environment, the second is an intermediary one that's used for additional testing, and the third one is the production environment.
- A visual app project will also often have one VB development environment in which visual apps are developed. However, there some cases that may have two development environments, for example one that is running a new VB release while the other is running a previous VB release. Both of these will be two separate environments in the visual app project.

The environment(s) must support the type of project you are working on. To create a workspace for a visual application, your project must be associated with a Visual Builder instance. To create a workspace for an extension, your project must be associated with an Oracle Cloud Application instance.

If the development environment isn't defined, you won't be able to create a workspace. You'll need to ask the project owner or an administrator to create one for you before you try to create a workspace.

> **Note:**
>
> Typically, the Visual Builder instance added to your visual application's environment uses the same identity domain as your Visual Builder Studio instance. If you choose a Visual Builder instance from a different identity domain as your deployment environment, you'll see a warning about setting up the Allowed Origins configuration. If you see this, you'll need to talk to your administrator to make sure your instance's domain is added to its list of allowed origins, as described in Allow Other Domains Access to Services.

Depending on what you need in your workspace, the type of template you select and the information you're asked to supply during creation varies considerably. Here are the options for creating a workspace:

- **Clone an existing Git repository that contains an extension or visual application**
  Most people will create a workspace using the **Clone from Git** option, once a repository with the visual application or extension that the team works on has been created. One team member will create the Git repository, so you and each member of your team will use the clone option to create at least one workspace so they can do their work. Or, perhaps you want to continue working on a branch that someone else has started. If you work on

multiple branches and want to move freely from branch to branch without having to make sure that you added/committed changes you made in a branch, then you'd likely use the clone option to create a new workspace on the same repository.

See Create a Workspace Using an Existing Repository if you're creating a visual application or Clone an Existing Repository if you're using this workspace for an extension.

- **Create a new visual application**
  See Create Workspace for a New Visual Application for instructions on how to use this option.
- **Create a new extension**
  See Create an Extension for instructions on how to use this option.
- **Import an exported archive that contains an extension or a visual application**

  You'd use this option when you want to base your new visual application on a valid existing artifact that you've already exported from Visual Builder or Visual Builder Studio to your local system. Or, if a team member gives you an archive of an extension, you can import it to create a workspace containing all the files in their branch of the extension's Git repository. When you create a workspace by importing an archive, you create a new Git repository and branch. By using the import and export functionality, you can share application sources and move applications between instances.

  When you select an archive to import, or drag it to the upload area in the dialog, VB Studio checks to see if the archive is valid for the operation (contains a visual application or an extension).

  See Import an App Extension Archive if you're creating an extension or Create a Workspace by Importing a Visual Application if you're using this workspace for a visual application.

## What is a Scratch Repository?

When you create a workspace, you have the option to create a scratch repository, rather than creating a new repository or using a clone of the project's Git repository. You may want to create a scratch repository when you are experimenting and you're pretty sure you'll never want to merge your changes into an existing repository. A scratch repository is a private repository that only exists in your workspace. Only you can use the scratch repository, and it's deleted when you delete the workspace. If you want to let your team members use your scratch repository, you'll need to push the scratch repository to a **new** remote repository.

No build pipeline is set up for you if you create a scratch repository when creating your workspace. If you want to build and publish artifacts, you might want to create a new repository and branch instead of a scratch repository. When you create a new repository while creating a workspace, a build pipeline is automatically set up for you when the workspace is created.

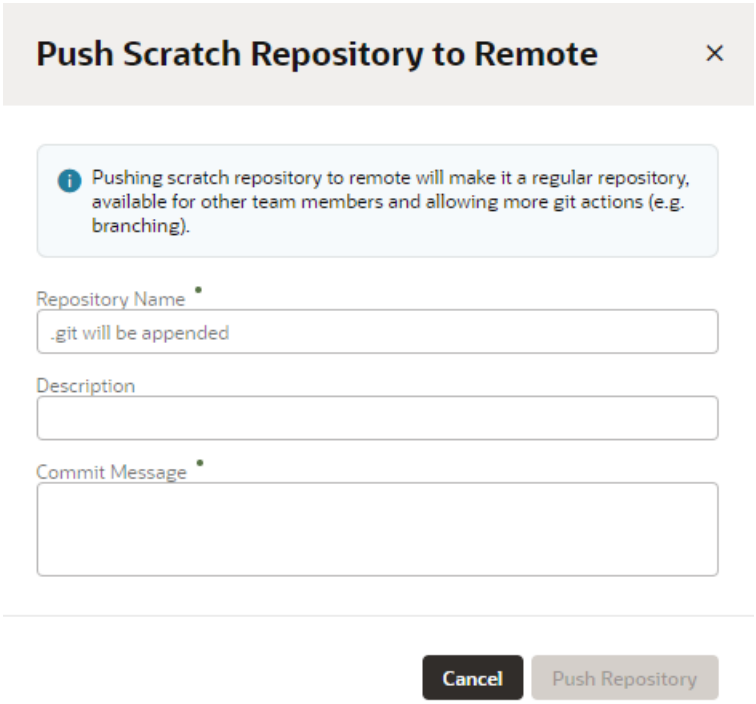
If you want to build and publish artifacts from a scratch repository, you'll need to first push the scratch repository to a new remote repository. After the new repository is created, you or your project administrator will need to set up build jobs for the repository.

## Push Your Scratch Repository to the Remote Repository

If you chose to use a scratch repository when creating your workspace, you'll need to push the scratch repository to the remote repository if you want other team members to see its contents. Pushing your scratch repository creates a **new** remote Git repository in the project.

1. Open your workspace.
2. In the header, click the arrow next to your Git repo and select **Push** in the menu.

3. In the Push Scratch Repository to Remote dialog, type a repository name. This name cannot be the same as an existing project repository.
4. Enter a commit message, and click **Push Repository**.




After you push your scratch repository to the new remote repository, you can create a pipeline with jobs for packaging the build artifacts from the repository and deploying the artifacts to your environment.

# Understanding Default Branch Names

When a new repository, both scratch or non-scratch, is created for a workspace, `main` is now used as the default branch.

When a repository is cloned, the default branch, which could be `master`, `main`, or anything that has been set manually, is used to determine whether it contains an application and can be cloned.

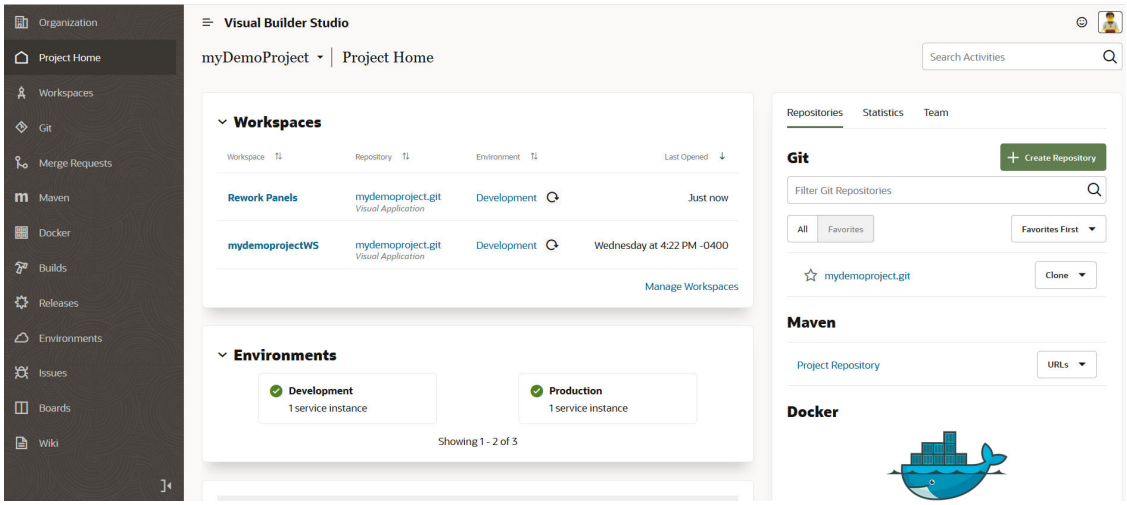
Publish actions always use the current default branch as the target branch. Existing workspaces behave as they always have — VB Studio won't try to automatically update the default branch to `main`. So, a scratch workspace that was created prior to this release will continue to use `master` as the repository default when it is pushed, whereas a newly created one will use `main`.

If you change the repository's default branch after the workspace's build jobs have been created, you'll need to manually update those jobs to use the new default branch name.

# Manage Workspaces

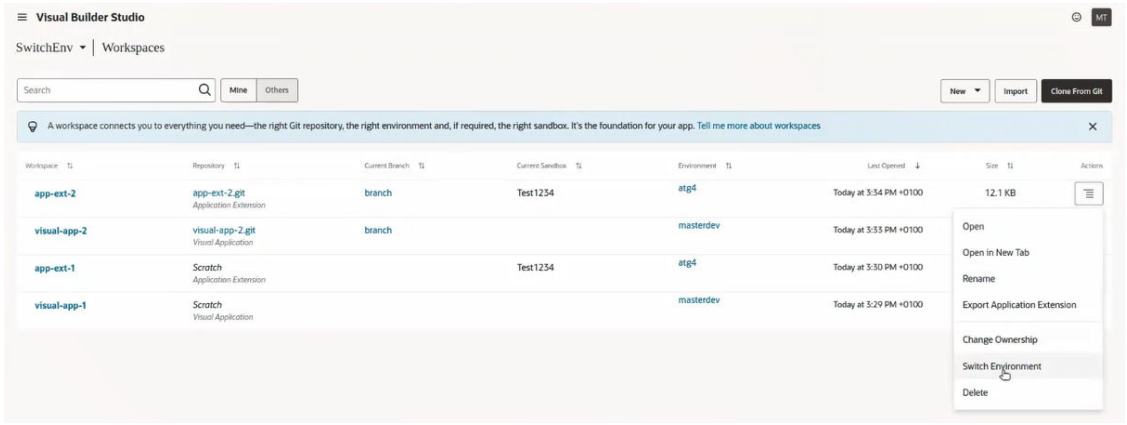
Some workspace management tasks can be done by an individual, while others must be performed by the project owners.

The Project Home page displays your personal workspaces associated with the selected project:




The Workspaces tile on that page has **Refresh** buttons for its environments, so you can reload them independently from each other. Click a workspace name in the tile to open the Designer in the context of that workspace (that is, with the correct Git branch and sandbox, if used).

Depending on your permissions, you can use the Actions menu on the Workspaces page to perform various workspace management tasks:




The activities stream on the Project Home page will display notifications about most, but not all, of these tasks. Notifications for opening workspaces, opening a workspace in new tab, and exporting workspaces won't be shown, but notifications for creating, deleting, importing, changing ownership, and renaming workspaces will be displayed.

Here's what you can do from the Workspaces page:

- Use **Mine** to view your personal workspaces only, or use **Others** to view all the workspaces associated with this project. If the list is long, you can use the search field to find specific user or workspace names.
- Open the workspace in a new tab. This is convenient when you want to work in another project area, such as Issues, without closing and leaving the page where you opened the workspace. Both pages can be open simultaneously.
- Delete a workspace.

As an individual working on extensions or visual applications, it's a good idea to delete a workspace once you've finished with it (that is, once you've used the Publish action to push your changes into the main Git branch).

> **Note:**
>
> If there are uncommitted changes to the workspace being deleted, the Confirm Delete dialog will indicate this. The dialog will also indicate if the workspace contains changes that were committed but not pushed.

If you're a project owner, you can delete workspaces that are or were associated with projects that you own, even workspaces that you don't own or didn't create. Try not to let inactive workspaces accrue in your project, as they still count against your total resource allocation and thus have a (hidden) cost.

- Both the project owner and the workspace owner can rename a workspace, or export it to an archive that can later be imported to another project. See Export a Visual Application for more information.
- Both the project owner and the workspace owner can assign a new owner for a workspace by selecting **Change Ownership** in the Actions menu. The Developer and Developer Limited roles can reassign ownership of their own workspace only, but the project owner can reassign ownership of any workspace. This may be necessary for making changes, like resolving conflicts or pushing changes. A project owner may also need to assign a new workspace owner if the previous owner left the company or organization, leaving the workspace with no owner.

> **Note:**
>
> If you're transferring a workspace's ownership to a user in different identity stripe, you'll see the "Switch environment required" warning message. Click the message to display the **Switch Environment** dialog. The dialog indicates the current environment you're using. Use the selector to choose an environment that's in the same identity stripe as the new user. The environment currently associated with the workspace won't be listed.

Once a workspace has a new owner, it behaves as if the new owner created the workspace. Only the new owner (and the project owner) can work on that workspace going forward. The original owner could still see the workspace by setting **Others**, if he or she is a project owner, but would lose all access to the workspace.

- You may find it necessary to switch the environment that your workspace is using to another environment because the old one may be down, decommissioned, or has been repurposed or replaced. By switching environments, you can resume development work using the same workspace, without having to first push any changes back to the remote repository and then create a new workspace that uses the new environment.
  You can change your workspace's environment by selecting **Switch Environment** in the Actions menu. This dialog notifies you which environment you are using and presents a list of environments that contain an Oracle Cloud Application instance or a Visual Builder instance, omitting the one currently associated with the workspace. You're prompted to choose another environment. After an environment has been selected from the list, its health is checked.

> **Note:**
>
> The list shows all environments that contain an Oracle Cloud Application instance or a Visual Builder instance, not just those that are compatible with the current workspace's contents. The Switch Environment dialog relies on you to know what type of workspace (visual app or app extension) it is. If the workspace type is a visual application, for example, and you try to select an environment that contains an Oracle Cloud Application instance, you'll see a warning. When you see this kind of message, you shouldn't make the switch. (You'll be able to select **Switch**, after choosing an incompatible environment, but a backend check prevents the switch and notifies you by displaying an error message.)

If you select an environment that contains an Oracle Cloud Application instance, you'll see the sandbox menu so you can select a sandbox (optional).

If the workspace includes Business Object data, you'll see a message warning that switching environments may cause the loss of data. To avoid losing any data, you need to export the business objects (see Export the Data to a File from the Data Manager) before switching environments and then importing them (see Import Data from a File Using the Data Manager) after you make the switch.

7

# Build Your Applications

After uploading the source code files to Git repositories, you can use the **Builds** page to create and configure build jobs and pipelines, run builds and generate artifacts.

## Configure and Run Project Jobs and Builds

Oracle Visual Builder Studio (VB Studio) includes continuous integration services to build project source files. You configure the builds from the **Builds** page.

The **Builds** page, also called the Jobs Overview page, displays information about all the project's build jobs and provides links to configure and manage them.

### What Are Jobs and Builds?

A **job** is a configuration that defines your application's builds. A **build** is the result of a job's run.

A job defines where to find the source code files, how and when to run builds, and the software and environment required to run builds. When a build runs, it packages application archives that can be deployed to a web server. A job can have hundreds of builds, each generating its own artifacts, logs, and reports.

Here are some terms that this documentation uses to describe the build features and components:

| Term | Description |
|---|---|
| Build system | Software that automates the process of compiling, linking and packaging the source code into an executable form. |
| Build executor template | A build executor template defines the operating system and software packages installed on a VM build executor. A build executor template must be created before VM build executors can be added to it.<br>See What Are VM Build Executors and Build Executor Templates?. |
| VM build executor | A VM build executor is an OCI or OCI Classic VM compute instance dedicated to running builds of jobs that organization members define in VB Studio projects. A VM build executor is always associated with a build executor template. Each build uses one VM build executor.<br>See What Are VM Build Executors and Build Executor Templates?. |
| Build artifact | A file generated by a build. It could be a packaged archive file, such as a .zip or .ear file, that you can deploy to a build server. |
| Trigger | A condition to run a build. |

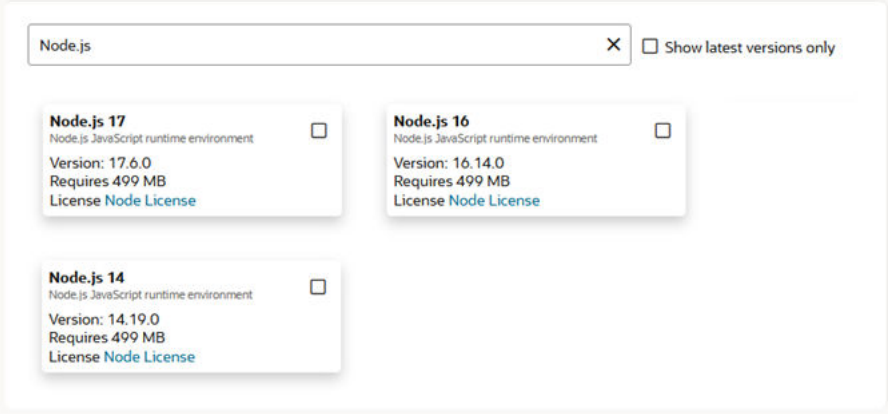
### What Are Software Packages?

There are multiple versions of some software, such as Node.js and Java, listed in the Software Catalog. This software is referred to as software packages.

The version number of a package can be categorized into two: the major version and the minor version. If a software's version is `1.2.3`, then `1` is its major version and `2.3` is its minor version.

In a software's tile, the major version number is displayed in the title of the package. In Configure Software page, the number shown in **Version** is the installed version, which includes both major and minor versions.

Here's an example. In this image, Node.js 17, 16, and 14 are shown in the software catalog. In the Node.js 17 tile, `17` is the major version and `6.0` is its minor version. The installed version of the software is `17.6.0`.




When a new minor version of a software package is available in the Software Catalog, all build executor templates using that software package are updated automatically. For example, assume that Node.js 14.17.5 is available in the Software Catalog for the Node.js 14 package. When Node.js 14.19.0 is made available in the Software Catalog, all build executor templates using the Node.js 14 package update automatically to use Node.js 14.19.0. If there's an incompatibility between the upgraded software and other installed software of the build executor template, an error is reported with suggestions about the cause of the error.

When a new major version of a software package is available in the catalog, build executor templates using the older versions of the software package aren't updated automatically. The new major version of the software is added to the catalog as a separate package. For example, when Node.js 17 is available in the Software Catalog, all build executor templates using Node.js 14 or Node.js 16 aren't updated automatically. To use the new version, you must manually update the build executor templates to use the new package.

When a major version of a software is removed from the catalog, all build executor templates using that software version are updated automatically to use the next higher version. For example, when Node.js 12 was phased out, build executor templates that were using Node.js 12 were automatically updated to use Node.js 14.

# Create and Manage Jobs

You can use the **Jobs** tab on the Builds page to create jobs that run builds and generate deployable artifacts.

> **Note:**
>
> If you're creating a new job or renaming an existing one and see a message that the job name you entered matches a pattern that is restricted for you, you don't have access to any job name that matches the glob pattern that restricts the job. Enter a name that doesn't match the name restricted by the glob pattern or contact the job owner and request to be added as an authorized user (or member of an authorized group). See Configure Job Protection Settings.

> **Note:**
>
> If you're trying to access a renamed or deleted job by using a link from the activities feed (or from a bookmarked link), you will be unable to do so because that job no longer exists. A page with a message indicating that there was an error loading the Jobs page is displayed, along with some text that says that the job could not be found because it may have been deleted or renamed. Click the **Take me back to Jobs** button and the Builds page's **Jobs** tab will be displayed.

| Action | How To |
|---|---|
| Create a blank job | 1. In the left navigator, click **Builds**.<br>2. In the **Jobs** tab, click **+ Create Job**.<br>3. In the New Job dialog box, in **Name**, enter a unique name.<br>4. In **Description**, enter the job's description.<br>5. In **Template**, select the build executor template.<br>6. Click **Create**. |
| Copy an existing job | There may be times that you want to copy parameters and a job configuration from one job to another. You can do that when you create a job. You cannot copy the configuration of an existing job to another existing job.<br>After you create the new job, you can modify the copied parameters and configuration:<br>1. In the left navigator, click **Builds**.<br>2. In the **Jobs** tab, click **+ Create Job**.<br>3. In the New Job dialog box, in **Name**, enter a unique name.<br>4. In **Description**, enter the job's description.<br>5. Select the **Copy From Existing** check box.<br>6. In **Copy From**, select the source job.<br>7. In **Template**, select the build executor template.<br>8. Click **Create**. |
| Create a job that accepts build parameters and will be associated with a merge request | 1. In the left navigator, click **Builds**.<br>2. In the **Jobs** tab, click **+ Create Job**.<br>3. In the New Job dialog, in **Name**, enter a unique name.<br>4. In **Description**, enter the job's description.<br>5. Select the **Use For Merge Request** check box.<br>6. In **Template**, select the build executor template.<br>7. Click **Create**. |
| Create a job using YAML | In VB Studio, you can use a YAML file to create a job and define its configuration. The file is stored in a project's Git repository. See Configure Jobs and Pipelines with YAML. |
| Run a build of a job | In the **Jobs** tab, click the **Actions** menu and select **Build Now**. |

| Action | How To |
|---|---|
| Configure a job | The job configuration page opens immediately after you create a job. You can also update an existing job from the **Jobs** tab. Click the **Actions** ⋯ menu and select **Configure**. |
| View a job's log | In the **Jobs** tab, click the **Actions** ⋯ menu and select **View Log**. The log for the most recent run is displayed. |
| Disable a job | In the **Jobs** tab, click the **Actions** ⋯ menu to disable a job, and select **Disable**. A message indicating that the job has been disabled successfully is displayed. The **Disabled** icon in the job's Status column indicates that the job has been disabled. It cannot be run unless it is enabled again.<br>The **Disable** option is unavailable if the job does not afford write permission. |
| Enable a disabled job | In the **Jobs** tab, click the **Actions** ⋯ menu to enable a disabled job, and select **Enable**. A message indicating that the job has been enabled successfully is displayed.<br>The **Enable** option is unavailable if the job does not afford write permission. |
| Delete a job | In the **Jobs** tab, click the **Actions** ⋯ menu and select **Delete**. The Delete Job dialog is displayed and you'll see the message, "Do you really want to delete job xx1 and all its builds and data? This operation cannot be undone." Click the **I understand that this job will be permanently deleted** button, and then click **Delete** button. |
| Update multiple jobs | 1. In the **Jobs** tab, select the checkboxes for the jobs you'd like to update.<br>2. Click **Update Selected Jobs**, then select **Enable**, **Disable**, or **Delete**. |
| View status of all job builds | In the **Jobs** tab, click the **View Job Statistics** button. A pie graph showing all of the build jobs is displayed. The status is shown using percentages. |

You can use the **Job Queue** tab on the Builds page to cancel a single running job, or multiple running jobs.

| Action | How To |
|---|---|
| Cancel a single running job | There are two ways to cancel a single running job. You can cancel the job without selecting its check box:<br>1. From the Job Queue page, for the job you want to cancel, click the **Actions** ⋯ column to the right.<br>2. Select **Cancel**.<br>The **Cancel Build** dialog is shown and you'll see the prompt, “Do you really want to cancel job xx build x?"<br>3. Click the **Cancel Build** button.<br>You can also cancel the job by selecting its checkbox:<br>1. From the Job Queue page, select a build job using its checkbox in the leftmost column.<br>2. Select the **Update Selected** dropdown menu.<br>3. Select the **Cancel Selected Builds** menu item.<br>The Cancel Selected Builds dialog is shown and you'll see the prompt, "Do you really want to cancel the 1 selected builds?"<br>4. Click the **Cancel Build** button. |
| Cancel multiple running jobs | 1. From the Job Queue page, select multiple build jobs using the checkboxes in the leftmost column.<br>2. Select the **Update Selected** dropdown menu.<br>3. Select the **Cancel Selected Builds** menu item.<br>The Cancel Selected Builds dialog is shown and you'll see the prompt, "Do you really want to cancel the 3 selected builds?"<br>4. Click the **Cancel Builds** button. |

You can use the **Build Notification** tab on the Builds page to manage email notifications for multiple jobs.

| Action | How To |
|---|---|
| Enable or disable job notifications | To enable or disable job notifications, click the toggle switch in the Enable column for the applicable jobs.<br>You can't edit notification settings for private jobs unless you're an authorized user. Private jobs are indicated with a lock symbol in the Private column.<br>Job notifications must also be enabled in your user preferences. Click your user image, then select **Preferences**.<br>• Click **Profile** and confirm that your email address has been verified – if not, click **Re-send email** to verify.<br>• Click **Notifications** and select **Build activities** and **Include my own updates**. |
| Subscribe to Successful or Failure | Use the toggle switches in the Subscribe to Successful and Subscribe to Failure columns to enable or disable those subscriptions. |

# Configure a Job

You can create, manage, and configure jobs from the Jobs tab on the **Builds** page.

To open a job's configuration page, go to the **Jobs** tab on the **Builds** page, click the job's **Actions** menu, and select **Configure**.

Click the **Show/Hide Sidebar** icon on right to hide or show the sidebar buttons (**Save**, **Cancel**, and page action) on the right side of the Git, Parameters, Before Build, Steps, and After Build pages. This provides a wider page for you to see all the steps (or settings) with less scrolling. In addition, if any of the job configuration pages is long enough to have a scrollbar, you can use the **Go To Top** or **Go To Bottom** buttons in the sticky header to traverse the pages with minimal scrolling. With a single click, you can go to the bottom of the page or return to the top without using the scrollbar.

## Configure Job Protection Settings

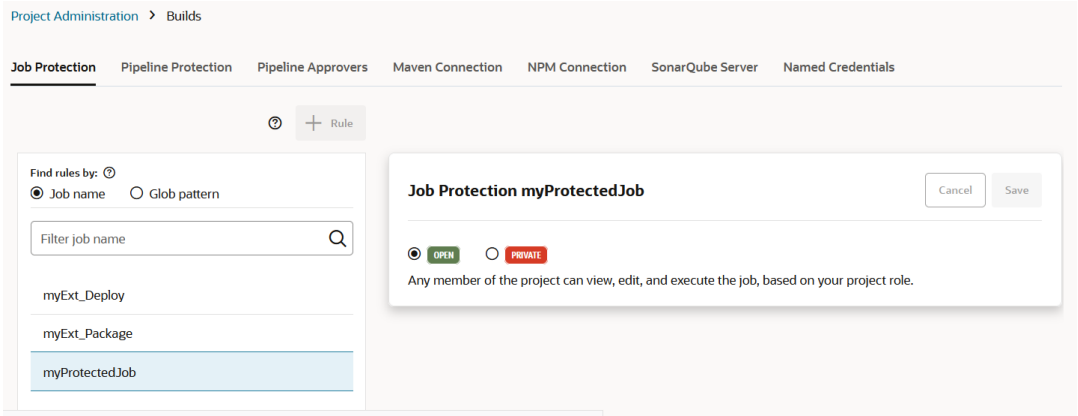
The project owner can mark a job as private to restrict who can see it, edit it, delete its configuration, or run its build:

1. In the left navigator, click **Project Administration**.
2. Click **Builds**.
3. Click the **Job Protection** tab.
4. From the jobs list, select the job.

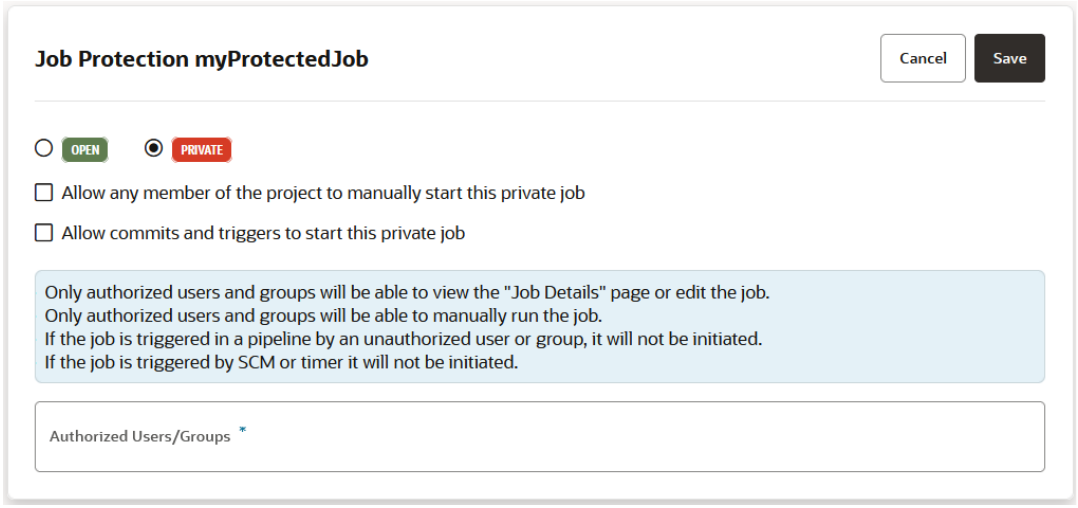
   If your project has many jobs, you may have difficulty finding the specific job you want to protect. Use the **Search** bar to quickly locate the job to which you want to add the restricted settings.

   If a job in the list of jobs to the left has a lock icon next to it, it has already been protected. A protected job's restrictions can still be modified, removed, or the list of authorized users and groups can still be changed.

The **Protection Settings** screen is displayed.



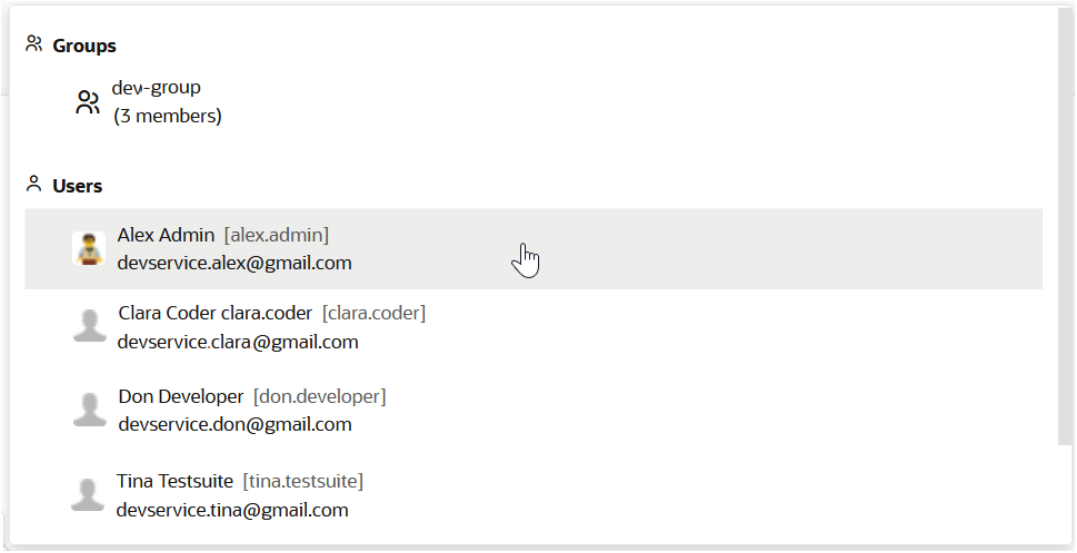

5. Select the **Private** option.




With just this option selected, only authorized users and groups will be able to view the Job Details page, edit the job, or manually run it. If the job is triggered in a pipeline by an unauthorized user or group, or if it is triggered by SCM or a timer, it will not be initiated.

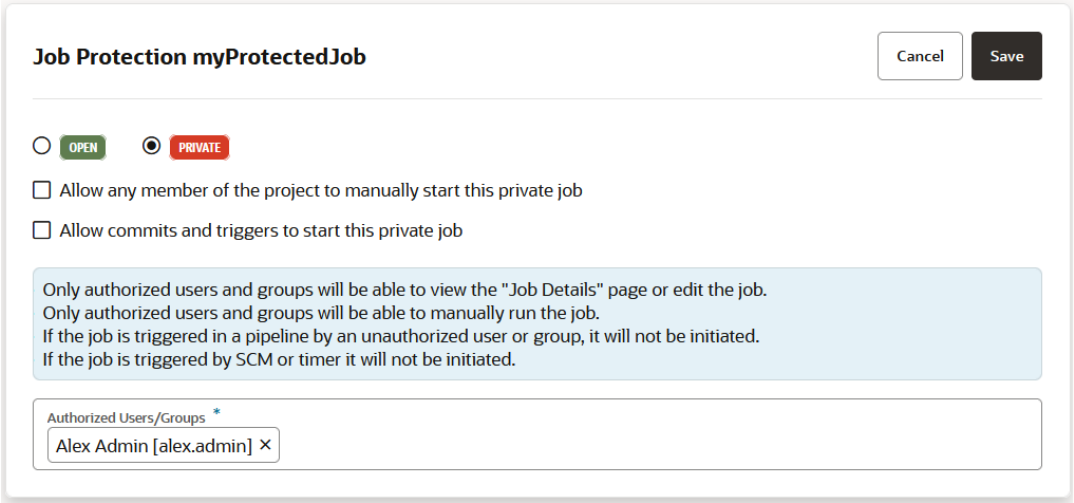
6. Click in the **Authorized Users/Groups** field to display a dialog that lists the project's **Groups** and **Users** you can select from.

   Under Users, you can see a flattened list of all users that are members of the group(s) as well as ones that were added individually. For example, the dev-group members (Clara Coder, Don Developer, and Tina Testsuite) appear in the Users list, along with Alex Admin, who was added individually. From the list, select one or more groups and/or users. Don't forget to add yourself.

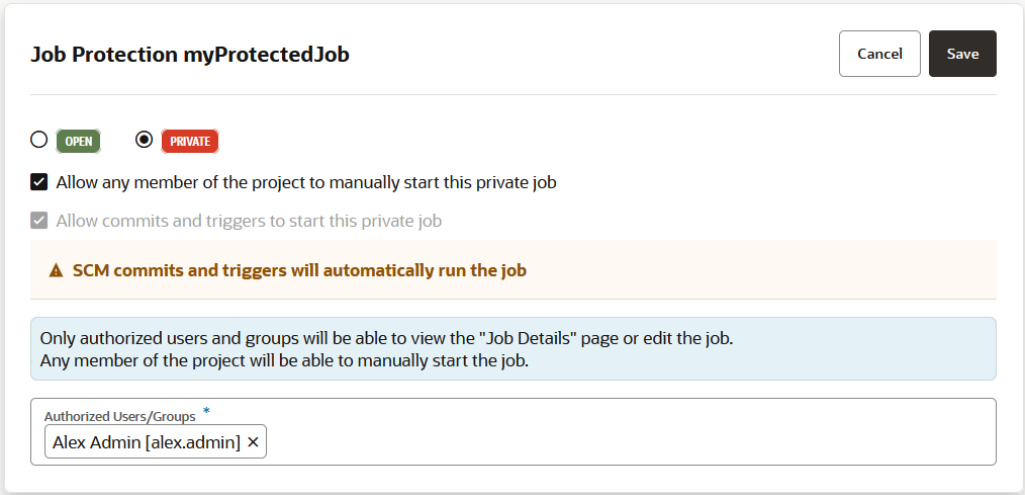



> **Note:**
>
> Users in Oracle Cloud Application environments that have multiple VB Studio instances in different identity stripes have username strings that include the environment name, where that user has been defined. Since a unique user may have been defined for multiple environments, this format ensures that one identity can be distinguished from that user's other unique identities. This should help you select the correct user to add.

Alex Admin was selected.

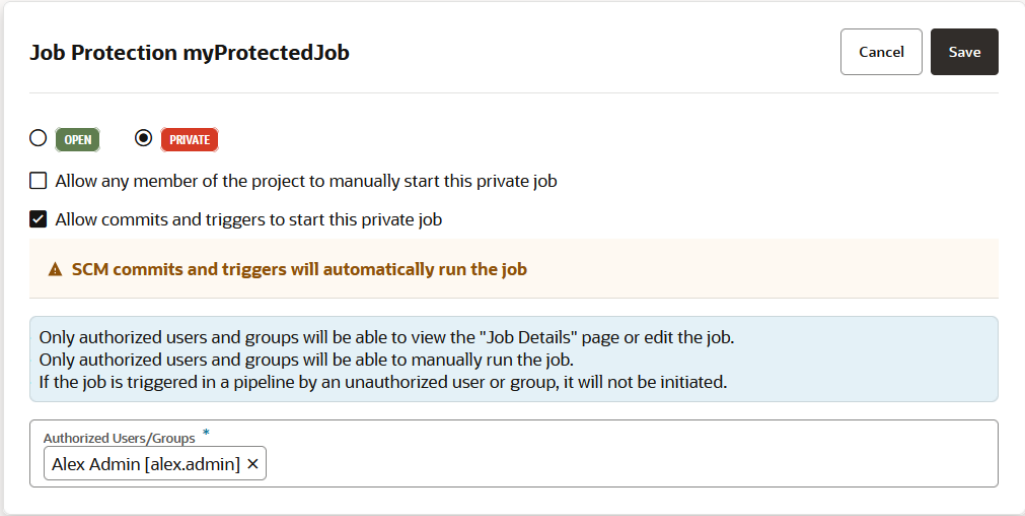



7. Select the checkboxes to allow project members to manually start private jobs and/or allow commits and triggers to automatically start private jobs:
   - Select the **Allow any member of the project to manually start this private job** checkbox to allow any project member, not just authorized users, to manually start the job.




Notice that when you select the first checkbox, VB Studio automatically selects the second checkbox, which allows commits and triggers to start the private job, and grays it out. With this setting, only authorized users and groups can view the Job Details page or edit the job, but any project member can start and run the job. In addition, SCM commits or triggers will also automatically start and run the job.

- Select just the **Allow commits and triggers to start this private job** checkbox if you want SCM commits and triggers to be able to automatically run this job.




With just this checkbox selected, periodic triggers will run any job or pipeline, including private jobs set to allow commits and triggers to start the private job. However, if a pipeline includes a private job with this option selected and a non-authorized user attempts to run the pipeline manually, the private job won't run but periodic triggers and SCM commits will.

Leave the checkbox unselected if you don't want the job to be started when it is triggered by an SCM commit or timer.

> **Note:**
>
> **Best Practice:**
>
> If you use the checkbox to enable the protected build to be triggered by an SCM commit, you need to protect the branch that the build job is tied to. If you don't do this, anyone can trigger the protected build by making a commit to trigger the build.

8. Click **Save**.

   The activity stream displays all changes to a job's protection status, like changing the job protection from public to private, or private to public, or changing a private job to allow commits and triggers.

You can see if a job is private from several places in the VB Studio user interface. A private job is indicated by a **Lock** icon:

- In the jobs list found on the **Project Administration** tile's **Builds** page's **Job Protection** tab, to the right of each protected job's name.
- In the **Private** column on the **Builds** page's **Jobs** tab.
- In the jobs shown in the the **Builds** page's **Pipelines** tab.

An unauthorized user can't run a private build job manually, or through a pipeline, or via an SCM/periodic trigger.

## Access a Project's Git Repositories

You can configure a job to access a project's Git repositories and their source code files:

1. Open the job's configuration page.
2. Click the **Git** tab.
3. From the **Add Git** list, select **Git**.
4. In **Repository**, select the Git repository to track.
   When you created the job, if you selected the **Use for Merge Request** check box, the field is automatically populated with the `${GIT_REPO_URL}` value. Don't change it.
5. In **Branch**, select the branch name in the repository to track. By default, `main` is set.
   When you created the job, if you selected the **Use for Merge Request** check box, **Branch** is automatically populated with the `${GIT_REPO_BRANCH}` value. Don't change it unless you don't want to link the job with a merge request.
6. Click **Save**.

If you specify multiple Git repositories, make sure that you set **Local Checkout Directory** for all Git repositories.

### Run a Build Automatically on an SCM Commit

You can configure a job to monitor a Git repository and trigger a build automatically each time a commit is pushed.

There are different use cases in which an automatically-triggered build will work and other cases where you should set up a polling job instead:

- A trigger that's set up to build automatically on an SCM commit only works with repositories that are defined and stored in the same project. In this case, commits to Git repositories in your project are sufficient triggers for automatically initiating a build.
- An automatic trigger for a build job isn't recommended for an external repository. In this case, a polling job should be set up. Only a `git push` operation will result in a build.
- An automatic trigger for a build job isn't recommended for an external repository that has been cloned as an internal repository either. In this case, a polling job should be set up. Only a `git push` operation will result in a build.

To set up a polling-based build trigger, see Trigger a Build Automatically According to an SCM Polling Schedule.

Here's how to configure a job that monitors a Git repository in your project and triggers a build automatically when a Git commit is pushed to the repository being tracked:

1. Open the job's configuration page.
2. Click the **Git** tab and either use the dropdown to select the repository you want to monitor or type the name of the repository in the entry field.
3. For the Git repository you want to monitor, select the **Automatically perform build on SCM commit** check box.
4. To include or exclude files when tracking changes in the repository, click the **Include** or **Exclude** button, then enter the names of one or more branches, a pattern (like `valid.-*`), or both in the **Branches** field. See Generate Cron Expressions for more information on entering patterns.

> **Note:**
>
> If you choose to run this job manually using the **Build Now** button, we'll show you a list of all the branches in your repo that match the Include/Exclude criteria so you can pick one to build. For example, if your Include criteria is `branch*` and `test*`, then we'll show you all the branches that begin with those terms. If you specified Exclude instead, we'll show all the branches that *don't* begin with those terms—likely quite a few. If we don't find a branch that matches, we'll build your default branch instead.

5. Click **Save**.

## Trigger a Build Automatically According to an SCM Polling Schedule

SCM polling enables you to configure a job to periodically check the job's Git repositories for any commits pushed since the job's last build. If updates are found, it triggers a build. You can configure the job and specify the schedule in Coordinated Universal Time (UTC), the primary time standard by which the world regulates clocks and time. If you're not a Cron expert, use the novice mode and set the schedule by specifying values. If you're a Cron expert, use the Expert mode.

You can specify the schedule using Cron expressions:

1. Open the job's configuration page.
2. In the **Git** tab, add the Git repository.
   To include or exclude files when tracking changes in the repository according to a Cron expression, see Include or Exclude Files to Trigger a Build.

3. Click **Settings** .
4. Click the **Triggers** tab.
5. Click **Add Trigger** and select **SCM Polling Trigger**.
6. To use the expert mode, select the **Expert mode** check box and enter the schedule in the text box.
   The default pattern is `0/30 * * * *`, which runs a build every 30 minutes.

   After you edit the expression, it's validated immediately when the cursor moves out of the text box. Note that other fields of the section aren't available when the check box is selected.
7. To use the novice mode, deselect the **Expert mode** check box and specify the schedule information. The page displays the generated Cron expression next to the **Expert mode** check box.
8. To use the novice mode, deselect the **Expert mode** check box and specify the schedule information in **Minute**, **Hour**, **Day of the Month**, **Month**, and **Day of the Week**.
   Click **Toggle Recurrence** to add or remove `0/`or `1/` at the beginning of the value in the Cron expression.

   The page displays the generated Cron expression next to the **Expert mode** check box.

   > **Tip:**
   >
   > To check the job's Git repositories every minute, deselect all check boxes. Remember that this may consume large amounts of system resources.

9. If necessary, in **Comment**, enter a comment.
10. To view and verify the build schedule of the next ten builds, from the timezone drop-down list, select your time zone and then click **View Schedule**.
11. Click **Save**.

To see the SCM poll log of the job after the build runs, in the job's details page or the build's details page, click **SCM Poll Log** .

## Generate Cron Expressions

You can use Cron expressions to define periodic build patterns.

For more information about Cron, see `http://en.wikipedia.org/wiki/Cron`.

You can specify the Cron schedule information in the following format:

```
MINUTE HOUR DOM MONTH DOW
```

where:

- `MINUTE` is minutes within the hour (0-59)
- `HOUR` is the hour of the day (0-23)
- `DOM` is the day of the month (1-31)
- `MONTH` is the month (1-12)
- `DOW` is the day of the week (1-7)

To specify multiple values, you can use the following operators:

- `*` to specify all valid values
- `-` to specify a range, such as `1-5`
- `/` or `*/X` to specify skips of X's value through the range, such as `0/15` in the `MINUTE` field for `0,15,30,45`
- `A,B,...,Z` can be used to specify multiple values, such as `0,30` or `1,3,5`

## Include or Exclude Files to Trigger a Build

When you've configured a job to monitor a Git repository, you can use fileset patterns to include or exclude files when tracking changes in the repository. Then, each time a change is committed, only changes to files that match these patterns determine whether a build is triggered or not.

Fileset patterns work as filters to include or exclude files when tracking changes in a repository. They take effect only when you've enabled a build to be triggered either on each SCM commit or according to a polling schedule. Once these settings are enabled, each time changes are committed to the repository, the filter is applied and a build either runs or not based on the specified filter.
Here's how you specify fileset patterns that use changes to files that match these patterns to determine whether a build is triggered or not:

1. Open the job's configuration page.
2. Click the **Git** tab.
3. Expand **Advanced Git Settings** and select either **Include** or **Exclude**.
   - Click **Include** to specify a list of files and directories in the repository that you want to track for changes. By default, all files are included for tracking (`**/*`), meaning changes to any file or directory in the repository will trigger a build.
     To change the default configuration, select **Include** and specify the fileset to be included in **Directories/Files**. You can use regular expressions (regex) or glob patterns to specify the fileset. Each entry must be separated by a new line.

     You can extend this configuration to specify **Exceptions** to the included fileset. If changes occur only in the fileset specified as an exception, a build won't run.

     Here are some glob pattern examples:

| Desired Outcome | In Directories/Files, enter: | In Exceptions, enter: | Result |
|---|---|---|---|
| Trigger a build following changes to `.html`, `.jpeg`, or `.gif` files in the `myapp/src/main/web/` directory: | `myapp/src/main/web/*.html`<br>`myapp/src/main/web/*.jpeg`<br>`myapp/src/main/web/*.gif` | Leave blank | A build runs when a `.html`, `.jpeg`, or `.gif` file is changed in the `myapp/src/main/web/` directory. |
| Trigger a build following changes to `.java` files, but not `.html` files: | `*.java` | `*.html` | A build runs when any `.java` file is changed, except when all changed files are `.html` files. |

| Desired Outcome | In Directories/Files, enter: | In Exceptions, enter: | Result |
|---|---|---|---|
| Trigger a build following changes to `.java` files, but not `test.java`: | `*.java` | `test.java` | A build runs when any `.java` file is changed, except when `test.java` is the only changed file. |

- Click **Exclude** to specify a list of files and directories in the repository that you don't want to track for changes. If all changes are only in the specified files, a build won't be triggered. By default, no files are excluded, meaning all files and directories are tracked and therefore, changes to any file or directory in the repository will trigger a build.
  To change the default configuration, select **Exclude** and specify the fileset to be excluded in **Directories/Files**. You can use regular expressions (regex) or glob to specify an excluded fileset. Each entry must be separated by a new line.

  Optionally, specify files or directories within the excluded fileset that you want to include as **Exceptions**. If changes occur in the fileset specified as an exception, a build will be triggered.

  Here are some glob pattern examples:

| Desired Outcome | In Directories/Files, enter: | In Exceptions, enter: | Result |
|---|---|---|---|
| Don't trigger a build when only `.java` files are changed: | `*.java` | Leave blank | A build won't run when all changed files are `.java` files, but changes in any other file (say, `test.txt` and `test.html`) triggers a build. |
| Don't trigger a build when `.java` files in the `/myapp/mobile/` directory are changed, with the exception of `test.java`: | `/myapp/mobile/*.java` | `test.java` | A build won't run when all changes are in `.java` files other than `test.java` in the `/myapp/mobile/` directory. But a build runs when `test.java` in the `/myapp/mobile/` directory is the only changed file. |
| Don't trigger a build for changes to any file, except `.sql` files: | `**/*` | `*.sql` | A build runs only when `.sql` files are changed. |
| Don't trigger a build when only `.html`, `.jpeg`, or `.gif` files in the `myapp/src/main/web/` directory are changed: | `myapp/src/main/web/*.html`<br>`myapp/src/main/web/*.jpeg`<br>`myapp/src/main/web/*.gif` | Leave blank | A build won't run when only `.html`, `.jpeg`, or `.gif` files in the `myapp/src/main/web` directory are changed. |
| Don't trigger a build when `.gitignore` files are changed: | `*.gitignore` | Leave blank | A build won't run when the only changed files are `.gitignore` files. |

4. Click **Save**.

## Use External Git Repositories

If you use an *external* Git repository to manage source code files, you can configure a job to access its files when a build runs:

- If the external Git repository is a public repository, mirror it in the project or use its direct URL in the job configuration.
- If the external Git repository is a private repository, you must mirror it in the project. See Mirror an External Git Repository.

To configure a job to use an external Git repository:

1. Open the job's configuration page.
2. Click the **Git** tab.
3. From the **Add Git** list, select **Git**.
4. If the external Git repository is mirrored with the project, in **Repository**, select the repository name. The build executor will use the mirrored repository's internal address URL.

   If the external Git repository isn't mirrored with the project, enter the repository's direct URL. Don't enter the direct URL of a private repository.
5. Configure other fields of the page and click **Save**.

To trigger a build on an update to the external Git repository, set up SCM polling according to the frequency of commits. VB Studio can't trigger a build immediately on an update to the external Git repository.
Before you set SCM polling, note that if you use the internal address URL of a mirrored repository, there's a wait time of at least 5 minutes. If you use the external address URL or the direct URL of the repository, there's a wait time of at least 1 minute. Remember that polling every few minutes consumes large amounts of system resources.

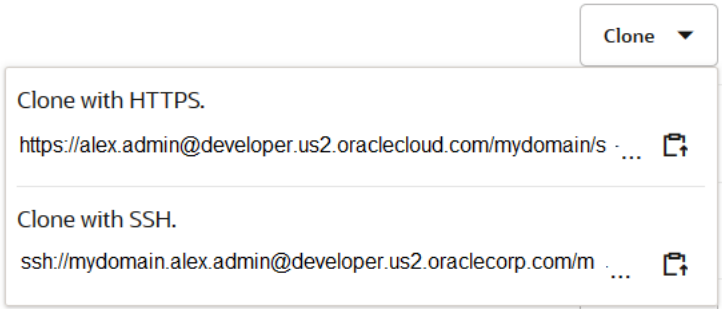
## Access Files in a Git Repository's Private Branch

To access a Git repository's private branch, configure the job to use SSH:

1. On the computer that you'll use to access the Git repository, generate a SSH key pair and upload its private key to VB Studio. See Upload Your Public SSH Key. Make sure that the private key on your computer is accessible to the Git client.

   Ignore this step if you've already uploaded the SSH public key.
2. Copy the Git repository's SSH URL.

   On the Git page, from the **Repositories** drop-down list, select the Git repository. From the **Clone** drop-down list, click **Copy to clipboard** and copy the SSH URL:




3. Open the job's configuration page.
4. Click the **Git** tab.
5. From the **Add Git** list, select **Git**.
6. In **Repository**, paste the SSH URL of the Git repository.
7. In **Branch**, select the private branch.
8. Click the **Before Build** tab.
9. Click **Add Before Build Action** and select **SSH Configuration**.
10. In **Private Key** and **Public Key**, enter the private and public key of your SSH Private-Public key pair.

    Leave the **Public Key** empty to use fingerprint.
11. In **Pass Phrase**, enter the pass phrase of your SSH Private-Public key pair. Leave it empty if the keys aren't encrypted with a pass phrase.
12. Continue to configure the job, as desired.
13. Click **Save**.

## Publish Git Artifacts to a Git Repository

You can set up a post-build action to publish Git artifacts, such as tags, branches, and merge results, to a Git repository.

Before publishing any artifacts to a Git repository, you're encouraged to review Best Practices for Storing Large Files and Binaries. The information should help you determine the best way to handle generated artifacts in your project, especially if they are generated frequently.
To publish Git artifacts to a Git repository:

1. Open the job's configuration page.
2. In the **Git** tab, add the Git repository where you want to publish Git artifacts.
3. Click the **After Build** tab.
4. Click **Add After Build Action** and select **Git Publisher**.
5. To push Git artifacts to the Git repository only if the build succeeds, select the **Publish only if build succeeds** check box.
6. To push merges back to the target remote name, select the **Merge results** check box.
7. To push a tag to the remote repository, in **Tag to push**, specify the Git repository tag name. You can also use environment variables. In **Target remote name**, specify the target remote name of the Git repository where the tag is pushed. By default, `origin` is used.

   The push fails if the tag doesn't exist. Select the **Create new tag** check box to create the tag and enter a unique tag name.

8. To push a branch to the remote repository, in **Branch to push**, specify the Git repository branch name. You can also use environment variables. In **Target remote name**, specify the target remote name of the Git repository where the branch is pushed. By default, `origin` is used.
9. Click **Save**.

## Advanced Git Options

When you configure a job's Git repositories, you can also configure several advanced Git options, such as changing the remote name of the repository, setting the checkout directory in the workspace, and specifying whether or not the workspace is cleaned before a build runs.

You can select and enable these configuration actions from the **Git** tab of the job configuration page. To see all configuration actions, expand the **Advanced Repository Options** and **Advanced Git Settings** sections.

| Action | How To |
|---|---|
| Change the remote name of a repository | For the Git repository, specify the new name in **Name**. The default remote name is `origin`. |
| Specify a reference specification of a repository | A reference repository helps to speed up the builds of the job by creating a cache in the workspace and hence reducing the data transfer. When a build runs, instead of cloning the Git repository from the remote, the build executor clones it from the reference repository.<br>To create a reference specification of a Git repository, specify the name in **Ref Spec**.<br>Leave the field empty to create a default reference specification. |
| Specify a local checkout directory | The local checkout directory is a directory in the workspace where the Git repository is checked out when a build runs.<br>To specify the directory of a Git repository, specify the path in **Local Checkout Directory**. Be sure that directory names don't contain a forward slash or null character. You can optionally include parameters in the path, such as:<br>`data/$myDir/$myDir2/stuff`<br>If **Local Checkout Directory** is left empty, the Git repository is checked out on the root directory of the workspace. |
| Include or exclude a list of files and directories to determine whether to trigger a build or not | When you've enabled a build to be triggered either on each SCM commit or according to a polling schedule, select **Include** or **Exclude**.<br>• To include a list of files and directories to track for changes and trigger a build, click **Include** and specify a fileset in **Directories/Files**. Default is `**/*`, indicating that all files and directories in the repository are tracked and changes to any file or directory will trigger a build.<br>If you don't want a build to be triggered for some files and directories in the included fileset, specify **Exceptions**. For example, to trigger a build following changes to all but `.java` files, enter `**/*` in **Directories/Files** and `*.java` in **Exceptions**.<br>• To exclude a list of files and directories from tracking and prevent a build from being triggered, click **Exclude** and specify a fileset in **Directories/Files**. If all changes occur only in the specified files or directories, a build won't run. Default is an empty list, indicating no files are excluded.<br>If you want a build to be triggered for some files and directories in the excluded fileset, specify **Exceptions**. For example, to trigger a build only when `.sql` files are changed, enter `**/*` in **Directories/Files** and `*.sql` in **Exceptions**.<br>For more examples, see Include or Exclude Files to Trigger a Build. |

| Action | How To |
|---|---|
| Check out the remote repository's branch and merge it into a local branch | In **Merge another branch**, specify the branch name to merge to. If specified, the build executor checks out the revision to build as `HEAD` on this branch.<br>If necessary, in **Checkout revision**, specify the branch to checkout and build as `HEAD` on the value of **Merge another branch**. |
| Configure Git user.name and user.email variables | In **Config user.name** and **Config user.email**, specify the user name and the email address. |
| Merge to a branch before a build runs | Select the **Merge from another repository** check box.<br>In **Repository**, enter or select the name of the repository to be merged. In **Branch**, enter or select the name of the branch to be merged. If no branch is specified, the default branch of the repository is used.<br>The build runs only if the merge is successful. |
| Prune obsolete local branches before running a build | Select the **Prune remote branches before build** check box. |
| Set the depth for cloning | Select the **Use shallow clone** check box and then choose from 1-8192 commits to grab when you clone. For example, to clone with the three most recent commits, select a value of 3. The Depth default is 1, except for YAML jobs, where the default depth is 0. Selecting a low depth can speed up your build process as less data needs to be fetched from the Git repository before the build.<br>The default setting is a full depth job. |
| Skip the internal tag | When a build runs, the build executor checks out the Git repository to the local repository of the workspace and applies a tag to it. To skip this process, deselect **Skip internal tag** check box. |
| Remove untracked files before running a build | Select the **Clean after checkout** check box. |
| Retrieve sub-modules recursively | Select the **Recursively update submodules** check box. |
| Display commit's author in the log | By default, the Git change log shows the commit's `Committer` . To display commit's `Author`, select the **Use commit author in changelog** check box. |
| Delete all files of the workspace before a build runs | Select the **Wipe out workspace before build** check box. |

## View the SCM Changes Log

The SCM changes log displays files that were added, edited, or removed from the job's Git repositories before the build was triggered.

You can view the SCM changes log from the job's details page and a build's details page. The **Recent SCM Changes** page that you open from the job's details page shows SCM commits from the last 20 builds, in reverse order. The **SCM Changes** page that you open from a build's details page shows SCM commits that happened after the previous build.

The log shows the build ID, commit messages, commit IDs, and affected files.

# Trigger a Build Automatically on a Schedule

You can configure a job to run builds on a specified schedule specified in either Coordinated Universal Time (UTC), the primary time standard by which the world regulates clocks and time,

or in a timezone, which takes Daylight Savings Time (DST) into account so you don't need to manually change the clock twice each year.

> **Note:**
>
> Regardless of how you set up the schedule, your builds could be delayed if no VM build executors of the job's build executor template are free to run builds at the scheduled time, or if any of the VM build executors are in the Stopped/Pending state.

1. Open the job's configuration page.
2. Click **Settings** .
3. Select the **Triggers** tab.
4. Click **Add Trigger** and select **Periodic Build Trigger**.
   You can specify the schedule using Cron expressions:
   - Use the Expert mode (see steps 5 then 6) if you're comfortable using Cron expressions.
   - Use the novice mode and set the schedule by specifying values (see step 5 then 7) if you're less familiar with Cron expressions.
5. In Time zone, select **UTC** (default) or **Another time zone** to specify the time setting used for the schedule.
   If you select **Another time zone**, use the drop down menu to select a location-based time zone to use. Notice that the Time Zone Schedule to the right changes to reflect your selection. There are a few advantages to setting the timezone. You don't need to manually change the clock twice each year (ahead in the Spring and back in the Fall) and you can see the times in the timezone you selected without doing any calculation to know when the trigger will happen in your selected time zone.

   > **Tip:**
   >
   > Click **Compare Schedules** and select a different time zone to compare the scheduled times in that time zone to the times in the selected time zone.

6. To use the expert mode, select the **Expert mode** check box, and enter the schedule in the text box.

   The default pattern is `0/30 * * * *`. This setting runs a build every 30 minutes.

   After you edit the expression, it's validated as soon as you move the cursor outside the text box. Note that other fields of the section aren't available when the check box is selected.
7. To use the novice mode, deselect the **Expert mode** check box and specify the schedule information in **Minute**, **Hour**, **Day of the Month**, **Month**, and **Day of the Week**.

   Click **Toggle Recurrence** to add or remove `0/` or `1/` at the beginning of the value in the Cron expression.

   The page displays the generated Cron expression next to the **Expert mode** check box.
8. If necessary, in **Comment**, enter a comment.
9. To view and verify the build schedule of the next ten builds, from the timezone drop-down list, select your time zone and then click **View Schedule**.

10. Click **Save**.

## Use Build Parameters

You can use build parameters for passing additional information to a build when it runs when that information isn't available at job configuration time.

You can configure a job to use a parameter and its value as an environment variable or through variable substitution in other parts of the job configuration. When a build runs, a Configure Parameters dialog box opens so you can enter or change the default values of the parameters:

1. Open the job's detail page.
2. Click **Configure**.
3. Click the **Parameters** tab.
4. From the **Add Parameter** drop-down list, select the parameter type.
   You can add these types of build parameters:

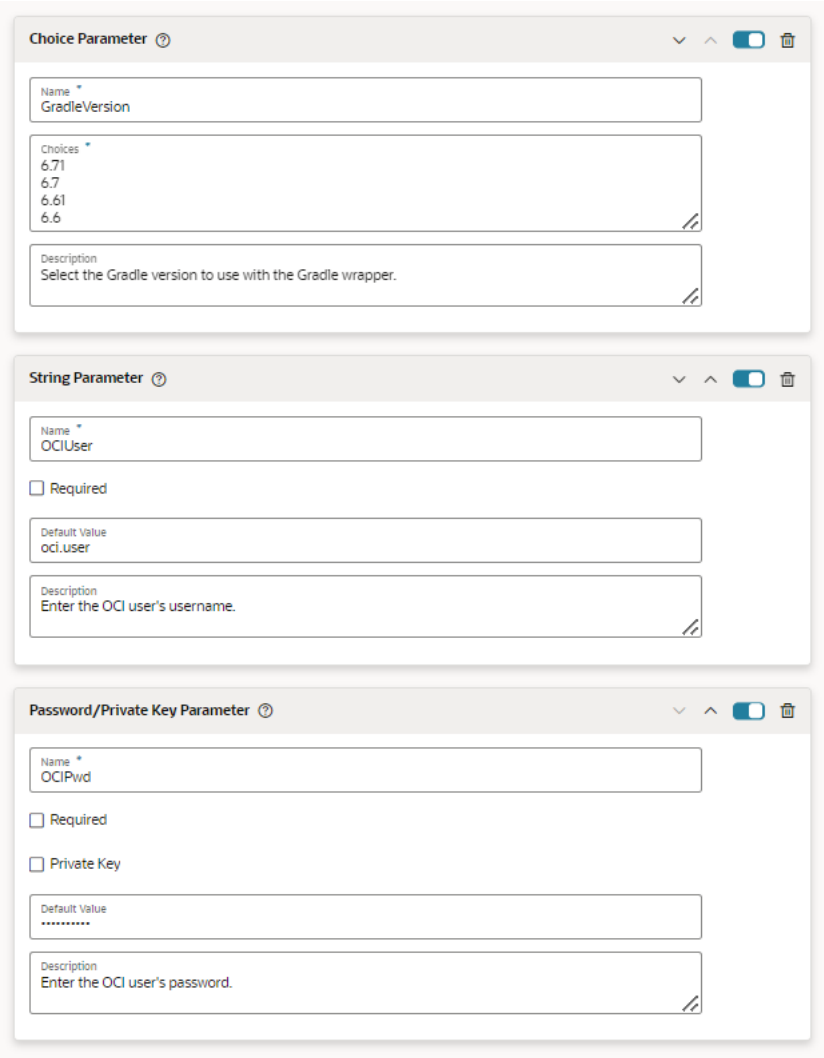
| Use this parameter type ... | To: |
|---|---|
| **String** | Accept a string value from the user when a build runs. The parameter field appears as a text box in the Configure Parameters dialog.<br>Select the **Required** checkbox to ensure that no empty value is allowed when starting a build. If the parameter is set to required, a default value is mandatory and parameters in automatic builds always have non empty values. Using required parameters reduces build and pipeline failures caused by missing values for parameters. |
| **Password/Private Key** | Accept a password or private key value from the user when a build runs. The parameter field appears as a password box in the Configure Parameters dialog.<br>Select the **Required** checkbox to ensure that no empty value is allowed when starting a build. If the parameter is set to required, a default value is mandatory and parameters in automatic builds always have non empty values. Using required parameters reduces build and pipeline failures caused by missing values for parameters.<br>It's important to note that the password/private key setting isn't a toggle. If you change the selection from password to private key (or private key to password), you'll need to re-enter the password/private key value. |
| **Boolean** | Accept `true` or `false` as input from the user when a build runs. The parameter field appears as a check box in the Configure Parameters dialog. |
| **Choice** | Accept a value from a list of values when a build runs. The parameter field appears as a drop-down list in the Configure Parameters dialog. The first value is the default selected value. |
| **Merge Request** | Accepts string values for the Git repository URL, the Git repository branch name, and the merge request ID as input. The parameter fields appear as a text box in the Configure Parameters dialog.<br>Use this parameter when you want to link an existing job with a merge request. |

5. Enter values, such as name, default value, password/private key, and description.

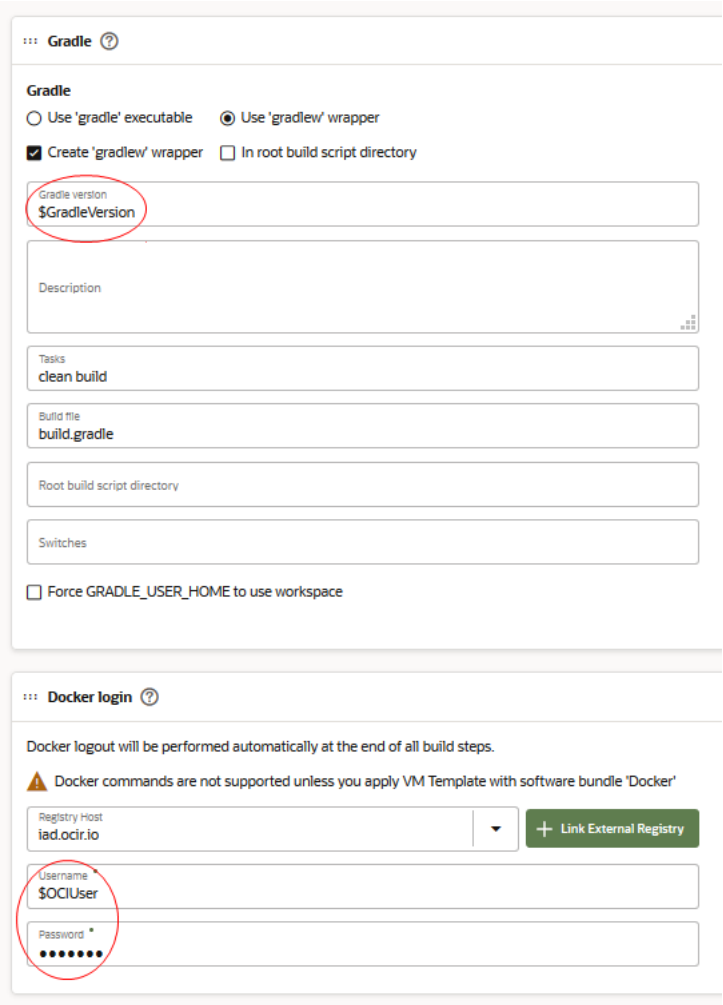
> **Note:**
>
> Parameter names must contain letters, numbers or underscores only. They can't begin with a number and they aren't case-sensitive (the names "job", "JOB", and "Job" are all treated the same).
>
> You can't use hyphens in build parameter names. When the build system encounters a script or a command with a hyphenated build parameter name in a UNIX shell build step, it removes the portion of the name preceding the hyphen. If you try to use a hyphen in a build parameter name in a job, you won't be able to save the job configuration that includes it.
>
> In addition, you shouldn't use an underscore by itself or any of the system or other environmental variable names listed in Reserved Words that Shouldn't Be Used in Build Parameter Names as build parameter names. There could be unintended consequences if you do.

6. Click **Save**.

For example, if you want a job to change the default values for the Gradle version, the OCI username, and the OCI user password when a build runs, in a Build step, create the Choice, String, and Password/Private Key build parameters to accept the values. Notice that the value for the Password/Private Key parameter isn't displayed in the input field.

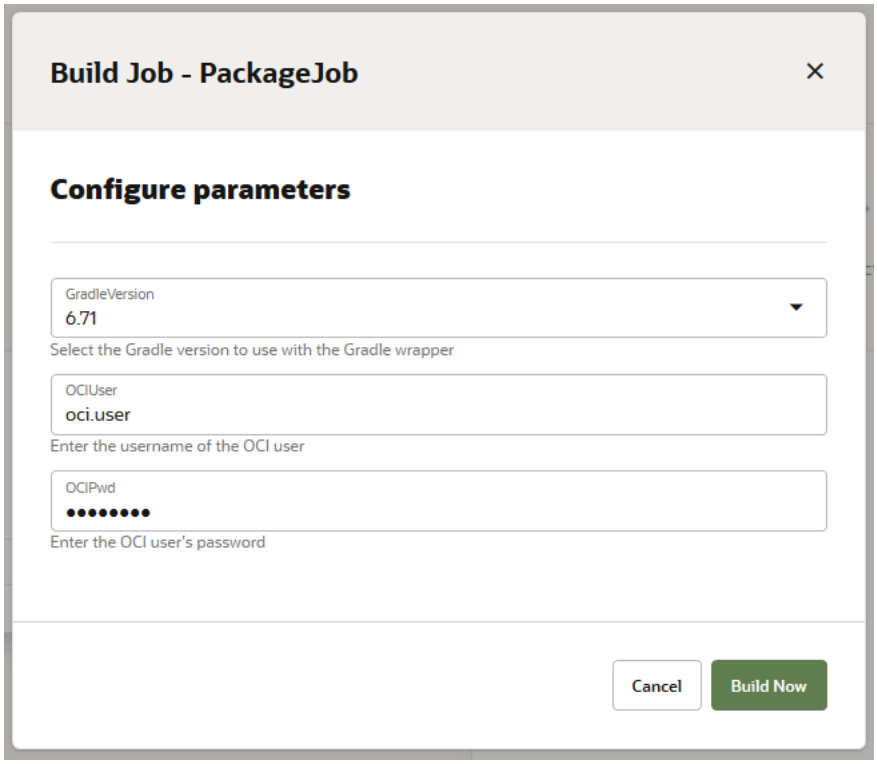



Use the `$BUILD_PARAMETER` format when you're using build parameters. (The `${BUILD_PARAMETER}` format can be used too.) For example, this screen shot shows the Gradle version, OCI username, and OCI password parameters used in the build step fields of a job. Notice that the password/private key parameter's variable name isn't displayed:

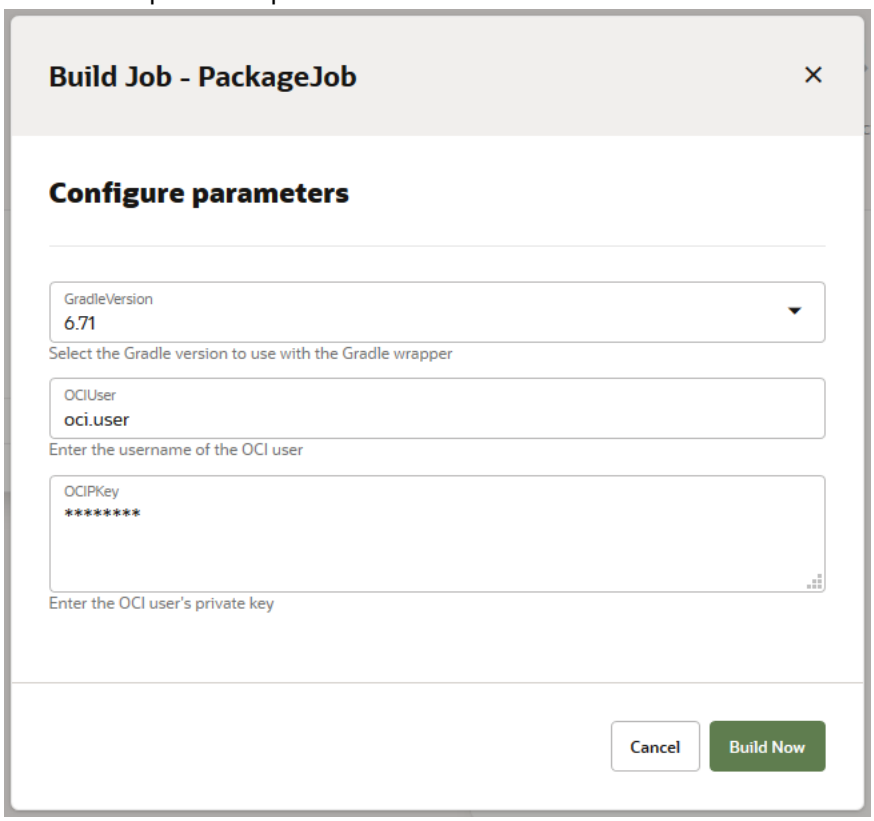



### Configure Parameters Dialog

When a build runs, the Configure Parameters dialog opens where you can enter or change the default values of parameters. All parameter values, except the Password/Private Key parameter's value, are displayed as string in the dialog box, and subsequently in the build log. This screenshot shows the dialog for a job configured to use a password parameter:




This screenshot shows the dialog for a similar job configured to use a private key parameter instead of a password parameter:



Notice the difference between the OCIPkey private key parameter's default value (asterisks) and that for the OCIPwd password parameter's default value (black dots) shown in the previous screenshot.

**Filepaths with Parameter(s) in Pathnames**

In the Job Configuration pages (for example, in the Before Build step's **Target Directory** field in the Copy Artifacts section or in the Git tab's **Local Checkout Directory** field), VB Studio allows and validates a filepath with parameter(s) in the pathname. For example, the `$myDir` and `${myDir2}` parameters that define directories can be used in the pathname, as in `<server>/**$myDir**/**${myDir2}**/<directory>`.

**Merge Request Parameters**

If you selected the **Use for Merge Request** check box while creating the job, GIT_REPO_URL, GIT_REPO_BRANCH, and MERGE_REQ_ID Merge Request parameters are automatically added to accept the Git repository URL, the Git repository branch name, and the merge request ID as input from the merge request, respectively. The GIT_REPO_URL and GIT_REPO_BRANCH variables are automatically set in the **Repository** and **Branch** fields of the **Git** tab.

**Job Parameters in Pipelines**

When a job in a pipeline runs, there is no way to enter or change the default values of the parameters. Job parameters in pipelines exhibit the following implicit behaviors:

- Upstream job parameters are passed to downstream jobs.
  For example, in a pipeline that flows from Job A to Job B to Job C, if parameter P1 is defined in Job A and parameter P2 is defined in Job B, then parameter P1 will be passed to Job B and parameters P1 and P2 will be passed to Job C.
- An upstream job with the same named parameter as a downstream job will overwrite the default value of the named parameter from the downstream job.
  For example, if parameters P1 and P2 are defined in Job A and parameters P2 and P3 are defined in Job B, then the value of parameter P2 from Job A will overwrite the default value of parameter P2 in Job B. If there was a Job C downstream from Job B, then the initial default value of P2 (from Job A) plus the values of P1 and P3 would be passed to Job C.
- When a build of the pipeline runs, the Configure Parameter dialog box displays all parameters of the jobs in the pipeline. Duplicate parameters are displayed once and its value is used by all jobs that use the parameter. The default value of a duplicate parameter comes from the first job in the pipeline where it is defined.
  For example, in a pipeline that flows from Job A to Job B to Job C, if parameter P1 is defined in Job A; parameters P2 and P3 are defined in Job 2; and parameters P1 and P4 are defined in Job C; then when the pipeline runs, it displays parameters P1, P2, P3, and P4 once in the Configure Parameter dialog box though parameter P1 is defined in two jobs. The default value of P1 would come from Job 1 and is passed to subsequent jobs of the pipeline.

  In the pipeline, if the **Auto start when pipeline jobs are built externally** is selected, then the Configure Parameter dialog box isn't displayed when a build of a pipeline's job runs. In the pipeline, the subsequent jobs of the job that trigger the build use the default values of their parameters. If a parameter is duplicate, then the job uses the default value of the first job where the parameter was defined.

  For example, in a pipeline that flows from Job A to Job B to Job C, if parameter P1 is defined in Job A; parameters P2 and P3 are defined in Job B; and parameters P1 and P4 are defined in Job C; then when a build of Job A runs, it passes the default value of P1 to Job B and Job C, and overwrites the default of P1 in Job C. If a build of Job B runs, then the builds use the default values of P2, P3, P1 (defined in Job C) and P4.

### Build Parameters in Shell Build Steps

To learn about how to use build parameters in a Shell build step, see the GNU documentation on Shell Parameter Expansion at `https://www.gnu.org/software/bash/manual/html_node/Shell-Parameter-Expansion.html`.

## Reserved Words that Shouldn't Be Used in Build Parameter Names

A system environment variable shouldn't be used as a parameter name. If you use one of the following system environment variable names, the build might run incorrectly or even fail unexpectedly:

- home
- hostname
- lang
- mail
- path
- pwd
- shell
- term
- user
- username

In addition, you should avoid using the following environment variables, listed alphabetically, that may be used elsewhere, to avoid interfering with the plugin or the process that introduced them:

- _ (underscore)
- ant_home
- build_dir, build_id, build_number
- cvs_rsh
- dcspassbuildinfofeaturecurrentorg, dcspassbuildinfofeaturecurrentproject
- g_broken_filenames, git_repo_branch, git_repo_url, gradle_home
- histcontrol, histsize, http_proxy, http_proxy_host, http_proxy_port, https_proxy, https_proxy_host, https_proxy_port
- isdcspassbuildinfofeatureenabled
- java_home, java_vendor, javacloud_home, javacloud_home_11_1_1_7_1, javacloud_home_11g, javacloud_home_soa, javacloud_home_soa_12_2_1, job_name
- lessopen, logname
- m2_home, merge_req_id, middleware_home, middleware_home_11_1_1_7_1, middleware_home_11g, middleware_home_soa, middleware_home_soa_12_2_1
- no_proxy, no_proxy_alt, node_path
- oracle_home, oracle_home_11_1_1_7_1, oracle_home_11g, oracle_home_soa, oracle_home_soa_12_2_1
- qtdir, qtinc, qtlib
- shlvl, ssh_askpass

- tool_path
- wls_home, wls_home_11_1_1_7_1, wls_home_11g, wls_home_soa, wls_home_soa_12_2_1, workspace

## Use a Named Password/Private Key

A **named password/private key** is a variable that users can use across a project's build job configurations, in any password/private key field in the job configuration, including external Git repositories as well as in SQLcl and Docker configurations.

When the value of the password or private key changes, you can edit and reset it and the new value will be applied to all jobs and configurations where the variable is used. However, if you change the selection from password to private key (or the other way around), you must you re-enter a new value for the password or private key.

Note that the named password/private key is *not* an environment variable. To use a named password/private key as an environment variable, create a **Password/Private Key** build parameter and set it to use the named password/private key.

### Create and Manage Named Passwords/Private Keys

If you're a project owner, you can create, edit, and delete named passwords/private keys:

| Action | How To |
|---|---|
| Create a named password/private key | 1. In the left navigator, click **Project Administration** .<br>2. Click **Builds**.<br>3. Click the **Named Passwords/Private Key** tab.<br>4. Click **+ Create Named Password/Private Key**.<br>5. In **Name**, enter a name for the variable and then do one of the following:<br>• In **Password**, enter the password.<br>• Click the **Private Key** checkbox, then paste the private key into the **Private Key** field.<br>6. Click **Create**.<br>After you create the named password or named private key, share its name with your project users. |

| Action | How To |
|---|---|
| Edit a named password/ private key | 1. In the left navigator, click **Project Administration** ⚙.<br>2. Click **Builds**.<br>3. Click the **Named Passwords/Private Key** tab.<br>4. Click the password/private key name and then click **Edit** ✎.<br>5. Update the password in the **Edit Named Password** dialog or the private key in the**Edit Private Key** dialog.<br>**Note:** You can't change a named password's name.<br>6. Click **Update**. |
| Delete a named password/private key | 1. In the left navigator, click **Project Administration** ⚙.<br>2. Click **Builds**.<br>3. Click the **Named Passwords/Private Key** tab<br>4. Click the named password/private key name, then click **Delete** ⊗.<br>5. In the **Delete Named Password** or the **Delete Private Key** dialog, click **Delete**.<br>After you delete the named password or private key, let your project users know that it's no longer available. |

## Configure a Job to Use a Named Password/Private Key

Here's how you can configure a job that uses a named password/private key:

1. Open the job's configuration page.
2. Click the **Parameters** tab.
3. Click **Add Parameter** and select **Password/Private Key Parameter**.
4. In the **Name** field, enter a name for the parameter, then do one of the following:
   a. If you're defining a password, in the **Default Value** field, enter the Named Password/ Private Key, if previously defined, in the format `#{password_name}`.
   b. If you are defining a private key, select the **Private Key** checkbox and either enter the Named Password/Private Key, if previously defined, in the format `#{pkey_name}` or paste the private key directly into the **Private Key** field, for example:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAzr32xevtOevm1ZKQ3alPWK888...
-----END RSA PRIVATE KEY----
```

   When the named variable for the private key was created, it must have been set as a private key, not a password. The named variable for the private key will then be displayed as `*****`.

5. Click **Save**.

## Access an Oracle Cloud Service Using SSH

You can configure a job to use SSH to access any Oracle Cloud service instances that has SSH enabled, such as Oracle Cloud Infrastructure Compute Classic VMs.

You can configure the job to use any of the following options, or both:

- Create an SSH tunnel to access a process running on a remote system, including an on-premise system, via the SSH port. The SSH tunnel is created at the start of the build job and is destroyed automatically when the job finishes.
- Set up the default `~/.ssh` directory with the provided keys in the build's workspace for use with the command-line tools. The modifications revert after the job finishes.

To connect to the Oracle Cloud service instance, you need IP address of the server, credentials of a user who can connect to the service instance, and local and remote port numbers:

1. Open the job's configuration page.
2. Click the **Before Build** tab.
3. Click **Add Before Build Action** and select **SSH Configuration**.
4. In **Private Key** and **Public Key**, enter the private and the public key of your SSH Private-Public key pair.

   Leave the **Public Key** empty to use the fingerprint.

   Use the ssh-keyscan tool to get the SSH public key from the VM itself. More than one key may be returned and, although you could use any of them, the most commonly used one is the ssh-rsa key.
5. In **Pass Phrase**, enter the passphrase of your SSH Private-Public key pair. Leave it empty if the keys aren't encrypted with a passphrase.

   > **Note:**
   >
   > If you want to access the Oracle Cloud service using a command or a Shell script from the UNIX Shell step, do not use a key protected by a passphrase, or SSH will interactively prompt for a passphrase during the build.

6. In **Host Key**, enter the public key of the SSH server.

   If you're using a command-line SSH tool, note that the host name and the IP address must match. The host name and the IP address can be comma separated. Example: `ssh1.example.com,10.0.0.13 ssh-rss ...`.

   Leave the field empty to skip host verification. For command-line tools, such as `ssh`, add the `-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null` option explicitly to skip host verification.
7. To use an SSH tunnel, select the **Create SSH Tunnel** check box.

   SSH tunnel provides an additional layer of security and can only be set up between trusted hosts. After you select the check box, enter the SSH server details:

   - **Username**: Name of the user who can connect to the SSH server.

- **Password**: Password of the SSH user. Leave the field empty to use the key based authentication.
- **Local Port**: Port number of the client used for local port forwarding.
- **Remote Host**: Name of the remote host, or an interface on the SSH server.
- **Remote Port**: Port number of the remote host or interface.
- **SSH Server**: Name or IP address of the target SSH server.
- **Connect String**: Displays the connect string to be used to set up the SSH tunnel.

8. To use command line tools (such as `ssh`, `scp`, or `sftp`), select the **Setup files in ~/.ssh for command-line ssh tools** check box.

   When a build runs, necessary files with the information that you've provided are created for you in the `known_hosts` file of the `~/.ssh` directory in the build system workspace. The files are removed automatically after the build is complete.

   When a build runs, necessary files with the information that you've provided are created for you in the `known_hosts` file of the `~/.ssh` directory in the build system workspace. The files are removed automatically after the build is complete.

   > **Note:**
   >
   > If you have a build job that uses an SSH proxy to run SSH commands, including scp, host verification will be performed for the proxy server as well as the target server. Host certificates for both servers must be added to the Host Keys field in the SSH Configuration.
   >
   > For example, if a build script contains an scp command such as this one:
   >
   > ```
   > scp -o 'ProxyCommand=ssh  -W %h:%p -p 22 opc@x.x.x.x' target/sample.war
   >     opc@y.y.y.y:/tmp
   > ```
   >
   > The host keys should have two lines, one for proxy server `x.x.x.x` and another for target server `y.y.y.y`
   >
   > Alternatively, you could disable host key verification for one or both of the servers:
   >
   > ```
   > scp -o StrictHostKeyChecking=no -o 'ProxyCommand=ssh   -o
   > StrictHostKeyChecking=no  -W %h:%p -p 22 opc@x.x.x.x' target/sample.war
   > opc@y.y.y.y:/tmp
   > ```

9. Click **Save**.

## Access the Oracle Maven Repository

The Oracle Maven Repository contains artifacts, such as ADF libraries, provided by Oracle. You may require these artifacts to compile, test, package, perform integration testing, or deploy your applications. For more information about the Oracle Maven repository, see `https://maven.oracle.com/doc.html`.

To build your applications and access the Oracle Maven Repository, you configure the job and provide your credentials to access the repository:

1. Open `https://www.oracle.com/webapps/maven/register/license.html` in your web browser, sign in with your Oracle Account credentials, and accept the license agreement.

2. Configure the POM file and add the Oracle Maven Repository details:

   a. Add a `<repository>` element that refers to `https://maven.oracle.com`:

   ```
   <repositories>
       <repository>
           <name>OracleMaven</name>
           <id>maven.oracle.com</id>
           <url>https://maven.oracle.com</url>
       </repository>
   </repositories>
   ```

   b. Depending on your application, you may also want to add the `<pluginRepository>` element and make it refer to `https://maven.oracle.com`:

   ```
   <pluginRepositories>
       <pluginRepository>
           <name>OracleMaven</name>
           <id>maven.oracle.com</id>
           <url>https://maven.oracle.com</url>
       </pluginRepository>
   </pluginRepositories>
   ```

   c. If you are going to use token-based authentication in your project, follow the steps in Set Up Token-Based Authentication to create a token with expiration and permission settings appropriate for your Maven project. (Don't forget to copy the token and paste it in a text file so you can access later.)

   Then add the `<server>` element and replace **`{token}`** with the token you created.

   ```
   <server>
       <id>internalRepo</id>
       <configuration>
           <httpConfiguration>
               <all>
                   <headers>
                       <property>
                           <name>Authorization</name>
                           <value>Bearer {token}</value>
                       </property>
                   </headers>
               </all>
           </httpConfiguration>
       </configuration>
   </server>
   ```

3. Commit the POM file to the project's Git repository.
4. If you're the project owner, set up Oracle Maven Repository connections for your project's team members.

   See Create and Manage Oracle Maven Repository Connections.
5. Create and configure a job to access Oracle Maven Repository.

   See Configure a Job to Connect to the Oracle Maven Repository.

## Create and Manage Oracle Maven Repository Connections

If your project users access the Oracle Maven Repository frequently, you can create a pre-defined connection for them. Project users can then configure a job and use the connection to access the artifacts of the Oracle Maven Repository while running builds.

You must be a project owner to add and manage Oracle Maven Repository connections.

To create, edit, and delete a connection, you'll need the Oracle Technology Network (OTN) Single Sign-On (SSO) credentials of a user who has accepted the Oracle Maven Repository license agreement:

| Action | How To |
|---|---|
| Add an Oracle Maven Repository connection | 1. In the left navigator, click **Project Administration** ⚙.<br>2. Select the **Builds** tile.<br>3. Click the **Maven Connection** tab.<br>4. Click **+ Create Maven Connection**.<br>5. In the Create Maven Connection dialog, in **Connection Name**, enter a unique name.<br>6. In **OTN Username** and **OTN Password**, enter the credentials of a user who has accepted the Oracle Maven Repository license agreement.<br>7. In **Server Id**, if necessary, enter the ID to use for the `<server>` element in the Maven `settings.xml` file or use the default `maven.oracle.com` ID.<br>8. Click **Create**. |
| Edit a connection and change the connection's user credentials or provide another server ID | 1. In the left navigator, click **Project Administration** ⚙.<br>2. Select the **Builds** tile.<br>3. Click the **Maven Connection** tab.<br>4. Click the connection name and then click the **Edit** icon.<br>5. In the Edit Maven Connection dialog box, if necessary, enter the credentials of a user with valid SSO user name.<br>In **Server Id**, if necessary, enter the ID to use for the `<server>` element in the Maven `settings.xml` file. If not provided, the ID defaults to `maven.oracle.com`.<br>6. Click **Update**. |
| Delete the connection | 1. In the left navigator, click **Project Administration** ⚙.<br>2. Select the **Builds** tile.<br>3. Click the **Maven Connection** tab.<br>4. Click the connection name and then click ⊗.<br>5. In the Delete Maven Connection dialog, click **Delete**. |

## Configure a Job to Connect to the Oracle Maven Repository

Here's how you can set up a job using a predefined connection to connect to the Oracle Maven Repository:

1. Open the job's configuration page.
2. Click the **Before Build** tab.
3. Click **Add Before Build Action** and select **Oracle Maven Repository Connection**.
4. From **Use Existing Connection**, select a pre-defined connection. Your project owner has created a connection so that you don't have to worry about setting it up.

If there's no pre-defined connection available or you want set up your own connection, click the toggle button. In **OTN Username** and **OTN Password**, enter the credentials of a user who has accepted the Oracle Maven Repository license agreement.

5. In **Server Id**, if required, enter the ID to use for the `<server>` element of the Maven `settings.xml` file, or use the default `maven.oracle.com` ID.
6. If you're using a custom `settings.xml` file, in **Custom settings.xml**, enter the file's path.
7. Click **Save**.

## Generate a Dependency Vulnerability Analysis Report

You can configure a job to generate a Dependency Vulnerability Analysis (DVA) report for a Maven, Node.js/Javascript, or Gradle application. This report can help you analyze any publicly known vulnerabilities in the application's dependencies.

When a build runs, VB Studio scans the job's POM file (Maven), `package.json` file (Node.js/Javascript), or `build.gradle` file (Gradle) and checks the direct and transitive dependencies against the National Vulnerability Database (NVD). See `https://nvd.nist.gov/` to find more about NVD.

Dependencies in Node.js and Javascript projects are also checked for vulnerabilities against the following sources:

- **NPM**: Data may be retrieved from the NPM Public Advisories, `https://www.npmjs.com/advisories`.
- **RetireJS**: Data may be retrieved from the RetireJS community, `https://retire.js/github.io/retire.js/`.
- **Sonatype OSS Index**: Data may be retrieved from the Sonatype OSS Index. `https://sonatype.ossindex.org`.

For any vulnerabilities found, you can configure the job to mark the build as failed or file an issue. If email notifications have been enabled or if a Slack webhook has been configured, you can be notified about these actions through email or Slack.

To configure a job to scan for security vulnerabilities:

1. Open the job's configuration page.
2. Click the **Before Build** tab.
3. From **Add Before Build Action**, select **Dependency Vulnerability Analyzer**.

   After adding the Dependency Vulnerability Analyzer build action, make sure it's enabled. You can disable the DVA report generation by disabling the build action.
4. In **Threshold at or above**, select the score threshold.

   The scores capture the principal characteristics of a vulnerability and reflect its severity.

> **Note:**
>
> The threshold and confidence settings have different mappings and values, depending on the type of project (Node.js/Javascript, Maven, or Gradle). The Common Vulnerability Scoring System (CVSS) score is for vulnerabilities from the NVD database only. Vulnerabilities in Node.js and Javascript projects can come from sources (NPM, RetireJS, Sonatype OSS Index) in addition to those that come from NVD.
>
> For more information about how CVSS scores are calculated, see `https://nvd.nist.gov/vuln-metrics/cvss`.

This table explains how the levels (Low, Medium, High) are defined for each vulnerability source. The Analyzer reports vulnerabilities when the value for the level you choose is met or exceeded.

| Source | Project | Low | Medium | High |
|---|---|---|---|---|
| NVD | Maven, NodeJS, Gradle | Score range 0.0-3.9 | Score range 4.0-6.9 | Score range 7.0-10.0 |
| NPM | NodeJS | Low | Moderate | High or Critical |
| RetireJS | NodeJS | Low | Medium | High or Critical |
| Sonatype OSS Index | NodeJS | Score range 0.0-3.9 | Score range 4.0-6.9 | Score range 7.0-10.0 |

For example, if you select **Medium**, any vulnerability with a CVSS score of 4.0 or above (and a Moderate NPM level, a Medium RetireJS level, or a Sonatype OSS Index score greater than 4.0 for a Node.js project) is detected and reported.

5. In **CPE Confidence**, select the confidence rating the DVA has for the identified Common Platform Enumeration (CPE).

   CPE is a structured naming scheme for describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. To find more about CPE, see `https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe/`.

   The CPE Confidence rating helps you filter the Common Vulnerabilities and Exposure (CVE) identifiers based on the confidence level. CVE is a list of common identifiers for publicly known cybersecurity vulnerabilities. To find more about CVE, see `https://cve.mitre.org/`.
   For example, select **Medium** to filter out the low confidence CVE identifiers from the report.

   > **Note:**
   >
   > CPE confidence levels are not supported for NPM, RetireJS, and Sonatype OSS Index sources. For these sources, the Analyzer reports all vulnerabilities, regardless of the level you choose.

6. To fail the build if a vulnerability is detected, select the **Fail Build** check box.
7. To automatically file an issue for every build file where a vulnerability is detected, select the **Create issue for every affected build file** check box.

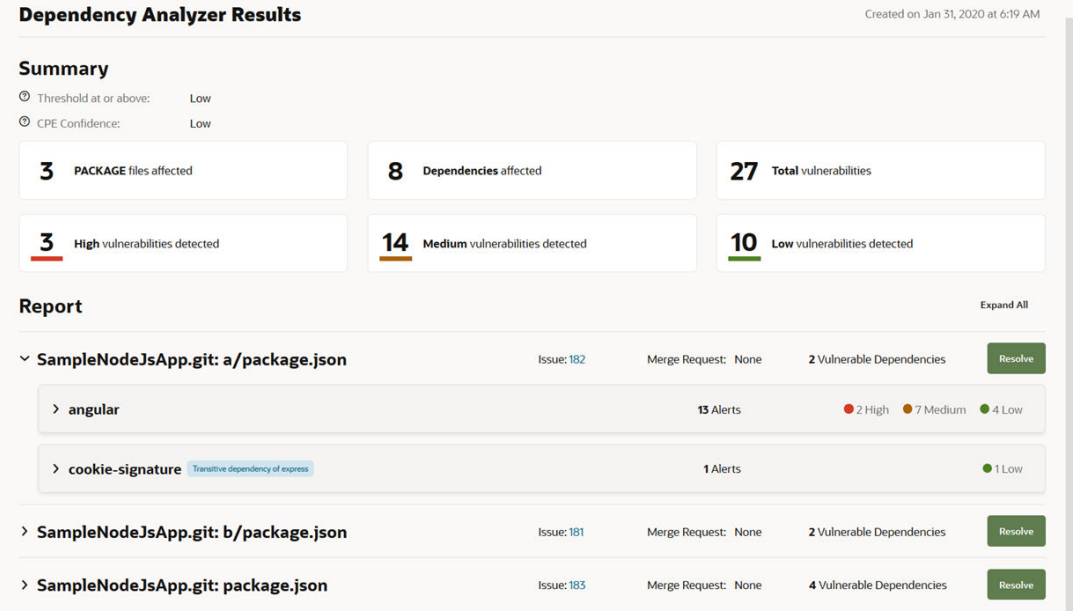
In **Product** and **Component**, select the issue's product and component.

8. Click the **Steps** tab.
9. Add a **Unix Shell** step (or appropriate step, such as a **Maven** step to build the POM file).

   For example, if you add a UNIX Shell step, enter the following command to build the `pom.xml` file in the `app_dir` directory in the job's Git repository:
   ```
   mvn -install -fae -f app_dir/pom.xml -X
   ```
10. Click **Save**.
11. Run a build of the job.

    To trigger the build manually, in the Job Details page, click **Build Now**.
12. After the build is complete and a vulnerability is detected, click **Vulnerabilities** ⚠ to view the vulnerabilities report. If no vulnerabilities were detected, **Vulnerabilities** will be disabled.
13. On the Dependency Analyzer Summary page, review the affected files, dependencies, and detected vulnerabilities.

    Expand the **Report** section to view the files of your application where vulnerabilities are found (in the POM file, in this example):




After the DVA report is generated, expand each file in the **Report** section to view these details:

- Issue ID, if the **Create issue for every affected file** check box was selected. Click the issue link to open it.
  You can also open the **Project Home** page and check the recent activities feed about the issue's create notification. You should see a message that an issue was created, such as **System created Defect 2: Vulnerabilities in -MavenJavaApp**. If an issue was previously created for the vulnerability, a comment will be added to the issue and a message like **System commented Defect 2: Vulnerabilities in - MavenJavaApp** will be added to the activities feed.
- Merge request ID, if the **Resolve** button was clicked to resolve the vulnerabilities. Click the merge request link to open it.
- Number of vulnerabilities

- Name of each dependency where a vulnerability is found
- Each dependency's type (direct or transitive)
  A transitive dependency displays a **Transitive** label next to the name. A direct dependency displays no label.
- Number of alerts and alert categories of vulnerabilities (High, Medium, or Low)
- Expand each dependency to view its vulnerabilities
  To mute a vulnerability's alerts, expand the vulnerability in the **Report** section, and click **Mute** in **Alerts**. In the Mute Vulnerability dialog box, review the details, and click **Mute**. The muted vulnerability won't be reported during the next run and it will not cause the build to fail. It will simply be included in the report as a muted vulnerability that should be used only for reference or to be unmuted and dealt with at some future time.

  Muted vulnerabilities will only show up in a report for the latest build, not in reports for any previous builds.

To fix a reported vulnerability, use **Resolve** and the dropdown menu in the analysis tool to change the dependency's version to one that doesn't have the vulnerability.

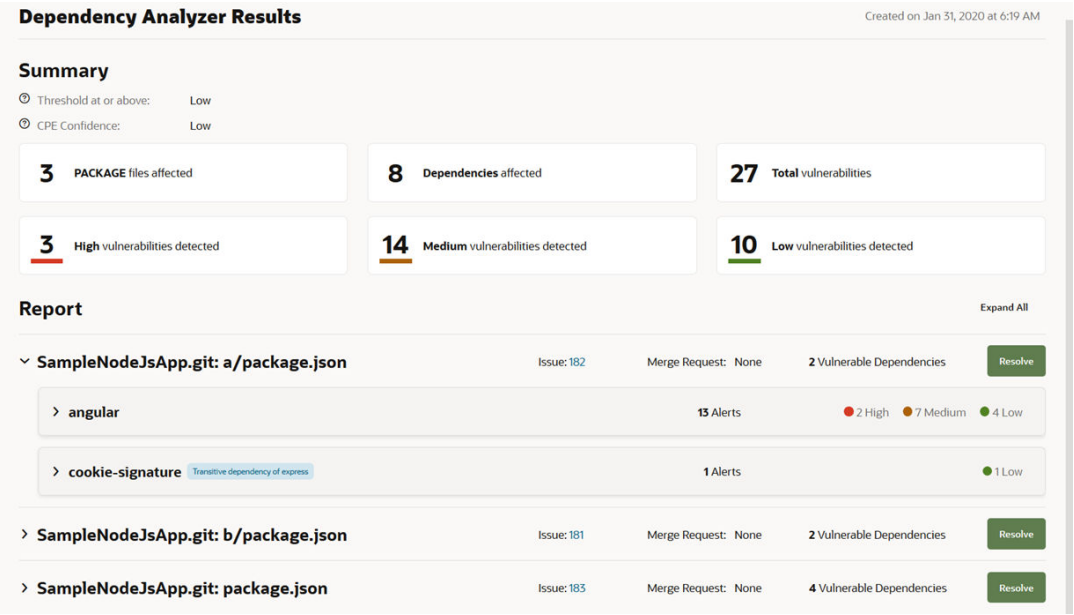
## Resolve Reported Vulnerabilities Automatically

After the Dependency Vulnerability Analysis (DVA) report for the Maven, Node.js, Javascript, or Gradle application has been generated, review the report to identify the vulnerabilities in the flagged files, and click the **Resolve** button to resolve them.

The **Resolve** button simplifies and automates the process for resolving vulnerabilities found in the direct as well as transitive dependencies of the application's build file. The **Resolve** button isn't available in the DVA reports of older builds of the job. It is only available in the latest build of the job. The **Resolve** button is also disabled if a `package.json` file in a Node.js or Javascript application has vulnerabilities in transitive dependencies only. Transitive dependencies in Node.js and Javascript applications must be resolved manually, by editing the direct dependencies in the `package.json` file and rerunning the analyzer.

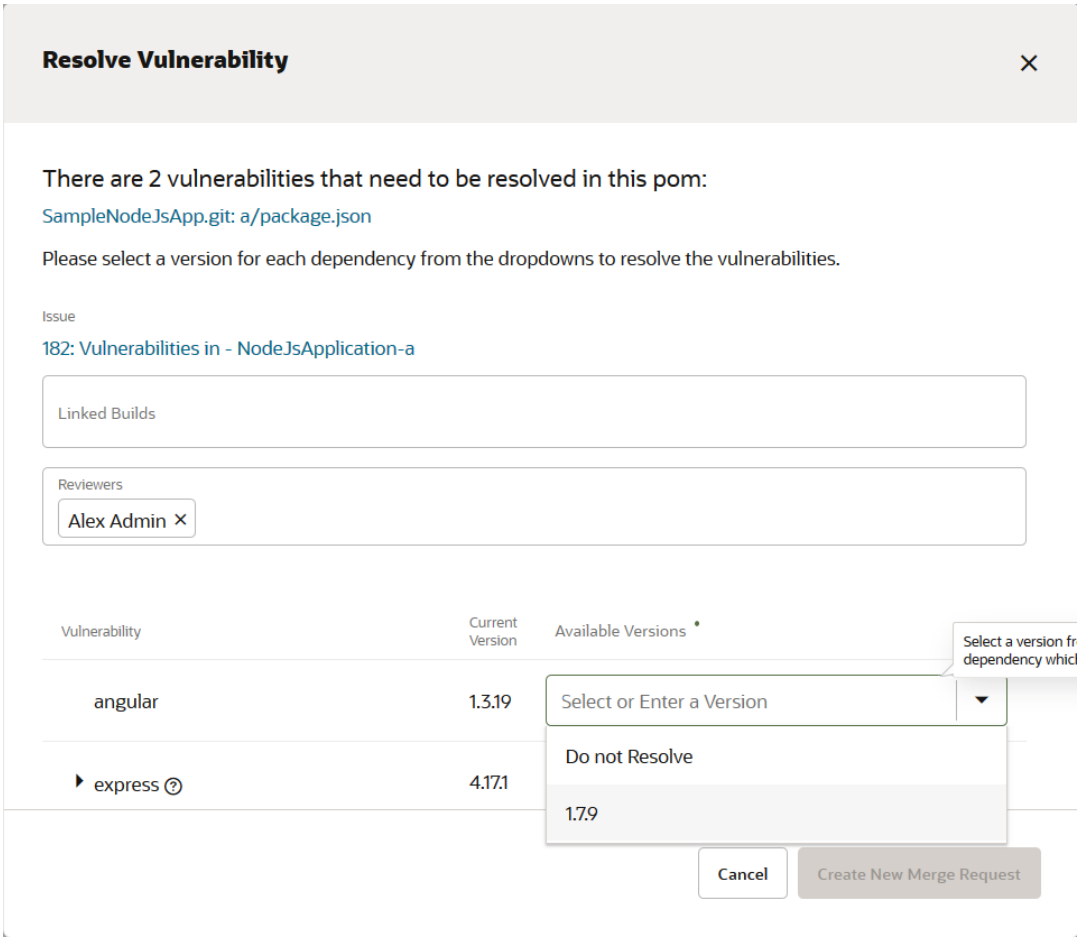
Click the **Resolve** button to resolve any direct and transitive dependencies that were found:

1. In the **Report** section of the vulnerability analysis report, expand the affected build file (POM is shown):



2. Click **Resolve**.
   If a merge request exists, you can cancel the dialog and use it or continue to create another merge request.
3. In the Resolve Vulnerability dialog box, review the reported vulnerabilities.
4. If an issue was created when the report was generated, its ID is displayed. If no issue was created, select the **Create issue to track this resolution** check box to create it.
   In **Linked Builds**, add an existing build to link it to the merge request.

   In **Reviewers**, add team members to review the merge request:

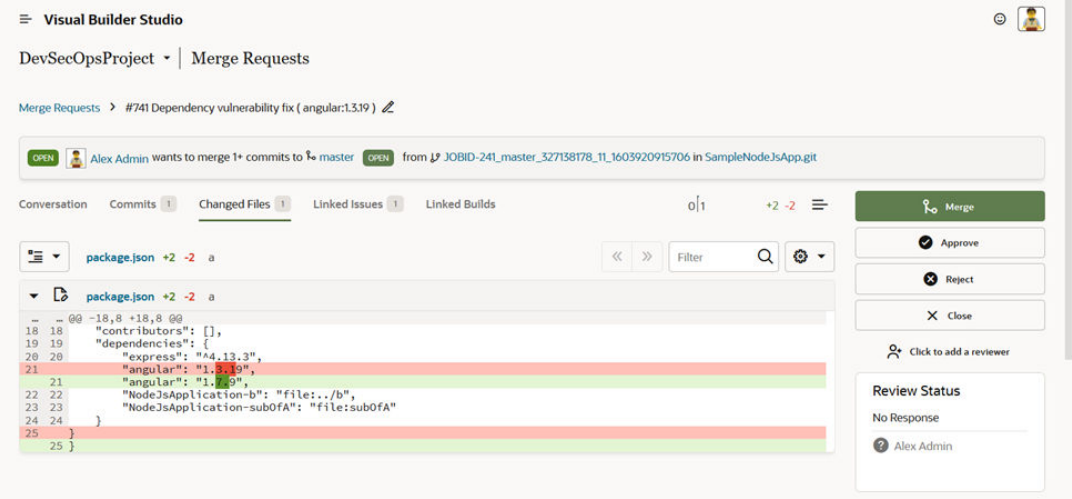



5. For each vulnerability, in **Available Versions**, select a version of the direct dependency or dependency with transitive dependencies that doesn't have the reported vulnerability.
   If you don't want to resolve the dependency or no versions are available, select **Do Not Resolve**.
6. Click **Create New Merge Request**.
   When you click the button, VB Studio does the following:
   a. Creates a merge request with details about the vulnerabilities found.
   b. Creates a branch with the job's Git repository branch as the base branch, and then sets it as the review branch of the merge request.
   c. Sets the job's Git repository branch as the target branch of the merge request.
   d. Updates the review branch's application build file to use the specified versions of the dependencies.

For example, if the job that generated the vulnerability report uses the `JavaMavenApp` Git repository and its `release1.1` branch, then a new branch is created in `JavaMavenApp` using `release1.1` as the base branch and is used as the review branch of the merge request. The `release1.1` branch is used as the target branch.

If a merge request with same review and target branches was created in an older build of the job, VB Studio uses the same merge request to merge the application build file updates.

7. Click the merge request link to open it in another tab or window of the browser, and click **OK**.
8. In the Merge Request, review the details of the vulnerabilities in the **Conversation** tab and the application build file changes (POM is shown) in the **Changed Files** tab:




9. If you've invited other reviewers, wait for their feedback.
10. If you've linked a build job to the merge request, in the **Linked Builds** tab, run a build and verify its stability.
11. When you're ready to merge the application build file updates, click **Merge**.
12. In the Merge dialog box, to delete the review branch, select the **Delete branch** check box. To resolve linked issues, select the **Resolve linked issues** check box and the check boxes of issues you want to resolve.
13. Click **Create a merge request**.
14. Run a build of the job that reported dependency vulnerabilities and verify that the application build file's update has fixed the vulnerability.
If a vulnerability is still found, repeat the preceding steps to create another merge request after selecting a different dependency version.

## Move Oracle Integration Projects, Artifacts, Packages, and Lookups Between Instances

When you want to share your code between different Oracle Integration instances, you can use VB Studio's CI/CD capabilities to configure build jobs that export and import Integration projects, Integration artifacts (known as Integrations), packages (collections of Integrations), and lookups from one Oracle Integration instance to another. This capability is useful to

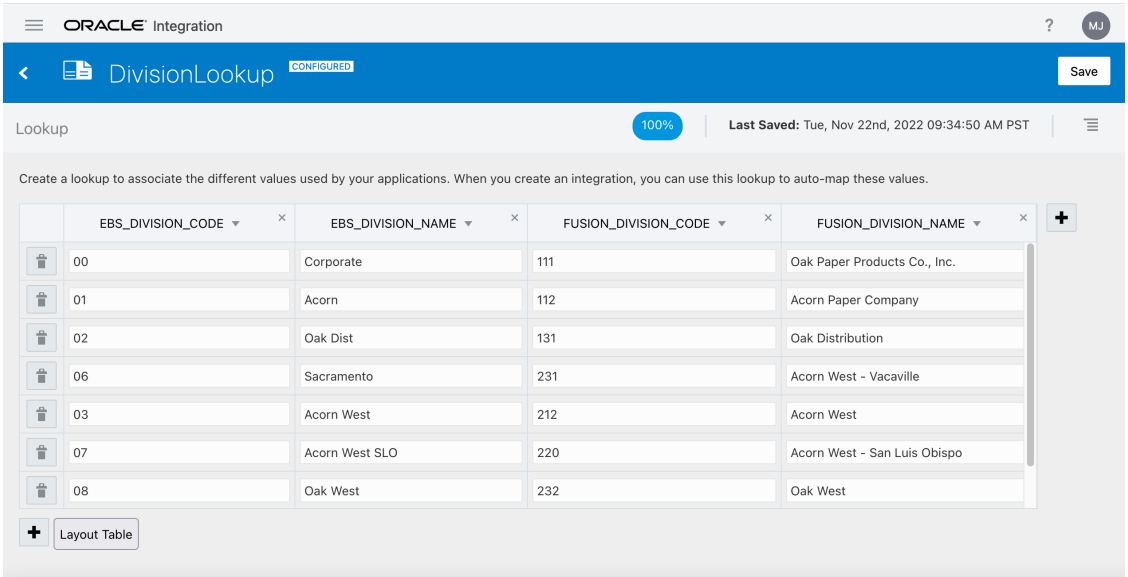
promote your code from lower environments to higher ones, typically from a development to a test and finally to a production environment.

Oracle Integration projects can contain integrations, lookups, connections, and JavaScript libraries.

Integrations are connections to applications with which you want to share data and are created from the Oracle Integration user interface. Each integration includes dependent artifacts such as lookup tables, JavaScript libraries, and connection types. Active integrations in Oracle Integration instances have connections with configured endpoints and credentials. An exported integration archive doesn't include those connection endpoints or credentials for security reasons.

Integrations can be grouped into collections in a package so, when you import or export the package to or from Oracle Integration, all integrations in that package are imported or exported.

Lookups are reusable tables that map the different terms used to describe the same item across your applications. For example, one application uses a specific set of codes to describe countries, while another uses a different set of codes to describe the same countries. You can use lookups for items like mapping gender codes, nationality codes, currency codes, or any type of information your applications share with each other but represent differently. The lookup below shows this for EBS division names and codes as well as for Fusion division names and codes.




To share your code between different Oracle Integration instances, you'll need to export and then import individual integrations, packaged integrations, and lookups from your source environment to the target environment—a task that VB Studio can automate for you. You can set up export and import jobs to move an Integration archive (IAR file) or package (PAR file) from one Oracle Integration instance to another and you can do this with standalone build jobs or as part of a build pipeline.

## Manage Integrations

These are the steps you'll need to use to import and export Integration artifacts via build jobs in VB Studio:

| Step | Description | See this topic |
|---|---|---|
| Configure a build job to export Integration artifacts from an Oracle Integration instance. | Creates and executes a job to<br>• Export an Integration archive (IAR file) from a source Oracle Integration instance and store it in a VB Studio repository.<br>• Allow export integrations with asserter recordings. Oracle's asserter testing framework is used to record tests of integration instances and replay them to reproduce potential issues. See Test Integration Instances for more information. | Configure a Job to Export an Integration |
| Configure a build job to import Integration artifacts to an Oracle Integration instance. | Creates and executes a job to:<br>• Reference a previously exported Integration archive (IAR file) and import it to another Oracle Integration instance.<br>• Optionally, activate the Integration, after which you can also contribute integration mappings to Oracle Recommends, enable asserter recordings, and enable tracing. If you enable tracing, you can also include the payload.<br>• Optionally, include asserter recordings, if there are any. | Configure a Job to Import an Integration |
| Configure a standalone build job to activate a new integration or reactivate an existing one on the target instance after importing artifacts. | Creates and executes a job to: | Configure a Job to Activate an Integration |
| Optional: Configure a build job to delete an Integration artifact that's no longer needed or one that causes a conflict. | Creates and executes a job to delete an Integration archive (IAR file) that exists on a particular Oracle Integration instance. | Configure a Job to Delete an Integration |

## Configure a Job to Export an Integration

You can create and execute a job that exports an Integration archive (IAR file) from a source Oracle Integration instance and stores it. You'll need to copy the artifact (IAR) from your export job into your import build job. Optionally, you can set up the job to add recordings to the exported artifact.

To configure a build job that exports an Integration artifact from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.

5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Export Integration**.
9. In **Source Instance** , select the Oracle Integration instance from which you'll be performing the export operation.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.

    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
11. In **Identifier**, provide the identifier for the integration as defined in the Oracle Integration service instance.

    You can also click **Search** and open the **Search Integrations** window to get a list of integrations in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "COUNTRYINFO", a list of Integrations that contain "COUNTRYINFO" will be displayed.
12. In **Version**, provide the version for the integration as defined in the Oracle Integration service instance.

    If you selected an integration by using the **Search Integrations** dialog, the **Version** and **Identifier** information will be filled in automatically.

    > **Tip:**
    >
    > Alternatively, you could specify a parameter, such as `$version`, here which, when combined with the example identifier in the previous step, would result in an output filename of `COUNTRYINFO_${version}.iar`. You'll need to copy the generated filename or enter `COUNTRYINFO_${version}.iar` when you import the Integration archive.
    > See Use Build Parameters for information about adding build parameters.
13. (Optional) The **Include asserter recordings in exported file** checkbox isn't selected by default. If you want to include recordings in the archive, select it.
14. Select the **After Build** tab and add **Artifact Archiver** as an **After Build Action**.

15. In **Files to archive**, ensure that the IAR file in the **Export Integration** build step's **Archive filename** is shown.
16. Click **Save**.

## Configure a Job to Import an Integration

You can create and execute a job that references a previously exported Integration archive (IAR file) and import it to another Oracle Integration instance. Optionally, you can set up the job to activate the Integration after importing it.

To configure a build job that imports a previously exported Integration artifact from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. From the **Before Build** tab select **Before Build Action** and then select **Copy Artifacts**.
   a. In **From job**, enter the previous Integration Export job you want to import.
   b. In **Which build**, select **Last successful build**.
8. Click the **Steps** tab.
9. From **Add Step**, select **Oracle Integration**, and then select **Import Integration**.
10. In **Import to Target Instance** , select the environment that points to the Oracle Integration instance to which the archive will be imported.
11. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
12. In **Import Archive Filename**, enter the name and version number of the archive file (with .iar extension) that contains the integration to import.

    You could enter the name of the same file that was created in the export. Or, if you specified the identifier, such as `COUNTRYINFO`, and used a parameter, such as `$version`, for the version to generate the `COUNTRYINFO_${version}.iar` output filename in the export operation, you'd enter the identifier and `${version}` for the version here, as in

`COUNTRYINFO_${version}.iar`. You could also copy and use the generated archive name from the import operation.

See Use Build Parameters for information about adding build parameters.

13. In the Import Method section:
    - Select the **Add** checkbox to import an integration with a new ID/version.
    - Select the **Replace** checkbox to import a previously imported integration.
    - Select the **Automatic** checkbox to either add or replace, depending on whether this integration already exists in the target instance.
14. If you chose **Replace** or **Automatic** in the previous step, in the Deactivation section select the **Deactivate existing integration when replacing it** checkbox to deactivate the current integration on the target instance before replacing it.

    If the integration to be replaced has been activated, the import will fail unless you choose this option.
15. (Optional) In the Activation section, check the **Activate new integration on the target instance after importing** checkbox to activate the imported integration in the Oracle Integration service instance.

    If you select **Activate new integration on the target instance after importing**, you can optionally contribute integration mappings to Oracle Recommends, enable asserter recording, and enable tracing. If **Enable tracing** is selected, you also select the **Include payload** option.
16. (Optional) In the Recordings section, the **Include asserter recordings (if any)** checkbox is selected by default. Leave it selected to import any recordings that were written to the exported Integration archive. To ignore the exported recordings, deselect the checkbox.
17. Click **Save**.

## Configure a Job to Activate an Integration

You might want to configure a standalone build job to deactivate and reactivate an OIC Integration. You might want to add this step after performing an Update Connection step to configure connection endpoints and credentials after importing an integration. The modified integration needs to be activated (or reactivated) after the connections(s) have been configured.

1. **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.
   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Activate Integration**.
9. In **Target Instance** , select the environment that points to the Oracle Integration instance with the Integration artifact(s) you want to deactivate and activate.

10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.

    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).

    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.

11. In **Identifier**, provide the identifier for the integration as defined in the Oracle Integration service instance.

    You can also click **Search** and open the **Search Integrations** window to get a list of integrations in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "Hello", a list of Integrations that contain "Hello" will be displayed.

    If you selected an integration by using the **Search Integrations** dialog, the **Version** and **Identifier** information will be filled in automatically.

12. In **Version**, provide the version for the integration as defined in the Oracle Integration service instance.

13. In the Deactivation section, select the **Deactivate then activate integration** checkbox to deactivate the current integration on the target instance before activating it.

    > **Note:**
    >
    > If the integration has already been activated, the activation will fail unless you choose this option.

    If you select **Deactivate then activate integration** on the target instance after importing, you can optionally contribute integration mappings to Oracle Recommends.

14. In the Oracle Asserter section, select the **Enable asserter recording** checkbox to allow Oracle's asserter testing framework to test the integration instance and records it, so you can replay it to reproduce any potential issues.

15. In the Tracing section, select the **Enable tracing** checkbox to display tracing Integration activity in the Oracle Integration Activity Stream.

16. Click **Save**.

## Configure a Job to Delete an Integration

You might want to configure a build job to delete Integration artifacts from an Oracle Integration instance, especially when an existing artifact could cause a conflict.

To configure a build job that deletes Integration artifacts from an Oracle Integration instance:

1. In the left navigator, click **Builds** .

2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Delete Integration**.
9. In **Delete from Target Instance** , select the environment that points to the Oracle Integration instance from which you want the Integration artifact(s) removed.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
11. In **Identifier**, provide the identifier for the integration as defined in the Oracle Integration service instance.

    You can also click **Search** and open the **Search Integrations** window to get a list of integrations in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "Hello", a list of Integrations that contain "Hello" will be displayed.

    If you selected an integration by using the **Search Integrations** dialog, the **Version** and **Identifier** information will be filled in automatically.
12. In **Version**, provide the version for the integration as defined in the Oracle Integration service instance.
13. Click **Save**.

## Manage Oracle Integration Projects

These are the steps you'll need to use to import and export Oracle Integration projects using build jobs in VB Studio:

| Step | Description | See this topic |
|---|---|---|
| Configure a build job to export an Integration project from an Oracle Integration instance. | Creates and executes a job to export an Integration project from an Oracle Integration instance. | Configure a Job to Export an Integration Project |

| Step | Description | See this topic |
|---|---|---|
| Configure a build job to import an Integration project to an Oracle Integration instance. | Creates and executes a job to import an Integration project to an Oracle Integration instance. | Configure a Job to Import an Oracle Integration Project |
| Optional: Configure a build job to delete an Integration project that's no longer needed. | Creates and executes a job to delete an Integration project on an Oracle Integration instance. | Configure a Job to Delete an Integration Project |

## Configure a Job to Export an Integration Project

You can create and execute a job that exports an Integration project from a source Oracle Integration instance to a target Oracle Integration instance.

To configure a build job that exports an Integration project from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Export Project**.
9. In **Source Instance** , select the Oracle Integration instance from which you'll be performing the export operation.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
11. In **Project Id**, provide the identifier of the project you'd like to export from the integration as defined in the source Oracle Integration service instance.

    You can also click **Search** and open the **Search Projects** dialog to get a list of integration projects in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "COUNTRYINFO", a list of integration projects that contain "COUNTRYINFO" will be displayed.

If you selected an integration by using the **Search Projects** dialog, the Project Id will be filled in automatically.

12. The **Archive Filename** is generated based on the **Project Id**.
13. Click **Save**.

## Configure a Job to Import an Oracle Integration Project

You can create and execute a job that imports an Oracle Integration project from one Oracle Integration instance to another Oracle Integration instance.

To configure a build job that imports an Oracle Integration project from another Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. From the **Before Build** tab select **Before Build Action** and then select **Copy Artifacts**.
   a. In **From job**, enter the previous Integration Export job you want to import.
   b. In **Which build**, select **Last successful build**.
8. Click the **Steps** tab.
9. From **Add Step**, select **Oracle Integration**, and then select **Import Project**.
10. In **Target Instance** , select the Oracle Integration instance to which the project will be imported.
11. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
12. In **Import Archive Filename**, enter the name and version number of the archive file (with .car extension) that contains the integration project to import.
13. Click **Save**.

## Configure a Job to Delete an Integration Project

You may want to configure a build job to delete an Integration project from an Oracle Integration instance.

To configure a build job that deletes an Integration project from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Delete Project**.
9. In **Target Instance**, select the environment that points to the Oracle Integration instance from which you want the Integration project removed.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
11. In **Project Id**, provide the identifier for the integration project as defined in the Oracle Integration service instance.

    You can also click **Search** and open the **Search Projects** dialog to get a list of integration projects in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "Hello", a list of Integration projects that contain "Hello" will be displayed.

    If you selected an integration by using the **Search Projects** dialog, the Project Id will be filled in automatically.
12. Click **Save**.

## Manage Oracle Integration Packages

These are the steps you'll need to use to import and export Oracle Integration packages using build jobs in VB Studio:

| Step | Description | See this topic |
|---|---|---|
| Configure a build job to export a package of Integration artifacts from an Oracle Integration instance. | Creates and executes a job to<br>• Export a package (PAR file) of integrations from a source Oracle Integration instance and store it in a VB Studio repository.<br>• Allow exported packages with integrations that have asserter recordings. | Configure a Job to Export an Integration Package |
| Configure a build job to import a package of Integration artifacts to an Oracle Integration instance. | Creates and executes a job to:<br>• Reference a previously exported package (PAR file) of integrations and import it to another Oracle Integration instance.<br>• Choose an import method (add a new package; replace an existing package; or automatically add or replace the package, depending on whether it already exists on the target instance or not).<br>• Deactivate any current integrations on the target instance before importing and reactivate after the import.<br>• Include asserter recordings, if any were exported. | Configure a Job to Import an Oracle Integration Package |
| Optional: Configure a build job to delete a package of Integration artifacts that's no longer needed. | Creates and executes a job to delete a package (PAR file) of integrations on a particular Oracle Integration instance. By default, these integrations will be automatically deactivated before they are deleted. | Configure a Job to Delete an Integration Package |

## Configure a Job to Export an Integration Package

You can create and execute a job that exports an Integration package (PAR file) of integrations from a source Oracle Integration instance and stores it in a VB Studio repository. Optionally, you can set up the job so the export operation adds artifacts with asserter recordings to the artifact.

To configure a build job that exports an Integration package from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

The job configuration page is displayed.

7. Optionally, click the **Parameters** tab and define any parameters, like `$Username` (using a string parameter) and `$Password` (using a password parameter), for example, that you'll use in this build step.
8. Click the **Steps** tab.
9. From **Add Step**, select **Oracle Integration**, and then select **Export Package**.
10. In **Export from Source Instance** , select the Oracle Integration instance from which you'll be performing the export operation.
11. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
12. In **Package Name**, provide the name for the package as defined in the Oracle Integration service instance.

    You can also click **Search** and open the **Search Integrations** window to get a list of packages in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "COUNTRYINFO", a list of packages that contain "COUNTRYINFO" will be displayed.
13. (Optional) Select the **Include asserter recordings in exported file** checkbox if you want to include integrations with asserter recordings in the package.
14. Click **Copy** to copy the PAR file name, You'll use it in other steps.
15. Select the **After Build** tab and add **Artifact Archiver** as an **After Build Action**.
16. In **Files to archive**, paste the PAR file in the **Export Package** build step's **Archive filename**.
17. Click **Save**.

## Configure a Job to Import an Oracle Integration Package

You can create and execute a job that references a previously exported Integration package (PAR file) and import it to another Oracle Integration instance. Optionally, you can set up the job to include integrations with asserter recordings, if any were written to the exported package.

To configure a build job that imports a previously exported package from an Oracle Integration instance:

1. In the left navigator, click **Builds** .

2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. From the **Before Build** tab select **Before Build Action** and then select **Copy Artifacts**.
   a. In **From job**, enter the previous Package Export job you want to import.
   b. In **Which build**, select **Last successful build**.
8. Click the **Steps** tab.
9. From **Add Step**, select **Oracle Integration**, and then select **Import Package**.
10. In **Import to Target Instance** , select the environment that points to the Oracle Integration instance to which the package will be imported.
11. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
12. In **Import Archive Filename**, enter the name of the file (with .par extension) that contains the package of integrations to import.

    You could enter the name of the archive file that was created in the export package job or the name of a previously exported archive.
13. In the Import Method section, select one of the following:
    - Choose **Add** to import a package with a new name.
    - Choose **Replace** to import a previously imported package.
    - Choose **Automatic** to either add or replace the package, depending on whether it already exists on the target instance.
14. If you've chosen **Replace** or **Automatic** in the previous step, select the **Deactivate existing integrations when replacing the package** checkbox in the Deactivate section to deactivate any current integrations on the target instance before replacing the package.

> **Note:**
>
> If any integration to be replaced has been activated, the import will fail unless you choose this option.

15. After replacing the package on the target instance, select the **Reactivate existing integrations after replacing the package** checkbox in the Reactivate section to reactivate any integrations that were deactivated for the import.
16. (Optional) Include asserter recordings, if any were written to the exported package.

    Oracle's asserter testing framework tests integration instances and records them, which you can replay to reproduce any potential issues.
17. Click **Save**.

## Configure a Job to Delete an Integration Package

You might want to configure a build job to delete an Integration package from an Oracle Integration instance. This action will delete the package and all integrations included in that package.

To configure a build job that deletes an Integration package (or packages) from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Delete Package**.
9. In **Delete from Target Instance** , select the environment that points to the Oracle Integration instance from which you want the package removed.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.

11. In **Package Name**, enter the name for the package as defined in the Oracle Integration service instance. If you copied the PAR file name in the Export Package build step, you can paste that name here, but you'll need to remove the .par file extension.

    You can also click **Search** and open the **Search Packages** window to get a list of packages in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field. For example, if you enter "Hello", a list of Integrations that contain "Hello" will be displayed.

    If you selected a package by using the **Search Packages** dialog, the **Package Name** will be filled in automatically.

12. Optionally, in **Deactivation**, select the **Automatically deactivate integrations in the package** checkbox.

    You won't be able to delete the package if any integrations in it are active.

13. Click **Save**.

## Manage Integration Lookups

These are the steps you'll need to use to import and export Integration lookups via build jobs in VB Studio:

| Step | Description | See this topic |
|---|---|---|
| Configure a build job to export Integration lookups from an Oracle Integration instance. | Creates and executes a job to<br>• Export an OIC lookup from a source Oracle Integration instance and store it in a VB Studio repository.<br>• In the source Integration instance, identify the name of the lookup you want to export.<br>• See the archive that contains the exported lookup. | Configure a Job to Export an Integration Lookup |
| Configure a job (or build step) to import an Oracle Integration lookup to a target Oracle Integration instance. | Creates and executes a job to:<br>• Reference a previously exported archive containing the Oracle Integration lookup to import.<br>• Choose how to import the lookup: import with a new name, replace a previously imported lookup, or automatically add or replace the lookup, depending on whether it already exists on the target instance. | Configure a Job to Import an Integration Lookup |
| Optional: Configure a job (or build step) to delete an Integration lookup that's no longer needed or one that causes a conflict. | Creates and executes a job to delete an OIC lookup on a particular Oracle Integration instance. | Configure a Job to Delete an Integration Lookup |

## Configure a Job to Export an Integration Lookup

You can create and use a build step to export an Integration lookup and use an **After Build** step to archive the exported file(s) to Object Storage. Later, you can import the archive containing the exported lookup(s) to a different target Oracle Integration instance.

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.
   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Export Lookup**.
9. In **Source Instance** , select the environment that points to the Oracle Integration instance from which you want to export the lookup(s).
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
11. In **Lookup Name**, enter or search for the name of the lookup you want to export from the source Oracle Integration instance.

    To search for the lookup name, click . The Search Lookups dialog is displayed. Press **Enter** in the **Search** field to see the list of all lookups on your target instance (if you used Basic authentication, you'll need to specify credentials to see this list). Select the lookup from the list. Instead of selecting the lookup from a list, you can type in a lookup name for the specific lookup, typing in either the full name, or part of it, then selecting it from the narrowed list of choices. The drop-down list initially shows all the instance's lookups available to you.

    The Archive Filename shows the archive file that'll contain the exported lookup(s). Copy the filename (`DivisionLookup.csv` , for example). You'll use it in the next step.
12. In the **After Build** tab, click **Add After Build Action** and select **Artifact Archiver**.
13. Paste the filename you copied in step 11 into the **Files to archive** field.
14. Click **Save**.

## Configure a Job to Import an Integration Lookup

You can create and use a build step to import an Integration lookup exported from another Oracle Integration instance.

1. In the left navigator, click **Builds** .

2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.
   The job configuration page is displayed.
7. In the **Before Build** tab, click **Add Before Build Action** , and select **Copy Artifacts**.
8. In the **From job** field, select your Import Lookup build job or enter its name.
9. Click the **Steps** tab.
10. From **Add Step**, select **Oracle Integration**, and then select **Import Lookup**.
11. In **Target Instance** , select the environment that points to the Oracle Integration instance into which you want to import the lookup(s).
12. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
13. In **Import Archive Filename**, enter the name of the archive file (with .csv extension) containing the lookup to import.
14. In **Import method**, select how to import the lookup:
    - Choose **Add** to import a lookup with a new name.
    - Choose **Replace** to import a previously imported lookup.
    - Choose **Automatic** to either add or replace the lookup, depending on whether it already exists on the target instance.
15. Click **Save**.

## Configure a Job to Delete an Integration Lookup

You can create and use a build step to delete an Integration lookup from an Oracle Integration instance.

To configure a build job that deletes an Integration lookup from an Oracle Integration instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.

5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.
   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Delete Lookup**.
9. In **Target Instance** , select the environment that points to the Oracle Integration instance from which you want to remove the lookup(s).
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.
11. In **Lookup Name**, enter or search for the name of the lookup you want to delete from the target Oracle Integration instance.

    To search for the lookup name, click . The Search Lookups dialog is displayed. Press **Enter** in the **Search** field to see the list of all lookups on your target instance (if you used Basic authentication, you'll need to specify credentials to see this list). Select the lookup from the list. Instead of selecting the lookup from a list, you can type in a lookup name for the specific lookup, typing in either the full name, or part of it, then selecting it from the narrowed list of choices. The drop-down list initially shows all the instance's lookups available to you.
12. Click **Save**.

## Manage Oracle Integration Connections

These are the build steps you'll need to use to update an Oracle Integration connection's security and connection properties and upload a property attachment for a specific Oracle Integration connection. You can make updates to a connection's security and connection properties, agentGroupId, or to the securityPolicy property. Other properties are ignored. Supported security properties depend on the securityPolicy that is selected. The connection type determines which securityPolicy values are supported.

| Step | Description | See this topic |
|---|---|---|
| Configure a build job to update an OIC connection's security and connection properties and more. | Creates and executes a job to:<br>• Identify the connection in the Oracle Integration instance that you want to update.<br>• Specify the path and filename of the file containing updates to the connection details or lets you change the details directly using inline JSON content. | Configure a Job to Update an Oracle Integration Connection |
| Optional: Configure a build job to upload a property attachment for an OIC connection. | Creates and executes a job to upload an attachment (such as a ZIP or WSDL file) to a connection and associates the attachment with a specific connection property. This is how SOAP adapters are configured. | Configure a Job to Upload a Property Attachment for an Oracle Integration Connection |

## Configure a Job to Update an Oracle Integration Connection

You can create and use job (or a build step) to update an Oracle Integration connection's security properties, connection properties, agentGroupId, or securityPolicy.

To configure a build job that updates an Oracle Integration connection's details:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.
   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Update Connection**.
9. In **Target Instance** , select the environment that points to the Oracle Integration instance with the connection details you want to modify.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.
    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).

- If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.

11. In **Identifier**, select the identifier of the connection in the target Oracle Integration instance that you want to update.

    You can also click **Search** and open the **Search Connections** window to get a list of connections in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field.

12. In the Source section, select and complete one of the following:

    - Choose **JSON file** and, in **JSON File Path**, enter the path and the filename of the file containing updates to the connection details.
    - Choose **Inline JSON** and enter the JSON updates directly in the text box.

13. Click **Save**.

## Configure a Job to Upload a Property Attachment for an Oracle Integration Connection

You can create and use a job (or a build step) to upload a property attachment (a zip or WSDL file) that updates a single property for an Oracle Integration connection.

To configure a build job that uploads an attachment (a zip or WSDL file) that modifies one Oracle Integration connection property:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.
   The job configuration page is displayed.
7. Click the **Steps** tab.
8. From **Add Step**, select **Oracle Integration**, and then select **Upload Connection Property Attachment**.
9. In **Target Instance** , select the environment that points to the Oracle Integration instance with the connection property you want to modify.
10. In the Authorization section, select **Use OAuth** (or **Use Basic**) to specify the type of authorization used to run this build step. OAuth is the only authentication method supported for Oracle Integration 3 instances.

    - If you selected **Use OAuth**, the `Authorization is required` message indicates that this build step needs to be authorized. Click **Authorize** and provide the credentials of a user who has permissions to perform Oracle Integration operations; you can also run the job manually and enter the credentials when prompted. Once authorized, the `Authorization has been provided` message displays.

      If the OAuth token used to authorize the connection expires during extended periods of inactivity (say, when you're away on vacation), you can click **Renew Authorization** to renew the token (or you can run the job manually, so you're prompted to authorize any expired OAuth tokens).
    - If you selected **Use Basic**, in **Username** and **Password**, provide the credentials of a user who has permissions to perform Oracle Integration operations.

11. In **Identifier**, select the identifier of the connection in the target Oracle Integration instance to receive the file.

    You can also click **Search** and open the **Search Connections** window to get a list of connections in your instance (if you used Basic authentication, you'll need to specify credentials to see this list), or narrow your search by entering a prefix in the Search field.

12. In **Property Name**, enter the name of the connection property to receive the file.
13. In **File Path**, enter the name of the file to upload, such as a zip or WSDL file.
14. If you're uploading a zip file, in **Service WSDL**, enter the file path for the WSDL file within the zip file.
15. Click **Save**.

## Run Unix Shell Commands

You can configure a job to run a Unix shell script or execute commands when a build runs:

1. Open the job's configuration page.
2. Click the **Steps** tab.
3. From **Add Step**, select **Common Build Tools**, then select **Unix Shell**.
4. In **Script**, enter the shell script or commands.

   The script runs, using the workspace as the current directory. If the shell script doesn't specify a header line, such as `#!/bin/sh`, the system shell will be used. You can also use the header line to write a script in another language, such as Perl (`#!/bin/perl`), or control the options that shell uses.

   You can also use Kubernetes, Docker, and OCIcli commands in the shell script. Make sure that you have the required software in the job's build executor template before you run a build.

5. To show the values of the variables and hide the input-output redirection in the build log, select the **(-x) Expand variables in commands, don't show I/O redirection** option.

   To show the command as-it-is in the build log, select the **(-v) Show commands exactly as written** option.

6. Click **Save**.

**Tip:**

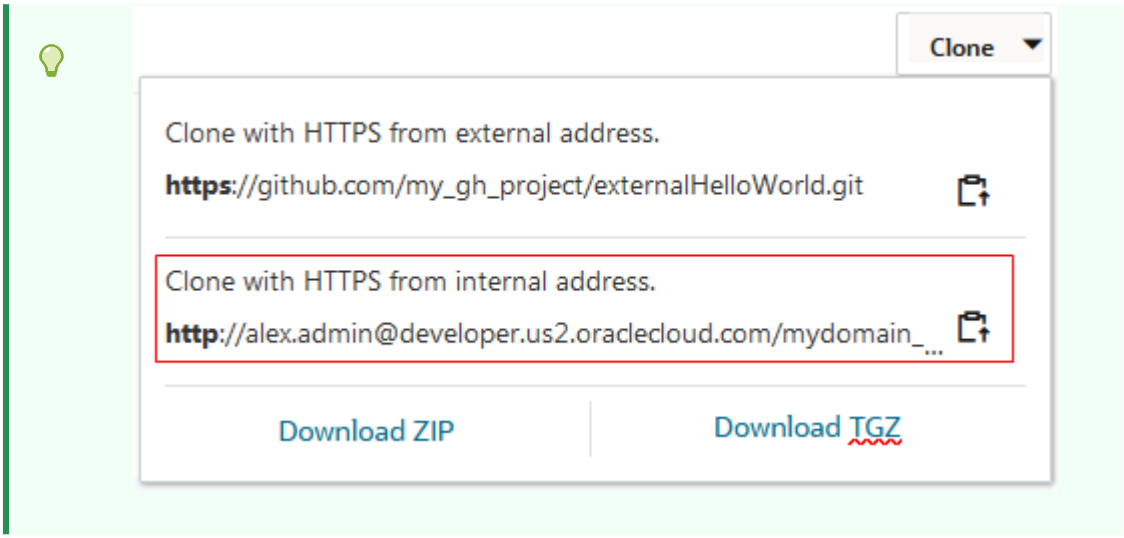
- By default, when a build runs, it invokes the shell with the `-ex` option. It prints all commands before they run. The build will fail if any command exits with a non-zero exit code. To change this behavior, add the `#!/bin/...` line in the shell script.
- If you have a long script, create a script file, add it to the Git repository, and then run it using a command, such as `bash -ex /myfolder/myscript.sh`.
- To run Python 3, create an isolated environment using venv. See `https://docs.python.org/3/library/venv.html`.

  For example, to create a virtual environment, add these commands as a Unix Shell build step:

  ```
  pip3 list
  cd $WORKSPACE
  python3 -m venv mytest
  cd mytest/bin
  ./pip3 list
  ./pip3 install --upgrade pip requests setuptools selenium
  ./pip3 list
  ./python3 -c 'import requests; r=requests.get('\''https://
  www.google.com'\''); print(r.status_code)'
  ./pip3 uninstall -y requests
  ./pip3 list
  ```
- To provide Python 3 capabilities to build jobs, use the Python 3 packages that are included with the build executor template by default. If you need specific capabilities that aren't available by default, you'll need to add the Python 3 version that has those capabilities to the build executor template.
- If both Python 2 and Python 3 are available in the job's build executor template, to call Python, use these commands:

| Command | Version |
|---|---|
| `python` | The `python` command refers to the OS-specific Python version that comes pre-installed with the software package:<br>– Python 2 (OL7)<br>– Python 3 (OL8) |
| `python2` | Python 2 |
| `python3` | The `python3` command refers to the Python 3 version installed with the software package. |
| `pip` | `pip` of Python 3 |
| `pip3` | `pip` of Python 3 |

- To clone an external Git repository using a shell command, use the internal URL of the external Git repository. To copy the URL, open the **Git** page and, from the **Repositories** drop-down list, select the external Git repository. From the **Clone** menu, click **Copy to clipboard** of the **Clone with HTTPS from internal address URL**.




## Use Docker-In-Docker with Shell Scripts

In VB Studio, Docker-in-Docker functionality is implemented using a methodology known as "sibling" containers, which means that a build creates images and containers in the deployment VM's Docker environment. Since multiple Docker executors share the same deployment VM, the images and containers will be shared among builds.

> **Note:**
>
> If your organization's builds use Docker executors and if those builds create Docker images and Docker containers, they'll be managed by the Docker environment in the deployment VM. This allows builds to interact with images and containers from other builds. If your project contains sensitive data and requires its build to run isolated in a VM, you should set up the build using VM executors instead.

Using a simple command, such as `docker rm $(docker container -q)`, in a shell script in a build could have the unintended consequence of killing containers that were created by other builds. To prevent this from happening, follow these recommendations to create and remove Docker images and containers:

- When you create a Docker container, use a unique name, by adding $TASKID to the container label. This distinguishes the container created by the build from a shared container.
- Containers that are created must be scoped to the build. They must be stopped and removed when the build completes.
- Images that are created in a build may be used across many builds, to avoid recreating the image during every build. However, take care to not consume all of the disk space in the deployment VM.
- *Don't* issue a Docker command, such as `docker rmi $(docker image -q)`, that deletes *all* Docker images. Instead, use a command, like `docker rmi <my_image>`, that only deletes *specific images* that were created by the build.
- *Don't* issue Docker commands like `docker stop $(docker ps -q)` and `docker rm $(docker ps -q)` that stop and delete *all* Docker containers.

Instead, use commands like `docker stop <my_container>` and `docker rm <my_container>` that stop and remove *specific containers* that were created by the build.

Here's an example that uses container *some_name*`_$TASKID`, with "_$TASKID" appended to the name. By using $TASKID with the container name, you can be sure that it is specific to your job and won't affect any other:

```
DOCKER_IMAGE=some_image

# Pull and run the container
docker pull ${DOCKER_IMAGE}
CONTAINER_ID=$(docker run --network=host --name some_name_$TASKID -it -d $
{DOCKER_IMAGE})

# Use your container

# Stop and remove the container
docker stop ${CONTAINER_ID}
docker rm ${CONTAINER_ID}
```

## Build Maven Applications

Using Apache Maven, you can automate your build process and download dependencies, as defined in the POM file:

1. Upload the Maven POM files to the project Git repository.
2. Open the job's configuration page.
3. In the **Git** tab, add the Git repository where you uploaded the build files.
4. Click the **Steps** tab.
5. From **Add Step**, select **Maven**.
6. In **Goals**, enter Maven goals, or phases, along with their options. By default, `clean` and `install` goals are added.
   For more information about Maven goals, see the Maven Lifecycle Reference documentation at `http://maven.apache.org`.
7. In **POM file**, enter the Maven POM file name and path, relative to the workspace root. The default value is **pom.xml** at the Git repository root.
8. If necessary, specify the Advanced Maven Settings:

| Action | How To |
|---|---|
| Use a private repository for builds | Select the **Use Private Repository** check box.<br>You may want to use it to make sure that other Maven build artifacts don't interfere with the artifacts of this job's builds. When a build runs, it creates a Maven repository `.maven/repo` directory in the build executor workspace. Remember selecting this option consumes more storage space of the workspace. |
| Use a private temporary directory for builds. | Select the **Use Private Temp Directory** check box.<br>You may want to use it to create a temporary directory for artifacts or temporary files. When a build runs, it creates a `.maven/tmp` directory in the workspace. The temporary files may consume large amount of storage, so, remember to clean up the directory regularly. |

| Action | How To |
|---|---|
| Work offline and don't access remote Maven repositories | Select the **Offline** check box. |
| Activate Maven profiles | In **Profiles**, enter a list of profiles, separated by commas.<br>For more information about Maven profiles, see the Maven documentation at `http://maven.apache.org`. |
| Set custom properties | In **Properties**, enter custom system properties in the `key=value` format. Specify each property on its own line.<br><br>**Tip:**<br>Don't surround properties that have multi-word values, like `key2=say hello`, with single or double quotes. Multi-word property assignments will be automatically wrapped with double quotes.<br>Also, notice that arguments, which are generated for the `mvn` command, have an extra space in them between "-D" and the argument, like this:<br>`mvn ... -D key1=value1 -D "key2=say hello" ...`<br><br>When a build runs, the properties are passed to the build executor in the standard way (for example, `-Dkey1=value1 -Dkey2=value2`) without the extra spaces. |
| Include build parameters in the Properties list | Leave checked (default setting) to include all build parameters in the Properties list.<br>Deselect to stop automatically adding the parameters to the list. |
| Set the Maven verbosity level | From **Verbosity**, select the level.<br>You may want to use it to set the verbosity of the Maven log output to the build log. |
| Set the checksum mode | From **Checksum**, select the mode.<br>You may want to use it to set the check-sum validation strictness when the build downloads artifacts from the remote Maven repositories. |
| Set handling of the SNAPSHOT artifacts | From **Snapshot**, select the mode. |
| Include other Maven projects to the reactor | In **Projects**, enter the comma or space separated list of Maven project jobs to include in the reactor. The reactor is a mechanism in Maven that handles multi-module projects. A project job can be specified by `[groupId]:artifactId` or by its relative path. |
| Resume a Maven project from the reactor | In **Resume From**, enter the Maven job project name from where you would like to resume the reactor. The Maven job project can be specified by `[groupId]:artifactId` or by its relative path. |
| Set the failure handling mode | From **Fail Mode**, select the mode.<br>You may want to use it to set how the Maven build proceeds in case of a failure. |
| Set the Make-like reactor mode | From **Make Mode**, select the mode. You may want to use it enable Make-like build behavior. |
| Configure the reactor threading model | In **Threading**, enter the value for experimental support for parallel builds. For example, a value of 3 indicates three threads for the build. |

| Action | How To |
|---|---|
| Pass parameters to Java VM | In **JVM Options**, enter the parameters. The build passes the parameters as `MAVEN_OPTS`. |

9. Click **Save**.

## Use the WebLogic Maven Plugin

The WebLogic server includes a Maven plugin that you can use to perform various deployment operations against the server, such as deploy, redeploy, and update. The plugin is available in the VB Studio build executor. For more information about how to use the WebLogic Maven plugin, see *Fusion Middleware Deploying Applications to Oracle WebLogic Server* in *Oracle Fusion Middleware Online Documentation Library*.

When a build runs, the build executor creates an empty Maven repository in the workspace.

> **Note:**
>
> Make sure that your build VM template includes the WebLogic Server (WLS) software bundle. For example, the Oracle JDeveloper Studio 12 software bundle includes the JDeveloper Studio 12.2.1.3 (or 12.2.1.4) software.

To install the WebLogic plugin every time a build starts, in the job configuration, add a Unix Shell command to install the plugin and then deploy it:

1. Open the job's configuration page.
2. Click the **Steps** tab.
3. From **Add Step**, select **Unix Shell**.
4. In **Script**, enter this command:

```
# Use sync to install deps
mvn install:install-file \
    -DpomFile=`find $MIDDLEWARE_HOME -name "oracle-maven-sync*.pom"` \
    -Dfile=`find $MIDDLEWARE_HOME -name "oracle-maven-sync*.jar"`
mvn com.oracle.maven:oracle-maven-sync:push \
    -DoracleHome=${MIDDLEWARE_HOME}

# Add additional arguments for deployment
mvn com.oracle.weblogic:weblogic-maven-plugin:deploy
```

5. Click **Save**.

## Upload to or Download Artifacts from the Project Maven Repository

To upload artifacts to the Maven repository, you'll use the `distributionManagement` snippet in the POM file. To download artifacts from the Maven repository, use the `repositories` snippet in the POM file:

1. To upload a build artifact to the Maven repository, copy the `distributionManagement` snippet of the project's Maven repository and add it to the POM file:
   a. In the left navigator, click **Maven**.
   b. On the right side of the page, click **Browse**.

c. In the **Artifact Details** section, expand **Distribution Management**.

d. In the **Maven** tab, click **Copy** to copy the `<distributionManagement>` code snippet to the clipboard.

e. Open the POM file of your project in a code editor (or a text editor) and paste the contents of the clipboard under the `<project>` element:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example.employees</groupId>
    <artifactId>employees-app</artifactId>
    <packaging>war</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>employees-app Maven Webapp</name>
    <url>http://maven.apache.org</url>
    <properties>
        <tomcat.version>7.0.57</tomcat.version>
    </properties>

  <distributionManagement>
    <repository>
      <id>Demo_repo</id>
      <name>Demo Maven Repository</name>
      <url>https://developer.us2.oraclecloud.com/profile/my-org/s/my-org_demo_12345/maven/</url>
    </repository>
    <snapshotRepository>
      <id>Demo_repo</id>
      <name>DemoMaven Repository</name>
      <url>https://developer.us2.oraclecloud.com/profile/my-org/s/my-org_demo_12345/maven/</url>
    </snapshotRepository>
  </distributionManagement>
```

2. To download an artifact from the Maven repository, use the `repositories` snippet of the project's Maven repository:

a. In the left navigator, click **Maven**.

b. On the right side of the page, click **Browse**.

c. In the **Artifact Details** section, expand **Distribution Management**.

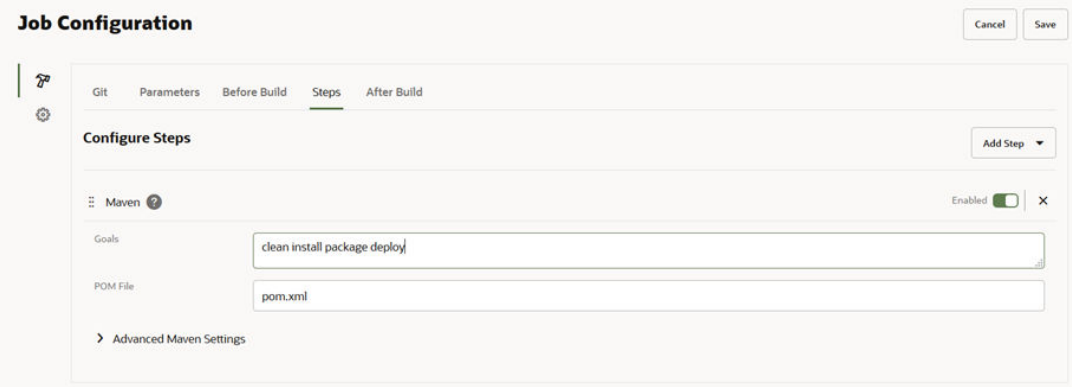
d. In the **Maven** tab, copy the `<repository>` element of the **Distribution Management** to the clipboard.

e. Open the POM file of your project in a code editor (or a text editor) and paste the `<repository>` element in the`<repositories>` element under `<project>`:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example.employees</groupId>
    <artifactId>employees-app</artifactId>
    <packaging>war</packaging>
    <version>0.0.1-SNAPSHOT</version>
    <name>employees-app Maven Webapp</name>
    <url>http://maven.apache.org</url>

    <repositories>
      <repository>
           <id>Demo_repo</id>
           <name>Demo Maven Repository</name>
           <url>http://developer.us2.oraclecloud.com/profile/my-org/s/my-
org_demo_12345/maven/</url>
         </repository>
    </repositories>
.
.
.
</project>
```

3. Save the file, commit it to the Git repository, and then push the commit.
4. Configure the job to add a Maven step and add the required Maven goals:

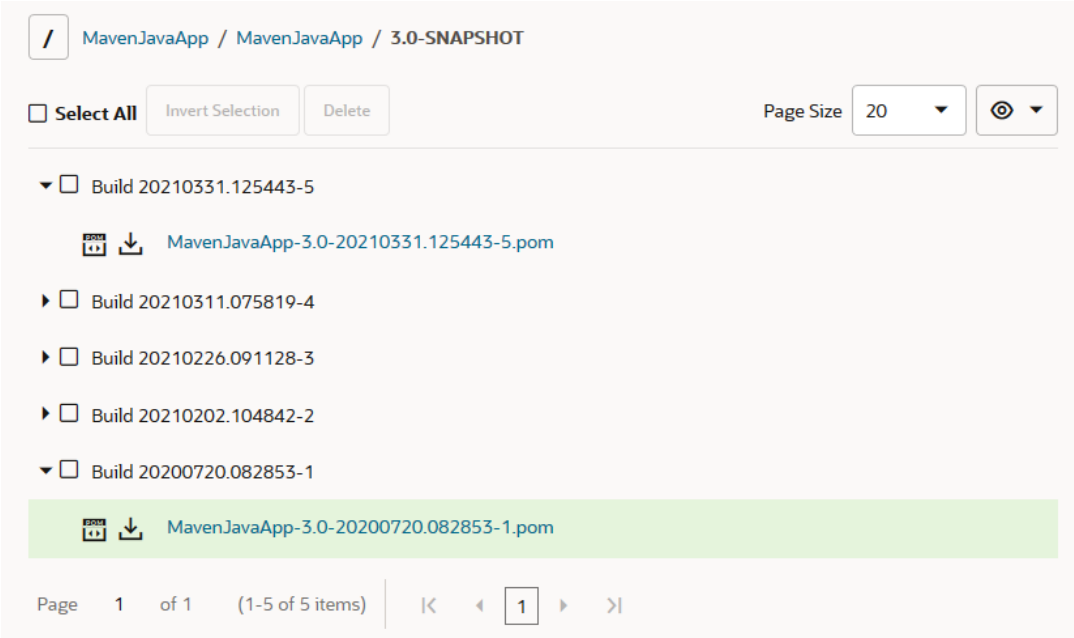



> **Tip:**
>
> Use the `deploy` goal to upload Maven artifacts to the project's Maven repository.

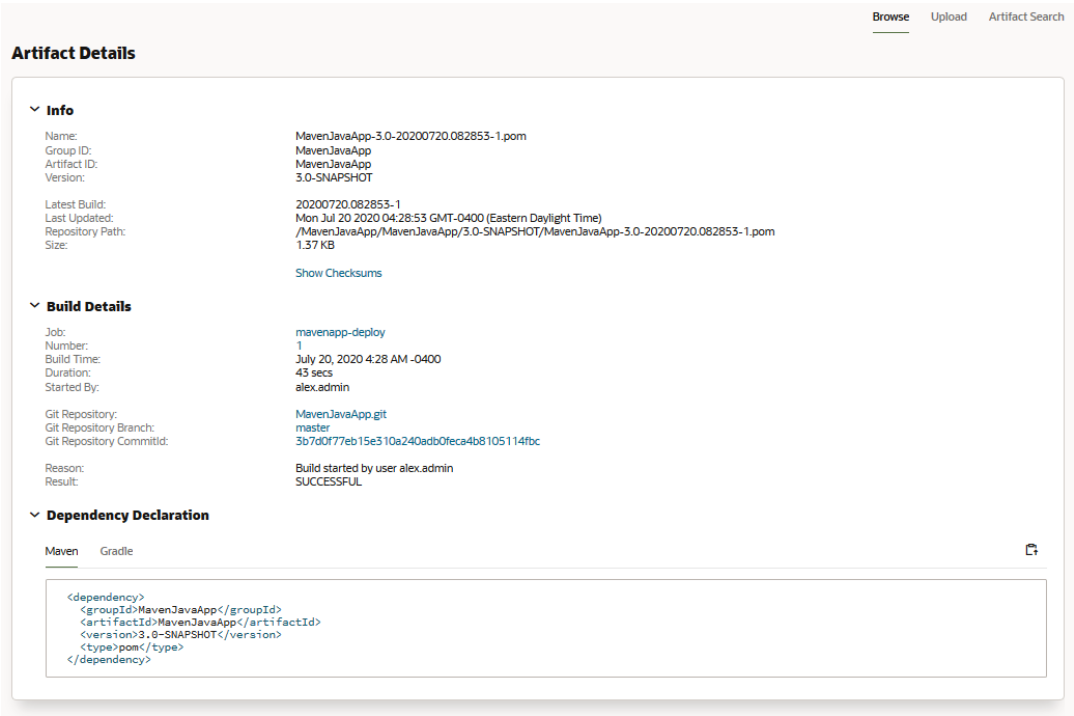
5. If you want to view build information for an artifact uploaded to the Maven repository with a build job or pipeline, make sure that the build executor template you select from the Software catalog in the **Software** tab includes the latest Maven and Gradle versions.
6. Run a build of the job.

> **Note:**
>
> You don't have to provide the credentials in `settings.xml` to access the project's Maven repository when you run a build. Build jobs have full access to the project's Maven repository for uploads and downloads.

7. If you configured the job to upload artifacts to the project's Maven repository, after the build is successful, verify the artifacts in the Maven page:




8. If you selected a build executor template that includes Maven and Gradle from the Software catalog in the **Software** tab, you can view build information for the artifact:




## Build Ant Applications

You can use Apache Ant to automate your build processes, as described in its build files:

1. Upload the Ant build files (such as `build.xml` and `build.properties`) to the project Git repository.
2. Open the job's configuration page.
3. In the **Git** tab, add the Git repository where you uploaded the build files.

4. Click the **Steps** tab.
5. From **Add Step**, select **Ant**.
6. In **Targets**, specify the Ant targets or leave it empty to run the default Ant target specified in the build file.
7. In **Build File**, specify the path of the build file.
8. If necessary, in **Properties**, specify the values for properties used in the Ant build file:

   ```
   # comment
   name1=value1
   name2=$VAR2
   ```

   When a build runs, these values will be passed to Ant as `-Dname1=value1 -Dname2=value2`. You should always use `$VAR` for parameter references instead of using `%VAR%`). Use a double backslash (`\\`) to escape a backslash (\). Avoid using double-quotes ("). To define an empty property, use `varname=` in the script.
9. If your build requires a custom `ANT_OPTS`, specify it in **Java Options**. You may use it to specify Java memory limits (example: `-Xmx512m`). Don't specify other Ant options here (such as `-lib`), but specify them in **Targets**.
10. Click **Save**.

For more information, see `https://ant.apache.org/`.

## Build Gradle Applications

Using Gradle, you can automate your build processes as defined in its build script. For more information about Gradle, see `https://gradle.org/`.

In VB Studio, Gradle 5 is available. To use another version of Gradle, use Gradle Wrapper in the Gradle build step. Gradle recommends using Gradle Wrapper as the preferred way to run a Gradle build. To learn more about using Gradle Wrapper, see `https://docs.gradle.org/current/userguide/gradle_wrapper.html`.

### Set Up a VM Build Executor and a Build Executor Template with Gradle

Before you can create a build step that uses Gradle commands, your organization administrator must create a build executor template that includes the Gradle software and add a VM build executor that uses that build executor template. The build executor template can be created from scratch or software can be added to an existing build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After the organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that build executor template and add Gradle commands.

## Configure a Job to Run Gradle Commands

Here's how you create and configure a job that runs Gradle commands:

1. Upload the `build.gradle` file to a project's Git repository.
2. Open the job's configuration page.

   If you're creating a job, in **Template** in the New Job dialog box, select the Gradle build executor template. Jump to step 5.
3. Click **Settings** .
4. In the **Software** tab, select the Gradle build executor template.
5. Click **Configure** .
6. In the **Git** tab, add the Git repository where you uploaded the build file.
7. Click the **Steps** tab.
8. From **Add Step**, select **Gradle**.
9. To use the Gradle installation that comes with the build executor, use the default settings and don't select anything. To use the Gradle wrapper, select **Use 'gradlew' wrapper**:
   - If you selected **Use 'gradlew' wrapper**, you can deselect the **Create 'gradlew' wrapper** check box if you don't want to create a new Gradle wrapper when a build runs. If the check box isn't selected, make sure that the `gradlew` executable is in the `$WORKSPACE` directory. If the `gradlew` executable is in the root build script directory, select the **In root build script directory** check box.
   - To use a different Gradle version, select **Use Gradle version** and specify the version in the **Gradle version** field. The Gradle version is optional.

     > **Tip:**
     >
     > To change the Gradle version when a build runs, add a build parameter, perhaps GRADLE_VERSION, and use it here. When a build runs, you can change the default version of Gradle and specify another version.
   - To specify a different distribution mirror, select **Use Gradle distribution URL** and enter the address in the **Gradle distribution URL** field. You may want to do this if, for example, you experience intermittent download timeouts. Your company may offer regional mirrors that you can use as an alternative.
10. In **Tasks**, enter Gradle tasks.
11. In **Build File**, enter the name and path of the Gradle `build.gradle` file. This path must be relative to the root build script directory, if specified, else relative to the `$WORKSPACE` directory.
12. In **Root build script directory**, enter the directory path that contains the top-level `build.gradle` file and serves as the project root. The path must be relative to the `$WORKSPACE` directory.

    If left empty, the path defaults to `build.gradle` in the root directory.
13. In **Switches**, enter Gradle switches.

14. If you're using a build executor that is shared by other jobs or users, select the **Force GRADLE_USER_HOME to use workspace** check box to set `GRADLE_USER_HOME` to the workspace.

    By default, `GRADLE_USER_HOME` is set to `$HOME/.gradle`, so with this option you can avoid encountering unwanted changes in the default shared directory.

15. Click **Save**.

## Build Node.js Applications

Using Node.js, you can develop applications that run JavaScript on a server. For more information, see `https://nodejs.org`.

### Set Up a VM Build Executor and a Build Executor Template with Node.js

Before you can create a build step that uses Node.js, your organization administrator must create a build executor template that includes the Node.js software and add a VM build executor that uses that build executor template. The build executor template can be created from scratch or software can be added to an existing build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After the organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that build executor template and add a Node.js script.

### Configure a Job to Build a Node.js Application

Here's how you create and configure a job that builds a Node.js application:

1. If you have a Node.js script, upload it to the project Git repository.
2. Open the job's configuration page.

   If you're creating a job, in **Template** in the New Job dialog box, select the Node.js build executor template. Jump to step 5.

3. Click **Settings** .
4. In the **Software** tab, select the Node.js build executor template.
5. Click **Configure** .
6. In the **Git** tab, add the Git repository where you uploaded the script file.
7. Click the **Steps** tab.
8. From **Add Step**, select **Node.js**.
9. To specify the script file, in **Source**, select **NodeJS File**. In **NodeJS File Path**, specify the file path in the Git repository.

To specify the script, in **Source**, select **Script**. In **NodeJS Script**, enter the script.

10. To speed up build execution time, you can use a Unix Shell step to install NPM packages globally on your VM build executor(s), by running NPM commands with the `--global` option.

    Modules such as Gulp, Grunt, Bower, and Oracle DB Node package come preinstalled on a Compute VM. Not all modules are available across all versions of Node and these packages get out of date and are superseded by newer versions rather quickly.

    By using the `--global` option, you can install the NPM packages you need on a VM build executor and doing so will also make those packages available to subsequent builds that run on the same VM build executor. This results in a significant time saving over installing the same packages locally, which requires them to be reinstalled in every subsequent build.

    Global packages are important, but you should avoid using them if they aren't needed. In general, the rule of thumb is:

    - If you're installing something to be used in your program, make it required, then install it locally, at the root of your project.
    - If you're installing something to be used in your shell or on the command line, install it globally, so its binaries end up in your PATH environment variable.

11. Click **Save**.

## Access an Oracle Database Using SQLcl

Using SQLcl, you can run SQL statements from a build to connect and access an Oracle Database. You can use SQLcl to access any publicly available Oracle Database that you can connect to using a JDBC connect string. You can run DML, DDL, and SQL Plus statements. You can also use SQLcl in a test scenario and run SQL scripts to initialize seed data or validate database changes.

To learn more about SQLcl, see `http://www.oracle.com/technetwork/developer-tools/sqlcl/overview/index.html`. Also see the HELP command description in Entering and Executing Commands in *Oracle SQL Developer Command-Line Quick Reference*.

### Set Up a VM Build Executor and a Build Executor Template with SQLcl

Before you can create a build step that uses SQLcl commands, your organization administrator must create a build executor template that includes the SQLcl software and add a VM build executor that uses that build executor template. The build executor template can be created from scratch or software can be added to an existing build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After the organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that build executor template and add SQLcl commands.

## Configure a Job to Run SQLcl Commands

Before you configure the job, you need to be aware of the following information:

- VB Studio doesn't support SQL commands to edit buffer (such as `set sqlformat csv`) or edit console.
- VB Studio doesn't support build parameters in the SQL file.
- If you are using Oracle REST Data Services (ORDS), some SQLcl commands, such as the BRIDGE command, requires a JDBC URL:
  ```
  BRIDGE table1 as "jdbc:oracle:thin:DEMO/demo@http://examplehost.com/ords/demo"(select * from DUAL);
  ```
- To mark a build as failed if the SQL commands fail, add the `WHENEVER SQLERROR EXIT 1` line to your script.

Here's how you create and configure a job that runs SQLcl commands:

1. Open the job's configuration page.

   If you're creating a job, in **Template** in the New Job dialog box, select the SQLcl build executor template. Jump to step 5.
2. Click **Settings** .
3. In the **Software** tab, select the SQLcl build executor template.
4. From the **Java** drop-down list, select the version.
5. Click **Configure** .
6. In the **Git** tab, add the Git repository where you uploaded the script file.
7. Click the **Steps** tab.
8. From **Add Step**, select **SQLcl**.
9. In **Username** and **Password**, enter the user name and password of the Oracle Database account.

   You can also use build parameters in **Username** and **Password**.
10. To connect to Oracle Database Exadata Express Cloud Service, in **Credentials File**, enter the workspace path of the uploaded credentials zip file.
11. In **Connect String**, enter the JDBC or HTTP connection string of the Oracle Database account using any of the `host_name:port:SID` or `host_name:port/service_name` formats.

    Here's a JDBC example:

    ```
    test_server.oracle.com:1521:adt1100
    ```

    In this example, `adt1100` is the SID, and `ora11g` is the service name in `test_server.oracle.com:1521/ora11g`.

    Here's an HTTP example:

    ```
    http://test_server.oracle.com:8085/ords/demo
    ```

    You can also use build parameters in **Connect String**.
12. If the SQL statements are available in a file uploaded to the project Git repository, in **Source**, select **SQL File**. In **SQL File Path**, enter the Git repository path of the SQL file. You can copy the file's path from the **Git** page.

To enter SQL statements, in **Source**, select **Inline SQL**. In **SQL Statements**, enter the SQL statements. You can also use build parameters in **SQL Statements**.

13. In **Role**, if necessary, select the database role of the user.
14. In **Restriction Level**, if necessary, specify the restriction level on the type of SQL statements that are allowed to run.
15. In **JVM Options**, enter the parameters you want to pass to the Java VM and the build will pass the parameters as _JAVA_OPTIONS.

    You're setting this parameter (or these parameters) specifically for the session running SQLcl – if you set it globally, that would impact every Java program running on the machine, which you probably don't want to do.
16. Click **Save**.

## Use OCIcli to Access Oracle Cloud Infrastructure Services

You can use Oracle Cloud Infrastructure command line interface (OCIcli) commands to create and manage Oracle Cloud Infrastructure objects and services when a build runs.

For more information about OCIcli and its commands, see the Oracle Cloud Infrastructure Command Line Interface documentation.
To configure the job, you'll need this information:

- The User OCID
- A private key that has been set **with no passphrase**

> **Note:**
>
> You shouldn't use a passphrase for OCI public/private keys in an OCIcli build step. If you do, when the build job encounters the key you'll be prompted for the passphrase, but, since you can't interact with the job's shell to supply it, the build will fail and an error will be reported in the build job's log. To avoid this problem, you'll need to generate a public-private key pair **without a passphrase** and upload the public key to your user preferences.
>
> See Upload Your Public SSH Key for information about generating an SSH key and uploading the public SSH key to your VB Studio account.

- The fingerprint of a user who can create and access the resources
- The tenancy name

Contact the OCI administrator and get the required OCI input values. Get OCI input Values explains where these values can be found.

### Set Up a VM Build Executor and a Build Executor Template with OCIcli

Before you can create a build step that uses OCIcli commands, your organization administrator must create a build executor template that includes the OCIcli software and add a VM build executor that uses that build executor template. The build executor template can be created from scratch or software can be added to an existing build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After the organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that build executor template and add OCIcli commands.

## Configure a Job to Run OCIcli Commands

Here's how you create and configure a job that runs OCIcli commands:

1. Open the job's configuration page.

   If you're creating a job, select **Template** in the **New Job** dialog and then select the OCIcli build executor template. After creating the job, proceed to step 4.

2. Click **Settings** .
3. In the **Software** tab, select the OCIcli build executor template.
4. Click **Configure** .
5. Click the **Steps** tab.
6. From **Add Step**, select **OCIcli**.
7. In **User OCID**, enter the OCID of the user who can access or create OCI resources.
8. In **Fingerprint**, enter the public key fingerprint of the user.
9. In **Tenancy**, enter the tenancy OCID.
10. In **Private Key**, enter the private key of the user.

    Use a private key that was set with **no passphrase**. If you don't have one, generate a public-private key pair **without a passphrase** and upload the public key to your user preferences. See Upload Your Public SSH Key for information about generating an SSH key and uploading the public SSH key to your VB Studio account.

11. In **Region**, select the Oracle Cloud Infrastructure tenancy's region.
12. Scroll up and from **Add Step**, select **Unix Shell**.
13. In **Script**, enter the OCIcli commands on separate lines.
14. Click **Save**.

Add multiple **Unix Shell** steps to run additional sets of commands. Don't add another **OCIcli** build step.

## Run Docker Commands

You can configure a job to run Docker commands on a Docker container when a build runs.

You should use the Docker container for short tests and builds. Don't run a Docker container for long tests or builds, or the builds might not finish. For example, if you use a Docker image that's listening on a certain port and behaves like a web server, most likely the build won't exit.

For more information about Docker commands, see `https://docs.docker.com/`.

> **Tip:**
>
> If you face a network issue when you run Docker commands, try adding the `HTTP_PROXY` and `HTTPS_PROXY` environment variables in the Docker file.

## Set Up a VM Build Executor and a Build Executor Template with Docker

Before you can create a build step that uses Docker commands, your organization administrator must create a build executor template that includes the Docker software and add a VM build executor that uses that build executor template. The build executor template can be created from scratch or software can be added to an existing build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After the organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that build executor template and add Docker commands.

## Configure a Job to Run Docker Commands

Here's how you create and configure a job that runs Docker commands:

1. Open the job's configuration page.

   If you're creating a job, in **Template** in the New Job dialog box, select the Docker build executor template. Proceed to step 4.

2. Click **Settings** .
3. In the **Software** tab, select the Docker build executor template.
4. Click **Configure** .
5. Click the **Steps** tab.
6. From **Add Step**, select **Docker**, and then select the Docker command:

| Use this command ... | To ... |
|---|---|
| `login` | Log in to the Docker registry.<br>In **Registry Host**, select a pre-linked Docker registry, or enter the Docker registry's host name where the images are stored. Leave it empty to use Docker Hub.<br>In **Username** and **Password**, enter the credentials of the user who can access the Docker registry. |

| Use this command ... | To ... |
|---|---|
| `build` | Build Docker images from a Dockerfile.<br>Specify the registry host name, the Docker image name, its version tag, any Docker options, and the name and source of the Dockerfile. You can upload the Dockerfile in the Git repository and provide its path, add the Dockerfile code manually, or provide its URL if it's available on an external source.<br>To specify an external source, include the protocol. For example, include `http`) in the URL if you're referencing a remote TAR file, such as `http://55.555.555.555/me/mydocker.tar.gz`. If you're referencing a remote repository, ignore the protocol, as in `git://github.com/me/my.git#mybranch:myfolder`, for example.<br>To learn more about Docker build command options, see `https://docs.docker.com/engine/reference/commandline/build/`. |
| `tag` | Create a target image tag that refers to the source image.<br>Specify the registry host name, Docker image name, and its version tag name for the source and target images. |
| `push` | Push an image to the Docker registry.<br>To learn more about push options, see `https://docs.docker.com/engine/reference/commandline/push/`. |
| `images` | List available images.<br>To learn more about images, see `https://docs.docker.com/engine/reference/commandline/images/`. |
| `save` | Save an image to a `.tar` archive file.<br>In **Output File**, specify the relative path and name of the output `.tar` file in the workspace. |
| `load` | Load an image from a `.tar` archive file.<br>In **Output File**, specify the relative path and name of the output `.tar` file in the workspace. |
| `rmi` | Remove an image. You can remove new images, a specific image, or all images.<br>To remove a specific image, enter the host name of the registry where the Docker images are stored. Remember that the images are stored in the registry if they are pushed there. Until the images are pushed, the Registry Host is used to form the fully qualified name of the Docker image on the computer where the image is being created. |
| `version` | View the version of Docker on the build executor. |

7. Click **Save**.

The Docker `logout` command runs automatically after all Docker commands have run.

## Run Fn Commands

Fn, or Fn Project, is an open-source, container-native, serverless platform for building, deploying, and scaling functions in multi-cloud environments. To run Fn commands when a build runs, you must have access to a Docker container that has a running Fn server.

For more information about Fn, see `https://fnproject.io/`.

## Set Up a VM Build Executor and a Build Executor Template with Fn

Before you can create a build step that uses Fn commands, your organization administrator must create a build executor template that includes the Fn software and add a VM build executor that uses that build executor template. The build executor template can be created from scratch or software can be added to an existing build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After your organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that buiold executor template and add Fn commands.

## Configure a Job to Run Fn Commands

Here's how you create and configure a job that runs Fn commands:

1. Open the job's configuration page.

   If you're creating a job, in **Template** in the **New Job** dialog, select the Fn build executor template. Proceed to step 4.

2. Click **Settings** .
3. In the **Software** tab, select the Fn build executor template.
4. Click **Configure** .
5. Click the **Steps** tab.
6. From **Add Step**, select **Fn**, and then select the command:

| Use this option ... | To ... |
|---|---|
| `Fn Version` | Log the version of the Fn CLI being used and the version of the Fn Server referenced by the current context, if available, in the build log. |
| `Fn Build` | Build a new function.<br>Specify the relative path of the working directory to build the function, Fn build arguments, Docker registry host, and its user name. If you don't want to use the Docker registry's cache, deselect the **Use Docker Cache** check box. To display the command's log in the build log, select the **Verbose Output** check box. |
| `Fn Push` | Push the image to the Docker registry.<br>Specify the relative path of the working directory, Docker registry host, and its user name. To display the command's log in the build log, select the **Verbose Output** check box. |
| `Fn Bump` | Bump the version of the `func.yaml` file.<br>Specify the relative path of the working directory and the bump type (`Major`, `Minor`, or `Patch`). To display the command's log in the build's log, select the **Verbose Output** check box. |

| Use this option ... | To ... |
|---|---|
| `Fn Deploy` | Deploy functions to the function server. Using the deploy command, you can bump, build, push and update a function.<br>In **Deploy to App**, specify the Fn app name to deploy to. In other fields, specify the working directory, build arguments, Docker registry host, user name, API URL, and the Call URL. Select the desired check boxes, if necessary. |
| `Fn OCI` | Augments the OCI configuration provided by the OCIcli builder with two additional parameters that are needed for Oracle Functions, the Oracle version of the open source Fn server. These required OCI parameters are the Oracle Compartment ID and the provider.<br>See Oracle Functions Quick Start Guides for more information about these options. |

7. Click **Save**.

# Use SonarQube

SonarQube is open source quality management software that continuously analyzes your application. When you configure a job to use SonarQube, the build generates an analysis summary that you can view from the job or build details page.

To learn about SonarQube, see its documentation at `https://docs.sonarqube.org`.

## Create and Manage the Pre-Defined SonarQube Server Connection

You must be the project owner to add and manage SonarQube server connections.

To create the connection, you'll need the URL of a SonarQube server that's available on the public internet.
Here's how you can set up a SonarQube system for your project's users and then create and manage a pre-defined SonarQube connection that they can use:

| Action | How To |
|---|---|
| Add a SonarQube connection | 1. In the left navigator, click **Project Administration** .<br>2. Click **Build**.<br>3. Click the **SonarQube Server** tab.<br>4. Click **Add SonarQube Server Connection**.<br>5. In the **Create SonarQube Server** dialog box, enter a name for the server, provide the SonarQube server's URL, and specify the credentials of a user who has access to the server.<br>6. Click **Create**. |

| Action | How To |
|---|---|
| Edit a connection to change the user credentials or provide another server ID | 1. In the left navigator, click **Project Administration** .<br>2. Click **Build**.<br>3. Click the **SonarQube Server** tab<br>4. Click the connection name and then click **Edit** .<br>5. In the **Edit SonarQube Server** dialog box, update the SonarQube server's URL and the credentials of a user who can access the server.<br>6. Click **Update**. |
| Delete the connection | 1. In the left navigator, click **Project Administration** .<br>2. Click **Build**.<br>3. Click the **SonarQube Server** tab.<br>4. Click the connection name and then click .<br>5. In the **Delete SonarQube Server** dialog, click **Delete**. |

## Configure a Job to Connect to SonarQube

You can configure a job to use SonarQube from the **Before Build** tab and then add a post-build action (on the **After Build** tab) to publish its reports:

1. Open the job's configuration page.
2. Click the **Before Build** tab.
3. From **Add Before Build Action**, select **SonarQube Settings**.
4. From **Sonar Server**, select the pre-configured SonarQube server.

   The **Username**, **Password**, and **SonarQube Server URL** display the selected user's details. To add a server, contact the organization administrator.
5. To provide the SonarQube project name and the SonarQube project key, expand **Advanced SonarQube Settings**, and update the values. Make sure that the SonarQube project key is unique.

   By default, the project key is set to `<organization>_<projectname>.<jobname>`and the project name is set to `<projectname>.<jobname>`.
6. Click the **After Build** tab.
7. From **Add After Build Action**, select **SonarQube Result Publisher**.
8. To use the SonarQube Quality Gate status as the build status, select **Apply SonarQube quality gate status as build status**.

   If the SonarQube Quality Gate status is `Passed`, the build is marked as successful. If the SonarQube Quality Gate status is `Failed`, the build is marked as failed. To learn about SonarQube Quality Gates, see `https://docs.sonarqube.org/display/SONAR/Quality+Gates`.
9. To create an archive file that contains the SonarQube analysis files, select the **Archive Analysis Files** check box.
10. Click **Save**.

To view the SonarQube analysis summary after a build, from the job's details page, click **SonarQube Analysis Summary** . The SonarQube Analysis Summary displays SonarQube server URL for the job and the analysis summary.

## Enable SonarQube for Gradle Applications

Use the following steps to enable SonarQube for your Gradle application.

1. Add a SonarQube connection.

   See Create and Manage the Pre-Defined SonarQube Server Connection.
2. Create a build job, using the Gradle Linux 7 VM build executor.
3. In the Job Configuration page, select the **Git** tab, then select **Git** from the **Add Git** dropdown.

   Add a Git repository for your Gradle project files.
4. Select the **Before Build** tab, then select **SonarQube Settings** from the **Add Before Build Action** dropdown.
5. Select the SonarQube server you set up in step 1.
6. Expand the **Advanced SonarQube Settings** and examine the **Project Name** entry.

   If the entry contains spaces, enclose it with double quotes.
7. In the **Steps** tab, select **Add Step**, then select **Gradle** from the **Common Build Tools** dropdown list.
   a. Select **Use 'gradlew' wrapper**.
   b. Enter the Gradle version needed to run your project.
   c. In **Tasks**, add `clean build`.
   d. Enter the location of your build file, `build.gradle`.
   e. Click **Turn on SonarQube**.
8. In the **After Build** tab, add the **SonarQube Result Publisher** action.
9. Select **Apply SonarQube quality gate status as build status** and **Archive Analysis Files**.
10. Click **Save** to save the build configuration.
11. From the Build Details page, click **Build Now** to run the build job.
12. Once the build completes, click **SonarQube Analysis Summary** to display the **SonarQube Analysis Summary** page, which shows the SonarQube server URL for the job and the analysis summary.

## Use a Unix Shell Script to Enable SonarQube for Gradle Applications

You can use a Unix Shell script to enable SonarQube for Gradle applications.

1. Add a SonarQube connection.

   See Create and Manage the Pre-Defined SonarQube Server Connection.
2. Create a build job, using the Gradle Linux 7 VM build executor.
3. In the Job Configuration page, select the **Git** tab, then select **Git** from the **Add Git** dropdown.

Add a Git repository for your Gradle project files.

4. Select the **Before Build** tab, then select **SonarQube Settings** from the **Add Before Build Action** dropdown.
5. Select the SonarQube server you set up in step 1.
6. Expand the **Advanced SonarQube Settings** and examine the **Project Name** entry.

   If the entry contains spaces, enclose it with double quotes.
7. In the **Steps** tab, select **Add Step**, then select **Unix Shell** from the **Common Build Tools** dropdown list.
   a. Click **Turn on SonarQube**.
   b. In the **For Gradle** tab, copy the Gradle command line content and paste it into the Unix Shell script area, at the top of the screen. For example:

```
gradle clean build --build-file=sonarqube-scanner-gradle-multimodule/
build.gradle sonarqube -Dsonar.host.url=$SONAR_URL -
Dsonar.login=$SONAR_LOGIN -Dsonar.password=$SONAR_PASSWD -
Dsonar.projectName=$SONAR_PROJECT_NAME -
Dsonar.projectKey=$SONAR_PROJECT_KEY
```

8. In the **After Build** tab, add the **SonarQube Result Publisher** action.
9. Select **Apply SonarQube quality gate status as build status** and **Archive Analysis Files**.
10. Click **Save** to save the build configuration.
11. From the Build Details page, click **Build Now** to run the build job.
12. Once the build completes, click **SonarQube Analysis Summary** to display the **SonarQube Analysis Summary** page, which shows the SonarQube server URL for the job and the analysis summary.

## Enable SonarQube for Maven Applications

Use the following steps to enable SonarQube for your Maven application.

1. Add a SonarQube connection.

   See Create and Manage the Pre-Defined SonarQube Server Connection.
2. Create a build job, using the Required Components Linux 7 VM build executor.
3. In the Job Configuration page, select the **Git** tab, then select **Git** from the **Add Git** dropdown.

   Add a Git repository for your Maven project files.
4. Select the **Before Build** tab, then select **SonarQube Settings** from the **Add Before Build Action** dropdown.
5. Select the SonarQube server you set up in step 1.
6. Expand the **Advanced SonarQube Settings** and examine the **Project Name** entry.

   If the entry contains spaces, enclose it with double quotes.
7. In the **Steps** tab, select **Add Step**, then select **Maven** from the **Common Build Tools** dropdown list.
   a. In **Tasks**, add `clean install`.

b. Enter the location of your POM file, `pom.xml`.

c. Click **Turn on SonarQube**.

8. In the **After Build** tab, add the **SonarQube Result Publisher** action.
9. Select **Apply SonarQube quality gate status as build status** and **Archive Analysis Files**.
10. Click **Save** to save the build configuration.
11. From the Build Details page, click **Build Now** to run the build job.
12. Once the build completes, click **SonarQube Analysis Summary** to display the **SonarQube Analysis Summary** page, which shows the SonarQube server URL for the job and the analysis summary.

## Use a Unix Shell Script to Enable SonarQube for Maven Applications

You can use a Unix Shell script to enable SonarQube for Maven applications.

1. Add a SonarQube connection.

   See Create and Manage the Pre-Defined SonarQube Server Connection.
2. Create a build job, using the Required Components Linux 7 VM build executor.
3. In the Job Configuration page, select the **Git** tab, then select **Git** from the **Add Git** dropdown.

   Add a Git repository for your Maven project files.
4. Select the **Before Build** tab, then select **SonarQube Settings** from the **Add Before Build Action** dropdown.
5. Select the SonarQube server you set up in step 1.
6. Expand the **Advanced SonarQube Settings** and examine the **Project Name** entry.

   If the entry contains spaces, enclose it with double quotes.
7. In the **Steps** tab, select **Add Step**, then select **Unix Shell** from the **Common Build Tools** dropdown list.

   a. Click **Turn on SonarQube**.

   b. In the **For Maven** tab, copy the Maven command line content and paste it into the Unix Shell script area, at the top of the screen. For example:

   ```
   mvn clean install -f sonarqube-scanner-maven/maven-basic/pom.xml
   sonar:sonar -Dsonar.host.url=$SONAR_URL -Dsonar.login=$SONAR_LOGIN -
   Dsonar.password=$SONAR_PASSWD -Dsonar.projectName=$SONAR_PROJECT_NAME -
   Dsonar.projectKey=$SONAR_PROJECT_KEY
   ```

8. In the **After Build** tab, add the **SonarQube Result Publisher** action.
9. Select **Apply SonarQube quality gate status as build status** and **Archive Analysis Files**.
10. Click **Save** to save the build configuration.
11. From the Build Details page, click **Build Now** to run the build job.
12. Once the build completes, click **SonarQube Analysis Summary** to display the **SonarQube Analysis Summary** page, which shows the SonarQube server URL for the job and the analysis summary.

## Create a Sonarqube Analysis Report for a VB Studio Project with Javascript Sources

The VB Studio build system supports Sonarqube analysis for Java using Maven and Gradle during building and packaging. This is for Java apps, not visual applications. It doesn't provide built-in support for analyzing Javascript sources. If you need to perform a Sonarqube analysis for Javascript sources, such as those created by VB Studio, you'll need to create your own Sonarqube analysis report by using the Unix Shell builder and then uploading the results to Sonarqube.

There are two ways to create a Sonarqube analysis report on Javascript sources in VB Studio:

- If the project being built is a Maven project, you'll need to direct the Sonar Scanner Maven plugin to include the JS files for analysis.
- If the project is a VB Studio project that is purely Javascript, you'll need to install and use the sonar-scanner command line tool to do the analysis.

### Analyze Javascript Sources in a Maven Project

If the project being built is a Maven project, by default, the Sonar Scanner Maven plugin will include only the Java sources from `src/main/java`. You'll need to make sure the plugin also includes the Javascript files for analysis:

1. Use the `-Dsonar.source` parameter on the command line to explicitly include the path to the Javascript files, as shown in this example:

   ```
   mvn clean install sonar:sonar
   -Dsonar.host.url=$SONAR_URL
   -Dsonar.login=$SONAR_LOGIN
   -Dsonar.password=$SONAR_PASSWD
   -Dsonar.sources=src/main/java,src/main/webapp
   -Dsonar.projectName=$SONAR_PROJECT_NAME
   -Dsonar.projectKey=$SONAR_PROJECT_KEY
   -f UiServer/pom.xml
   ```

   In the example, `-Dsonar.sources=src/main/java,src/main/webapp` is used to explicitly add Java sources from `src/main/java` and Javascript sources from `src/main/webapp`.

2. The log will show that the Javascript sources were analyzed, as were HTML and CSS files:

   ```
   [2021-04-01 21:31:30] [INFO] Sensor CSS Metrics [cssfamily]
   [2021-04-01 21:31:30] [INFO] Sensor CSS Metrics [cssfamily] (done) | time=29ms
   [2021-04-01 21:31:30] [INFO] Sensor CSS Rules [cssfamily]
   [2021-04-01 21:31:31] [INFO] 12 source files to be analyzed
   [2021-04-01 21:31:31] [INFO] 12/12 source files have been analyzed
   [2021-04-01 21:31:31] [INFO] Sensor CSS Rules [cssfamily] (done) | time=1446ms

   [2021-04-01 21:31:31] [INFO] Sensor JavaScript analysis [javascript]
   [2021-04-01 21:31:34] [INFO] 13 source files to be analyzed
   [2021-04-01 21:31:36] [INFO] 13/13 source files have been analyzed
   [2021-04-01 21:31:36] [INFO] Sensor JavaScript analysis [javascript] (done) |
   time=4971ms

   [2021-04-01 21:31:36] [INFO] Sensor HTML [web]
   [2021-04-01 21:31:36] [INFO] Sensor HTML [web] (done) | time=137ms
   ```

### Analyze a VB Studio Project That Contains Javascript Sources Only

For a VB studio project that contains just Javascript sources, you can create a Unix Shell step that downloads and installs the sonar-scanner command line tool, then uses it to perform the analysis:

1. Open the job's configuration page.
2. Click the **Steps** tab.
3. From **Add Step**, select **Unix Shell**.
4. In **Script**, enter the following commands:
   a. Download the sonar-scanner command line tool from SonarQube website:

      ```
      curl -o sonar-scanner-cli-4.6.0.2311-linux.zip https://binaries.sonarsource.com/
      Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.0.2311-linux.zip
      ```

   b. Unzip the tool:

      ```
      unzip sonar-scanner-cli-4.6.0.2311-linux.zip
      ```

   c. Run the scanner to perform the analysis, after explicitly specifying which Javascript sources you want it to analyze, as in `-Dsonar.sources=UiServer/src/main/webapp`::

      ```
      sonar-scanner-4.6.0.2311-linux/bin/sonar-scanner
      -Dsonar.host.url=$SONAR_URL
      -Dsonar.login=$SONAR_LOGIN
      -Dsonar.password=$SONAR_PASSWD
      -Dsonar.sources=UiServer/src/main/webapp
      -Dsonar.projectName=$SONAR_PROJECT_NAME
      -Dsonar.projectKey=$SONAR_PROJECT_KEY
      ```

   d. Click **Save**.
5. Run the build and check the build log to make sure that the analysis was successful:

```
[2021-04-01 22:12:20] INFO: ------------- Run sensors on module
Project1.Sonar_8_8_sonar_scanner
[2021-04-01 22:12:20] INFO: Load metrics repository
[2021-04-01 22:12:20] INFO: Load metrics repository (done) | time=486ms
[2021-04-01 22:12:22] INFO: Sensor CSS Metrics [cssfamily]
[2021-04-01 22:12:22] INFO: Sensor CSS Metrics [cssfamily] (done) | time=50ms
[2021-04-01 22:12:22] INFO: Sensor CSS Rules [cssfamily]
[2021-04-01 22:12:23] INFO: 12 source files to be analyzed
[2021-04-01 22:12:23] INFO: 12/12 source files have been analyzed
[2021-04-01 22:12:23] INFO: Sensor CSS Rules [cssfamily] (done) | time=1292ms
[2021-04-01 22:12:23] INFO: Sensor JaCoCo XML Report Importer [jacoco]
[2021-04-01 22:12:23] INFO: 'sonar.coverage.jacoco.xmlReportPaths' is not defined.
Using default locations: target/site/jacoco/jacoco.xml,target/site/jacoco-it/
jacoco.xml,build/reports/jacoco/test/jacocoTestReport.xml
[2021-04-01 22:12:23] INFO: No report imported, no coverage information will be
imported by JaCoCo XML Report Importer
[2021-04-01 22:12:23] INFO: Sensor JaCoCo XML Report Importer [jacoco] (done) |
time=3ms
[2021-04-01 22:12:23] INFO: Sensor JavaScript analysis [javascript]
[2021-04-01 22:12:26] INFO: 13 source files to be analyzed
[2021-04-01 22:12:28] INFO: 13/13 source files have been analyzed
[2021-04-01 22:12:28] INFO: Sensor JavaScript analysis [javascript] (done) |
time=4827ms
[2021-04-01 22:12:28] INFO: Sensor C# Project Type Information [csharp]
[2021-04-01 22:12:28] INFO: Sensor C# Project Type Information [csharp] (done) |
time=1ms
[2021-04-01 22:12:28] INFO: Sensor C# Properties [csharp]
[2021-04-01 22:12:28] INFO: Sensor C# Properties [csharp] (done) | time=0ms
[2021-04-01 22:12:28] INFO: Sensor JavaXmlSensor [java]
[2021-04-01 22:12:28] INFO: Sensor JavaXmlSensor [java] (done) | time=1ms
[2021-04-01 22:12:28] INFO: Sensor HTML [web]
[2021-04-01 22:12:28] INFO: Sensor HTML [web] (done) | time=151ms
```

```
[2021-04-01 22:12:28] INFO: Sensor VB.NET Project Type Information [vbnet]
[2021-04-01 22:12:28] INFO: Sensor VB.NET Project Type Information [vbnet] (done) |
time=1ms
[2021-04-01 22:12:28] INFO: Sensor VB.NET Properties [vbnet]
[2021-04-01 22:12:28] INFO: Sensor VB.NET Properties [vbnet] (done) | time=1ms
[2021-04-01 22:12:28] INFO: ------------ Run sensors on project
[2021-04-01 22:12:28] INFO: Sensor Zero Coverage Sensor
[2021-04-01 22:12:28] INFO: Sensor Zero Coverage Sensor (done) | time=20ms
[2021-04-01 22:12:28] INFO: SCM Publisher SCM provider for this project is: git
[2021-04-01 22:12:28] INFO: SCM Publisher 25 source files to be analyzed
[2021-04-01 22:12:29] INFO: SCM Publisher 25/25 source files have been analyzed
(done) | time=223ms
[2021-04-01 22:12:29] INFO: CPD Executor 5 files had no CPD blocks
[2021-04-01 22:12:29] INFO: CPD Executor Calculating CPD for 15 files
[2021-04-01 22:12:29] INFO: CPD Executor CPD calculation finished (done) | time=49ms
[2021-04-01 22:12:29] INFO: Analysis report generated in 165ms, dir size=202 KB
[2021-04-01 22:12:29] INFO: Analysis report compressed in 86ms, zip size=78 KB
[2021-04-01 22:12:30] INFO: Analysis report uploaded in 599ms
[2021-04-01 22:12:30] INFO: ANALYSIS SUCCESSFUL, you can browse http://
server123.mycorp.com:9000/dashboard?id=qa-dev_project1_1.Sonar_8_8_sonar_scanner
[2021-04-01 22:12:30] INFO: Note that you will be able to access the updated
dashboard once the server has processed the submitted analysis report
[2021-04-01 22:12:30] INFO: More about the report processing at http://
server123.mycorp.com:9000/api/ce/task?id=AXiPfrh9RniEjxk9KTc9
[2021-04-01 22:12:40] INFO: Analysis total time: 32.061 s
```

## Publish JUnit Results

JUnit test reports provide useful information about test results, such as historical test result trends, failure tracking, and so on.

If you use JUnit to run your application's test scripts, you can configure your job to publish JUnit test reports:

1. Upload your application with test script files to the Git repository.
2. Open the job's configuration page.
3. Click the **After Build** tab.
4. From **Add After Build Action**, select **JUnit Publisher**.
5. In **Include JUnit XMLs**, specify the path and names of XML files to include. You can use wildcards to specify multiple files:
   a. If you're using Ant, you could specify the path as `**/build/test-reports/*.xml`.
   b. If you're using Maven, you could specify the path as `target/surefire-reports/*.xml`.

   If you use this pattern, make sure that you don't include any non-report files.
6. In **Exclude JUnit XMLs**, specify the path and names of XML report files to exclude. You can use wildcards to specify multiple files.
7. To see and retain the standard output and errors in the build log, select the **Retain long standard output/error** check box.

   If you don't select the check box, the build log is saved, but the build executor truncates it to save space. If you select the check box, every log message is saved, but this might increase memory consumption and can slow the performance of the build executor.
8. To combine all test results into a single table of results, select the **Organize test output by parent location** check box.

If you use multiple browsers, the build executor will categorize the results by browser.

9. To mark the build as failed when JUnit tests fail, select the **Fail the build on fail tests** check box.
10. To archive videos and image files, select the **Archive Media Files** check box.
11. Click **Save.**

After a build runs, you can view its test results.

## View a Build's JUnit Test Results

See the Test Results page to examine a build's JUnit test results:

| Action | How To |
|---|---|
| View the last build's test results | 1. Open the job's details page.<br>2. Click **Tests** .<br>The Test Results page displays, showing the metrics for the most recent run. |
| View a particular build's test results | 1. Open the job's details or summary page.<br>2. In the Build History list, click the build number.<br>3. Click **Tests** . |
| View a test suite's details | On the Test Results page, click the **All Tests** toggle button. From the **Suite Name**, click the suite name.<br>The test suite details page is displayed. |
| View a test's details | Open the test suite details page and click the test name.<br>To view details of a failed test, on the Test Results page, click the **All Failed Tests** toggle button, and then click the test name. |
| View test results history | On the Test Results page, click **View Test Results History**. |

If you configure the job to archive videos and image files:

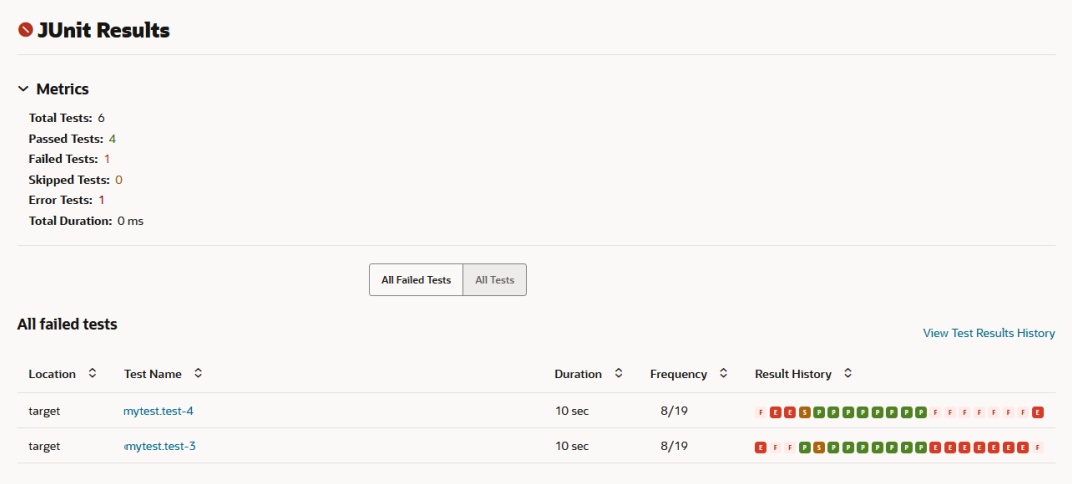
- Click **Show** to download the test image.
  Supported image formats include `.png`, `.jpg`, `.gif`, `.tif`, `.tiff`, `.bmp`, `.ai`, `.psd`, `.svg`, `.img`, `.jpeg`, `.ico`, `.eps`, and `.ps`.
- Click **Watch** to download the test video file.
  Supported video formats include `.mp4`, `.mov`, `.avi`, `.webm`, `.flv`, `.mpg`, `.gif`, `.wmv`, `.rm`, `.asf`, `.swf`, `.avchd`, and `.m4v`.

### Access and Analyze Your JUnit Test Result History

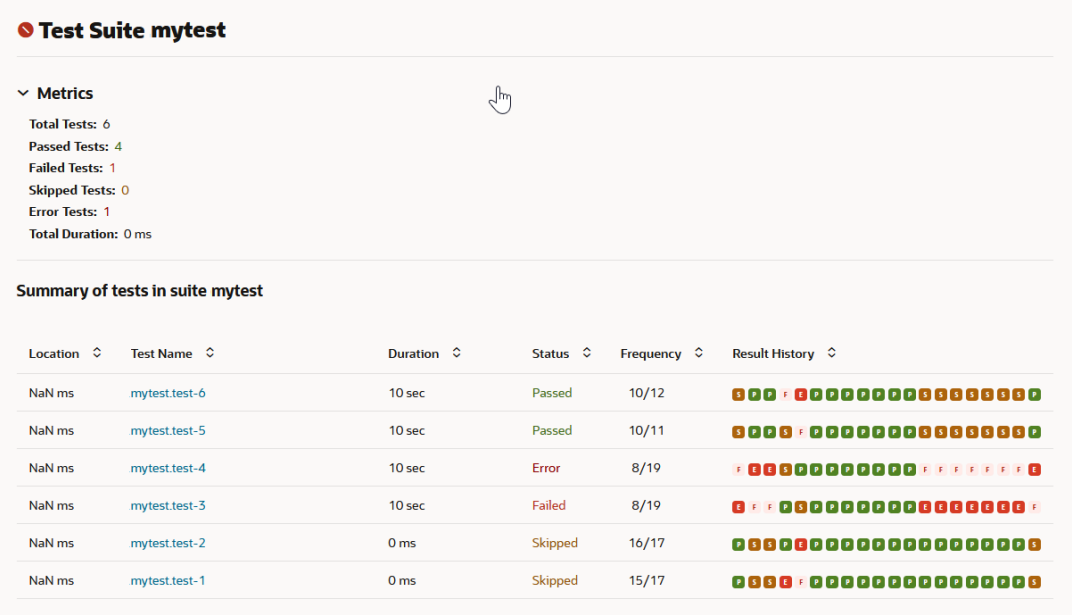
Some notable changes have come to the Test result history pages. The Frequency column is new. It shows the number of passing results compared to the number of times the test has been run. In addition, the Order column's name was changed to Result History.

You can see the Test result history in three places:

- In Failed tests




- In the Test information listed in the suite




- In the Test details page

# Use the Xvfb Wrapper

Xvfb is an X server that implements the X11 display server protocol and can run on machines that don't have physical input devices or a display.

## Set Up a VM Build Executor and a Build Executor Template with Xvfb

Before you can use Xvfb in a build step, your organization administrator must first create a build executor template with the minimum required software and then add a VM build executor that uses the build executor template. Your organization administrator can create a new build executor template or use any existing Oracle Linux 7 build executor template.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

After the organization administrator adds a VM build executor to the build executor template, you can create and configure a job to use that build executor template and Xvfb.

## Configure a Job to Run Xvfb

Create and configure a job that runs Xvfb commands:

1. Open the job's configuration page.

   If you're creating a job, in **Template** in the New Job dialog box, select the Xvfb build executor template. Proceed to step 4.

2. Click **Settings** .
3. In the **Software** tab, select the Xvfb build executor template or any minimum required build executor template.
4. Click **Configure** .
5. Click the **Before Build** tab.
6. From **Add Before Build Action**, select **Xvfb Wrapper**.
7. In **Display Number**, specify the ordinal number of the display the Xvfb server is running on. The default value is 0. If left empty, a random number is chosen when the build runs.
8. In **Screen offset**, specify the offset for display numbers. The default value is 0.
9. In **Screen Size (WxHxD)**, specify the resolution and color depth of the virtual frame buffer in the WxHxD format. The default value is `1024x758x24`.
10. In **Additional options**, specify additional Xvfb command line options, if necessary. The default options are `-nolisten inet6 +extension RANDR -fp /usr/share/X11/fonts/misc`.
11. In **Timeout in seconds**, specify the timeout duration for the build to wait before returning control to the job. The default value is 0.

12. If you don't want to log the Xvfb output in the build log, deselect the **Log Xvfb output** check box. The check box is selected by default.
13. If you don't want to keep the Xvfb server running for post-build steps, deselect the **Shutdown Xvfb with whole job, not just with the main build action** check box. The check box is selected by default.
14. Click **Save**.

## Publish Javadoc

If your application source code files are configured to generate Javadoc, you can configure a job to publish Javadocs when a build runs:

1. Open the job's configuration page.
2. Click the **After Build** tab.
3. From **Add After Build Action**, select **Javadoc Publisher**.
4. In **Javadoc Directory**, specify the workspace path where the build executor would publish the generated Javadoc. By default, the path is set to `target/site/apidocs` .
5. To configure the build executor to retain Javadoc for each successful build, select the **Retain Javadoc for each build** check box.

   You may want to enable this otion if you have a need to browse Javadoc of older builds, but be cognizant that this practice will consume more disk space that not retaining those older Javadocs. By default, the check box isn't selected.
6. Click **Save**.

## Archive Artifacts

You can manually download archived artifacts. By default, build artifacts are kept as long as the build log is.

If you want a job's builds to archive artifacts, you can do so as an after build action:

1. Open the job's configuration page.
2. Click **Configure** .
3. Click the **After Build** tab.
4. Click **Add After Build Action** and select **Artifact Archiver**.
5. In **Files to archive**, enter a comma-separated list of file paths, such as `env/,SQL/,target/`, using the path relative to the workspace, not the full file path.

   You can use wildcards to archive multiple files. For example, you could use `env/**` to archive all files in all subdirectories of the `env` directory. Or, you could use `env/**/*.bin` to archive all files that end with the `.bin` extension in all subdirectories of the `env` directory.

   Here are some more examples:

   - `**/*` or `**` archives all files in all directories and subdirectories
   - `**/*.sql` archives all files that have a `.sql` file extension, in all directories and subdirectories
   - `env/*` matches all files in the `env` folder itself, but doesn't include any files in any subdirectories

The patterns can be more complex too. For example, you could use `**/target/*.jar` to archive `.jar` files in all `target` directories your workspace.

6. In **Files to exclude**, enter a comma-separated list of files, including the path, as described in the previous step.

   A file that matches the exclude pattern won't be archived even if it matches the pattern specified in **Files to archive**.

7. If your application is a Maven application and you want to archive Maven artifacts, select **Archive Maven Artifacts**.

   To archive the Maven POM file along with the Maven artifacts, select **Include POM.xml**.

8. Click **Save**.

### Discard Old Builds and Artifacts

To save storage space, you can configure a job to discard its old builds and artifacts:

1. Open the job's configuration page.
2. Click **Settings** .
3. Click the **General** tab, if necessary.
4. If not selected, select **Discard Old Builds**.
5. Configure the discard options.
6. Click **Save**.

Old builds will be discarded after you save the job configuration and after a job has been built.

## Copy Artifacts from Another Job

If your application depends on artifacts from another job, you can configure the job to copy those artifacts when a build is run:

1. Open the job's configuration page.
2. Click **Configure** .
3. Click the **Before Build** tab.
4. Click **Add Before Build Action** and select **Copy Artifacts**.
5. In **From Job**, select the job whose artifacts you want to copy.
6. In **Which Build**, select the build that generated the artifacts.
7. If you select the **Use last successful build if not run in pipeline** option, the last successful (other) build will be used if this build isn't run in a pipeline.

   The build will fail if you don't select the option but do select the upstream build in the previous step and don't run the build in a pipeline.

8. In **Artifacts to copy**, specify the artifacts to copy. When a build runs, the artifacts are copied with their relative paths.

   If you don't specify a value, the build will copy all artifacts. The `archive.zip` file is never copied.

9. In **Target Directory**, specify the workspace directory where the artifacts will be copied.

   Be sure that directory names don't contain a forward slash or null character. You can optionally include parameters in the path, such as:

```
data/$myDir/$myDir2/stuff
```

If **Target Directory** is left empty, the artifacts will be copied to the root directory of the workspace.

10. To flatten the directory structure of the copied artifacts, select **Flatten Directories**.
11. By default, if a build can't copy artifacts, it'll be marked as failed. If you don't want the build to be marked as failed, select **Optional (Do not fail build if artifacts copy failed)**.
12. Click **Save**.

## Configure General and Advanced Job Settings

You can configure several general and advanced job settings, such as name and description, the Java version used in the build, discarding old and running concurrent builds, adding timestamps to the build log, and more:

> **Note:**
>
> If you're trying to access a renamed job by using a link from the activities feed (or from a bookmarked link), you will be unable to do so because that job no longer exists. A page with a message indicating that there was an error loading the Jobs page is displayed, along with some text that says that the job could not be found because it may have been deleted or renamed. Click the **Take me back to Jobs** button and the Builds page's **Jobs** tab will be displayed.

| Action | How To |
|---|---|
| Rename the job or update its description | 1. Open the job's configuration page.<br>2. Click **Settings** ⚙.<br>3. Click the **General** tab.<br>4. In **Name** and **Description**, update the job name and description.<br>5. Click **Save**. |
| Check the software available on the job's build executor template | 1. Open the job's configuration page.<br>2. Click **Settings** ⚙.<br>3. Click the **Software** tab.<br>4. See the software and their versions. You can change versions of some software, such as Java SE. Select the version from the drop-down list.<br>5. Click **Save**. |
| Run concurrent builds | 1. Open the job's configuration page.<br>2. Click **Settings** ⚙.<br>3. Click the **General** tab.<br>4. Select the **Execute concurrent builds if necessary** check box.<br>By default, only one build of a job runs at a time. The next build runs after the running build finishes.<br>5. Click **Save**. |

| Action | How To |
| --- | --- |
| Set a quiet period | 1. Open the job's configuration page.<br>2. Click **Settings** .<br>3. Click the **Advanced** tab.<br>4. Select the **Quiet period** check box and specify the amount of time (in seconds) a new scheduled build of the job will wait before it runs.<br>If the build executor is busy with too many builds, setting a longer quiet period can reduce the number of builds.<br>5. Click **Save**. |
| Set a retry count | 1. Open the job's configuration page.<br>2. Click **Settings** .<br>3. Click the **Advanced** tab.<br>4. Select the **Retry Count** check box.<br>5. In **Build Retries** specify the number of times the build executor tries the build. By default, the build executor tries five times to run a build that fails. You can increase or decrease the count.<br>In **SCM Retries** specify the number of times the build executor tries the build to checkout files from the Git repository. You can increase or decrease the default count.<br>6. Click **Save**. |
| Abort a build if it's stuck for some duration | 1. Open the job's configuration page.<br>2. Click **Settings** .<br>3. Click the **Advanced** tab.<br>4. Select the **Abort the build if it is stuck** check box.<br>5. In **Hours** and **Minutes**, specify the duration.<br>If a build doesn't complete in the specified amount of time, the build is terminated automatically and marked as aborted. Select the **Fail the build on abort** check box to mark the build as failed, rather than aborted.<br>6. Click **Save**. |
| Remove timestamps from the build log | 1. Open the job's configuration page.<br>2. Click **Settings** .<br>3. Click the **Advanced** tab.<br>4. Deselect the **Add Timestamps to the Console Output** check box.<br>By default, build logs are timestamped. This selection configures the job to remove them from the log.<br>5. Click **Save**. |

| Action | How To |
|---|---|
| Set the maximum size of the console log | 1. Open the job's configuration page.<br>2. Click **Settings** .<br>3. Click the **Advanced** tab.<br>4. In **Max Log Size (MB)**, set the size. The default value is 50 MB and the maximum value is 1000 MB.<br>5. Click **Save**. |

## Manage Build Actions

You can manage build actions in job configurations, including disabling/enabling, reordering, or removing build actions. These operations apply to build actions on the **Git**, **Parameters**, **Before Build**, **Steps**, and **After Build** tabs (under Configure) and build actions on the **Triggers** tab (under Settings).

Here are the job configuration build actions that you can manage:

| Action | How To |
|---|---|
| Disable a build action | In any tab on the Job Configuration page, for any enabled build action, change the toggle from **Enabled** to **Disabled** and click **Save**.<br>Use this toggle to disable the build step or action temporarily. If a step or action is disabled, it'll be skipped when the job is run.<br>If you see a validation error while trying to save a job configuration after adding, then disabling, a new build action, make sure that you filled out all required fields. Required fields are still required, even though the build action is disabled. You must either fill out the required field(s) in the disabled build action or remove the build action before trying to resave the job configuration. |
| Enable a disabled build action | In any tab on the Job Configuration page, for any disabled build action, change the toggle from **Disabled** to **Enabled** and click **Save**. |
| Expand/collapse build steps | In the **Steps** tab on the Job Configuration page, mouse over any build step header to collapse or expand the build step. |
| Reorder build actions | In any tab on the Job Configuration page that has multiple build actions, use the **Up**  and **Down**  icons with any build action to reorder and then click **Save**. |
| Remove a build action | In any tab on the Job Configuration page, for any enabled or disabled build action, click **Remove** and then click **Save**. |

## Change a Job's Java Version

Change the Java version used in a job:

1. Open the job's configuration page.
2. Click **Settings** .
3. Click the **Software** tab.
4. In **Available Software**, from the **Java** drop-down list, select the Java SE version (1.8.x, 11.x, 17.x, or 18.x) that you want to use.

> **Note:**
>
> Java 1.8.x is deprecated in the 22.10.0 release. Java 17.x has been added to all system build executor templates. Oracle recommends that you upgrade any of your jobs that currently use Java 1.8.x to this version.

Before changing a job's Java version, ask your organization administrator to add the version you want to use in the job's build executor template if you don't see the version already there. If an organization administrator adds multiple version of Java to a build executor template, users can select the Java version they want to use in a job from the job's configuration page.

Instead of selecting Java, you could select Java 1.8.x (Graal VM). GraalVM is a universal virtual machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Groovy, Kotlin, Clojure, and LLVM-based languages such as C and C++. To learn more about GraalVM, see `https://www.graalvm.org/docs/`.

5. Click **Save**.

## Change a Job's Build Executor Template

Contact the organization administrator to create a build executor template and add software packages.

The organization administrator creates the build executor template, selects and adds software packages to it, then creates an instance of the VM Build Executor. You specify the template when you create or configure your job. Then, when you run the job, the software that is specified in the build executor template is installed on the VM build executor.

> **Note:**
>
> To find your organization administrator, click **Contacts** under your user profile. Your administrator, or a list of administrators, will display.

See Create and Manage Build Executor Templates in *Administering Visual Builder Studio*.

Here's how you can change your job's build executor template after you create the job:

1. Open the job's configuration page.
2. Click **Settings** .
3. Click the **Software** tab.
4. Select the build executor template that you want to use for your builds.
5. Click **Save**.

# Run a Build

You can run a job's build manually or configure the job to trigger it automatically on an SCM commit or according to a schedule:

| Action | How To |
|---|---|
| Run a build manually | Open the job's details page and click **Build Now**.<br><br>**Note:**<br>If you've set up this job to run automatically on SCM commit, and you have specified Include/Exclude criteria, we'll show you a list of branches that match the pattern so you can pick one to build. For example, if your Include criteria is `branch*` and `test*`, then we'll show you all the branches that begin with those terms. If you specified Exclude instead, we'll show all the branches that *don't* begin with those terms—likely quite a few. If we don't find a branch that matches, we'll build your default branch instead.<br><br>You can also run a job's build from the **Jobs** tab on the Builds page. In the selected job's row, click the **Actions** ⋮ menu and select **Build**. |
| Run a build on SCM commit | See Run a Build Automatically on an SCM Commit. |
| Run a build on a schedule | See Trigger a Build Automatically on a Schedule. |

A job that takes more than eight hours to build will fail. If you know that a job's processes will take more than eight hours to execute, you should distribute those processes in multiple jobs and run them together in a pipeline.

# View a Job's Builds and Reports

From the Builds page, click a job name to open its details page, from which you can view a job's builds, reports, and build history, or perform actions such as running a build or configuring the job.

## View a Build's Logs and Reports

A build generates various types of reports and logs, such as SCM changes, test results, and action history. You open these reports from the Job Details or Build Details page by clicking the report icon and viewing its details.

Here are the types of reports that are generated by a build:

| Log/Report | Description |
|---|---|
| **Changes** | View all files that have changed in the build.<br>When a build is triggered, the build system checks the job's Git repositories for any changes to the SCM. If there are any updates, the SCM Change log displays the files that were added, edited or removed. |
| **Artifacts** | View the latest archived artifacts generated by the build. |
| **Javadoc** | View the build's Javadoc output.<br>The report is available only if the application's build generated Javadoc. |
| **Tests** | View the log of build's JUnit test results.<br>To open the Test Suite details page, on the Test Results page, click the **All Tests** toggle button and click the suite name in the **Suite Name** column.<br>To view details of a test, on the Test Results page, click the **All Failed Tests** toggle button and then click the test name link in the **Test Name** column. You can also click the **All Tests** toggle button, open the test suite details page, and then click the test name link in the **Test Name** column. |
| **Build Log** | View the last build's log. In the log page, review the build log. If the log is displayed partially, click the **Full Log** link to view the entire log. To download the log as a text file, click the **Download Log** link. |
| **Git Log** | View the Git SCM polling log of the builds that displays the log of builds triggered by SCM polling. The log includes scheduled builds and builds triggered by SCM updates.<br>In the Job Details page of a job, click **Latest SCM Poll Log** to view the Git SCM polling log of the last build. |
| **Audit** | View the Audit log of user actions.<br>You can use the Audit log to track the user actions on a build. Use the log to see who performed particular actions on the job. For example, you can see who canceled a build of the job, or who disabled the job and when was it disabled. |
| **SonarQube** | View the SonarQube analysis report for the job. |
| **Vulnerabilities** | View the Security Vulnerabilities report that identifies direct and transitive dependencies in the job's Maven, Node.js, Javascript, and/or Gradle projects. |

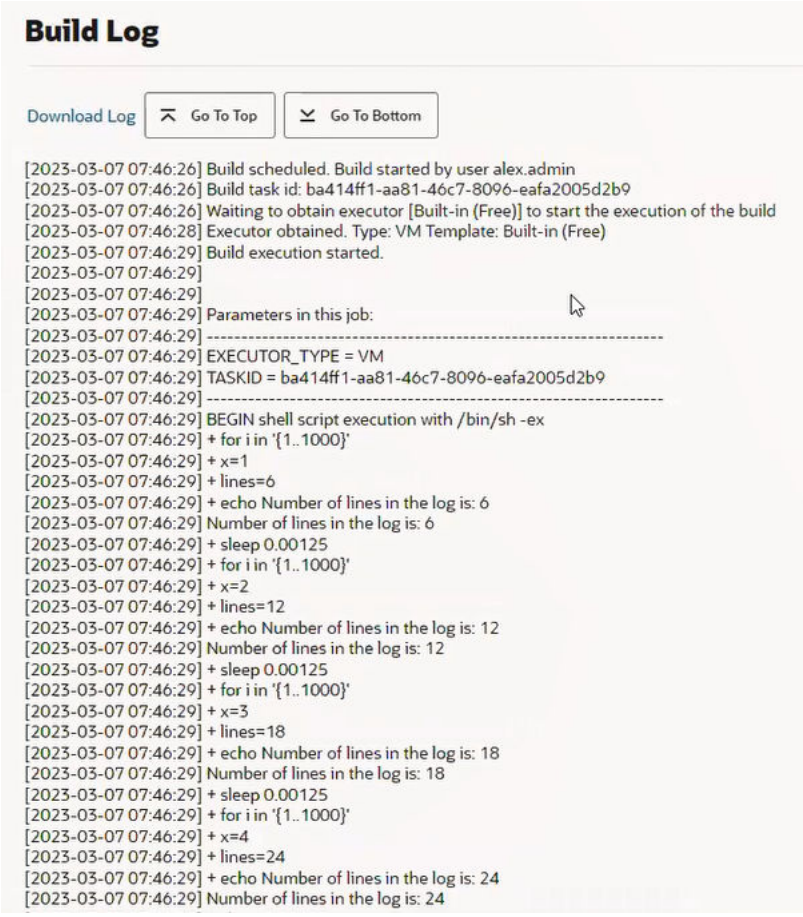
## Examine Large Log Files

Large log files can be quite cumbersome to work with. Often, log files are too big to display in browser, so you need to download them and open them in an editor. You may find that the key parts of the log are in the beginning of the file or at the end. Instead of endlessly scrolling to get to the bottom of the file, or back to the top of it again, VB Studio provides a few features that make this task much easier.

The most interesting information about a build is found in the beginning and end of the log. If, after you went to the bottom of the file, you scrolled up to see more information about the phase that failed, and the information you're looking for isn't in those lines, you'll need to download the log and open it in an editor.

1. From the job's details page, click **Build Log** to open the build log.

   The log file opens and displays the lines (up to 100,000) as it lazy loads.




2. Scroll down to examine the log or use the floating menu at the top.

The floating menu provides these options:

- The **Download Log** link is always available, so you can download the log locally and open it in your text editor. This is useful if you want to search for information.
- Click the **Go To Top** button to go to the beginning of the log.
- Click the **Go To Bottom** button to go to the end of the log.

3. Click the **Go To Bottom** button and check the status of the build.

4. If your log file exceeds 100,000 lines, scroll up from the end of the log and you'll see a message indicating how many lines were skipped.

5. Click the **Download Log** link and download the log file to your computer where you can open it in your text editor and scroll through the lines that were skipped in the log.
6. Click the **Go To Top** button to return to the beginning of the log and examine the settings and parameters that were used in the build.

The log lazy loads as you scroll. You can see as much as you want to and can scroll down to about 100,000 lines. If you scroll that far, you'll see the skipped lines notice plus the last 100 lines. If you click the **Go To Bottom** button, all those lines will be loaded, the skipped lines notice will be displayed, and then the last 100 lines will be shown.

Clicking the **Go To Top** button at that point will not change what is loaded. You'll still see the first 100,000 lines plus the last 100.

## View a Project's Build History

The Recent Build History page displays builds of all the project's jobs.

To see the build history, click the **View Recent Build History** link in the **Job Queue** panel on the **Builds** page. The history page displays the last 50 builds of the project. Click any job name to open its details page or click any build number to open its details page. Click to view the build log.

> **Tip:**
>
> To sort the table data by a column, right-click inside the build history table column and select the sort order from the **Sort** context menu.

# View a Job's Build History

A job's build history can be viewed in the Build history section of the Job Details page. It displays the status of running builds, and completed job builds in descending order (most recent first) along with their build numbers, date and time, and a link to the build's console output.

The build history shows how the build was triggered as well as its status, build number, and date-time stamp. In this view, you can click the **Actions** ⋯ menu and choose **View Log** to view the build log or **Delete** to delete the build.

When you review the build history, take note of these things:

- In the **By** column, the icons indicate the following:

| This icon ... | Indicates: |
|---|---|
| **User** | The build was initiated by a user. |
| **SCM Change** | The build was triggered by an SCM change. |
| **Pipeline** | The build was initiated by a pipeline. Click to open the build's pipeline instance. |
| **Periodic Build Trigger** | The build was triggered by a periodic build trigger. |
| **Build System** | The build was started or rescheduled by the build system. |

- Discarded and deleted jobs are not included in this list.
- In the **Build** column, an * in the build number indicates the build is annotated with a description. Mouse over the build number to see the description. You can also click a build number to see more detailed information about it.
- Hover over a row in the **Started** column to see more detailed information about the job, such as:
  - What time the job was queued
  - The exact time the job started
  - What time the job completed
- If a running build remains stuck in the queued state for a long time, you can mouse over the `Queued` status to display a message about the problem.

  If the build uses a VM build executor, you can contact the organization administrator to check its status.
- To sort the table data in ascending or descending order, click the header column name and then click the Previous or Next icon in the column header.

  As an alternative, you can right-click inside the table column and select the sort order from the **Sort** context menu.
- Only project members can delete builds. Non-members cannot.

## View a Job's History by User Name

Use the Audit log to see who impacted a job and when. For example, you can see who canceled a build of the job, or who disabled the job and when it was disabled.

To open the Audit log, click **Audit** on the job's details page.
The log displays information such as:

- Who initiated a build. The username and service instance name is shown in the Audit log or pipeline name.
- If a build job has been deleted, there won't be an Audit log, but the activity stream will show that the build job has been removed.
  A build can also be triggered by a timer, a commit to a Git repository, or an upstream job.
- Time and date the build was queued, started, and completed
- Whether the result was successful, rescheduled, aborted, or failed
- The duration of the build, in seconds
- User name and service instance name of the user who changed the configuration of a job, plus the timestamp
- User name and service instance name of the user who created, renamed, enabled, or disabled a job, plus the timestamp

## View a Build's Details

A build's details page shows its status, plus the time and date the build was queued, started, and completed, as well as total duration. The page also shows links to open build reports, download artifacts, and logs. To open a build's details page, click the build number in the Build History.

You can perform these common actions from a build's details page:

| Action | How To |
|---|---|
| Keep a build forever | A build that's marked "forever" isn't removed if a job is configured to discard old builds automatically. You can't delete it either.<br>To keep a build forever, click **Configure**, select the **Keep Build Forever** check box, and click **Save**. |
| Add a name and description to a build | Adding a description and a name is especially helpful if you mark a particular build to keep it forever and not get discarded automatically. When you add a description to a build, an * is added to the build number in the Build History table.<br>To keep a build forever, click **Configure**. In **Name** and **Description**, enter the details, and click **Save**. |
| Open a build's log | Click **Build Log**. |
| Delete a build | Click **Delete**. |

## Download Build Artifacts

Build artifacts are displayed in a directory tree structure. Click the link to download parts of the tree, including individual files, directories, and subdirectories.

If the job is configured to archive artifacts, you can download them to your computer and then deploy the artifact(s) to your web server:

1. Open the job's details page.
2. Click **Artifacts** .

   To download a particular build's artifacts, in the Build History, click the build number, and then click **Artifacts** .
3. Expand the directory structure and click the artifact link (file or directory) to download it.

   To download a zip file of all artifacts, click **(All files in zip)** .
4. Save the file to your computer.

## Watch a Job

You can subscribe to email notifications that you'll receive when a build of a job succeeds or fails.

To get email notifications, enable them in your user preferences, and then set up a watch on the job:

| Action | How To |
|---|---|
| Enable your email notifications preference | In your user preferences page, select the **Build Activities** check box. |
| Watch a job | 1. Open the job's details page.<br>2. Click the **On** toggle button, if necessary.<br>3. Click **CC Me**.<br>4. In the CC Me dialog box, to receive email when the build is successful, select the **Successful Builds** check box. Select **Failed Builds** to receive email when the build fails.<br>5. Click **OK**.<br><br>**Note:**<br>You can use the **Build Notification** tab on the Builds page to manage email notifications for multiple jobs. See Create and Manage Jobs. |
| Disable email notifications of the job to all subscribed members | 1. Open the job's details page.<br>2. Click the **Off** toggle button, if necessary. |

## Build Executor Environment Variables

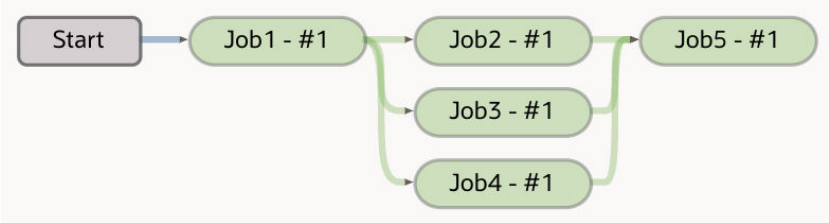
When you run a build job, you can use environment variables in your shell scripts and commands to access software on the VM build executor.

To use a variable, use the `$VARIABLE_NAME` syntax, such as `$BUILD_ID`.

### Common Environment Variables

Here are some common environment variables:

| Environment Variable | Description |
|---|---|
| *BUILD_ID* | The current build's ID. |
| *BUILD_NUMBER* | The current build number. |
| *BUILD_DIR* | The build output directory. |
| *JOB_NAME* | The name of the current job. |
| *STARTED_BY* | The user or system that initiated the build. Returned values can be USER, SYSTEM: PIPELINE, SYSTEM: SCM, or SYSTEM: PERIODIC-TRIGGER. For example, if you start a pipeline, SYSTEM: PIPELINE will be followed by the pipeline name that started the build. If you add this line to a shell script in a Unix Shell build step in the job whose log you are examining:<br><br>`echo "Started By =  " $STARTED_BY`<br><br>The log file will show this information:<br><br>`[2023-08-10 18:22:31] Build scheduled. Build started by pipeline Protection02`<br>`echo 'Started By =  ' SYSTEM-PIPELINE: Protection02`<br><br>To get the following, a timestamp that shows when the build started and a build number, you need to add two additional environment variables in your shell script:<br><br>`echo "TimeStamp =  "  $BUILD_ID`<br>`echo "Build Number =  " $BUILD_NUMBER` |
| *STARTED_BY_ORIGIN* | Who or what started or triggered the pipeline run. The value represents what the origin of the entire pipeline run was and what was the origin of the build happening. It can be a USER, SYSTEM: SCM, or SYSTEM: PERIODIC-TRIGGER. Whatever started the first build that started the pipeline is returned. |

| Environment Variable | Description |
|---|---|
| *STARTED_BY_UPSTREAM* | A list of the running build's direct upstream jobs that started it with build numbers. An asterisk (*) is shown to the right of the build that triggered the current build. In the current job's build log, the Started by line at the top shows the incoming upstream job(s), with asterisk (*) after the trigger that started the build of the current job.<br>Let's look at an example, using this pipeline, MyPipeline.<br>  <br><br>The log for Job5, which executed after the last build of the three parallel jobs (Job2, Job3, and Job4) finished running, shows that Job3 of the three parallel jobs was built last and triggered the build for Job5, the last job in the pipeline.<br><br>**[2023-08-23 19:20:40] Build scheduled. Build started by pipeline MyPipeline [Upstream Jobs: Job2 Build: #1, Job3 Build: #1*, Job4 Build: #1]**<br>[2023-08-23 19:20:40] Build task id: cde8a1b7-954b-4802-bf21-890cb5d559e4<br>[2023-08-23 19:20:40] Waiting to obtain executor [DCS QA Req Comp Linux 7] to start the execution of the build<br>[2023-08-23 19:20:41] Executor obtained. Type: VM Template: DCS QA Req Comp Linux 7 Executor VM ID: 3982<br>[2023-08-23 19:20:42] Build execution started.<br>[2023-08-23 19:20:42]<br>[2023-08-23 19:20:42]<br>[2023-08-23 19:20:42] Parameters in this job:<br>[2023-08-23 19:20:42]<br>---------<br>[2023-08-23 19:20:42] EXECUTOR_TYPE = VM<br>[2023-08-23 19:20:42] TASKID = cde8a1b7-954b-4802-bf21-890cb5d559e4<br>[2023-08-23 19:20:42]<br>---------<br>[2023-08-23 19:20:42] BEGIN shell script execution with /bin/sh -ex<br>[2023-08-23 19:20:42] + echo 'Started By Upstream = ' '[Job2' Build: '#1,' Job3 Build: '#1*,' Job4 Build: '#1]'<br>**[2023-08-23 19:20:42] Started By Upstream = [Job2 Build: #1, Job3 Build: #1*, Job4 Build: #1]**<br>[2023-08-23 19:20:42] END shell script execution<br>Build log size 706 B (706)<br>[2023-08-23 19:20:50]<br>[2023-08-23 19:20:50] Build completed. |

| Environment Variable | Description |
|---|---|
| | `[2023-08-23 19:20:50] Status: DONE Result: SUCCESSFUL Duration: 8.0 sec`<br><br>The first line in bold indicates that the pipeline run was started by the pipeline (MyPipeline), a trio of jobs were built in parallel, and the build of Job3 finished after the other two, so it has the asterisk next to it, indicating that it was the trigger for starting the build of Job5. The other bold line, the Started By Upstream line, shows the same information. This information comes from a shell script that was added to a Unix Shell build step in Job5. You can see in the log where the script was run.<br><br>**Note:**<br>Only the direct upstream builds are listed, as they are the direct conditions to start the job whose log is being examined. For example, for the pipeline with Job1 followed by Job2 followed by Job3, the log for Job3 will list only Job2, which triggers the build of Job3. If all the pipeline jobs build consecutively, not in parallel, when you examine the Started by line in the log for the current build, you'll see just one build listed, with an asterisk next to it, indicating that it was the build that triggered the current build. |
| *HTTP_PROXY* | The HTTP proxy for outgoing connections. |
| *HTTP_PROXY_HOST* | The HTTP proxy host for outgoing connections. |
| *HTTP_PROXY_PORT* | The HTTP proxy port for outgoing connections. |
| *HTTPS_PROXY* | The HTTPS proxy for outgoing connections. |
| *HTTPS_PROXY_PORT* | The HTTPS proxy host for outgoing connections. |
| *HTTPS_PROXY_PORT* | The HTTPS proxy port for outgoing connections. |
| *NO_PROXY* | A comma separated list of domain names or IP addresses for which the proxy should not be used. You can also specify port numbers. |
| *NO_PROXY_ALT* | A pipe ( \| ) separated list of domain names or IP addresses for which the proxy should not be used. You can also specify port numbers. |
| *PATH* | The *PATH* variable, set in the VM build executor, specifies the path of executables in the VM build executor.<br>Executables from the software packages are available on the VM build executor's *PATH* variable, which is set to `/usr/bin`, and can be invoked directly from the **Unix Shell**. You should use the *PATH* variable and other environment variables to access the installed software.<br>See Software for Build Executor Templates in *Administering Visual Builder Studio* for more information. |
| *WORKSPACE* | The absolute path of the VM build executor's workspace. |

### Software Environment Variables

| Environment Variable | Description |
|---|---|
| *GRADLE_HOME* | The path of the Gradle directory. |

| Environment Variable | Description |
|---|---|
| *JAVA_HOME* | The path of the directory where the Java Development Kit (JDK) or the Java Runtime Environment (JRE) is installed.<br>If your job is configured to use a specific Java version, the build executor sets the variable to the path of the specified Java version. When the variable is set, *PATH* is also updated to have *$JAVA_HOME*`/bin`. |
| *NODE_PATH* | The path of the Node.js modules directory. |

**Tip:**

- You can run the `env` command as a Shell build step to view all environment variables of the build executor.
- Some Linux programs, such as curl, only support lower-case environment variables. Change the build steps in your job configuration to use lower-case environment variables:

```
export http_proxy="$HTTP_PROXY"
export https_proxy="$HTTPS_PROXY"
export no_proxy="$NO_PROXY"
curl -v http://www.google.com
```

### Software Environment Variables for SOA

To access SOA, use these environment variables that are defined for you when you include a SOA package in your template:

- Use *JAVACLOUD_HOME* variables to access the Java SDK
- Use *MIDDLEWARE_HOME* variables to access Oracle Fusion Middleware.
  The `MIDDLEWARE_HOME` directory includes the WebLogic Server installation directory and the Oracle Common library dependencies.
- Use *WLS_HOME* variables to access the WebLogic server binary directory

Make sure that you have the right software available in your job's build executor template:

| Software | Variables |
|---|---|
| SOA 12.2.1.4 | *JAVACLOUD_HOME_SOA*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/jdeveloper/cloud/oracle-javacloud-sdk/lib<br>*JAVACLOUD_HOME_SOA_12_2_1*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/jdeveloper/cloud/oracle-javacloud-sdk/lib<br>*MIDDLEWARE_HOME_SOA*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0<br>*MIDDLEWARE_HOME_SOA_12_2_1*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0<br>*ORACLE_HOME*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/jdeveloper<br>*ORACLE_HOME_SOA*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/jdeveloper<br>*ORACLE_HOME_SOA_12_2_1*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/jdeveloper<br>*WLS_HOME_SOA*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/wlserver<br>*WLS_HOME_SOA_12_2_1*=/opt/Oracle/MiddlewareSOA_12.2.1.4.0/wlserver |

### Software Environment Variables for Oracle JDeveloper

To access Oracle JDeveloper, use these environment variables that are defined for you when you include a JDeveloper package in your template:

- Use *JAVACLOUD_HOME* variables to access the Java SDK
- Use *MIDDLEWARE_HOME* variables to access Oracle Fusion Middleware. The `MIDDLEWARE_HOME` directory includes the WebLogic Server installation directory and the Oracle Common library dependencies.
- Use *WLS_HOME* variables to access the WebLogic server binary directory.

Make sure that you have the right software available in your job's build executor template:

| Software | Variables |
|---|---|
| JDeveloper 12.2.1.4 | *JAVACLOUD_HOME*=/opt/Oracle/Middleware_12.2.1.4.0/jdeveloper/cloud/oracle-javacloud-sdk/lib<br>*MIDDLEWARE_HOME*=/opt/Oracle/Middleware_12.2.1.4.0<br>*ORACLE_HOME*=/opt/Oracle/Middleware_12.2.1.4.0/jdeveloper<br>*WLS_HOME*=/opt/Oracle/Middleware_12.2.1.4.0/wlserver |

# Run Jobs in a Pipeline

You can create, manage, and configure job pipelines from the **Pipelines** tab of the **Builds** page.

# What Is a Pipeline?

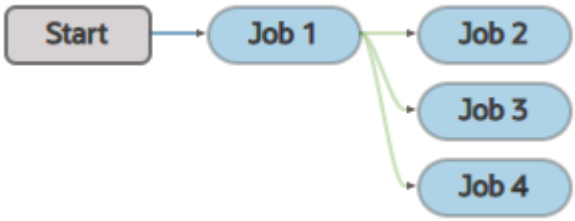
A Pipeline lets you define dependencies of jobs and create a path or a chain of builds. You can use pipelines to run continuous integration jobs.

To create a pipeline, you design a pipeline diagram where you define the dependencies of jobs. When you create a dependency of a job over another, you define the order of automatic builds of the dependent jobs. If required, the dependent jobs can be configured to use artifacts of the parent job too.

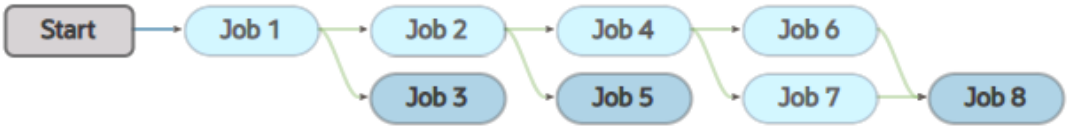
> **Note:**
>
> YAML pipelines are created in a .yml file that's stored in any of the project's Git repositories. You can view a YAML pipeline diagram layout, but you can't configure a YAML pipeline in the pipeline designer. See Configure Jobs and Pipelines with YAML.

In this diagram, Job 2 depends on Job 1 and runs after Job 1 is successful.

In this diagram, Job 2, Job 3, and Job 4 depend on Job 1 and run after Job 1 is successful. Job 2, Job 3, and Job 4 are scheduled in parallel. They can all run at the same time.




This diagram shows a complex example.

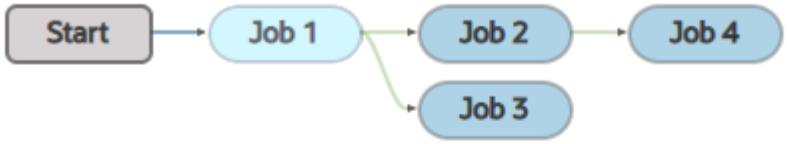



The above diagram defines these dependencies:

- Job 2 and Job 3 depend on Job 1 and run after Job 1 is successful
- Job 4 and Job 5 depend on Job 2 and run after Job 2 is successful
- Job 6 and Job 7 depend on Job 4 and run after Job 4 is successful
- Job 8 depends on Job 6 and Job 7 and runs after Job 6 and Job 7 are successful

- Job 1 is the initial job. Running it triggers a chain. All jobs after it in the chain (Job 2 through Job 8) run automatically, one after the other.

You can create multiple pipeline diagrams of jobs. If multiple pipelines have some common jobs, then multiple builds run some of those jobs. For example, in this figure, Pipeline 1 and Pipeline 2 have common jobs:




Pipeline 1

Pipeline 2

Let's assume that Pipeline 1 is defined first and Pipeline 2 is defined second. If both pipelines are triggered, the builds run in this order:

1. A build of Job 1 runs.
2. Builds of Job 2 and Job 3 of Pipeline 1 get in the build executor queue after Job 1 is successful. A build of Job 2 of Pipeline 2 also gets in the build executor queue after Job 1 is successful.
3. Builds of jobs in build executor queue run on first-come first-served basis. So, Job 2 and Job 3 of Pipeline 1 run first. Let's call the build as Build 1 of Job 2 and Job 3. Then, another build of Job 2 of Pipeline 2 runs. Let's call it Build 2 of Job 2.
4. A build of Job 4 of Pipeline 1 joins the build executor queue as soon as Job 2 is successful. A build of Job 3 of Pipeline 2 also joins the queue when Job 2 is successful.
5. As soon as the build executor is available, Build 1 of Job 4 runs and Build 2 of Job 3 also runs. Remember that Build 1 of Job 3 ran in Pipeline 1.
6. After a build of Job 3 of Pipeline 2 is successful, a build of Job 4 of Pipeline 2 joins the queue and runs when the build executor is available. Remember that this is Build 2 of Job 4 as Build 1 ran in Pipeline 1.

While creating multiple pipeline diagrams with common jobs, be careful if a job is dependent on artifacts of the parent job.

## Pipeline Layout

From a pipeline's Layout page, you can view the pipeline diagram and textual layout summary.

To access a pipeline's layout page:

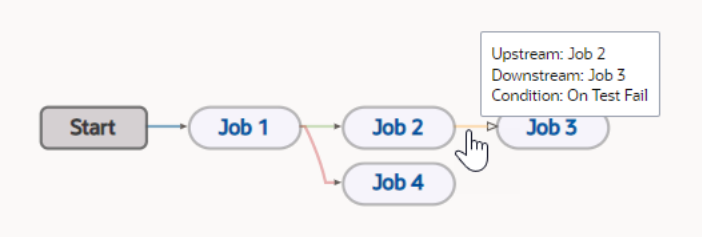
- In the **Pipeline Details** page, click the **View Layout** button.

- In the **Pipelines** tab, click the **Actions** *** menu for the pipeline you want to view and select **View Layout**.

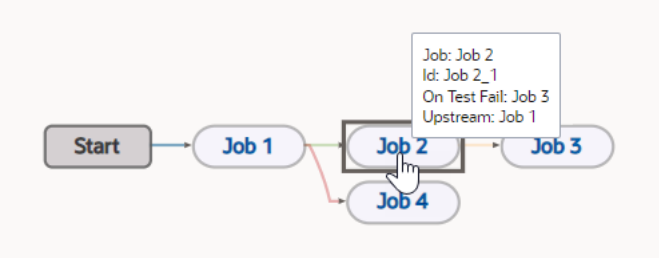
In a pipeline diagram, the arrow colors represent the dependencies between jobs in the flow.

- Green: The child job will run if the parent job is successful.
- Red: The child job will run if the parent job fails.
- Orange: The child job will run if the test run of the parent job fails.
- Purple: Two or more conditions point to the same node.

Hover over an arrow to view information on the conditions and dependencies between child and parent jobs.




Hover over a job note to view the job number, job ID, and dependencies between child and parent jobs.




To view a textual summary of the pipeline flow, click the **Pipeline Layout Summary** in the upper right corner of the page.

## Create and Manage Pipelines

From the Pipelines tab, you can create, run, and manage pipelines.

> **Note:**
>
> If you're creating a new pipeline or renaming an existing one and see a message that the pipeline name you entered matches a pattern that is restricted, you don't have access to any pipeline name matching the glob pattern that restricts the pipeline. Enter a name that doesn't match the name restricted by the glob pattern or contact the pipeline owner and request to be added as an authorized user (or member of an authorized group).
> See Protect Your Pipeline: Restrict Who Can Start It Manually or Edit Its Configuration.

> **Note:**
>
> If you're trying to access a renamed or deleted pipeline by using a link from the activities feed (or from a bookmarked link), you will be unable to do so because that pipeline no longer exists. A page with a message indicating that there was an error loading the Pipelines page is displayed, along with some text that says that the pipeline could not be found because it may have been deleted or renamed. Click the **Take me back to Pipelines** button and the Builds page's **Pipelines** tab will be displayed.

To access the Pipelines tab:

1. In the left navigator, click **Builds** .
2. Click the **Pipelines** tab.

| Action | How to |
|---|---|
| Create a pipeline | In the **Pipelines** tab:<br>a. Click **+ Create Pipeline**.<br>The **Create Pipeline** dialog is displayed.<br>b. In **Name** and **Description**, enter a unique name and a description, respectively.<br>c. (Optional) Select the appropriate checkboxes:<br>• **Auto start when pipeline jobs are built externally**: Select this checkbox to trigger a pipeline build when any job in the pipeline is triggered externally (that is, started from outside the pipeline).<br>In the pipeline, builds of jobs downstream from the started job will be run in the order shown in the diagram. Builds of jobs upstream from the externally-started job in the pipeline won't be run.<br>Note that triggering any job in the pipeline won't trigger a pipeline build, if this pipeline is ever disabled in the future.<br>• **Auto start only when trigger jobs are built**: If you selected the **Auto start** option in the previous step, you can select this checkbox to limit the jobs that can automatically start the pipeline to trigger jobs only. This effectively excludes all jobs that aren't trigger jobs.<br>If both options are selected, when a non-trigger pipeline job is started manually, it won't be shown in the **Pipeline Instances** page. It will be shown on the **Jobs Overview** page instead, because the **Trigger only** option was selected.<br>• **Disallow pipeline jobs to build externally when the pipeline is building**: Select this checkbox to disable manual or automatic builds of the jobs that are part of the pipeline when the pipeline is running.<br>d. Click **Create**.<br>e. In the **Pipeline Configuration** page, design the pipeline, and click **Save**. |

| Action | How to |
|---|---|
| Rename a pipeline or update its description | In the **Pipelines** tab:<br>a. Choose **Configure Pipeline** from the **Actions** *** menu in the **Pipelines** tab (or click the **Configure** button from a pipeline's details page).<br>b. Select **Settings** if it isn't already selected.<br>c. In **Name**, update the pipeline name.<br>d. Click **Save**. |
| Create a pipeline using YAML | In VB Studio, you can use a YAML file (a file with .yml extension) to store a job or pipeline configuration in any of the project's Git repositories. See Configure Jobs and Pipelines with YAML. |
| Configure a pipeline | The pipeline configuration page opens immediately after you create a pipeline. You can also configure an existing pipeline by choosing<br>**Configure Pipeline** from the **Actions** *** menu in the **Pipelines** tab or clicking the **Configure** button from a pipeline's details page.<br><br>**Note:**<br>If you are configuring an **edit restricted** pipeline:<br>• The Pipelines page's **Configure Pipeline** and **Delete Pipeline** options in the **Actions** *** menu are disabled.<br>• The Pipeline Details page doesn't display the **Configure** or **Delete** buttons.<br>• The **Delete** option in the **Run History** page's **Actions** *** menu is disabled.<br>• The **Delete** button is not displayed on the **Pipeline Run Instance** page.<br>• The lock icon will be displayed in the Private column in the list of pipelines on the **Pipelines** page.<br>• The **Pipeline Protection** page's pipelines list (accessed from the tab under the **Project Administration : Builds** tile) uses the same icon to identify protected pipelines. See Protect Your Pipeline: Restrict Who Can Start It Manually or Edit Its Configuration. |
| Run a pipeline | Choose **Run Pipeline** from the **Actions** *** menu in the **Pipelines** tab or click the **Run** button from a pipeline's details page.<br><br>**Note:**<br>There can be a maximum of five (5) instances of the same pipeline running simultaneously. If there is an attempt to exceed this limit, VB Studio will display a message that an additional instance of the pipeline failed to start because the maximum number of instances were already running. |
| View a pipeline's layout | Choose **View Layout** from the **Actions** *** menu or click the **View Layout** button from a pipeline's details page. |

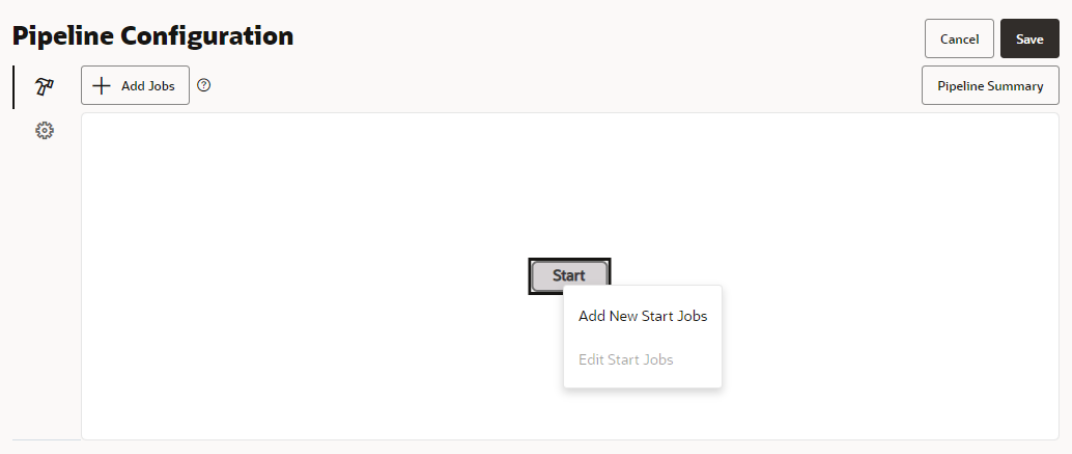
| Action | How to |
|---|---|
| View last run log | Choose **Last Run Log** from the **Actions** *** menu in the **Pipelines** tab or click the **Run Log** icon from a pipeline's details page. |
| Disable a pipeline | Choose **Disable Pipeline** from the **Actions** *** menu in the **Pipelines** tab or click the **Disable** button from a pipeline's details page.<br><br>The **Disabled** icon in the pipeline's Status column indicates that the pipeline has been disabled. It can't be run unless it's enabled again. |
| Enable a disabled pipeline | Choose **Enable Pipeline** from the **Actions** *** menu in the **Pipelines** tab or click the **Enable** button from a pipeline's details page. |
| Delete a pipeline | Choose **Delete Pipeline** from the **Actions** *** menu in the **Pipelines** tab or click the **Delete** button from a pipeline's details page. |

# Use the Pipeline Designer

You use the **Pipeline Configuration** page to create a pipeline diagram that defines dependencies between jobs and the order of their builds.

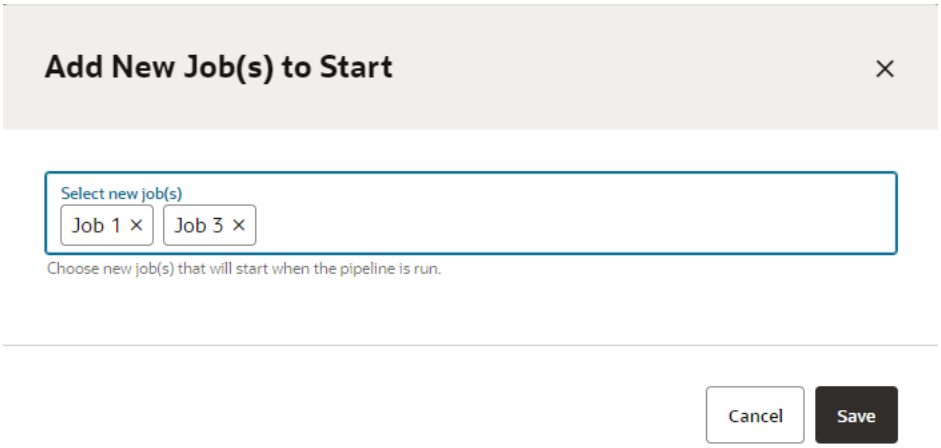
To create a new pipeline using the designer:

1. In the left navigator, click **Builds** .
2. Click the **Pipelines** tab.
3. Click the **Actions** *** menu for the pipeline that you want to configure and select **Configure Pipeline**.
4. Click the **Configure** button.
5. To add a new start job, right-click the **Start** node, and select **Add New Start Jobs**.

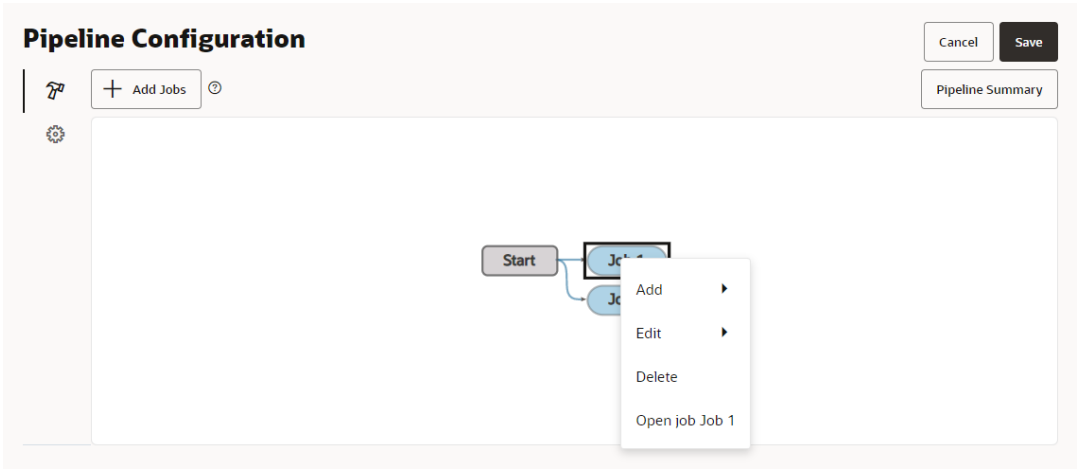



> **Tip:**
>
> You can also press the **Add Jobs** button to add jobs to the pipeline configuration diagram, then right-click **Start node** then select **Edit Start Jobs** to link the job you added to the Start node.

6. Search for and select the jobs that you want to add, then click **Save**.




7. Right-click a job to add or edit child jobs, delete the selected job, or open the selected job.




8. When you are finished building the pipeline, click **Save**.

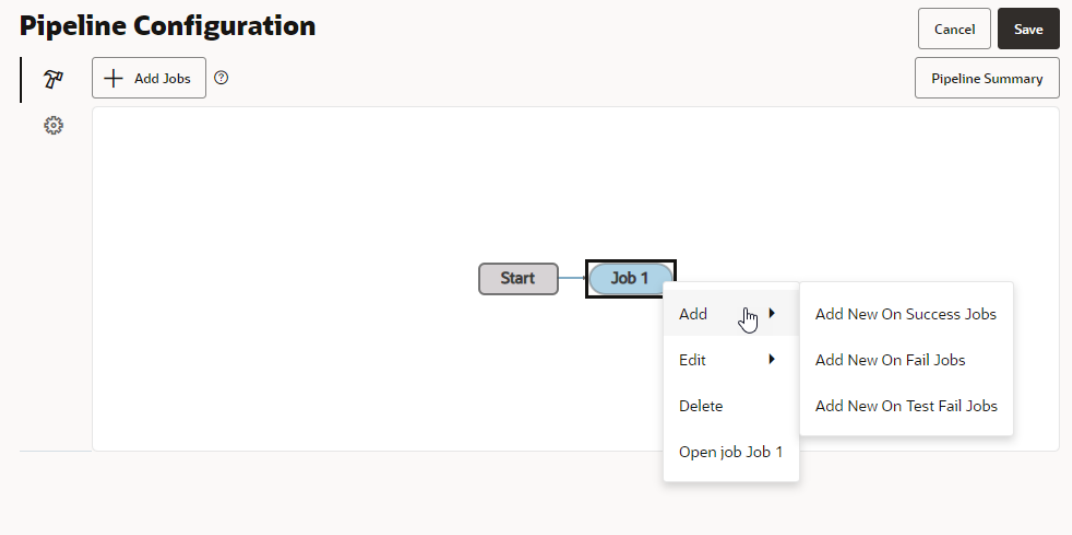
> **Note:**
>
> To add a job that triggers a pipeline, click **+Add Jobs**, then search for and add the trigger job.

## Create a One-to-One Dependency

A one-to-one dependency is formed between a parent and a child job. When a build of the parent job is successful, a build of the child job runs automatically.

To create a one-to-one dependency of a child job to its parent job:

1. Right-click the existing job and select **Add**, then select a dependency type for the new job.




2. Search for and select the child job, then click **Save**.
   A dependency is now formed.

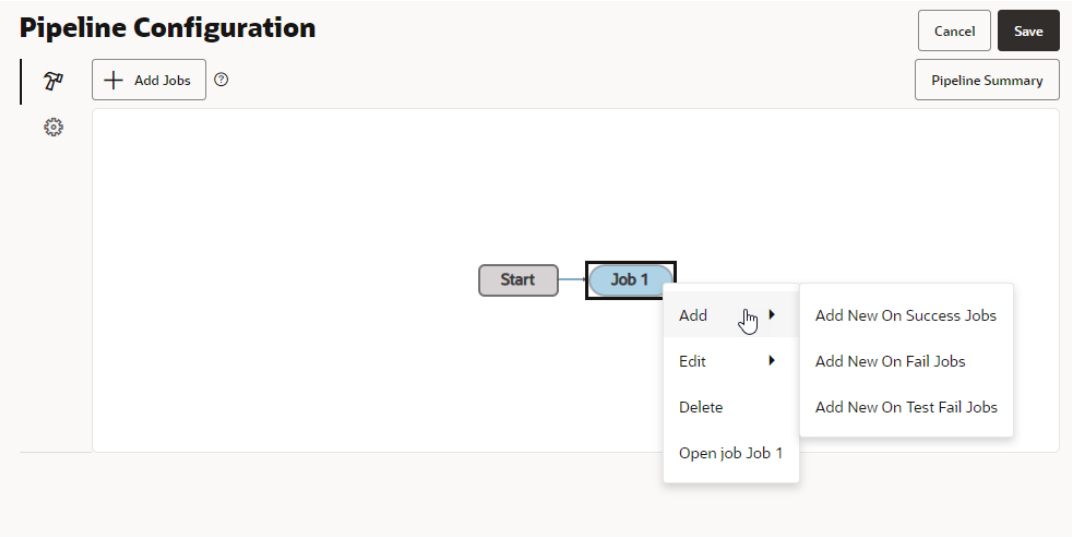
In the above example, `Job 2` is now dependent on `Job 1`. A build of `Job 2` will run automatically after every `Job 1` build is successful.

## Create a One-to-Many Dependency

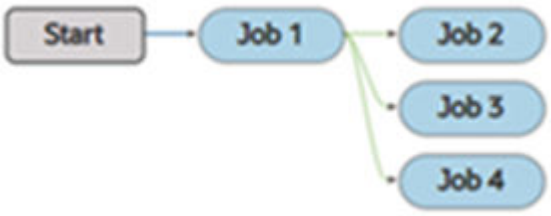
A one-to-many dependency is formed between one parent job and multiple child jobs. When a build of the parent job is successful, builds of child jobs run automatically.

To create a one-to-many dependency between jobs:

1. Right-click an existing job, and select **Add**, then select a dependency type for the new job.



2. Search for and select the child jobs, then click **Save**.



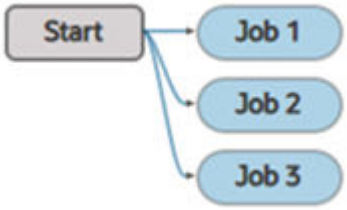

A dependency is now formed. In the above example, `Job 2`, `Job 3`, and `Job 4` are now dependent on `Job 1`. `Job 2`, `Job 3`, and `Job 4` run automatically after `Job 1` completes successfully.

## Create a Many-to-One Dependency

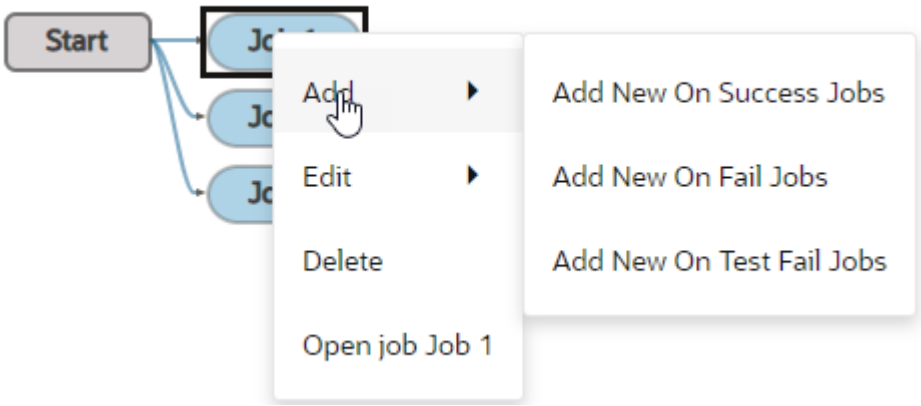
A many-to-one dependency is formed between multiple parent jobs and one child job. When builds of all parent jobs are successful, a build of the child job runs automatically.

To create a many-to-one dependency on parent jobs with a child job:

1. Add the parent jobs to the pipeline.

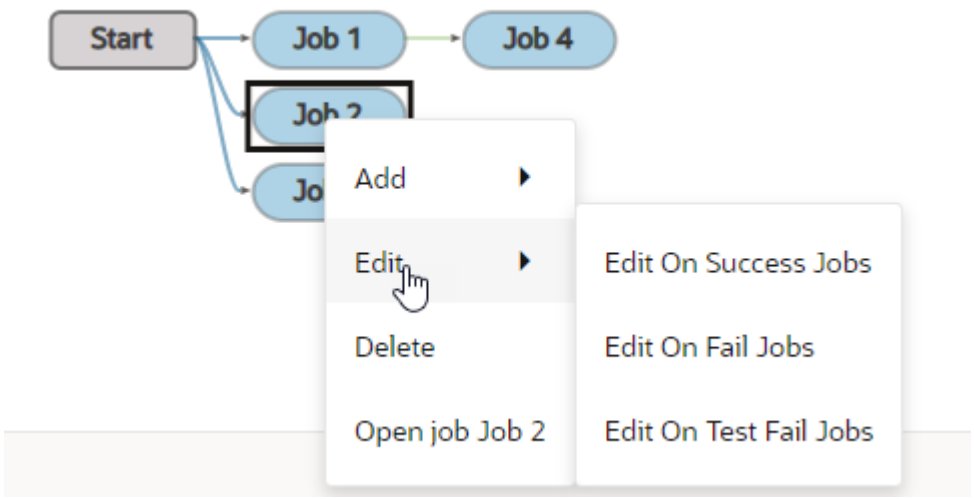



2. Right-click one of the parent jobs, select **Add**, then select a dependency type for the child job.

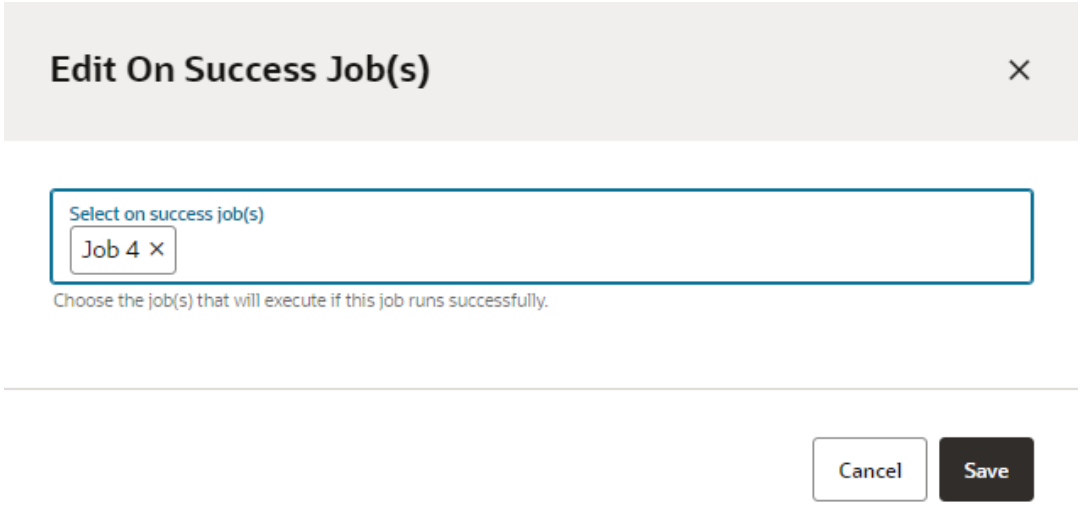



3. Search for and select the child job, then click **Save**.

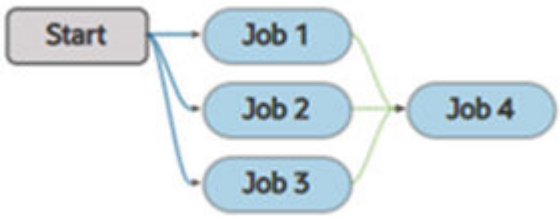
4. Right-click the second parent job, select **Edit**, then choose a dependency type for the child job.




5. Select the same child job that you added in Step 3.




6. Repeat Steps 4 and 5 for each parent job that the child job is dependent on.



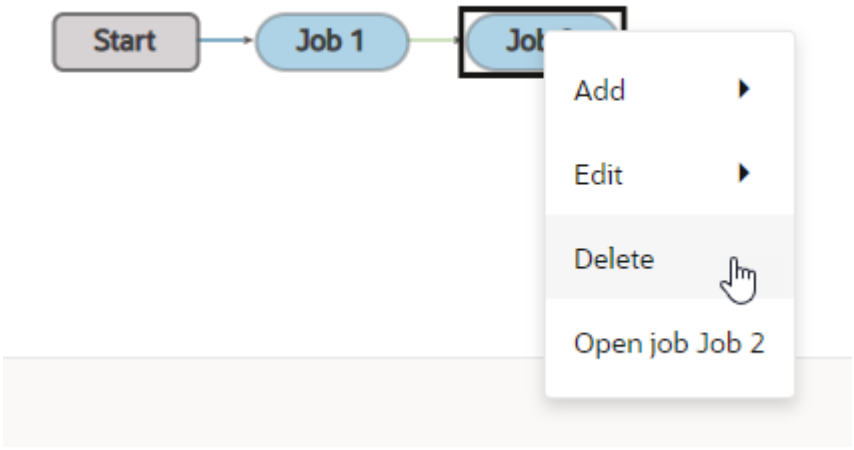
A dependency is now formed. In the above example, `Job 4` is dependent on `Job 1`, `Job 2`, and `Job 3`. A build of `Job 4` will run automatically after `Job 1`, `Job 2`, and `Job 3` are successful.

## Edit Pipeline Dependencies

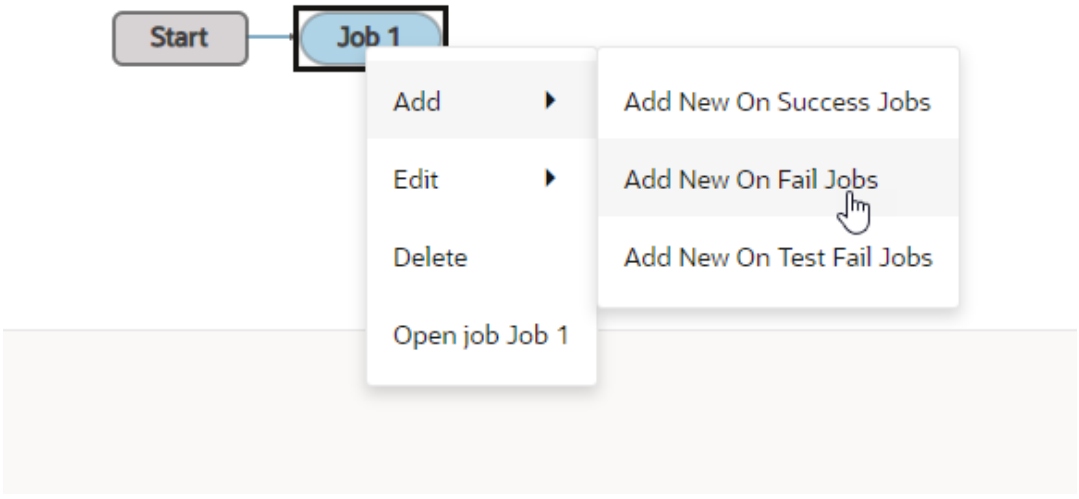
From the Pipeline Configuration page, you can update the dependencies between jobs in the pipeline.

To change the dependency condition of a child job, you first need to delete the job from the pipeline, then add it again with the new dependency.

1. In the pipeline designer, right-click the job that you want to update and select **Delete.**




2. Right-click the parent job, select **Add**, then select the dependency for the child job.




3. Search for and add the child job, then click **Save**.
The color of the dependency arrow will change to reflect the current dependency.

If you configure the pipeline using YAML, you'll have access to additional options that aren't available in the UI. However, the dependency conditions in YAML all map to the three that you have access to in the UI. See Set Dependency Conditions in Pipelines Using YAML.

> **Note:**
>
> You can also use this procedure to update the formatting of a pipeline. For example, if you a diagram that looks confusing because of flow arrows that cross over each other, you can rebuild the diagram by deleting child nodes and editing the dependencies.

## Protect Your Pipeline: Restrict Who Can Start It Manually or Edit Its Configuration

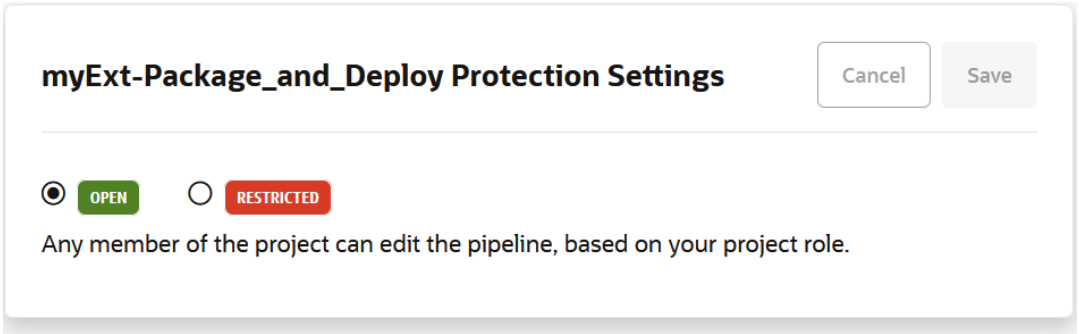
If you want to prevent unauthorized users from manually starting your pipeline or editing (and changing) its configuration, you can impose those restrictions from the **Pipeline Protection** tab on the **Project Administration Builds** page:

1. In the left navigator, click **Project Administration** .
2. Click **Builds**.
3. Click the **Pipeline Protection** tab.
4. From the pipelines list, select the pipeline.

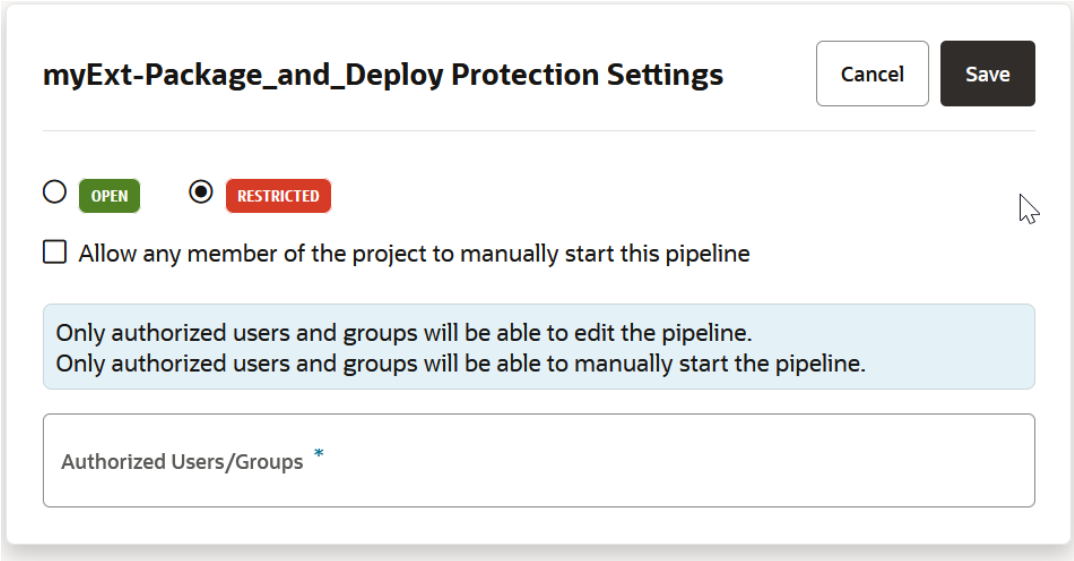
   If your project has many pipelines, you may have difficulty finding the specific pipeline you want to protect. Use the **Search** bar to quickly locate the pipeline to which you want to add the restricted settings.

   If a pipeline in the list of pipelines to the left has a lock icon next to it, it has already been protected. A protected pipeline's restrictions can still be modified, removed, or the list of authorized users and groups can still be changed.

   The **Protection Settings** dialog box is displayed.



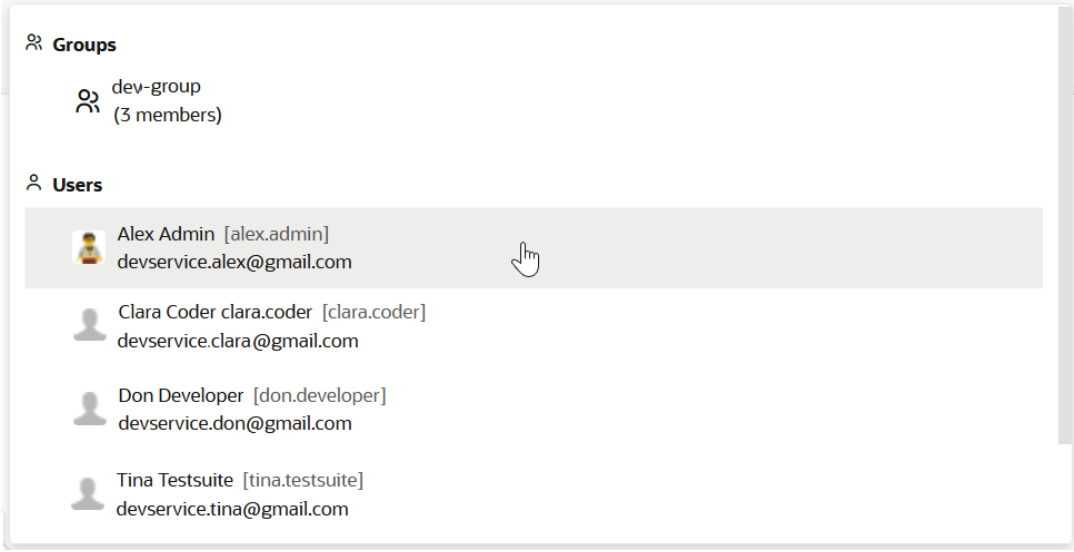
5. Select the **RESTRICTED** option.




With this setting, only authorized users and groups will be allowed to edit the pipeline configuration.

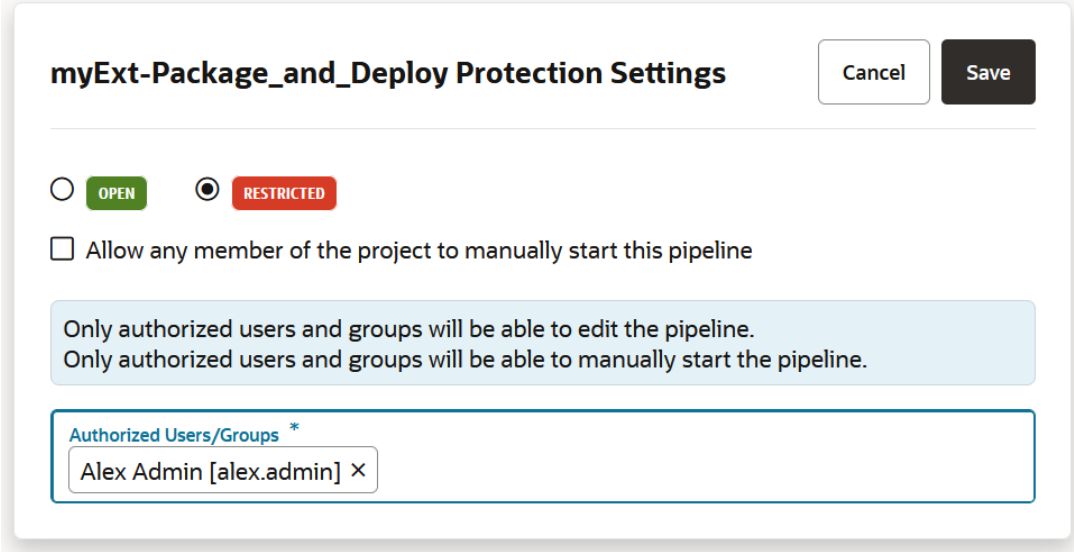
6. Click in the **Authorized Users/Groups** field to display a dialog that lists the project's **Groups** and **Users** you can select from.

   Under Users, you can see a flattened list of all users that are members of the group(s) as well as ones that were added individually. For example, the dev-group members (Clara Coder, Don Developer, and Tina Testsuite) appear in the Users list, along with Alex Admin, who was added individually. From the list, select one or more groups and/or users. Don't forget to add yourself.

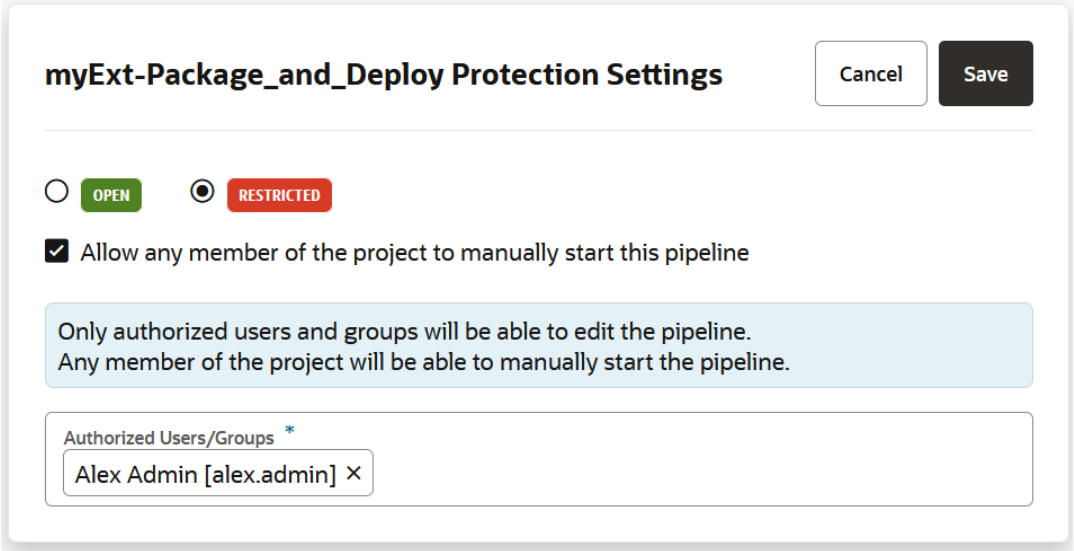


> **Note:**
>
> Users in Oracle Cloud Application environments that have multiple VB Studio instances in different identity stripes have username strings that include the environment name, where that user has been defined. Since a unique user may have been defined for multiple environments, this format ensures that one identity can be distinguished from that user's other unique identities. This should help you select the correct user to add.

Alex Admin was selected.



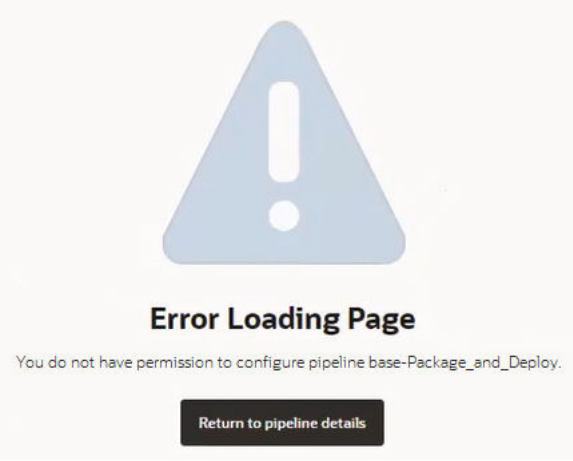

7. Select the **Allow any member of the project to manually start this pipeline** checkbox to leave the restriction for editing the pipeline by authorized users and groups in place but lift the restriction on who can manually start the pipeline.



If you select this checkbox, any project member with sufficient privileges can start the pipeline manually, even if they haven't been specified as an authorized user or are a member of a group that has been authorized. This effectively cancels any previous restriction for starting or running the pipeline manually.

8. Click **Save**.

When a pipeline is protected, the Pipeline Details page won't show the **Configure** and **Delete** buttons to unauthorized users for protected pipelines and the **Configure Pipeline** option in the **Actions** ⋯ menu on the Pipelines page won't be available. However, an unauthorized user could still inadvertently access a protected pipeline, perhaps by using a bookmarked URL that was saved before the user lost access, and this is what they'll see:




The lock icon in the **Private** column in the list of pipelines on the **Pipelines** page identifies protected pipelines.

The project's activity feed shows all edit-restricting activities, thereby providing a historical record of showing how the pipeline was protected. The pipeline log also records these protective activities and provides a historical accounting that can be referred to later, if needed.

> **Tip:**
>
> Protecting the pipeline prevents unauthorized users from *editing* the configuration but it does not prevent anyone from *running* the pipeline. The only way to limit that is to protect the initial job (the one that follows the Start node) by changing its Job Protection Settings to Private. That way, if the job is triggered in a pipeline by an unauthorized user or group, it won't be initiated. By default, the Job Protection settings don't allow commits and triggers to start a private job. You may, however, click the **Allow commits and triggers to start this private job** button to allow the job to be initiated if it is triggered by an SCM commit or by a timer. See Configure Job Protection Settings for more information.

## Add Manual Pipeline Approvals

You can set up and use a manual approval process to pause a pipeline run, so one or more authorized users can approve executing the rest of its run. This is done in the Pipeline Designer by adding an approval item to the pipeline where you want the pipeline execution to stop so someone with the necessary permissions can approve or reject executing the rest of

the pipeline. The pipeline run pauses, while it waits for approvals. Once the approvers approve, the pipeline run resumes building. If the approvers reject, the pipeline run stops and returns a failed status. The project's activity stream displays pipeline approval-related activities when a pipeline pauses for approval, an approver approves or rejects a pipeline, and when the pipeline resumes after approval.

Manual approval functionality involves four main tasks:

- Configuring the approvers
  See Create a Pipeline Approvers Group for Your Project.
- Adding the approval item to the pipeline during design time
  See Add a Manual Approval Item to Your Pipeline at Design Time.
- Notifying the approvers that they need to take action
  See Notify the Pipeline Approvers.
- Approving or rejecting the manual approval during run time
  See Approve the Manual Approval when the Pipeline Runs.

## Create a Pipeline Approvers Group for Your Project

The pipeline approvers group lets you approve all manual approval pipelines that are not restricted by specific groups.

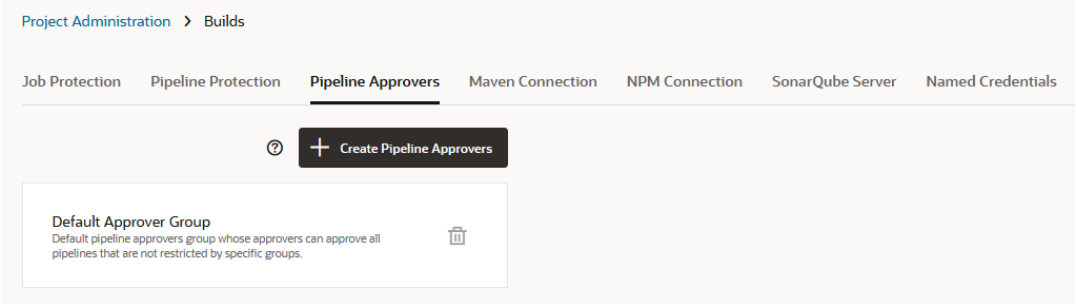
An approver is always part of a group. A group of approvers can approve certain pipelines only. Certain pipelines can be approved by certain groups of approvers only.

The approvers in the "Default Approver Group" can approve or reject any Manual Approval pipeline. The default pipeline approvers group is made up of individuals (approvers) who can approve any unrestricted pipeline (a pipeline that doesn't require approval from a specific group). So, you can add approvers to the default group but you can't limit that approval group to any specific pipeline.

To do that, you need to create a group to approve specific pipelines. This group allows you to add both approvers and pipelines. Then later, when you add the Manual Approval item in the pipeline designer, you can designate approvers as required approvers and specify the minimum number of approvers needed to approve the pipeline. See Add a Manual Approval Item to Your Pipeline at Design Time.

To add pipeline approvers to your project:

1. In the left navigator, click Project Administration .
2. Go to the **Pipeline Approvers** tab.




Notice that a Default Approver Group has already been set up for the project. This group has no members initially. Its approvers can approve any pipelines that isn't restricted by a

specific approval group. If the default group has no approvers, however, nobody will be allowed to add any approvers to any pipelines in the project. Note that pipelines can't be added to this group.

Add the default pipeline approvers now:

a. Select the default group.

   The **Configure Pipeline Approvers** dialog displays.

b. In the **Configure Pipeline Approvers** dialog, click in the **Approvers** field and select one or more approvers from the list and add them as default approvers.

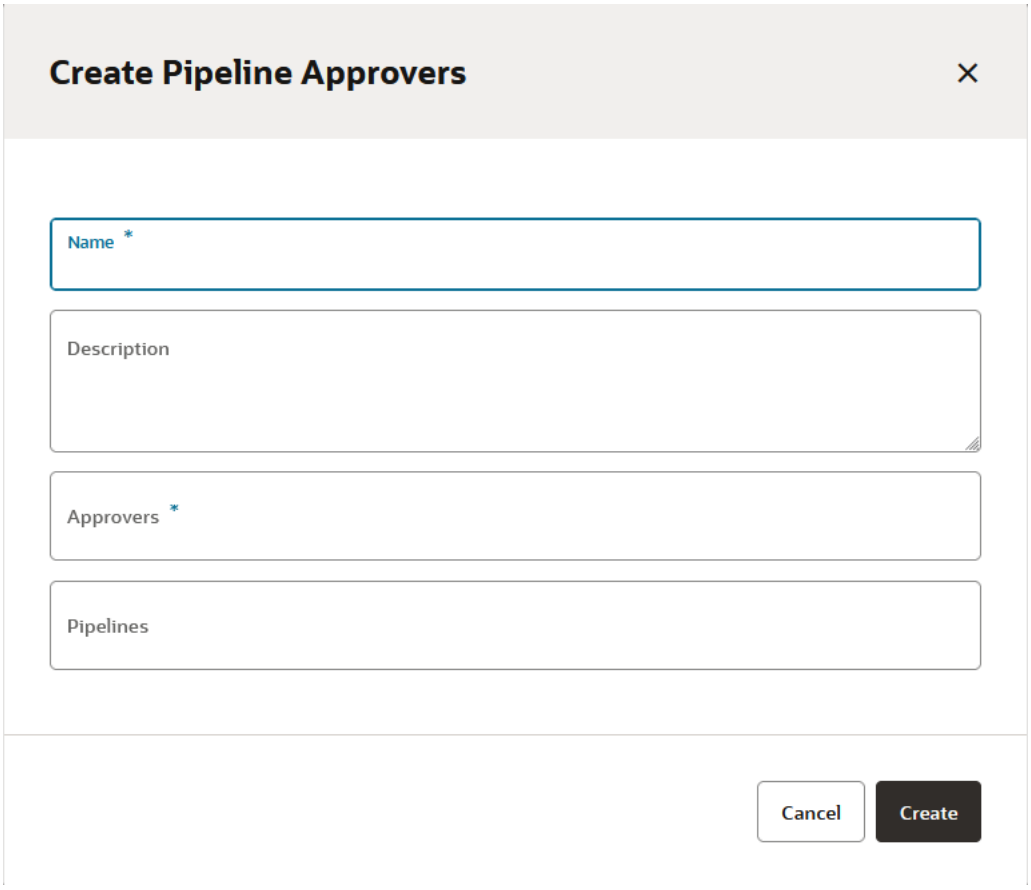
c. Click **Save**.

   Now, for any pipeline you create in the Pipeline Designer going forward in the project, you'll see and be able to select from the list of default approvers you added here.

> **Tip:**
>
> Using this approach only isn't an ideal solution for most organizations. A better approach would be to also create an approval group for a specific pipeline (or multiple pipelines). Once you do that, the default group no longer applies to that pipeline (or pipelines). Proceed to the next step and begin this process.

3. Click **+ Create Pipeline Approvers**.

   The Create Pipeline Approvers dialog displays.

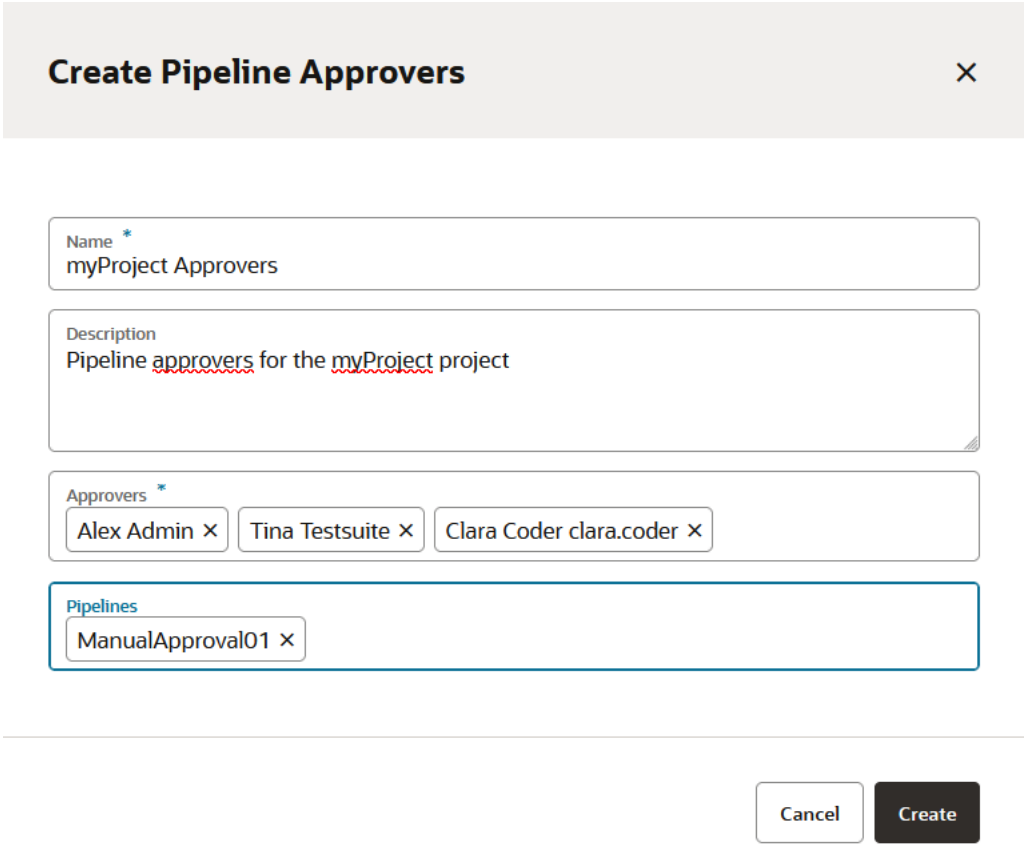


4. Enter a name for your group in the **Name** field (required) and optionally describe the group in the **Description** field.
5. Click in the **Approvers (Required)** area and select the project members you want to add as approvers. You can also add individual members from a *local group*, which are listed along the left panel. (Local groups are created at the Organizational level and added to the current project.) To add a team member from a local group, click the Approvers dropdown and select the member you want to add.

   This step is required. Approvers you add to this approvers group will also appear in the list of approvers that you can select in the Pipeline Designer, when you configure the Manual Approval item (node). See Add a Manual Approval Item to Your Pipeline at Design Time.

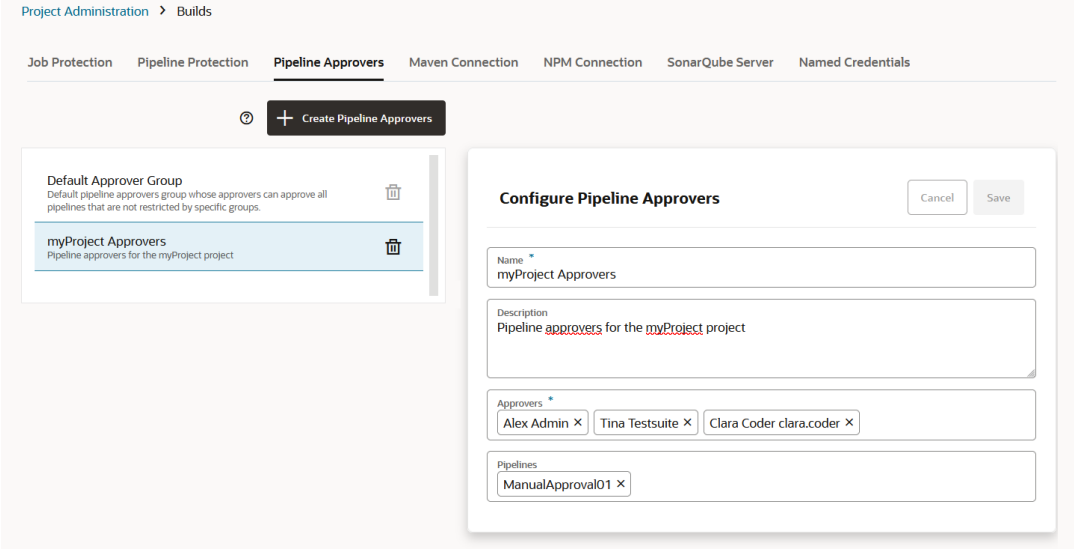
> **Note:**
>
> Users in Oracle Cloud Application environments that have multiple VB Studio instances in different identity stripes have username strings that include the environment name, where that user has been defined. Since a unique user may have been defined for multiple environments, this format ensures that one identity can be distinguished from that user's other unique identities. This should help you select the correct user to add.

6. Click in the **Pipelines** area and optionally add the pipeline(s) that will require pipeline approvals.

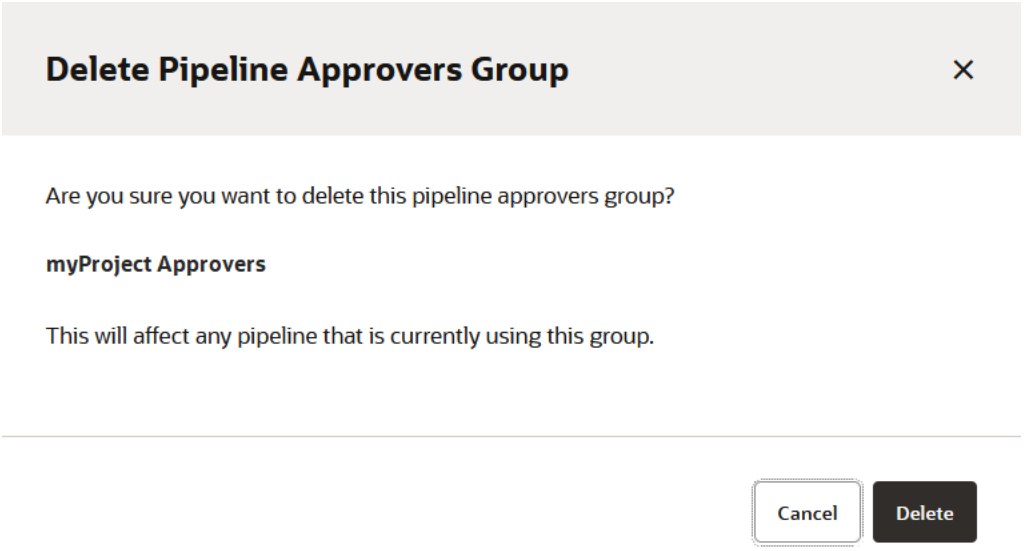



7. Click **Create**.

You can see the group you just added, on the left, under the Default Pipeline Approvers group.




When you no longer need the group, click **Delete** 🗑. You'll be prompted to verify that you do want to remove your pipeline approvers group.




Click the **Delete** button and remove the group. Otherwise, click **Cancel**.

## Add a Manual Approval Item to Your Pipeline at Design Time

As you're creating your pipeline in the Pipeline Designer, you can add a pipeline approval. You do this by right-clicking on an upstream Job node and selecting the **Add Manual Approval**

context menu item. When the node is added to the pipeline layout, its appearance is quite different from that of a Job node, so it's easy to differentiate between them.

Like a Job item, a Manual Approval item can be used with the fork-join layout pattern. That is, it can be a downstream item of a single upstream Job or a join of multiple upstream Jobs. Once it has been approved, it can trigger a single downstream job or fork multiple downstream jobs. However, although you can join multiple upstream jobs to and fork multiple downstream jobs from an Approval item, you can't use the Approval item itself as one of the joining items.

> **Note:**
>
> If you haven't defined any default approvers yet, you won't be able to use the manual approval functionality with any pipelines, because you can't add any approvers for that pipeline. The default approval group applies to *any* pipeline. Before you try to add approvals to a pipeline, you must add approvers to the default approvers group so you have a pool of approvers from which you can select when you define approvers for a pipeline in the Pipeline Designer or when you create an approval group and associate it with a specific pipeline (or pipelines).
> See Create a Pipeline Approvers Group for Your Project.

After you add a Manual Approval node to your pipeline layout, you'll be presented with a dialog to configure approvers for your pipeline. If you don't want to configure them now, you can right-click on the approval item later and select the **Configure** context menu item, which displays the dialog you use to configure approvers.)

You can use the Configure dialog to:

- Add or remove one or more approvers (only those who can approve this pipeline).
- View information about the group of approvers, making it easier for you to determine if you're picking up the correct approver.
- If you've selected multiple approvers, you can mark **All** or **n number of** approvers to enable approval for the approval item.
- Mark one or more approvers as required. Required approvers must approve; all other approvers are optional.
  That is, if three approvers (approver1, approver2, approver3) are selected and approver2 is marked as required, if at least 2 approvers must approve, then one of those two approvers must be approver2.

If you already configured approvers, when you hover over the Manual Approval node, you can view information about it. If you haven't configured any approvers for the pipeline, the warning marker next to the node indicates that you need to configure approvers. The tooltip you see when you mouse over it provides information that explains how to configure the required approvers.

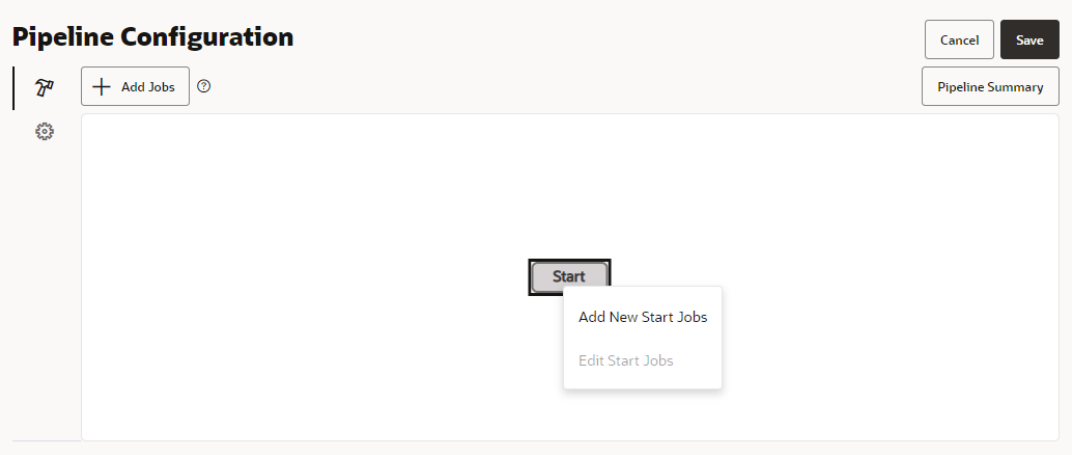
Let's step through this process using an example. We could start with a very simple pipeline that has two jobs and an approval but it may be more interesting and realistic to create one that has joins and forks. If you already have a pipeline with two jobs, each connected to the Start node, you can skip to step 8 and start there. Otherwise, start at the beginning.

1. In the left navigator, click **Builds** .
2. Click the **Pipelines** tab.
3. Click **+ Create Pipeline**.

The **Create Pipeline** dialog is displayed.

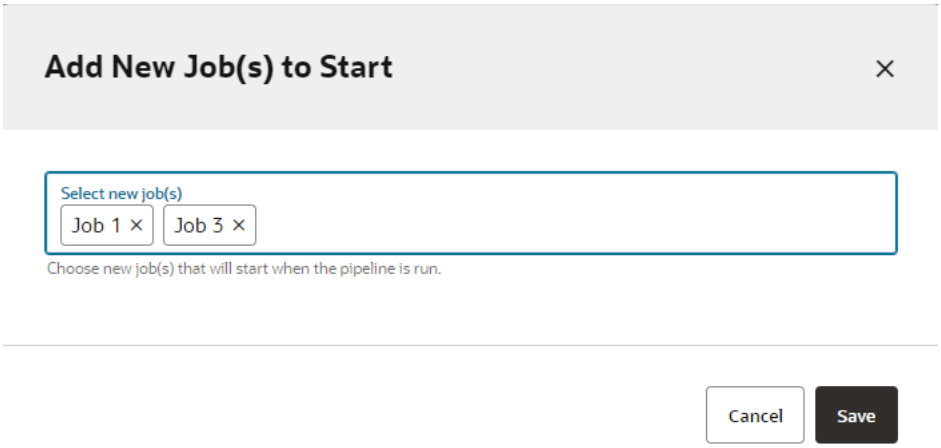
4. In **Name** and **Description**, enter a unique name and a description, respectively, then optionally select the appropriate check boxes for autostarting the pipeline and disable manual or automatic builds of the jobs that are part of the pipeline when the pipeline is running.
5. Click **Create**.

   The pipeline configuration page opens immediately after you create the pipeline.
6. To add a new start job, right-click the **Start** node, and select **Add New Start Jobs**.

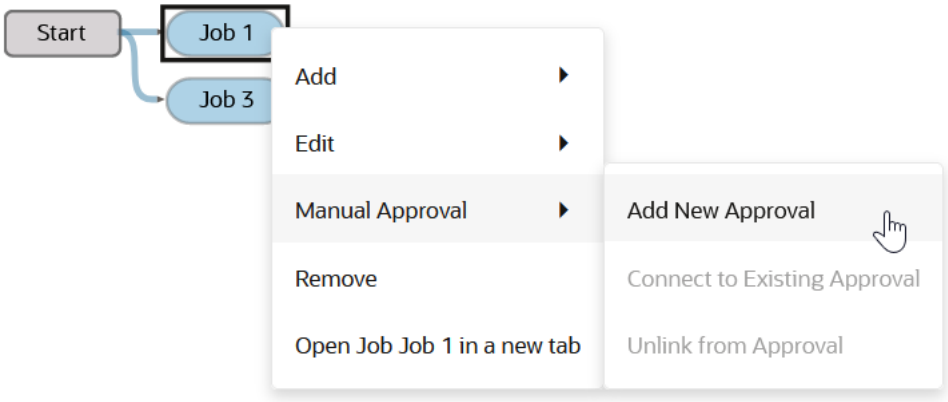



7. Search for and select the jobs that you want to add, then click **Save**.

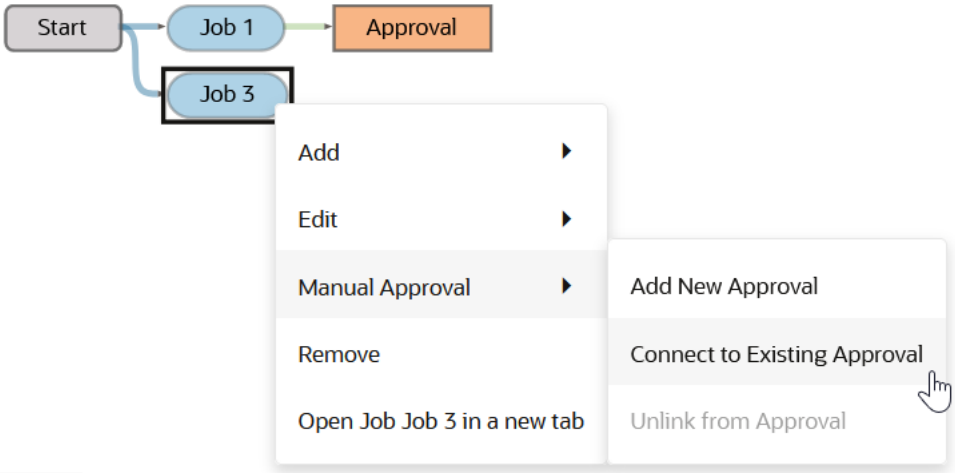
   Here we've selected Job 1 and Job 3.




8. Right click the first job (Job 1) and select **Manual Approval**, then select **Add New Approval**.

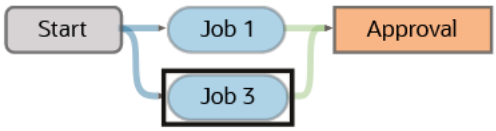



9. Right-click the second job (Job 3) and select **Manual Approval**, then select **Connect to Existing Approval**.




The **Connect to Existing Approval** dialog opens.

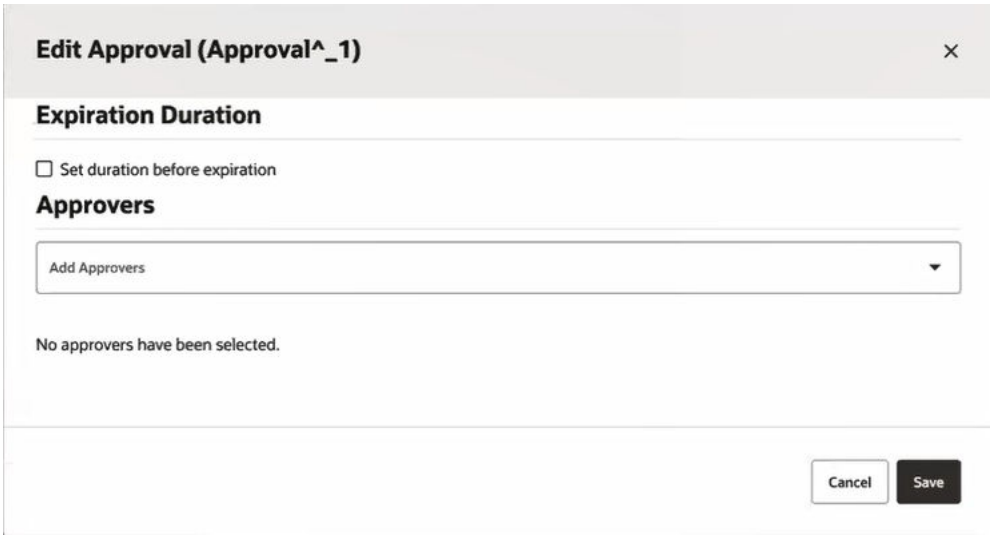
10. Click the **Select Approval** field and, from the dropdown list, select the approval to invoke when this job completes (Approval, in this example), then click **Save** to close the dialog.




The approval will wait for both jobs to finish before notifying any approvers you configure.

11. Right click the approval item, select **Edit Approval**.

The **Edit Approval** dialog is displayed.




12. In the Approvers section, click the **Add Approvers** field and use the selector to choose the approvers that can approve this manual approval from the list displayed.

    Here we've selected Alex Admin and Clara Coder.

> **Note:**
>
> Users in Oracle Cloud Application environments that have multiple VB Studio instances in different identity stripes have username strings that include the environment name, where that user has been defined. Since a unique user may have been defined for multiple environments, this format ensures that one identity can be distinguished from that user's other unique identities. This should help you select the correct user to add.

13. Toggle the **Required** setting to on to make the approver's approval mandatory.
14. Repeat for all the project members you want to be approvers.

    If you select more than one approver from the list, you'll see a field for setting the **Minimum number of approvers before the pipeline can proceed** field filled in with one approver, who is required.
    The minimum number must always be equal to or greater than the total number of required approvers. For example, if Alex Admin and Clara Coder are both selected as required

reviewers, then the minimum number of approvers must be at least two. If you wanted *either* Alex Admin or Clara Coder's approval to allow the pipeline to continue, then you would set both of them to *not required* and set the number of approvers to 1.

15. In the Expiration Duration section, click the **Set duration before expiration** check box.

16. Use the **Days**, **Hours**, and **Minutes** selectors or enter values to set the length of time to wait for an approval.

    If you enter or select a value of 72 in the **Hours** field, for example, it will be converted to a value of 3 in the **Days** field. Similarly, 25 hours will be converted to 1 day and 1 hour.

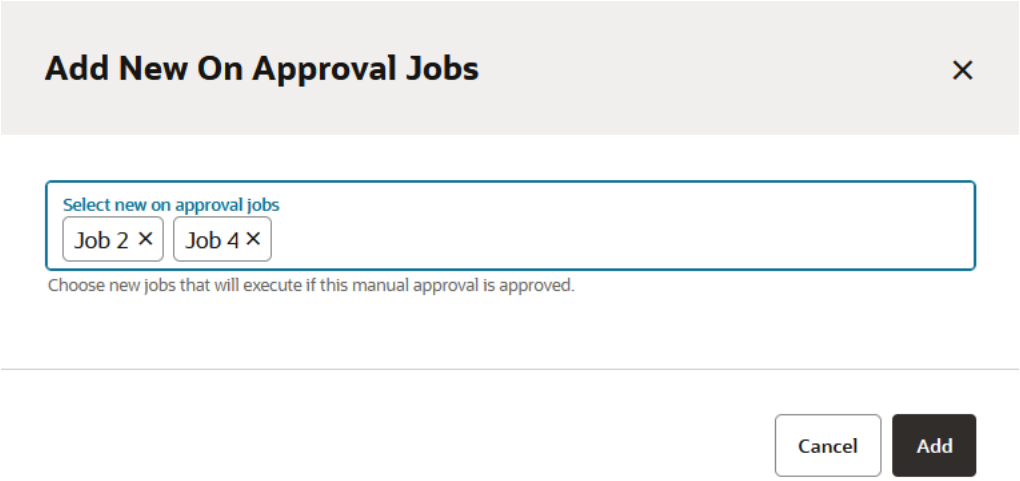
17. Click **Save** to exit the values you set.

18. Right-click the approval item again and select the **Add New On Approval Jobs** option.

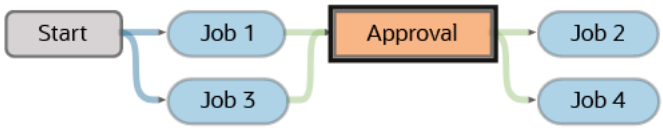
    The **Add New On Approval Jobs** dialog displays.

19. Select a new on approval job, or select multiple jobs.

    We're adding two jobs, Job 2 and Job 4.

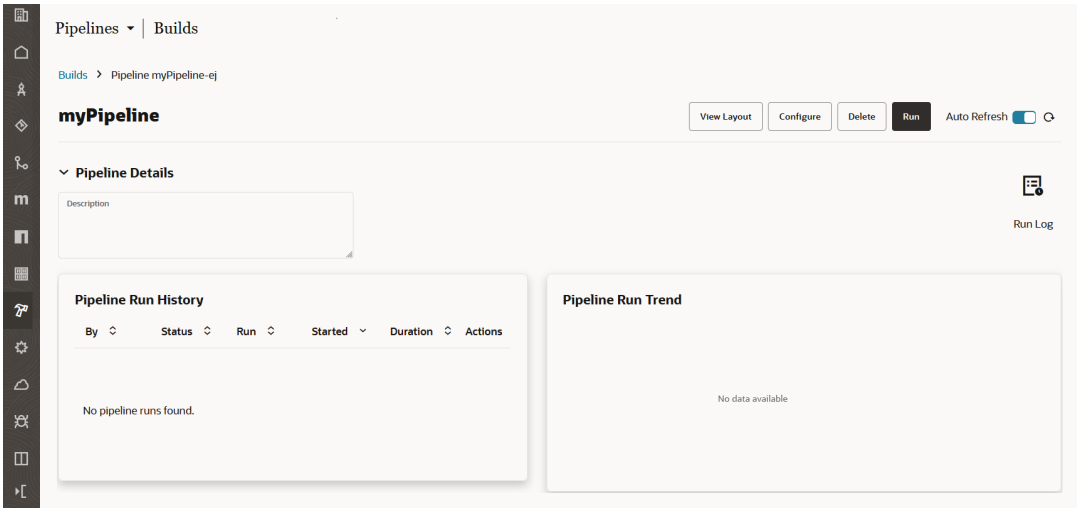



20. Click **Add** to add the two jobs to the pipeline.




21. Click **Save** to save the pipeline.

    The **Pipeline Details** screen is displayed.




22. Click **Run** to run the pipeline.

    If you don't see a notification, go to the **Profile** tab your user account and, under **User Preferences**, check that the user has been verified. See Add Manual Pipeline Approvals.

There are a few restrictions for manual approvals. You may encounter these when you are prevented from saving a newly created pipeline or one that's been modified:

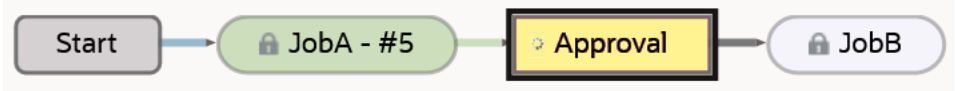
- You can't create an approval item from the Start node. You can only create it from a job (or jobs) downstream from the Start node.
- You can't configure parallel approvals. If you try, when you attempt to save the configuration, you'll see the message, "Only one manual approval can be in effect at any given point in time. Your pipeline has multiple paths which can result in more than one approval to be active at the same time." Click **OK** to dismiss.
- You can't have random jobs in the pipeline that aren't connected to start or to approvals or to other downstream jobs. "Pipelines with manual approval cannot have unconnected jobs or approvals. The following jobs or approvals are not connected to start: Job1".
- You can't introduce jobs that bypass the approval(s).

> **Note:**
>
> Currently, you can't resume a pipeline after it has been rejected. You need to address and correct the issue(s) that caused an approver to reject completing the pipeline run and then restart the pipeline and approval process.

## Approve the Manual Approval when the Pipeline Runs

If you're an approver, you can either approve or reject a Manual Approval action from the **Pipeline Run** page, where the graph or layout for the pipeline run is displayed. You can also add comments about why you approved or rejected the approval.

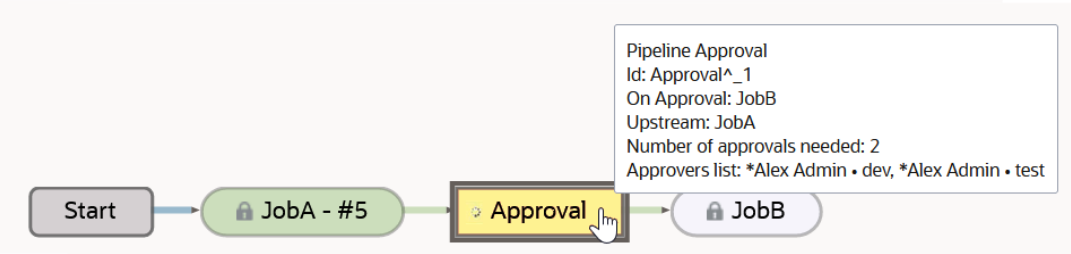
1. In the graph, locate the yellow Approval item.



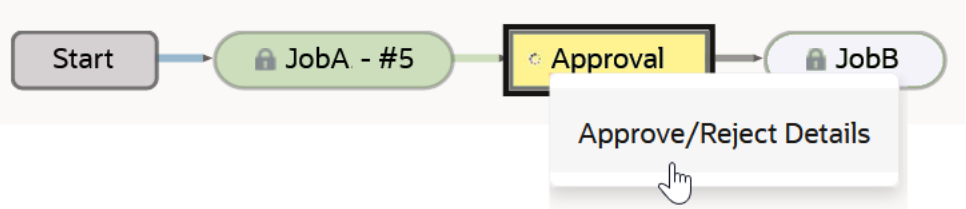
The approval node is yellow, which indicates that some action is needed. The spinner indicates that the approval process is ongoing and likely in a waiting state.

2. Hover over the approval node.

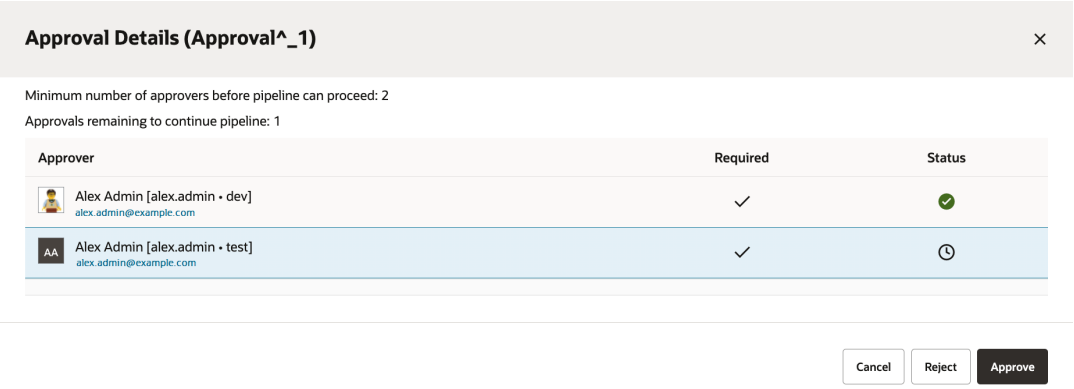
   You'll see that the pipeline is waiting for approval, how many approvals are needed, how many approvals have been approved, who the approvers are, how many approvals are pending, the length of time before the approval expires (and will be automatically rejected), and so on.




3. Right click the approval item and select **Approve/Reject Details**.




The Approval Details dialog is displayed.

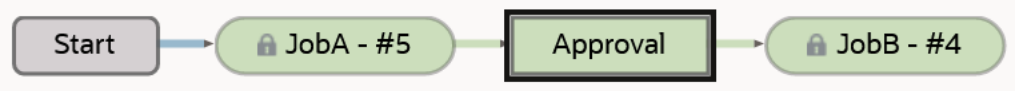


> **Note:**
>
> Users in Oracle Cloud Application environments that have multiple VB Studio instances in different identity stripes have username strings that include the environment name, where that user has been defined. Since a unique user may have been defined for multiple environments, this format ensures that one identity can be distinguished from that user's other unique identities. This should help you select the correct user to add.

4. Two approvals are needed. It looks like alex.admin-dev already approved, so now alex.admin-test (the blue stripe indicates that that's you) needs to approve or reject the run.

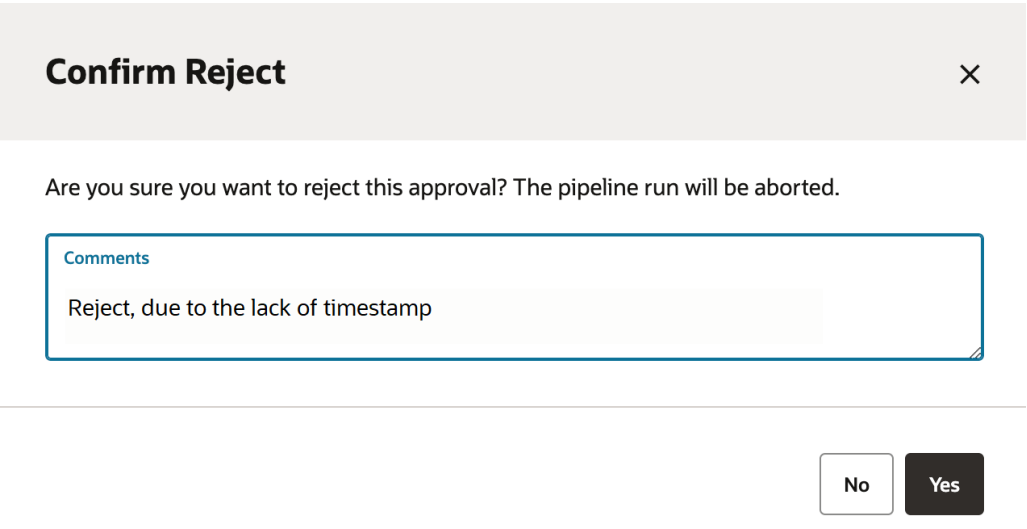
   Select one of the options:

   - Select **Approve**.
     A confirmation message shows that an approval has been submitted. The pipeline will continue executing. After the approval has been completed, the approval item will turn green, and any downstream jobs will too after they successfully execute.

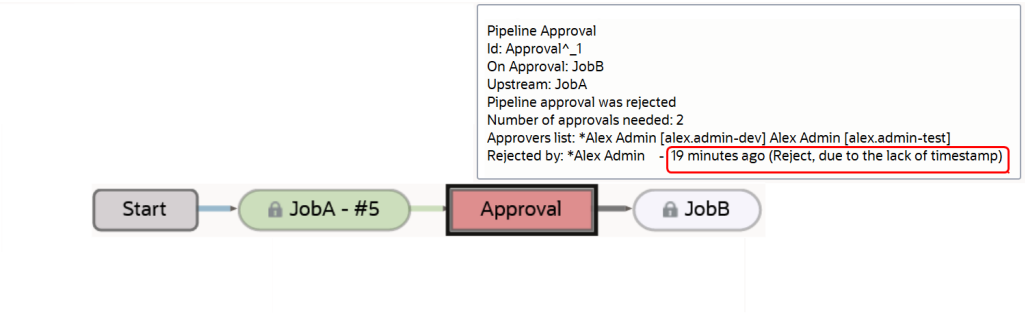
     
     

     If you check the status, you'll see a green check, indicating that the pipeline run was successful.

   - Select **Reject**.
     You'll be prompted to confirm that you want to reject the approval and will be allowed to add a comment explaining why you chose this action.

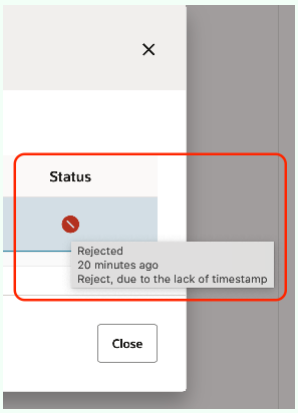
     The approval item turns red and the pipeline execution halts. No downstream jobs are executed so they remain white, instead of completing and turning green. When you mouse over the approval node, you'll see that it was rejected and the comment that was added in the confirmation window.




The red thumbs down icon , at the top of the screen, indicates that the pipeline run was rejected.

**Tip:**

If you mouse over the rejected icon in the Status column of the Approval Details screen, this is representative of what you may see.




Notice the timestamp that shows when the approval was rejected and any comments that were added explaining why.

If an expiration duration was set for the manual approval but nobody approves it during the specified interval, the build system automatically rejects the approval. The activities stream displays the reason for the rejection. In addition, the Pipeline Run Summary screen and the notification email that is sent to required approvers also indicate why the approval was rejected.

**Note:**

Currently, you can't resume a pipeline after it has been rejected. You'll need to address and correct the issue(s) that caused an approver to reject completing the pipeline run and then restart the pipeline and approval process.

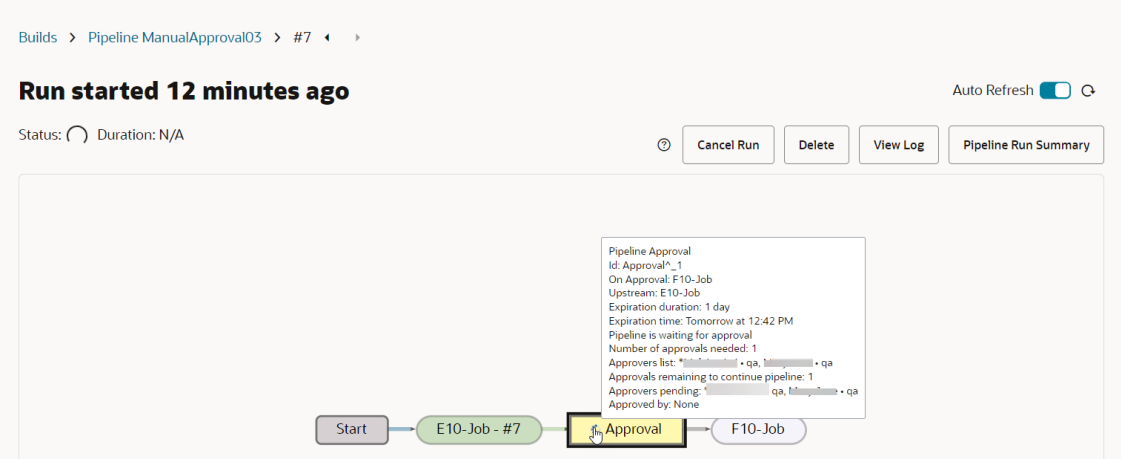
# Notify the Pipeline Approvers

Pipeline approvers are notified in email that an approval is awaiting their action.

When a pipeline encounters a Manual Approval item, the pipeline run pauses and an email notifying approvers that they need to take action is sent. The **Pipeline Queue** page shows who triggered the build (if triggered manually), the pipeline name, the run number, if it was scheduled, and progress, which shows, "Waiting for approval". If you hover over it, you'll see a different message, depending on your role:

- If you're not an approver, you'll see "Not waiting for your approval. Approvals pending:", followed by a list of the approvers.
- If you're an optional approver, you'll see "Waiting for approval. Go to pipeline `name` run `number`"
- If you're a required approver, you'll see "You are a required approver. Go to pipeline `name` run `number`"

If you're an approver, you can click **Waiting for approval** to go to the **Pipeline Run** page, which shows all the approvers, who has already approved, and so on, as shown:

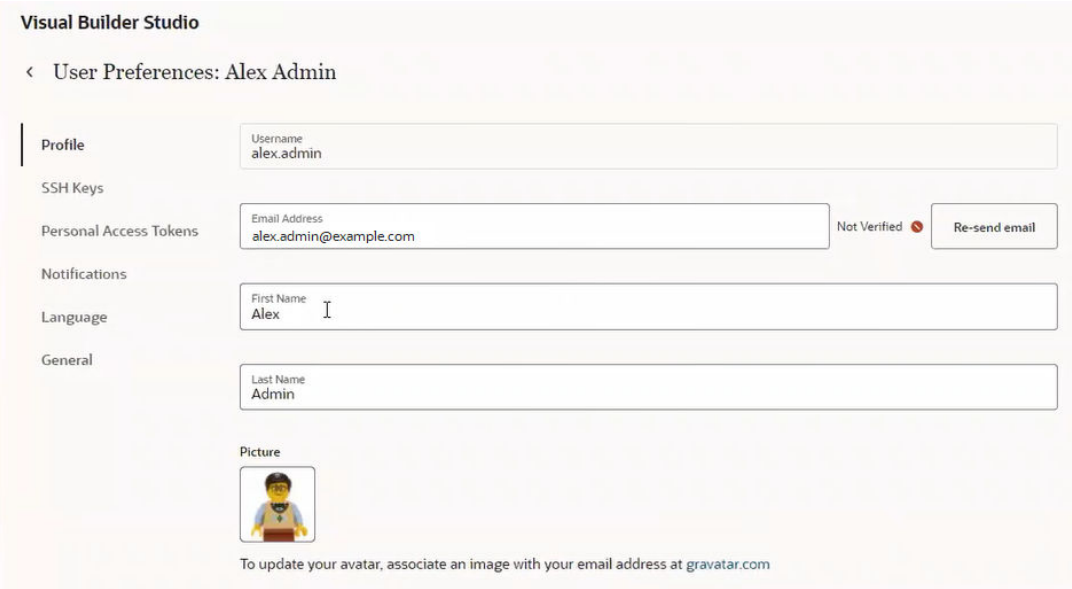



If you approve or reject from here (by right-clicking the **Approval** node), any comments you supply will be added to the final email that all approvers receive.

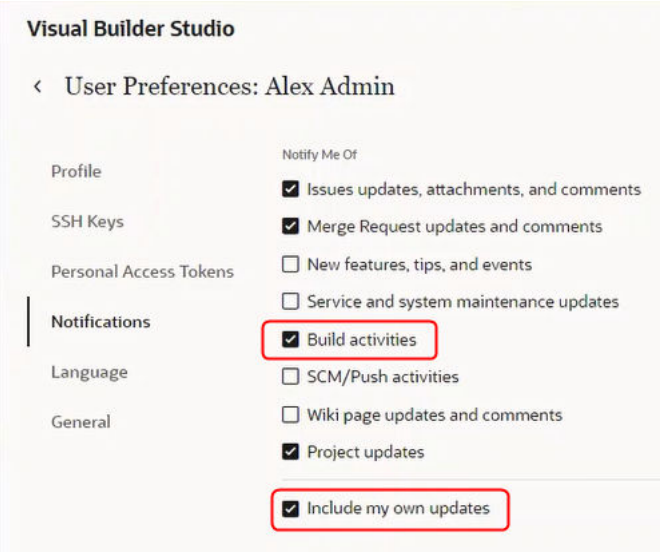
> **Note:**
>
> If you are not a required approver and you approve the job (either from here or through email), the pipeline remains paused until all *required* approvers have signed off. If you reject the job, this run of the pipeline is canceled.

As an approver, there are a few tasks that you need to do to ensure that you can receive email notifications about pipeline approvals:

- Make sure you verified your email address or you won't receive any VB Studio email notifications at all.
  You don't want to see this message.




- Make sure you enabled notifications for Build activities and that you're also set up to include your own updates.
  The check boxes encircled in red need to be selected.




To check both items, click the user avatar and select **Preferences**. See Configure Your Global Email Notifications for additional information about these items that you need to check and potentially address.

Click on link in email and it'll take you to the pipeline's Run page or from the Build page's **Pipeline Queue** tab go to the **Waiting for approval** link in the Progress column. Because you're the approver, you can see the link. If you're not one of the approvers, you'll see the same message but it won't be a link. If you are an approver and you click the link, you'll go to the Pipeline Designer graph. Hover over the approval item to see its status, then right-click and select the **Approve/Reject Details** option. If you approved it, the `Pipeline approve submitted` message is displayed. Once the required number of approvers have submitted an approval, the pipeline starts back up and continues running, the approval node turns green (signifying completion), and the pipeline continues by executing the next downstream task.

There are several places in the pipelines interface where approvers are subtly notified about pending approvals.

- They can see the pipeline's flow from the **Pipeline Run** page before approving or rejecting it.
- The **Pipeline Queue** tab shows pipelines runs that are either running or are waiting for some action, such as an approval. If a pipeline is in the waiting state, an approval (or multiple ones) is needed.
- From the **Pipeline Queue** page, you can sort by the **Progress** column, to see all the pipelines "Waiting for approval".
- The notification email's subject line may show, for example, "Pipeline run 4 for pipeline ManualApproval07 requires approval to proceed". If you click the "4" link in the email, you'll be taken to the page where you can right-click the Approval node and select the **Approve/ Reject Details** context menu item.

## Monitor Pipeline Status

You can view the status of all of a project's pipelines from the **Pipelines** tab.

The **Pipelines** tab displays this information for the latest run of each pipeline:

| Column | Details |
|---|---|
| Status | The status for the latest pipeline build is displayed:<br>• **Success**: Indicates that the latest build of the pipeline was successful.<br>• **Failed**: Indicates that the latest build of a job in the pipeline failed, causing the pipeline to fail too.<br>• **Test Failed**: Indicates that the latest test build of a job in the pipeline failed, causing the pipeline to fail too.<br>• **Canceled**: Indicates the latest build of the pipeline was canceled.<br>• **In Progress**: Indicates a pipeline is in progress.<br>• **Disabled**: Indicates a pipeline is disabled. It can't be run unless it's enabled again. |
| Pipeline | Click the pipeline link to view more information about the pipeline. See View Pipeline Run Details. |
| Weather | The weather for the recent builds are displayed:<br>• **All recent builds completed successfully**<br>• **All recent builds failed**<br>The weather also displays icons showing what percentage (%) of recent builds failed. |
| Last Successful | Date and time of the last successful build of the pipeline. You can click on the instance number to view more details about the pipeline instance. |
| Last Unsuccessful | Date and time of the last unsuccessful build of the pipeline. You can click on the instance number to view more details about the pipeline instance. |
| Duration | Duration of the latest pipeline build |
| # Jobs | Number of jobs in the pipeline. Click the number to view a list of the jobs. You can click on a job name to display details about the job. |
| Details | Click **Details** to view details and settings for the latest build. |
| Private | The lock icon identifies if the pipeline is protected. |

From the **Actions** menu in the **Pipelines** tab, you can also run or configure the pipeline, view the pipeline layout and last run log, and disable, enable, or delete the pipeline. See Create and Manage Pipelines.

## View Pipeline Run Details

From a pipeline's information page, you can view details about pipeline activity, including status of each pipeline run and pipeline run trends.

Click the name of a pipeline in the **Pipelines** tab to access details for the pipeline. Click on the column arrows to sort the rows.

The **Pipeline Run History** table provides the following information for each pipeline build:

| Column | Details |
|---|---|
| By | Who or what initiated the run. The icons represent the following:<br>• **User**: The build was initiated by a user.<br>• **Periodic Build Trigger**: The build was triggered by a periodic build trigger.<br>• **Build System**: The build was started or rescheduled by the build system. |
| Status | These are the possible statuses for each pipeline build:<br>• **Success**: Indicates that all the pipeline builds were successful.<br>• **Failed**: Indicates that a build of a job in the pipeline failed, causing the pipeline to fail too.<br>• **Test Failed**: Indicates that a test build of a job in the pipeline failed, causing the pipeline to fail too.<br>• **Canceled**: Indicates the pipeline was canceled.<br>• **In Progress**: Indicates a pipeline is in progress.<br>• **Disabled**: Indicates a pipeline is disabled. It can't be run unless it's enabled again. |
| Run | Click on the run number to view details about the pipeline run. From the run details page, you can view the pipeline run diagram, delete the run, view the run log, or view a summary of the run.<br>In the pipeline diagram, the color of job nodes indicates the job's status:<br>• **Green**: The last build of the job was successful<br>• **White**: A build of the job is running or hasn't run yet<br>• **Red**: The last build of the job failed<br>• **Orange**: The last test build failed. |
| Started | Displays the date and time the run started. Click the link to view the run details and pipeline diagram. Use the **Cancel Run**, **Delete**, **View Log**, and **Pipeline Run Summary** buttons to cancel a running pipeline, delete the run, view the run log, or view a summary of the run. |
| View the run log | Click the **Actions** menu and select **View Log**. |

From the pipeline details page, you can also view the pipeline run trends, run, configure, disable or enable, or delete the pipeline, and view the pipeline layout. See Create and Manage Pipelines.

## View Pipelines in Progress

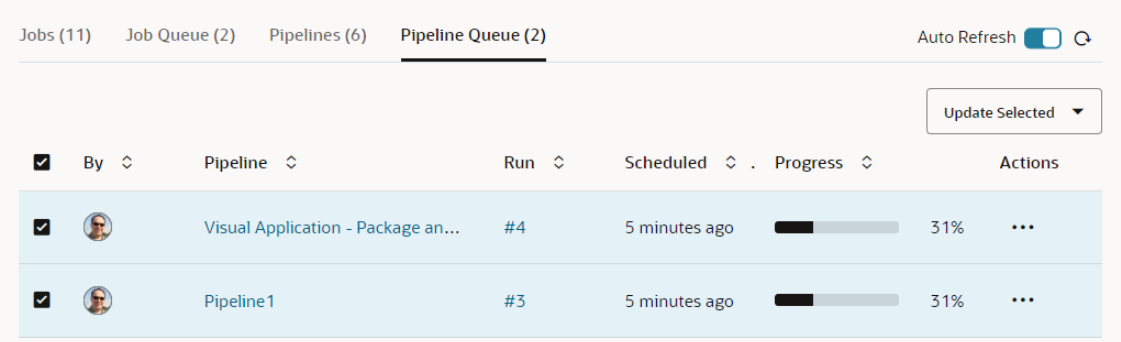
You can view the pipelines in progress from the **Pipelines Queue** tab.

The **Pipelines Queue** table provides the following information for each pipeline in the queue:

| Column | Details |
|---|---|
| By | Who or what initiated the run. The icons represent the following:<br>• **User**: The build was initiated by a user.<br>• **Periodic Build Trigger**: The build was triggered by a periodic build trigger.<br>• **Build System**: The build was started or rescheduled by the build system. |
| Pipeline | Click the name of the pipeline to view the **Pipeline Details** page. |
| Run | Click on the run number to view details about the pipeline build progress. From the pipeline details page, you can view the pipeline diagram, delete the run, view the run log, or view a summary of the run.<br>In the pipeline diagram, the color of job nodes indicates the job's status:<br>• **Green**: The job's last build was successful.<br>• **White**: A build of the job is either currently running or hasn't run yet.<br>• **Red**: The job's last build failed.<br>• **Orange**: The last test build failed. |
| Scheduled | Displays the time the build started. |
| Progress | Displays a progress bar and the percentage of the build that's been completed. |

To cancel a pipeline run, click the **Actions** menu, and select **Cancel**.

To cancel more than one pipeline run, select the pipelines that you want to cancel or click the box at the top left corner of the table to select all pipelines, then click **Update Selected** and select **Cancel Selected Runs**.




## Add or Export Parameters and Parameter Lists

You might want to set or change parameters in a pipeline job that can be used by downstream jobs in the same pipeline. Or, you might want to add parameters at the start of job execution, based upon data from a Git clone operation, when the cloned repository contains information that will become the value of new parameters or override the value of existing parameters. This

parameter can then be used to configure subsequently run jobs, and appear as environment variables in shell scripts run in the build. You might even want to export parameters at the end of job execution, based upon data that was calculated during the build.

Both added and exported parameters would be visible to downstream jobs, which could, in turn, modify a subset of the parameters and then pass them along.

There are two different ways to set parameters dynamically:

- From a list of parameter definitions that are written in the same manner that environment variables are set in a shell script, that is, from a file with one or more lines that contain `PARAMETER_NAME=value` definitions.
- With multi-line values, such as private keys for example, and parameters with sensitive contents like passwords and private keys. These include items that are more complex than those that can be specified using the simple definition format.

All jobs in a pipeline currently see job parameters that have been configured for all jobs in the pipeline. These parameters are collected from the jobs when the pipeline is started, and are added to all jobs that are downstream of the condition that started the pipeline, that is of a triggering job or the “Start” node of a manually-started or a periodically-triggered pipeline. Then, when a job completes, its parameters are extracted and are passed on to any downstream jobs it triggers.

These job parameters can be modified and new parameters can be added in subsequent build steps during a run. This simply adds a way to explicitly direct that parameters be added, both before and after Build steps, like shell scripts, run.

## Add a Parameter

The **Add a Parameter** task runs during build setup, after the Git steps finish running, but before any build steps are run. The task adds a single parameter at a time, potentially allowing a multi-line value (a private key, for example) read from a file, and allows the parameter to be marked sensitive which, by the way, means “Don't print these values in the build log.” You can configure zero or more of these in a job.

Here's how you configure a pre-build task that adds a parameter (or multiple parameters) to a build job:

1. In the left navigator, click **Builds** .
2. In the Jobs overview page, select the job you want to modify and the **Jobs Detail** page will display.
3. Click **Configure**.
   This displays the **Job Configuration** page.
4. In the **Git** tab, click **Add Git** and select the repository where the file with the parameter is stored.
5. Click the **Before Build** tab.
6. Click **Add Before Build Action** and select **Add Parameter**.
7. In **Parameter name**, enter the name of the parameter.
8. In **File containing parameter value**, enter the name of the file that contains the value for the parameter.
9. Select the **Sensitive** checkbox to prevent printing the value of parameters with sensitive contents, like passwords and private keys parameters, in the build log.
10. Repeat steps 6-8 to add multiple parameters.

11. Click **Save**.

## Add a Parameter List

The **Add a Parameter List** task runs during build setup, after the Git steps finish running, but before any build steps are run. The task reads a list, one per line, of one or more parameter definitions in the form `PARAMETER_NAME=value` and sets the job parameters accordingly. You can configure zero or more of these in a job.

Here's how you configure a pre-build task that adds a parameter list to a build job:

1. In the left navigator, click **Builds** .
2. In the Jobs overview page, select the job you want to modify and the **Jobs Detail** page will display.
3. Click **Configure**.
   This displays the **Job Configuration** page.
4. In the **Git** tab, click **Add Git** and select the repository where the file with the parameter list is stored.
5. Click the **Before Build** tab.
6. Click **Add Before Build Action** and select **Add Parameter List**.
7. Enter the name of the file that contains the parameter definitions.
8. Click **Save**.

## Export a Parameter

The **Export a Parameter** task runs after the build steps have been run. The task adds a single parameter at a time, to allow multi-line values ( a private key, for example) to be read from a file. It also allows the parameter to be marked sensitive which, by the way, means “don't print these values in the build log.” You can configure zero or more of these in a job.

Here's how you configure a post-build task that exports a parameter (or multiple parameters) that can be passed to a downstream build job:

1. In the left navigator, click **Builds** .
2. In the Jobs overview page, select the job you want to modify and the **Jobs Detail** page will display.
3. Click **Configure**.
   This displays the **Job Configuration** page.
4. In the **Git** tab, click **Add Git** and select the repository with the file where the value for the parameter will be written.
5. Click the **After Build** tab.
6. Click **Add After Build Action** and select **Export Parameter**.
7. In **Parameter name**, enter the name of the parameter to be exported.
8. In **File containing parameter value**, enter the name of the file to write the value for the parameter.
9. Select the **Sensitive** checkbox to prevent printing the value of parameters with sensitive contents, like passwords and private keys parameters, in the build log.
10. Repeat steps 6-8 to export multiple parameters.

11. Click **Save**.

## Export a Parameter List

The **Export a Parameter List** task runs after the build steps have been run. The task reads a list, one per line, of one or more parameter definitions in the form `PARAMETER_NAME=value` and sets job parameters accordingly. You can configure zero or more of these in a job.

Here's how you configure a post-build task that exports a parameter list that can be used by a downstream build job:

1. In the left navigator, click **Builds** .
2. In the Jobs overview page, select the job you want to modify and the **Jobs Detail** page will display.
3. Click **Configure**.
   This displays the **Job Configuration** page.
4. In the **Git** tab, click **Add Git** and select the repository with the file where the values for the parameter list will be written.
5. Click the **After Build** tab.
6. Click **Add After Build Action** and select **Export Parameter List**.
7. Enter the name of the file where the parameter definitions used in the build job will be written.
8. Click **Save**.

# Configure Jobs and Pipelines with YAML

YAML (YAML Ain't Markup Language) is a human-readable data serialization language that is commonly used for configuration files. To find more about YAML, see `https://yaml.org/`.

In VB Studio, you can use a YAML file (a file with `.yml` extension) to store a job or pipeline configuration in any of the project's Git repositories. The build system constantly monitors the Git repositories and, when it detects a YAML file, creates or updates a job or a pipeline with the configuration specified in the YAML file.

Here's an example with a YAML file that configures a job:

```
job:
  name: MyFirstYAMLJob
  vm-template: Basic Build Executor Template
  git:
  - url: "https://mydevcsinstance-mydomain.developer.ocp.oraclecloud.com/
mydevcsinstance-mydomain/s/mydevcsinstance-mydomain_my-project_902/scm/
employee.git"
    branch: main
    repo-name: origin
  steps:
  - shell:
      script: "echo Build Number: $BUILD_NUMBER"
  - maven:
      goals: clean install
      pom-file: "employees-app/pom.xml"
  after:
```

```
- artifacts:
    include: "employees-app/target/*"
settings:
  - discard-old:
      days-to-keep-build: 5
      builds-to-keep: 10
      days-to-keep-artifacts: 5
      artifacts-to-keep: 10
```

## What Are YAML Files Used for in VB Studio?

All YAML files must reside in the `.ci-build` directory in the root directory of any hosted Git repository's `main` branch. YAML files in other branches will be ignored. Any text file that has a `.yml` file extension and resides in the `main` branch's `.ci-build` directory is considered to be a YAML configuration file. Each YAML file can contain configuration data for exactly one job or one pipeline. You can have YAML files in multiple Git repositories, or use a separate Git repository to host all your YAML configuration files. You cannot, however, use an external Git repository to host YAML files. Because these configuration files are stored using Git, you can track changes made to the job or pipeline configuration and, if a job or pipeline is deleted, you can use the configuration file to recreate it.

The build system constantly monitors the project's Git repositories. When it detects an update to a file with the `.yml` extension in the `.ci-build` directory of a Git repository's `main` branch, it scans the file to determine if it is a job or a pipeline, and creates or updates the corresponding job or pipeline. First, it verifies whether the job or the pipeline of the same name (as in the configuration file) exists on the **Builds** page. If the job or the pipeline exists, it's updated. If the name of the job or pipeline has changed in the configuration file, it's renamed. If the job or the pipeline doesn't exist, it's created.

> **Note:**
>
> Jobs and pipelines created with YAML can't be edited on the **Builds** page. They must be edited using YAML. Similarly, jobs and pipelines created on the **Builds** page can't be edited using YAML.

YAML stores data as a key-value pair in the `field: value` format. A hyphen (-) before a field identifies it as an array or a list. It must be indented to the same level as the parent field. To indent, always use spaces, not tabs. Make sure that number of indent spaces before a field name matches the number of indented spaces in the template. YAML is sensitive to number of spaces used to indent fields. Also, the field names in a YAML file are similar to the field names in the job configuration user interface:

```
name: MyFirstYAMLJob
vm-template: Basic Build Executor Template
git:
- url: "https://mydevcsinstance-mydomain/.../scm/employee.git"
steps:
- shell:
    script: "echo Build Number: $BUILD_NUMBER"
- maven:
    goals: clean install
    pom-file: "employees-app/pom.xml"
```

If you're editing a YAML file on your computer, always use a text editor with the UTF-8 encoding. Don't use a word processor.

Here are some additional points to consider about YAML files before you begin creating or editing them:

- The `name` field in the configuration file defines the job's or pipeline's name. If no name is specified, the build system creates a job or a pipeline with name as `<repo-name>_<name>`, where `repo-name` is the name of the Git repository where the YAML file is hosted and `<name>.yml` is the name of the YAML file.
  For example, if the YAML file's name is **MyYAMLJob** and it's hosted in the **YAMLJobs** Git repository, then the job's or pipeline's name would be **YAMLJobs_MyYAMLJob**.

  If you add the `name` field later, the job or pipeline will be renamed. Its access URL will also change.
- Each job's configuration must define the `vm-template` field.
- When you define a string value, you can use quotes, if necessary. If any string values contain special characters, always enclose the values with quotes.
  Here are some examples of special characters: `*`, `:`, `{`, `}`, `[`, `]`, `,`, `&`, `#`, `?`, `|`, `-`, `<`, `>`, `=`, `!`, `%`, `@`, `` ` ``.

  You can use single quotes (`' '`) or double quotes (`" "`). To include a single quote in a single quoted string, escape the single quote by prefixing it with another single quote. For example, to set `Don's job` in the `name` field, use `name=Don''s job` in your YAML file. To use a double quote in a double quoted string, escape the double quote with a backslash (`\`) character. For example, to set `My "final" job` in the `name` field, use `name=My \"final\" job` in your YAML file. There's no need to escape backslashes in a single quoted string.
- Named Password/Private Key parameters must be specified in the format `#{PSSWD_Docker}` surrounded by quotes, as shown in bold in the following example:

```
params:
- string:
    name: myUserName
    value: "don.developer"
    description: My Username
steps:
- docker-login:
     username: $myUserName
     password: "#{PSSWD_Docker}"
```

  Password/Private Key parameters are specified using the format `$myPassword`, as shown in bold in the following example:

```
params:
- string:
    name: myUserName
    value: "don.developer"
    description: My Username
  - password:
      name: myPwd
      password: #{PSSWD_Docker}
      description: Defining the build password
steps:
  - docker-login:
       username: $myUserName
       password: $myPwd
```

- If you specify a field name but don't specify a value, YAML assumes the value to be `null`. This can cause errors. If you don't need to define a value for a field, you should remove the field name.
  For example, if you don't want to define Maven goals and use the default `clean install`, remove the `goals` field. The following YAML code can cause error because `goals` isn't defined:

```
steps:
- shell:
    script: "echo Build Number: $BUILD_NUMBER"
- maven:
    goals:
    pom-file: "employees-app/pom.xml"
```

- You don't need to define every one of the job's fields in the YAML file. Just define the ones you want to configure or change from the default values, and make sure that you're adding the parent field(s) when you define a child field:

```
steps:
- maven:
    pom-file: "employees-app/pom.xml"
```

- To run a build of the job automatically when its Git repository is updated, use the `auto` field or set `build-on-commit` to `true`.
  For the current Git repository, using `auto` is equivalent to setting `build-on-commit` to `true`. So, don't use `auto` and `build-on-commit: true` together.

  Here's an example that uses `auto`:

```
name: MyFirstYAMLJob
vm-template: Basic Build Executor Template
auto:
  branch: patchset_1
```

  If you use `auto`, don't specify the Git repository URL. The job automatically tracks the Git repository where the YAML file is committed.

  Here's an example that uses `build-on-commit`:

```
name: MyFirstYAMLJob
vm-template: Basic Build Executor Template
git:
- url: "https://mydevcsinstance-mydomain.developer.ocp.oraclecloud.com/
mydevcsinstance-mydomain/s/mydevcsinstance-mydomain_my-project_902/scm/employee.git"
    branch: patchset_1
    build-on-commit: true
```

  A commit when pushed to the `patchset_1` branch triggers a build of the **MyFirstYAMLJob** job.

- To add comments in the configuration file, precede the comment with the pound sign (`#`):

```
steps:
# Shell script
- shell:
    script: "echo Build Number: $BUILD_NUMBER"
```

- On the **Builds** page, to configure an existing job or a pipeline, click its **Configure** button or icon. If the job or the pipeline was created in YAML, VB Studio opens the YAML file in the code editor on the **Git** page so you can view or edit the configuration.
- The branch value is dependent on the default branch of the repository that is specified in the YAML. If the head of the Git repository is `main`, then that is the default. If the head is `master`, then that will be the default.
  The default behavior has been dependent on the head of the Git repository. Until this release, though, that has always been `master`.

# REST API for Accessing YAML Files

You can use an API testing tool, such as Postman, or curl commands to run REST API methods. To run curl commands, either download curl to your computer or use the Git CLI to run curl commands.

To create the REST API URL, you need your VB Studio user name and password, the base URL of your instance, the unique organization ID, and the project ID, which you can get from any of the project's Git repository URLs.

In a Git repository URL, the project's ID is located before `/scm/<repo-name>.git`. For example, if `https://alex.admin%40example.com@mydevcsinstance-mydomain.developer.ocp.oraclecloud.com/mydevcsinstance-mydomain/s/`**`mydevcsinstance-mydomain_my-project_123`**`/scm/NodeJSDocker.git` is the Git repository's URL in a project, the project's unique ID will be **mydevcsinstance-mydomain_my-project_123**.

## How Do I Validate a Job or Pipeline Configuration?

To validate a job (or pipeline) configuration, use this URL with the syntax shown, passing in the local (on your computer) YAML file as a parameter:

```
https://<base-url>/<identity-domain>/rest/<identity-domain>_<unique-projectID>/cibuild/v1/yaml/validate
```

Here's an example with a curl command that validates a job configuration on a Windows computer:

```
curl -X POST -H "Content-Type: text/plain" --data-binary @d:/myApps/myPHPapp/.ci-build/my_yaml_job.yml -u alex.admin@example.com:My123Password https://mydevcsinstance-mydomain.developer.ocp.oraclecloud.com/myorg/rest/myorg_my-project_1234/cibuild/v1/yaml/validate
```

Here's an example with a curl command that validates a pipeline configuration on a Windows computer:

```
curl -X POST -H "Content-Type: text/plain" --data-binary @d:/myApps/myPHPapp/.ci-build/my_yaml_pipeline.yml -u alex.admin@example.com:My123Password https://mydevcsinstance-mydomain.developer.ocp.oraclecloud.com/myorg/rest/myorg_my-project_1234/cibuild/v1/yaml/validate
```

## Create a Job or a Pipeline Without Committing the YAML File

You can create a job or pipeline without first committing its YAML file to your project's Git repository. To do so, use a URL with this syntax, passing in a local (on your computer) YAML file as a parameter:

```
https://<base-url>/<identity-domain>/rest/<identity-domain>_<unique-projectID>/
cibuild/v1/yaml/cibuild/yaml/import
```

VB Studio will read the YAML job (or pipeline) configuration and, if no errors are detected, create a new job (or pipeline). The job (or pipeline) must be explicitly named in the YAML configuration. After the job (or pipeline) has been created, you can edit its configuration on the **Builds** page. If errors are detected, the job (or pipeline) will not be created and the Recent Activities feed will display any error messages.

Here's an example that shows how to use a curl command with a YAML file on a Windows computer to create a job:

```
curl -X POST -H "Content-Type: text/plain" --data-binary @d:/myApps/myPHPapp/
my_PHP_yaml_job.yml -u alex.admin@example.com https://mydevcsinstance-
mydomain.developer.ocp.oraclecloud.com/myorg/rest/myorg_my-project_1234/
cibuild/v1/yaml/import
```

You'll be prompted for the password:

```
Enter host password for user 'alex.admin':
```

# How Do I Use YAML to Create or Configure a Job?

You can use YAML for creating a new job or configuring an existing one:

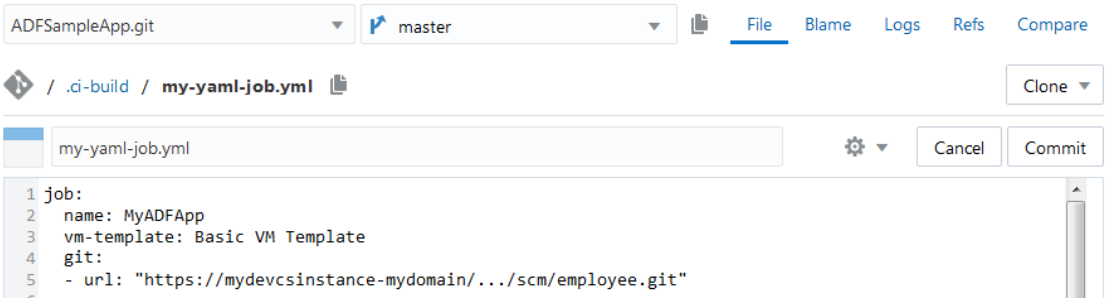
1. Clone the Git repository with the YAML file to your computer or to the location where you want to host it.
2. Create a file with the job's YAML configuration.

   See What Is the Format for a YAML Job Configuration?.
3. Save the file with the `.yml` extension in the `.ci-build` directory at the root of the cloned Git repository: `.ci-build/my_yaml_job.yml`
4. Validate the local YAML file. See How Do I Validate a Job or Pipeline Configuration?.

   Resolve any errors.
5. Commit and push the file to the project's Git repository.
6. Open the **Project Home** page and, in the Recent Activities Feed, verify that the YAML file' and job were created.

   If there are any validation issues with the YAML file, a notification with a **View Error** link is displayed. Click the **View Error** link to review the error messages. Then update the YAML file and commit it again.
7. Click the job's name to open it in the **Builds** page.

You can create the job configuration file using the code editor on the **Git** page too:



If you create the YAML file this way, you won't be able to validate it without committing it first. Commit the file and check the Recent Activities Feed on the **Project Home** page for any errors.

## What Is the Format for a YAML Job Configuration?

In a YAML job configuration, any field with a value of `""` accepts a string value that is empty by default. `""` is not a valid value for some fields, such as `name`, `vm-template`, and `url`.

When you configure a job, fields such as `name`, `description`, `vm-template`, and `auto` must precede groups like `git`, `params`, and `steps`.

Here's a job's YAML configuration format with the default values:

```
job:
  name: ""
  description: ""
  vm-template: ""          # required
  auto: false              # deprecated - true implies branch: master;
otherwise, set branch explicitly
  auto:
    branch: mybranch       # deprecated
                           # See Auto specification section below
  auto: mybranch           # automatically build a single branch on commit
  auto: "*"                # automatically build any branch on commit
  auto:
    include:               # array of branches or branch patterns to include,
for example
    - "*"                  # automatically build any branch on commit
    except:                # array of exceptions (optional)
    - ""                   # except these branches
  auto:
    exclude:               # array of branches or branch patterns to exclude
    - ""                   # default exclude nothing (include everything)
    except:                # array of exceptions (optional)
    - ""                   # but including these branches
  from-job: ""             # create job as copy of another job; ignored after
creation
  for-merge-request: false
  allow-concurrent: false  # if true, concurrent builds will be allowed if
necessary
  disabled: false          # if true, job will not build
                           #
                           # disabled=true/false can be specified for every
item in the job below
                           # e.g., for git, param, steps, etc. items
                           # for brevity, not shown below in every item
  git:
  - url: ""                # required
    branch: "master"       # branch: * is treated specially; see the Auto
build section above
    repo-name: "origin"
    local-git-dir: ""
    refspec: ""
    included-regions: ""   # deprecated - see trigger-when, file-pattern, and
exceptions
```

```
    excluded-regions: ""   # deprecated - see trigger-when, file-pattern, and
exceptions
    trigger-when: INCLUDE  # one of INCLUDE or EXCLUDE
    file-pattern: ""       # default is "**/*" for INCLUDE or "" for EXCLUDE
    exceptions: ""         # exceptions to INCLUDE or EXCLUDE file-pattern
    excluded-users: ""
    merge-branch: ""
    config-user-name: ""
    config-user-email: ""
    merge-from-repo: false
    merge-repo-url: ""
    checkout-revision: ""
    prune-remote-branches: false
    skip-internal-tag: true
    clean-after-checkout: false
    update-submodules: false
    use-commit-author: false
    wipeout-workspace: false
    build-on-commit: false
    shallow-clone: false   # Added 24.07.0. Optional, default false. Defines
if the git will use shallow cloning or not in which case the clone-depth will
be used
    clone-depth: 0         # Added 24.07.0. Optional, default 0. The depth of
the clone for that git if the shallow-clone is set to true
                           # When build-on-commit: true, the "auto" branch
can be specified as follows:
    include:               # A list of branches to include
    - "*"                  # Branch name, wildcard like "*" or regular
expressions like /.*/ are allowed
    except:                # Except do not include the branches in this list
    - "/^patchset_/"       # Branch name, example regular expression shown
                           # Or
    exclude:               # A list of branches to exclude (all branches not
excluded are included)
    - "/^patchset_/"       # Branch name, example regular expression shown
    except:                # Except do not exclude the branches in this list
    - patchset_21_07_0     # Branch name, example literal branch name shown
  params:
  # boolean, choice, and string parameters can be specified as string values
of the form - NAME=VALUE
  #   the VALUE of a boolean parameter must be true or false, e.g., -
BUILD_ALL=true
  #   the VALUE of a choice parameter is a comma-separated list, e.g., -
PRIORITY=NORMAL,HIGH,LOW
  #   the VALUE of a string parameter is anything else, e.g., - URL=https://
github.com
  # Alternatively, parameters can be specified as objects:
  - boolean:
      name: ""               # required
      value: true            # required
      description: ""
  - choice:
      name: ""               # required
      description: ""
      choices: []            # array of string value choices; at least one
required
```

```
  - merge-request:
      params:
      - GIT_REPO_BRANCH=""    # required
      - GIT_REPO_URL=""       # required
      - MERGE_REQ_ID=""
  - password:
      name: ""                # required
                              # one of password or private-key is required
                              # recommended to use named password/private key
reference like "#{NAME}"
      password: ""            # required, or
      private-key: ""         # required
      required: false         # if not present required is false. If required
is set to true then when the user will do a build now operation, he will have
to have a value
                              # for that field. The user will not be able to
submit the build with an empty value.
      description: ""
  - string:
      name: ""                # required
      value: ""               # required
      required: false         # if not present required is false. If required
is set to true then when the user will do a build now operation, he will have
to have a value
                              # for that field. The user will not be able to
submit the build with an empty value.
      description: ""
  before:
  - add-param:                # Add a parameter after git before rest of build
      parameter-name: ""      # required - name of added parameter
      file-path: ""           # required - file that contains value of
parameter
      sensitive: false        # true if sensitive, e.g., password or private
key
  - add-params:               # Add one or more parameters as above (cannot
be used to add password parameters)
      file-path: ""           # required - file that contains one or more
lines of the format
                              #   NAME=value
  - copy-artifacts:
      from-job: ""
      build-number: 1                # requires which-build: SPECIFIC_BUILD
      artifacts-to-copy: ""
      target-dir: ""
      which-build: "LAST_SUCCESSFUL"  # other choices: LAST_KEEP_FOR_EVER,
UPSTREAM_BUILD, SPECIFIC_BUILD, PERMALINK, PARAMETER
      last-successful-fallback: false
      permalink: "LAST_SUCCESSFUL"    # other choices: LAST, LAST_SUCCESSFUL,
LAST_FAILED, LAST_UNSTABLE, LAST_UNSUCCESSFUL
                                      # other choices require which-build:
PERMALINK
      param-name: "BUILD_SELECTOR"    # requires which-build: PARAMETER
      flatten-dirs: false
      optional: false
  - npm-registry-setup:
      use-current-project-registry: true    # true to use current project's
```

```
Built-in NPM registry
                                          # otherwise, specify one of
registry-url or connection
      connection: ""                    # required if use-current-project-
registry is false and registry-url is empty
      username: ""                      # required if registry at registry-url
requires authentication
      password: ""                      # required if username is specified
      registry-url: ""                  # required if use-current-project-
registry is false and connection is empty
      custom-npmrc: ""                  # optional path to a custom .npmrc from
the workspace
  - oracle-maven:
      connection: ""                    # required if otn-login or otn-password
is empty
      otn-login: ""                     # required if connection is empty
      otn-password: ""                  # required if connection is empty
      server-id: ""
      settings-xml: ""
  - security-check:
      perform-analysis: false           # true to turn on security dependency
analyzer of maven builds
      create-issues: false              # true to create issue for every
affected pom file
      fail-build: false                 # true to fail build if vulnerabilities
detected
      severity: "low"                   # low (CVSS >= 0.0), medium (CVSS >=
4.0), high (CVSS >= 7.0)
      confidence: "low"                 # low, medium, high, highest
      product: ""                       # required if create-issues true; "1"
for Default
      component: ""                     # required if create-issues true; "1"
for Default
  - ssh:
      config:
        private-key: ""                 # optional if ssh-tunnel: password
specified
        public-key: ""
        passphrase: ""
        server-public-key: ""           # leave empty to skip host verification
        setup-ssh: true.                # true if setup files in ~/.ssh for cmd
line tools
        ssh-tunnel: false
        username: ""                    # required if ssh-tunnel true
        password: ""                    # optional if ssh-tunnel true and
private-key specified
        local-port: 0                   # required if ssh-tunnel true
        remote-host-name: "localhost"   # optional if ssh-tunnel true
        remote-port: 0                  # required if ssh-tunnel true
        ssh-host-name: ""               # required if ssh-tunnel true (name or
IP)
  - sonarqube-setup:
      sonar-server: ""                  # required Server Name as configured in
Builds admin
  - xvfb:
      display-number: "0"
```

```
      screen-offset: "0"
      screen-dimensions: "1024x768x24"
      timeout-in-seconds: 0
      more-options: "-nolisten inet6 +extension RANDR -fp /usr/share/X11/
fonts/misc"
      log-output: true
      shutdown-xvfb-after: true
  steps:
  - cancel-configuration-set:
        configuration-set-id: ""         # required
        environment-name: ""             # required
        service-name: ""                 # required
        username:                        # required
        password:                        # required
  - ant:
      build-file: ""
      targets: ""
      properties: ""
      java-options: ""
  - application-ext-packaging:
      build-artifact: "extension.vx" # optional, defaults to 'extension.vx'
      version: ""
  - application-ext-delete:
      extension-id:                    # required
      extension-version:               # required
      environment-name:                # required
      service-name:                    # required
      auth-type: "OAUTH"               # optional, defaults to 'BASIC'
      username:                        # required if auth-type is 'BASIC'
      password:                        # required if auth-type is 'BASIC'
  - application-ext-audit:
      environment-name:                # required
      service-instance:                # required
      username:                        # required
      password:                        # required
      extension-id:                    # required
      extension-version:               # required
      options: |                       # optional, defaults to
'audit.outputfile=auditoutput.json', each option on its own line
        audit.outputfile=auditoutput.json
  - application-ext-test:
      karma-browser: "FirefoxHeadless" # optional, defaults to
'FirefoxHeadless'
      karma-log-level: "INFO"          # optional, defaults to 'INFO'
      mocha-timeout: 0                 # optional, defaults to 0
  - apply-configuration-set:
      configuration-set-id: ""         # required
      environment-name: ""             # required
      service-name: ""                 # required
      username:                        # required
      password:                        # required
  - bmccli:
      private-key: ""
      user-ocid: ""            # required
      fingerprint: ""          # required
      tenancy: ""              # required
```

```
      passphrase: ""
      region: "us-phoenix-1"   # current valid regions are: us-phoenix-1, us-
ashburn-1, eu-frankfurt-1, uk-london-1
                               # more may be added - check OCI configuration
  - docker-certificate:
      registry-host: ""        # required
      certificate: ""          # required
  - docker-build:              # docker commands require vm-template with
software bundle 'Docker'
      source: "DOCKERFILE"     # other choices: DOCKERTEXT, URL
      path: ""                 # docker file directory in workspace
      docker-file: ""          # Name of docker file; if empty use Dockerfile
      options: ""
      image:
        registry-host: ""
        registry-id: ""
        image-name: ""         # required
        version-tag: ""
      docker-text: ""          # required if source: DOCKERTEXT otherwise not
allowed
      context-root-url: ""     # required if source: URL otherwise not allowed
  - docker-image:
      options: ""
      image:
        registry-host: ""
        registry-id: ""
        image-name: ""
        version-tag: ""
  - docker-load:
      input-file: ""           # required
  - docker-login:
      registry-host: ""
      username: ""             # required
      password: ""             # required
  - docker-pull:
      options: ""
      timeout: null            # timeout pull request, in minutes
      image:
        registry-host: ""      # required
        registry-id: ""
        image-name: ""         # required
        version-tag: ""
  - docker-push:
      options: ""
      image:
        registry-host: ""      # required
        registry-id: ""
        image-name: ""         # required
        version-tag: ""
  - docker-rmi:
      remove: "NEW"            # other options: ONE, ALL
      options: ""
      image:                   # only if remove: ONE
        registry-host: ""      # required
        registry-id: ""
        image-name: ""         # required
```

```
        version-tag: ""
    - docker-save:
        output-file:             # required
        image:
          registry-host: ""      # if omitted Docker Hub is assumed
          registry-id: ""
          image-name: ""         # required
          version-tag: ""
    - docker-tag:
        source-image:
          registry-host: ""      # required
          registry-id: ""
          image-name: ""         # required
          version-tag: ""
        target-image:
          registry-host: ""      # required
          registry-id: ""
          image-name: ""         # required
          version-tag: ""
    - docker-version:
        options: ""
    - export-configuration-set:
        sandbox-name: ""                # required
        description: ""
        id-parameter-name: "CONFIGURATION_SET_ID"  # optional, defaults to
'CONFIGURATION_SET_ID'
        include-all-modules: false
        optional-modules: ""            # Comma-separated list of (zero or
more) Optional Module names or codes, eg. "CRM,BI". Default is empty string
meaning no module
        move-all-changes: false
        skip-target-check: false
        environment-name: ""            # required
        service-name: ""                # required
        username:                       # required
        password:                       # required
    - fn-build:
        build-args: ""
        work-dir: ""
        use-docker-cache: true
        verbose-output: false
        registry-host: ""
        username: ""
    - fn-bump:
        work-dir: ""
        bump: "--patch"          # other choices: "--major", "--minor"
    - fn-deploy:
        deploy-to-app: ""        # required
        build-args: ""
        work-dir: ""
        deploy-all: false
        verbose-output: false
        use-docker-cache: true
        no-version-bump: true
        do-not-push: true
        registry-host: ""
```

```
      username: ""
      api-url: ""              # required
  - fn-oci:
      compartment-id: ""       # required
      provider: ""
                               # Note: the passphrase field is no longer
required nor allowed
  - fn-push:
      work-dir: ""
      verbose: false
      registry-host: ""
      username: ""
  - fn-version: {}
  - gradle:
      use-wrapper: false
      wrapper-gradle-version: ""            # ignored unless use-wrapper: true
      make-executable: false                # ignored unless use-wrapper:
true, then default true
                                            # must set make-executable: true
if wrapper doesn't already exist
                                            # corresponds to Create 'gradlew'
wrapper
      from-root-build-script-dir: false     # ignored unless use-wrapper: true
      root-build-script: ""                 # ignored unless from-root-build-
script-dir: true; script directory
      tasks: "clean build"
      build-file: "build.gradle"
      switches: ""
      use-workspace-as-home: false
      description: ""
      use-sonar: false                      # if true sonarqube-setup must be
configured
  - import-configuration-set:
      configuration-set-id: ""         # required
      ignore-unpublished-sandboxes: false
      environment-name: ""             # required
      service-name: ""                 # required
      username:                        # required
      password:                        # required
  - maven:
      goals: "clean install"
      pom-file: "pom.xml"
      private-repo: false
      private-temp-dir: false
      offline: false
      show-errors: false
      recursive: true
      profiles: ""
      properties: ""
      verbosity: NORMAL                # other choices: DEBUG, QUIET
      checksum:  NORMAL                # other choices: STRICT, LAX
      snapshot:  NORMAL                # other choices: FORCE, SUPPRESS
      projects: ""
      resume-from: ""
      fail-mode:  NORMAL               # other choices: AT_END, FAST, NEVER
      make-mode:  NONE                 # other choices: DEPENDENCIES,
```

```
DEPENDENTS, BOTH
      threading: ""
      jvm-options: ""
      use-sonar: false                # if true, sonarqube-setup must be
configured
  - nodejs:
      source: SCRIPT                  # other choice: FILE
      file: ""                        # only if source: FILE
      script: ""                      # only if source: SCRIPT
  - oic-activate-integration:
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      identifier: ""                  # required, the uppercase integration
identifier
      version: ""                     # required, the integration version
      deactivate: true                # when replacing the integration,
deactivate it first if it is active
      oracle-recommends-flag: true    # see https://docs.oracle.com/en/cloud/
paas/integration-cloud/integrations-user/activate-and-deactivate-
integrations.html
      record-enabled-flag: false
      tracing-enabled-flag: false
      payload-tracing-enabled-flag: false
  - oic-delete-integration:
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      identifier: ""                  # required, the uppercase integration
identifier
      version: ""                     # required, the integration version
  - oic-delete-lookup:
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      lookup-name: ""                 # required, the name of the lookup to
delete
  - oic-delete-package:
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      package-name: ""                # required, the name of the package to
delete
      deactivate-integrations: false  # if true, automatically deactivate
```

```
integrations before deleting package
  - oic-export-integration:
      environment-name: ""              # required, identifies the environment
containing the OIC instance
      service-name: ""                  # required, the OIC instance for the
operation
      username: ""                      # required
      password: ""                      # required
      identifier: ""                    # required, the uppercase integration
identifier
      version: ""                       # required, the integration version
      include-recording-flag: false
  - oic-export-lookup:
      environment-name: ""              # required, identifies the environment
containing the OIC instance
      service-name: ""                  # required, the OIC instance for the
operation
      username: ""                      # required
      password: ""                      # required
      lookup-name: ""                   # required, the name of the lookup to
export
  - oic-export-package:
      environment-name: ""              # required, identifies the environment
containing the OIC instance
      service-name: ""                  # required, the OIC instance for the
operation
      username: ""                      # required
      password: ""                      # required
      package-name: ""                  # required, the name of the package to
export
      include-recording-flag: false
  - oic-import-integration:
      environment-name: ""              # required, identifies the environment
containing the OIC instance
      service-name: ""                  # required, the OIC instance for the
operation
      username: ""                      # required
      password: ""                      # required
      integration-archive: ""           # required, the filename of the
integration archive file (<IDENTIFIER>_<VERSION>.iar)
      import-method: "ADD"              # other choices: REPLACE, AUTOMATIC
      deactivate: true                  # when replacing the integration,
deactivate it first if it is active
      include-recording-flag: true      # include asserter recordings (if any)
      activate: false                   # see https://docs.oracle.com/en/cloud/
paas/integration-cloud/integrations-user/activate-and-deactivate-
integrations.html
      oracle-recommends-flag: true
      record-enabled-flag: false
      tracing-enabled-flag: false
      payload-tracing-enabled-flag: false
  - oic-import-lookup:
      environment-name: ""              # required, identifies the environment
containing the OIC instance
      service-name: ""                  # required, the OIC instance for the
operation
```

```
      username: ""                    # required
      password: ""                    # required
      lookup-archive: ""              # required, the filename of the lookup
archive file (<lookupname>.csv)
      import-method: "ADD"            # other choices: REPLACE, AUTOMATIC
  - oic-import-package:
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      package-archive: ""             # required, the filename of the
package archive file (<packagename>.par)
      import-method: "ADD"            # other choices: REPLACE, AUTOMATIC
      deactivate-integrations: true   # when replacing the package,
deactivate any of its active integrations first
      reactivate-integrations: true   # after replacing the package,
reactivate any integrations that were deactivated by the deactivate-
integrations option
      include-recording-flag: true    # include asserter recordings (if any)
  - oic-update-connection:
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      identifier: ""                  # required, the identifier of the
connection to update
      json-file: ""                   # required if json-text not specified,
the filename of the json file containing connection properties to update
      json-text: ""                   # required if json-file not specified,
the inline json text containing connection properties to update
  - oic-upload-connection-property-attachment
      environment-name: ""            # required, identifies the environment
containing the OIC instance
      service-name: ""                # required, the OIC instance for the
operation
      username: ""                    # required
      password: ""                    # required
      identifier: ""                  # required, the identifier of the
connection to update
      property-name: ""               # required, the name of the property
to attach the upload file to
      file-path: ""                   # required, specifies the zip file or
WSDL file to upload
      service-wsdl: ""                # required if file-path is a zip file,
specifies file path of WSDL file within the zip
  - oracle-deployment:                # currently Visual Applications,
Application Extensions, and JCS using REST are supported
      environment-name: ""            # required, scopes the service-name
      service-name: ""                # required, the service instance type
determines the deployment type
      auth-type: "OAUTH"              # optional, defaults to 'BASIC'
      username: ""                    # required if Visual Application
```

```
                                    # required if Application Extension
deployment and auth-type is 'BASIC'
                                    # required if JCS deployment, and then
it is the weblogic username
      password: ""                  # required if Visual Application
                                    # required if Application Extension
deployment and auth-type is 'BASIC'
                                    # required if JCS, and then it is the
weblogic user's password
      application-version: ""       # optional if Visual Application
(defaults from visual-application.json), else n/a
      application-profile: ""       # optional if Visual Application, else
n/a
      include-application-version-in-url: true  # required if Visual
Application, other choice: false
      data-management: "KEEP_EXISTING_ENVIRONMENT_DATA"     # required if
Visual Application, other choice: "USE_CLEAN_DATABASE"
      sources: ""                   # optional if Visual Application
(defaults to build/sources.zip), else unused
      build-artifact: ""            # optional if Visual Application
(defaults to build/built-assets.zip), else unused
                                    # required if Application Extension
                                    # required if JCS
      application-name: ""          # required if JCS, else n/a
      weblogic-version: ""          # for JCS (if specified, must be
12.2.x)
      https-port: "7002"            # required if JCS
      protocol: "REST"              # for JCS (if specified, one of REST,
REST1221)
      targets: ""                   # required if JCS, one or more names
of target service or cluster, comma-separated
  - application-ext-deployment:
      environment-name: ""          # required, scopes the service-name
      service-name: ""              # required, the service instance name
in the environment
      auth-type: "OAUTH"            # optional, defaults to 'BASIC'
      username: ""                  # required if auth-type is 'BASIC'
      password: ""                  # required if auth-type is 'BASIC'
      build-artifact: ""            # required
  - visual-app-deployment
      environment-name: ""          # required, scopes the service-name
      service-name: ""              # required, the service instance name
in the environment
      auth-type: "OAUTH"            # optional, defaults to 'BASIC'
      username: ""                  # required if auth-type is 'BASIC'
      password: ""                  # required if auth-type is 'BASIC'
      application-version: ""       # optional (defaults from visual-
application.json)
      application-profile: ""       # optional
      include-application-version-in-url: true  # required, other choice:
false
      data-management: "KEEP_EXISTING_ENVIRONMENT_DATA"     # required, other
choice: "USE_CLEAN_DATABASE"
      sources: ""                   # optional (defaults to build/
sources.zip)
      build-artifact: ""            # optional (defaults to build/built-
```

```
assets.zip)
  - jcs-deployment:
      environment-name: ""            # required, scopes the service-name
      service-name: ""                # required, the service instance name
in the environment
      username: ""                    # required, the weblogic username
      password: ""                    # required, the weblogic user's
password
      build-artifact: ""              # required
      application-name: ""            # required
      https-port: "7002"              # required
      targets: ""                     # required, one or more names of
target service or cluster, comma-separated
      library: false                  # optional, defaults to false
  - restore-configuration-set:
      configuration-set-id: ""        # required
      environment-name: ""            # required
      service-name: ""                # required
      username:                       # required
      password:                       # required
  - shell:
      script: ""
      xtrace: true
      verbose: false                  # both verbose and xtrace cannot be
true
      use-sonar: false                # if true sonarqube-setup must be
configured
  - sqlcl:
      username: ""
      password: ""
      credentials-file: ""
      connect-string: ""
      source: SQLFILE                 # other choice: SQLTEXT
      sql-file: ""                    # only if source: SQLFILE
      sql-text: ""                    # only if source: SQLTEXT
      role: DEFAULT                   # other choices: SYSDBA, SYSBACKUP,
SYSDG, SYSKM, SYSASM
      restriction-level: DEFAULT      # other choices: LEVEL_1, LEVEL_2,
LEVEL_3, LEVEL_4
  - vbappops-export-data:
      environment-name:               # required
      service-instance:               # required
      vb-project-id:                  # required
      vb-project-version:             # required
      username:                       # required
      password:                       # required
      app-data-file:                  # required
  - vbappops-import-data:
      environment-name:               # required
      service-instance:               # required
      vb-project-id:                  # required
      vb-project-version:             # required
      username:                       # required
      password:                       # required
      app-data-file:                  # required
  - vbappops-lock-app:
```

```
      environment-name:                # required
      service-instance:                # required
      vb-project-id:                   # required
      vb-project-version:              # required
      username:                        # required
      password:                        # required
  - vbappops-unlock-app:
      environment-name:                # required
      service-instance:                # required
      vb-project-id:                   # required
      vb-project-version:              # required
      username:                        # required
      password:                        # required
  - vbappops-undeploy-app:
      environment-name:                # required
      service-instance:                # required
      vb-project-id:                   # required
      vb-project-version:              # required
      username:                        # required
      password:                        # required
  - vbappops-rollback-app:
      environment-name:                # required
      service-instance:                # required
      vb-project-id:                   # required
      vb-project-version:              # required
      username:                        # required
      password:                        # required
  - visual-app-packaging:
      sources: "build/sources.zip"     # optional, defaults to 'build/
sources.zip'
      build-artifact: "build/built-assets.zip" # optional, defaults to 'build/
built-assets.zip'
      optimize: true                     # boolean
  - visual-app-audit:
      environment-name:                # required
      service-instance:                # required
      username:                        # required
      password:                        # required
      app-url-root:                    # required
      app-version:                     # required
      options: |                       # optional, defaults to
'audit.outputfile=auditoutput.json', each option on its own line
        audit.outputfile=auditoutput.json
  - visual-app-test:
      karma-browser: "FirefoxHeadless" # optional, defaults to
'FirefoxHeadless'
      karma-log-level: "INFO"          # optional, defaults to 'INFO'
      mocha-timeout: 0                 # optional, defaults to 0
  after:
  - artifacts:
      include: ""                    # required
      exclude: ""
      maven-artifacts: false
      include-pom: false             # ignored unless maven-artifacts: true
  - export-param:                    # Add a parameter after git before rest
of build
```

```
      parameter-name: ""              # required - name of added parameter
      file-path: ""                   # required - file that contains value
of parameter
      sensitive: false                # true if sensitive, e.g., password or
private key
  - export-params:                    # Add one or more parameters as above
(cannot be used to add password parameters)
      file-path: ""                   # required - file that contains one or
more lines of the format
                                      #   NAME=value
  - git-push:
      push-on-success: false
      merge-results: false
      tag-to-push: ""
      create-new-tag: false
      tag-remote-name: "origin"
      branch-to-push: ""
      branch-remote-name: "origin"
      local-git-dir: ""
  - javadoc:
      javadoc-dir: "target/site/apidocs"
      retain-for-each-build: false
  - junit:
      include-junit-xml: "**/surefire-reports/*.xml"
      exclude-junit-xml: ""
      keep-long-stdio: false
      organize-by-parent: false
      fail-build-on-test-fail: false
      archive-media: true
  - sonarqube:                            # sonarqube-setup must be configured
      replace-build-status: true          # Apply SonarQube quality gate
status as build status
      archive-analysis-files: false
  settings:
  - abort-after:
      hours: 0
      minutes: 0
      fail-build: false
  - build-retry:
      build-retry-count: 5
      git-retry-count: 5
  - discard-old:
      days-to-keep-build: 0
      builds-to-keep: 100
      days-to-keep-artifacts: 0
      artifacts-to-keep: 20
  - git-poll:
      cron-pattern: "0/30 * * * * #Every 30 minutes"
  - log-size:
      max: 50                     # megabytes
  - logger-timestamp:
      timestamp: true
  - periodic-build:
      cron-pattern: "0/30 * * * * #Every 30 minutes"
  - quiet-period:
      seconds: 0
```

```
  - versions:
      version-map:
        Java: "17"      # For templates the configurable options (with
defaults wrapped in '*' chars) are
                        #   Java: 8, 11, *17*, 8 (GraalVM)
                        # For the Built-in (Free) executors, the options are
                        #   Java: 8, 11, or *17*
                        #   nodejs: 0.12 or *10*
                        #   python3: 3.5, or *3.6*
                        #   soa: 12.1.3, or *12.2.1.1*
```

## YAML Job Configuration Examples

Here are several examples of YAML job configurations:

| Job Configuration | YAML Code |
|---|---|
| This configuration creates a job that runs Maven goals then archives the artifacts:<br>• Job Name: MyFirstYAMLJob<br>• Job's Build Executor Template: Basic Build Executor Template<br>• Git repository: employee.git<br>• Maven step:<br>– Goals: clean install<br>– POM file: employees-app/pom.xml<br>• After build action:<br>– Archived artifacts: employees-app/target/* | `job:`<br>`  name: MyFirstYAMLJob`<br>`  vm-template: Basic Build Executor Template`<br>`  git:`<br>`  - url: "https://mydevcsinstance-mydomain/.../scm/employee.git"`<br>`  steps:`<br>`  - maven:`<br>`  goals: clean install`<br>`  pom-file: "employees-app/pom.xml"`<br>`  after:`<br>`  - artifacts:`<br>`  include: "employees-app/target/*"` |

| Job Configuration | YAML Code |
|---|---|
| This configuration creates a job to run Docker steps that log in, build, and push an image to the OCI Registry:<br>• Job Name: MyDockerJob<br>• Job's Build Executor Template: Docker and Node.js Template<br>• Job Description: Job to build and push a Node.js image to the OCI Registry<br>• Git Repository: NodeJSMicroDocker.git<br>• Docker steps:<br>– Docker registry host: iad.ocir.io<br>– Username: myoci/ociuser<br>– Password: My123Password<br>– Image name: myoci/ociuser/mynodejsimage<br>– Proxy options: --build-arg https_proxy=http://my-proxy-server:80 | `job:`<br>`  name: MyDockerJob`<br>`  description: Job to build and push a Node.js image to OCI Registry`<br>`  vm-template: Docker and Node.js Template`<br>`  git:`<br>`  - url: "https://mydevcsinstance-mydomain/.../scm/NodeJSMicroDocker.git"`<br>`  steps:`<br>`  - docker-login:`<br>`  registry-host: "https://iad.ocir.io"`<br>`  username: "myoci/ociuser"`<br>`  password: My123Password`<br>`  - docker-build:`<br>`  source: "DOCKERFILE"`<br>`  options: "--build-arg https_proxy=https://my-proxy-server:80"`<br>`  image:`<br>`  image-name: "myoci/ociuser/mynodejsimage"`<br>`  version-tag: "1.8"`<br>`  registry-host: "https://iad.ocir.io"`<br>`  path: "mydockerbuild/"`<br>`  - docker-push:`<br>`  image:`<br>`  registry-host: "https://iad.ocir.io"`<br>`  image-name: "myoci/ociuser/mynodejsimage"`<br>`  version-tag: "1.8"`<br>`  - docker-image:`<br>`  options: "--all"` |
| This configuration creates a job that uses SQLcl to run SQL commands and a script:<br>• Job Name: RunSQLJob<br>• Job's Build Executor Template: Basic Build Executor Template<br>• SQL steps:<br>– Username: dbuser<br>– Password: My123Password<br>– Connect string: myserver.oracle.com:1521:db1234<br>– SQL commands:<br>`CD /home`<br>`select * from Emp`<br>– SQL script file: sqlcl/simpleselect.sql | `job:`<br>`  name: RunSQLJob`<br>`  vm-template: Basic Build Executor Template`<br>`  steps:`<br>`  - sqlcl:`<br>`  username: dbuser`<br>`  password: My123Password`<br>`  connect-string: "myserver.oracle.com:1521:db1234"`<br>`  sql-text: "CD /home\nselect * from Emp"`<br>`  source: "SQLTEXT"`<br>`  - sqlcl:`<br>`  username: dbuser`<br>`  password: My123Password`<br>`  connect-string: "myserver.oracle.com:1521:db1234"`<br>`  sql-file: "sqlcl/simpleselect.sql"`<br>`  source: "SQLFILE"` |

| Job Configuration | YAML Code |
|---|---|
| This configuration creates a job that runs Maven goals and archives the artifacts:<br>• Job Name: MyADFApp<br>• Job's Build Executor Template: JDev and ADF Template<br>• Git Repository: ADFApp.git<br>• Run a build on a push update to the patchset_1 branch: Yes<br>• Git repository branch: patchset_1<br>– Files to track for changes: myapp/src/main/web/.*\.java<br>– Files to ignore for changes: myapp/src/main/web/.*\.gif<br>– Remove untracked files before running a build: Yes<br>– Display the commit author in the log: Yes<br>• Copy artifacts from another job: ADFDependencies<br>– Artifacts: adf-dependencies.war<br>• Oracle Maven Repository connection:<br>– OTN username: alex.admin@example.com<br>– OTN password: My123Password<br>• Maven step:<br>– Goals: clean install package<br>– POM file: WorkBetterFaces/pom.xml<br>• After build steps:<br>– Artifacts to archive: WorkBetterFaces/target/*.ear<br>• Other settings:<br>– Java version: 17<br>– Discard old builds: Yes<br>* Number of builds to keep: 50<br>* Number of builds to keep: 10<br>– Periodic build trigger:<br>* Hour: 2<br>* Minutes: 30<br>– Build retry count: 5<br>– SCM retry count: 10<br>– Abort if the build is stuck: 1 hour | `job:`<br>`  name: MyADFApp`<br>`  vm-template: JDev and ADF Build Executor Template`<br>`  auto:`<br>`  branch: "patchset_1"`<br>`  git:`<br>`  - url: "https://mydevcsinstance-mydomain/.../scm/ADFApp.git"`<br>`  branch: patchset_1`<br>`  build-on-commit: true`<br>`  included-regions: "myapp/src/main/web/.*\\.java"`<br>`  excluded-regions: "myapp/src/main/web/.*\\.gif"`<br>`  clean-after-checkout: true`<br>`  before:`<br>`  - copy-artifacts:`<br>`  from-job: ADFDependecies`<br>`  artifacts-to-copy: adf-dependencies.war`<br>`  - oracle-maven:`<br>`  otn-login: "alex.admin@example.com"`<br>`  otn-password: My123Password`<br>`  steps:`<br>`  - maven:`<br>`  goals: clean install package`<br>`  pom-file: "WorkBetterFaces/pom.xml"`<br>`  after:`<br>`  - artifacts:`<br>`  include: "WorkBetterFaces/target/*.ear"`<br>`  settings:`<br>`  general:`<br>`  - discard-old:`<br>`  days-to-keep-build: 50`<br>`  builds-to-keep: 10`<br>`  software:`<br>`  - versions:`<br>`  version-map:`<br>`  Java: 17`<br>`  triggers:`<br>`  - git-poll:`<br>`  cron-pattern: "0/30 5 * 2 *"`<br>`  advanced:`<br>`  - abort-after:`<br>`  hours: 1`<br>`  - build-retry:`<br>`  build-retry-count: 5`<br>`  git-retry-count: 10` |

# How Do I Use YAML to Create or Configure a Pipeline?

You can use YAML for creating a new pipeline or configuring an existing one:

1. Clone the Git repository with the YAML file, to your computer or to the location where you want host it.

2. Create a file with the pipeline's YAML configuration.
3. Save the file with the `.yml` extension in the `.ci-build` directory at the root of the cloned Git repository: `.ci-build/my_yaml_pipeline.yml`
4. Validate the local YAML file. See How Do I Validate a Job or Pipeline Configuration?.

   Resolve any issues, if any were reported.
5. Commit and push the file to the project's Git repository.
6. Open the **Project Home** page and, in the Recent Activities Feed, verify the notification about the YAML file. You should see notifications that the YAML file and pipeline were created.

   If there were any issues with the YAML file, a notification with a **View Error** link would be displayed. Click the **View Error** link to download a JSON file with the error messages. Review the error, update the YAML file, and commit the file again.
7. To view the pipeline, go to the **Builds** page and open the **Pipelines** tab:
   - To run a build of the pipeline's jobs, click **Build**.
   - To view its instances, click the pipeline's name. To edit its YAML configuration, click **Configure**.

## What Is the Format for a YAML Pipeline Configuration?

Here's a pipeline's configuration with the default values in YAML format:

```
pipeline:
  name: ""                      # pipeline name - if omitted, name is
constructed from repository and file name
  description: ""               # pipeline description
  auto-start: true              # automatically start pipeline if any job in
pipeline is run
                                # if false, pipeline will start only if
manually started
                                #   or a trigger action item is activated
  auto-start:                   # implied true
    triggers-only: false        # if true, autostart only for jobs that have
no preceding jobs
  disabled: false               # if true, pipeline will not run
  allow-external-builds: true   # jobs in pipeline can run independently while
pipeline is running
  periodic-trigger: ""          # cron pattern with 5 elements (minute, hour,
day, month, year)
                                #   the pipeline is started (beginning with
the Start item) periodically
  triggers:                     # define trigger action items of periodic,
poll, or commit types
                                #   there may be one or more of each type
  - periodic:                   # define trigger action item of periodic type;
build pipeline every so often
      name: ""                  # required, trigger name - must be unique
trigger name; may not be "Start"
      cron-pattern: ""          # required, cron pattern specifying Minute
Hour Day Month Year, e.g., "? 0 * * *"
  - poll:                       # define trigger action item of poll type;
```

```
poll repository every so often
      name: ""                  # required, trigger name - must be unique
trigger name; may not be "Start"
      cron-pattern: ""          # required, cron pattern as above
      url: ""                   # required, git repository URL
      branch: ""                # required, git repository branch - trigger
activated if changes detected in branch
      exclude-users: ""         # user identifier of committer to ignore
                                #   if more than one user, use multi-line
text, one user per line
      trigger-when: INCLUDE     # activate only if change to files in file-
pattern; alternative EXCLUDE
                                #   if EXCLUDE, activate only for change to
files not in file-pattern
      file-pattern: ""          # file(s) to INCLUDE/EXCLUDE; may be ant or
wildcard-style file/folder pattern
                                #   if more than one pattern, use multi-line
text, one pattern per line
                                #   for example...
      exceptions: ""            # exceptions to file-pattern above
                                #   if more than one exception file pattern,
use multi-line text, one pattern per line
                                #
                                # Example of multi-line pattern - note that
these lines can't have comments, as they would be part of text
      file-pattern: |
                  README*
                  *.sql
  - commit:                     # define trigger action item of commit type;
automatically run pipeline on commit
      name: ""                  # required, trigger name - must be unique
trigger name; may not be "Start"
      url: ""                   # required, git repository URL for local
project repository
      branch: ""                # git repository branch name
                                #   required: must specify branch or include/
exclude/except branch patterns
      include: ""               # branch patterns to include; branch name,
wildcard or regex
                                #   if more than one pattern, use multi-line
text, one pattern per line
      exclude: ""               # branch patterns to ignore; specify either
include or exclude, not both
                                #   if more than one pattern, use multi-line
text, one pattern per line
      except: ""                # branch pattern exceptions to include or
exclude above
                                #   if more than one pattern, use multi-line
text, one pattern per line
      exclude-users: ""         # user identifier of committer to ignore
                                #   if more than one user, use multi-line
text, one user per line
      trigger-when: INCLUDE     # activate only if change to files in file-
pattern; alternative EXCLUDE
                                #   if EXCLUDE, activate only for change to
files not in file-pattern
```

```
      file-pattern: ""          # file(s) to INCLUDE/EXCLUDE; may be ant or
wildcard-style file/folder pattern
                                #   if more than one pattern, use multi-line
text, one pattern per line
      exceptions: ""            # exceptions to file-pattern above
                                #   if more than one exception file pattern,
use multi-line text, one pattern per line
  start:                        # required begins an array of job names, or
parallel, sequential, or on groups
  - JobName                     # this job runs first, and so on (start is a
sequential group)
                                # Groups:
  - parallel:                   # items in group run in parallel
  - sequential:                 # items in group run sequentially
  - on succeed,fail,test-fail: # items in group run sequentially if preceding
job result matches condition
                                # can specify one or more of conditions:
                                #   succeed (success), fail (failure), or test-
fail (post-fail)
                                # Examples:
  - parallel:                   # jobs A, B and C run in parallel, job D runs
after they all finish
    - A
    - B
    - C
  - D
  - on succeed:                 # if job D succeeds, E builds, otherwise F
builds
    - E
  - on fail, test-fail:
    - F
                                #
  start:                        # Jobs that trigger pipelines can be specified.
  - trigger:                    # A trigger section appears before the job(s)
it triggers
    - JobA                      # trigger is a "parallel" second - JobA and
JobB are independent
    - JobB
  - JobName                     # This job runs first. It can be started when
the pipeline is run, or if
                                # either of the trigger jobs JobA or JobB is
built successfully.
                                # Triggers assume that auto-start is true.
  start:
  - A
  - parallel:
    - B
    - sequential:               # A trigger cannot appear in a parallel section
      - trigger:                # The jobs triggered are the next in sequence
        - Trigger1
      - C
  - trigger:                    # But can appear anywhere in a sequential
section
    - Trigger2
  - D
  - trigger:                    # A trigger at the end of the start section is
```

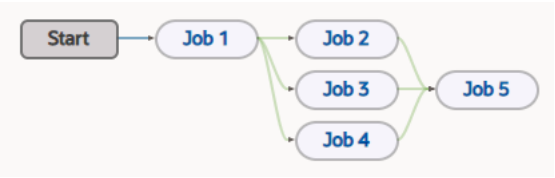
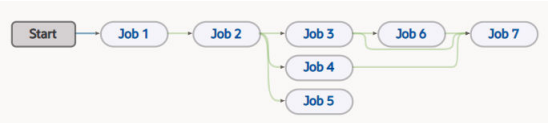
```
an independent graph
    - sequential:              # not connected to anything that precedes it.
      - X
      - Y
      - Z
                               #
                               # on sections can "join" - like an if/then/
else followed by something else
  start:
  - A
  - on fail:
    - F                        # If A fails, build F
  - on test-fail:
    - T                        # If the tests for A fail, build T
  - on succeed:
    - <continue>               # If A succeeds, fall through to whatever
follows the on conditions for A
  - B                          # B is built if A, F, or T succeed
                               #
                               # A job run in parallel (or conditionally) can
end the chain
  start:
  - A
  - parallel:                  # Run B, C, and D in parallel
    - B
    - C
    - end:
      - D                      # There is no arrow from D
  - E                          # E is run if B and C succeed
                               #
  start:
  - A
  - on fail:
    - end:
      - F                      # If A fails, build F and end the pipeline
  - on test-fail:
    - T                        # If the tests for A fail, build T
  - on succeed:
    - <continue>               # If A succeeds, fall through to whatever
follows the on conditions for A
  - B                          # B is built if A or T succeed
                               #
-------------------------------------------------------------------------------
-
                               # Not all pipelines you can draw can be
represented in hierarchical form as above.
                               # To allow a YAML definition of any pipeline
graph, you can use a graph notation
                               # similar to the digraph representation
supported by Dot/GraphViz.
                               # For example, the pipeline with triggers
above can be written as a graph.
  graph:                       # (Both graph: and start: cannot be used in
the same pipeline.)
  - JobA -> JobName            # There is a link from JobA to JobName
  - JobB -> JobName            # There is a link from JobB to JobName
```

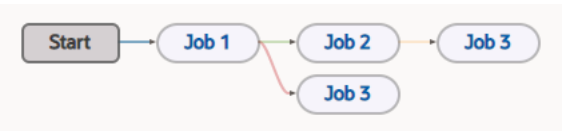
```
  - <Start> -> JobName          # There is a link from Start to JobName
                                # The representation <Start> distinguishes the
special "Start" node that
                                # appears in every pipeline from a job named
Start.
                                #
                                # Conditional links can be represented using
the ? and a list of one or more conditions.
                                # For example, the partial pipeline above
beginning with 'parallel' can be represented as:
  graph:
  - <Start> -> A
  - <Start> -> B
  - <Start> -> C
  - A -> D
  - B -> D
  - C -> D
  - D -> E ? succeed            # If D succeeds, E is built
  - D -> F ? fail, test-fail    # If D fails or tests fail, F is built
                                # "succeed" is the default when no ? is
specified.
                                #
                                # Any combination of succeed (success), fail
(failure), or test-fail (post-fail)
                                # can be written in a comma-separated list
after the question mark.
                                #
                                # Not every graph that can be specified in
this way is a valid pipeline.
                                # For example, graphs with cycles are not
allowed.
                                #
                                # "Joins" like the A, B, C converging on D
above only work (D gets built)
                                # if all of A, B, and C succeed. If, for
example, B fails, D will not be built.
                                # However, joins on nodes that are directly
downstream from [Start] are a
                                # special case. If any job triggers these
nodes, they will be run.
                                # This special case allows the triggers:
section to work as expected.
                                # (This is not new behavior in 22.01.0.)
  graph:
  - <Start> -> A
  - <Start> -> B
  - A -> C                      # Links from A to C and B to C are to the same
node (and job) C
  - B -> C
                                # on the other hand...
  graph:
  - <Start> -> A
  - <Start> -> B
  - A -> C                      # Links from A to C and B to C$2 are to the
same job C
  - B -> C$2                    # but to different nodes
```

```
                                    # In other words, the job C appears in two
different
```

## YAML Pipeline Configuration Examples

Here are some examples of different YAML pipeline configurations:

| YAML Definition | Pipeline Configuration |
|---|---|
| `pipeline:`<br>`  name: My Pipeline`<br>`  description: YAML pipeline configuration`<br>`  auto-start: true`<br>`  allow-external-builds: true`<br>`  start:`<br>`  - Job 1`<br>`  - Job 2`<br>`  - Job 3` | <br>Job 2 runs after Job 1 completes successfully. Then Job 3 runs after Job 2 completes successfully.<br>The pipeline automatically starts if any job in the pipeline is run. The default settings are used even if they all aren't listed in the YAML file:<br>• `auto-start` is implied to be `true` if it isn't listed, but it is included in this configuration.<br>• `triggers-only` is set to `false` by default and isn't listed here in this configuration. You could explicitly set it to `true`, in which case the autostart option will only be applied to jobs that have no preceding jobs.<br>• `allow-external-builds` is set to `true` and is included in this configuration, so jobs in the pipeline can run independently while the pipeline is running.<br>If you change these settings, you should expect different results. |
| `pipeline:`<br>`  name: My Pipeline`<br>`  auto-start: true`<br>`  start:`<br>`  - Job 1`<br>`  - parallel:`<br>`  - Job 2`<br>`  - Job 3`<br>`  - Job 4`<br>`  - Job 5` |  <br>Jobs 2, 3, and 4 run in parallel after Job 1 completes successfully. Job 5 runs after the three parallel jobs complete successfully.<br>The pipeline will start if any job in the pipeline runs. |
| `pipeline:`<br>`  name: My Pipeline`<br>`  start:`<br>`  - Job 1`<br>`  - Job 2`<br>`  - parallel:`<br>`  - sequential`<br>`  - Job 3`<br>`  - Job 6`<br>`  - Job 4`<br>`  - Job 5`<br>`  - Job 7` |  <br>Job 2 runs after Job 1 completes successfully. Jobs 3, 4, and 5 run in parallel after Job 2 completes successfully. Job 6 runs after Job 3 completes successfully. Job 7 runs after Jobs 6, 3, and 4 complete successfully. |

| YAML Definition | Pipeline Configuration |
| --- | --- |
| `pipeline:`<br>`  name: My Pipeline`<br>`  start:`<br>`  - Job 1`<br>`  - on succeed:`<br>`  - Job 2`<br>`  - on fail:`<br>`  - Job 3` | <br>If Job 1 runs successfully, Job 2 runs. If Job 1 runs and fails, Job 3 runs. |
| `pipeline:`<br>`  name: My Pipeline`<br>`  start:`<br>`  - Job 1`<br>`  - on succeed:`<br>`  - Job 2`<br>`  - on test-fail:`<br>`  - Job 3`<br>`  - on fail:`<br>`  - Job 3` |  <br>If Job 1 runs successfully, Job 2 is run. If Job 2 runs successfully but fails tests or any post build action, or if Job 1 fails, Job 3 is run. Job 3 *won't* run if Job1 completes successfully. |

## Set Dependency Conditions in Pipelines Using YAML

When you create a pipeline that includes a dependency between a parent and a child job, by default, the build of the child job will run after the parent job's build completes successfully. You can configure the dependency to run a build of the child job after the parent job's build fails too, either by using the pipeline designer or by setting an "on condition" in YAML to configure the result condition.

The pipeline designer supports **Successful**, **Failed**, or **Test Failed** conditions (see Configure the Dependency Condition). YAML supports additional conditions you can use. Here they are, with the build results they are mapped to:

- "succeed" and "success" map to a "SUCCESSFUL" build result
- "fail" and "failure" map to a "FAILED" build result
- "test-fail" and "post-fail" map to a "POSTFAILED" build result

None of these conditions match when a job is aborted, canceled, or restarted, so the pipeline never proceeds beyond that job.

See YAML Pipeline Configuration Examples to learn more about using and setting some of these dependency conditions in YAML. The fourth example shows how to use the "on succeed" and "on fail" settings. The fifth example shows how to use the "on succeed", "on fail", and "on post-fail" settings.

You can use the new public API to view the pipeline instance log to see what happened with the builds in the pipeline, after the fact. Use this format to get the log:

```
GET pipelines/{pipelineName}/instances/{instanceId}/log
```

## Define and Use Triggers in YAML

Triggers are artifacts in a pipeline that aren't jobs but are just nodes, called action items. A job or multiple jobs can be used as triggers, but there is an overhead cost associated with such use. Instead of using trigger jobs, you can specify a new category of action items in the YAML

pipeline configuration to define triggers. Trigger action items can start up on their own and then trigger the rest of the pipeline. This is a YAML only feature. To understand triggers, it helps to explore action items.

### What Are Action Items?

Action items, a category of special-purpose executable pipeline items, are used to automate actions when jobs and tasks are too heavyweight. An action is a short-running activity that's performed locally in the build system. This special-purpose long-lived executable entity appears as a node in a pipeline and can be started by a user action, an automated action, or by entry from an upstream item.

Actions are single-purpose where each action does one thing. When an action is started, it performs some action, and completes with a result condition. An action is configurable but not programmable and should *never* contain user-written code in any form. An action has a category, like “trigger”, and a sub-category, like “periodic” or “polling”, that defines the item type. Each action has a name that is unique within a pipeline. If the name isn't configured, a default name will be supplied based on the item type, for example, “PERIODIC-1”. The name “Start” is reserved. The name is required and represents a configuration of the action.

### What Is a Trigger?

A **trigger** is an action item that is based on some user or automated event that starts executing a pipeline at a specific point beginning with the nodes directly downstream of the trigger. If a trigger has upstream connections, and is invoked from an upstream connection, the trigger acts as a pass-through. It completes immediately and, if it has any downstream connections, the downstream items are initiated.

There are several subcategories of triggers:

- Periodic – The trigger is started periodically, based on a cron schedule.
- Polling – The trigger is started if SCM polling detects that commits have been pushed to a specified repository and branch since the last poll. Polling is based on a cron schedule. The repository URL and branch name are set as downstream parameter values with user-configurable names.
- Commit – The trigger is started if a commit is pushed to a specified local project repository and a specified set of branches. The repository URL and branch name are set as downstream parameter values with user-configurable names.
- Manual – The trigger is started manually. Start, the default manual trigger, is present in every pipeline. If Start is the only manual trigger, the pipeline starts there and executes the next downstream job(s). If a pipeline includes a manual trigger job, it can be started in the UI and execute its next downstream job, bypassing Start. If the pipeline has multiple trigger jobs, the user needs to choose which of them to initiate the pipeline run with.

### Periodic Triggers

Here's an example that shows how to use a periodic trigger:

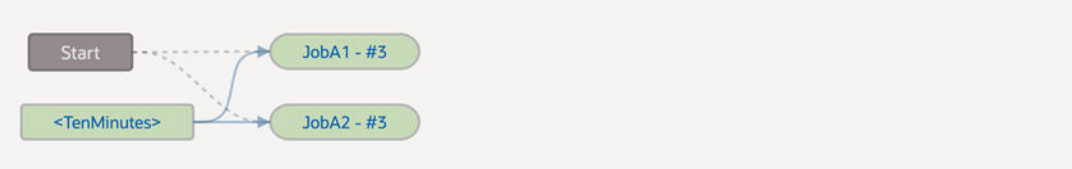
```
pipeline:
  name: PeriodicPipeline
  description: "Trigger defined in periodic, used in start"
  auto-start:
    triggers-only: true
  allow-external-builds: false
  triggers:
```

```
- periodic:
    name: MidnightUTC
    cron-pattern: "0 0 * * *"
start:
- trigger:
     - <MidnightUTC>
- JobA3
```

Notice that the periodic trigger is defined with a name and a cron pattern. The reference to the trigger action item is enclosed in angle brackets, <MidnightUTC> in this case, to differentiate it from a job, such as JobA3.

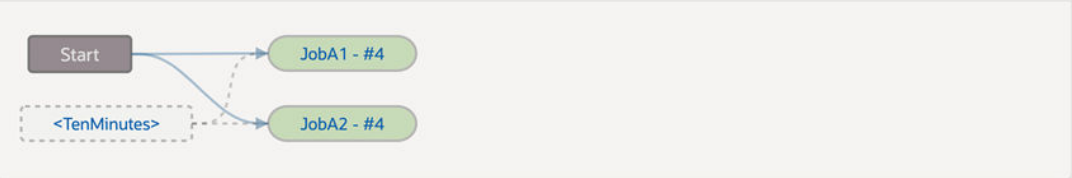
Let's look at how the UI graphically represents things:

- In the **Pipelines** tab on the **Builds** page, trigger action items are represented as squared off blocks, like the Start item or the <TenMinutes> item. The pipeline in this diagram was started by the action item with the periodic trigger <TenMinutes>. This trigger runs the pipeline every ten minutes.

  Notice that a solid line goes from it to JobA1 and JobA2, but a dotted line goes from Start through its trigger to its downstream jobs. This is so, because the pipeline wasn't initiated from Start. The trigger item and the executed jobs are shaded, indicating the pipeline's execution path.

- You could manually start the pipeline too, as this diagram shows.

  In this case, the execution passes from Start, the default trigger, to JobA1 and then to JobA2. The graphic representation shows the execution path with solid lines. The <TenMinutes> periodic trigger job, outlined with dotted lines, isn't shaded because it wasn't executed. The dotted lines from it to its downstream jobs further indicate an execution path not taken.

### Polling Triggers

This polling pipeline only runs at midnight UTC, as specified by the cron pattern. Additional parameters can be used too. See What Is the Format for a YAML Pipeline Configuration?.

```
pipeline:
  name: PollingPipeline
  auto-start: false
  triggers:
  - poll:
      name: Poller
      cron-pattern: "0 0 * * *"
      url: <git-repo-url>
      branch: main
```

```
  start:
  - <Poller>
  - A
```

When the pipeline is started manually, the execution flow goes through the trigger action item to job A. In the Pipeline Designer, the trigger has no hue and has a dotted line border. There are dotted lines from Start to the trigger item to job A. When the polling mechanism detects a change, the pipeline is started by the trigger. This is shown with a dotted line from Start to the trigger item, the trigger item has a dark hue, and there is a solid line from the trigger item to job A.

#### Commit Triggers

A commit trigger automatically runs when a commit happens.

```
pipeline:
  name: CommitPipeline
  auto-start: false
  triggers:
  - commit
      name: OnCommit
      url: <git-repo-url>
      branch: main
  start:
  - <OnCommit>
  - A
```

Additional parameters can be used too. See What Is the Format for a YAML Pipeline Configuration?.

## Control How a Pipeline Is Automatically Started

The auto-start option automatically starts a pipeline if any job in pipeline is run. The default setting is "true". If the option is set to "false", the pipeline will start only if it is manually started or if a trigger action item is activated. Starting pipelines in the middle can be problematic, since preceding or parallel steps in the pipeline could set up conditions for follow-on steps. This behavior can be controlled using the auto-start option.

The first pipeline job that follows Start is a trigger only if it is the only job triggered by Start. Either the entire pipeline or only parts of the pipeline that have defined trigger jobs will automatically start.

## View Logs for Pipelines Started by a Trigger Job

A triggered pipeline starts when the trigger job begins executing. For a pipeline that contains two jobs, Job A and Job B, where the Job A triggers the pipeline, the pipeline starts when the trigger job starts. The pipeline log reflects this:

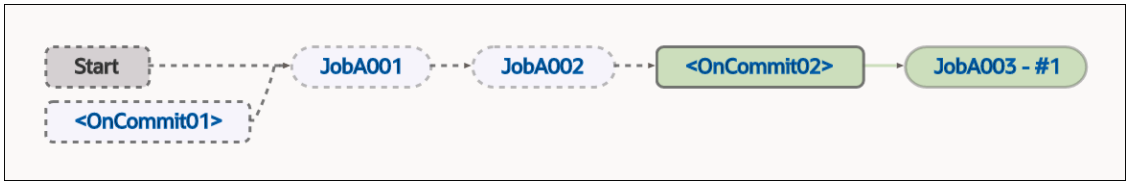
```
[2023-01-22 15:21:15] Started by job Job A build #1 (BUILDING} which was
initiated by maxyz
[2023-01-22 15:21:15] Job [Job A] finished build #1 (SUCCESSFUL}
[2023-01-22 15:21:15] Job [Job B] started build
[2023-01-22 15:21:40] Job [Job B] finished build #1 (SUCCESSFUL}
[2023-01-22 15:22:05] Pipeline end. Run time 53 sec
```

## Examine Pipeline Logs with Commit or SCM Polling Trigger Action Items

Poll logs for pipelines are very similar in format to logs for jobs. By examining these pipeline logs, you can determine how pipelines were triggered and see exactly what was executed during the run. SCM polling and commit logs show action items, but not embedded triggers.

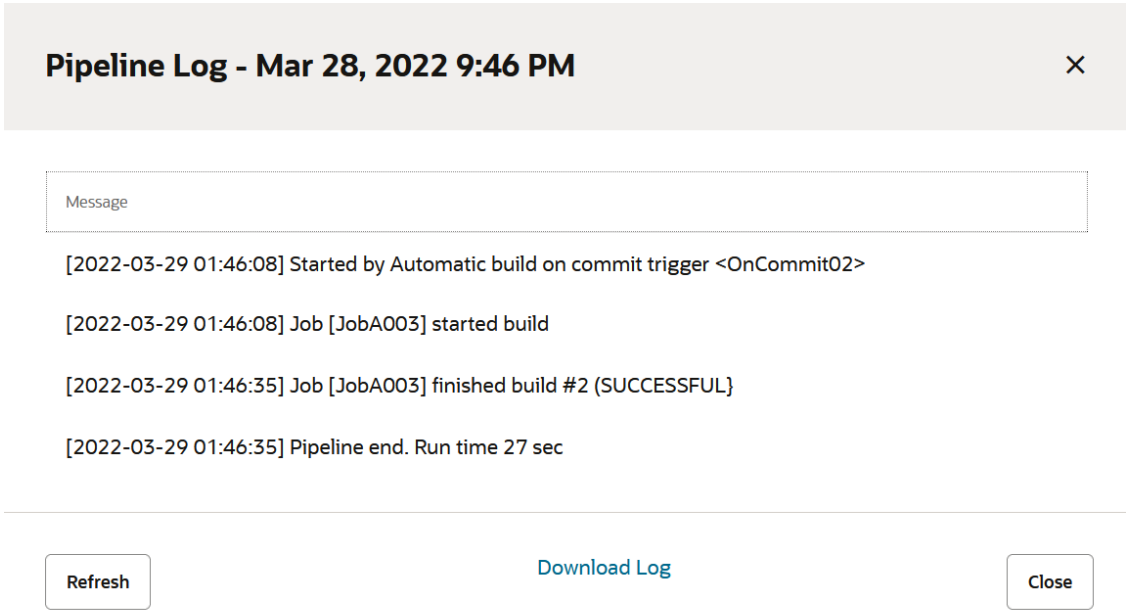
### Commit Pipeline Log

Here's a pipeline diagram with two triggers, <OnCommit01> and <OnCommit02>.




Notice that the first trigger (<OnCommit01>) is shown in gray and its downstream jobs are too? That's because they weren't executed.

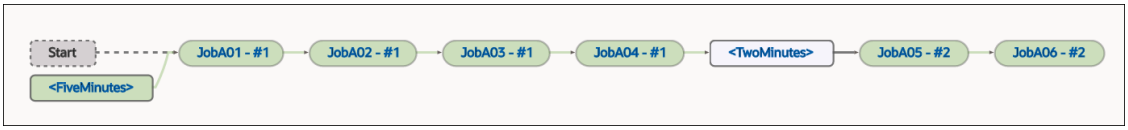
The first line in the log reveals how the pipeline was started - automatically by the commit trigger <OnCommit02>:




A commit automatically triggered the pipeline, which started build (build #2) that executed JobA003. The pipeline run was successful and took 27 seconds to run. The two jobs upstream from the second trigger (JobA001 and JobA002) were not executed because their trigger (<OnCommit01>) didn't initiate the pipeline run. Had that happened, the second trigger would have been a pass-through and the build would have executed three jobs in succession.

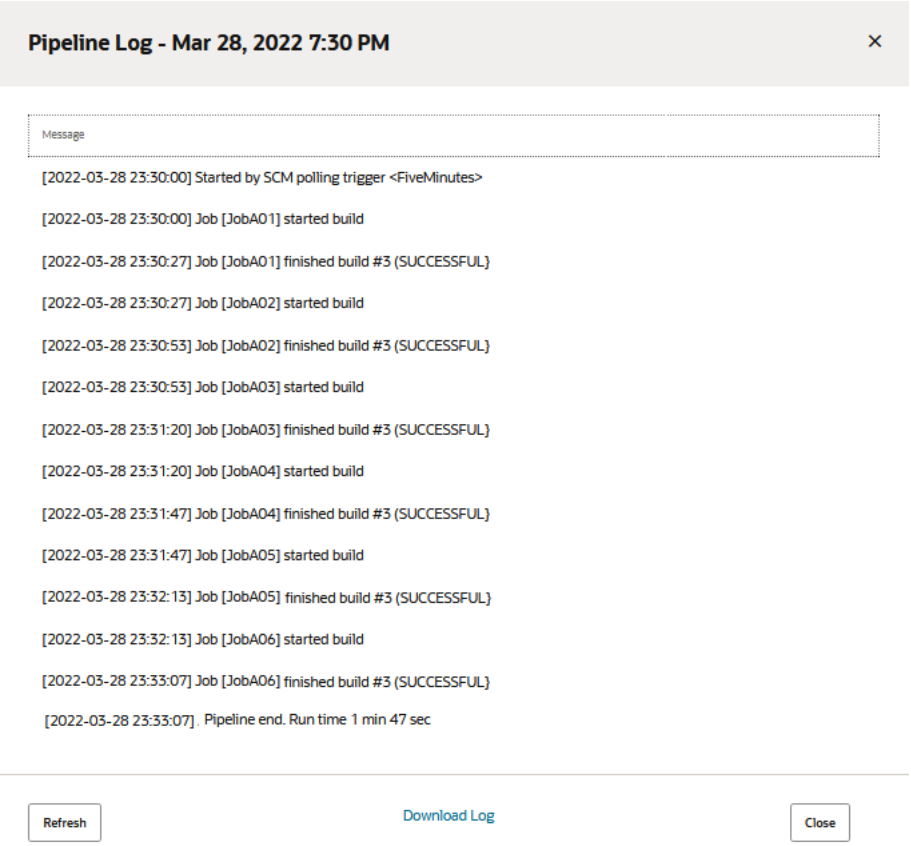
### SCM Poll Pipeline Log

Here's a pipeline diagram that has two triggers, <FiveMinutes> and <TwoMinutes>.




Notice that the pipeline's second trigger (<TwoMinutes>) is shaded with a lighter color. It wasn't executed, so it didn't have any effect on the pipeline run, even though it was in its path. In this case, the execution flow passed through it and continued executing downstream jobs.

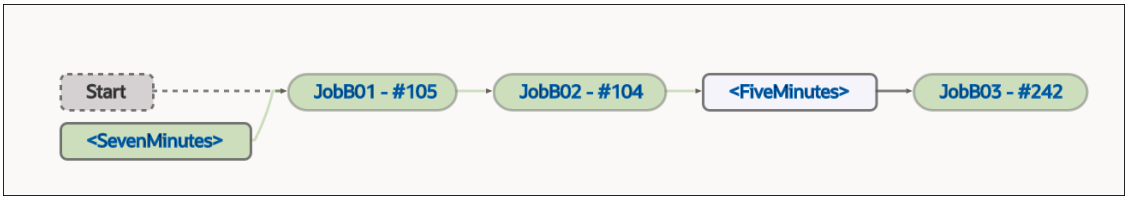
The first line in the log reveals how the pipeline was started - by the SCM polling trigger <FiveMinutes>:




We can see that the trigger started build #3 with JobA01 then, upon successful completion, started job JobA02. Successive jobs JobA03 through JobA06 ran after the previous job successfully completed. After the last job (JobA06) in build #3 finished, the pipeline ended. We can also see how long it took the pipeline to run and whether all the jobs in the build were successful or not. In this case, they were.

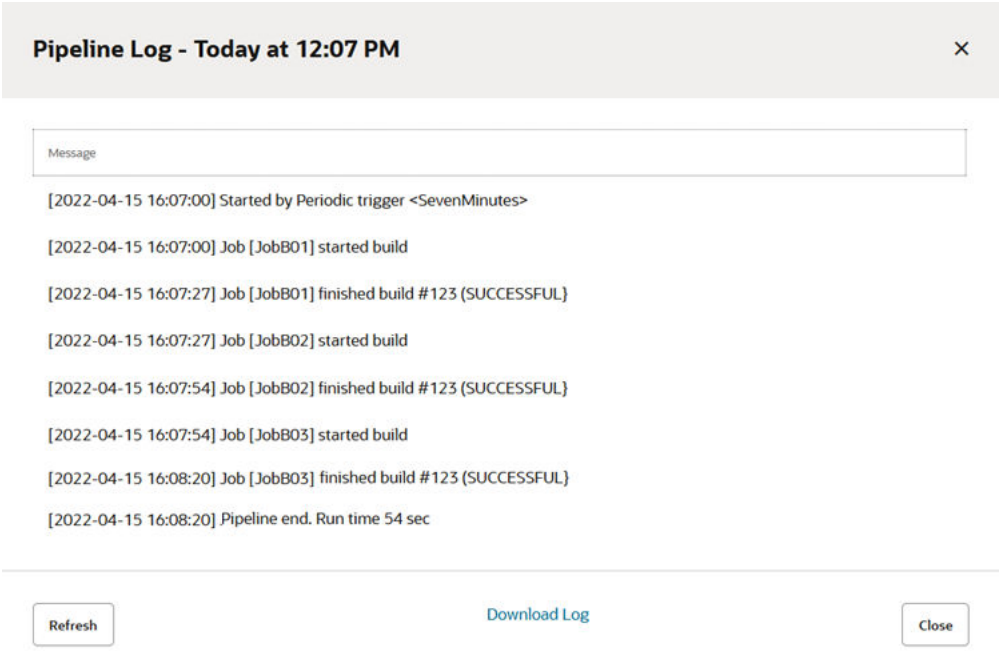
### Periodic Pipeline Log

As a bonus, here's a pipeline diagram that shows a build (build #123) that was started by a the first of two periodic triggers (<SevenMinutes>, not the second trigger (<FiveMinutes>).



As in the previous pipeline diagram, the second trigger, which wasn't executed, is shown in a lighter shade. Start, the default trigger, is shown in gray since it wasn't executed either. Had the pipeline been manually started, the first trigger would have been ignored but the job execution would've been the same as it was in the current run.

Here's the log:




We can see that all three jobs were successfully executed and the pipeline run lasted less than a minute. There was no mention of the pass-through trigger (<FiveMinutes>) that wasn't executed.

8

# Deploy and Manage Your Applications

By using a **Deploy** build step, you can deploy an application to an instance called a deployment target. You deploy extensions to Oracle Cloud Applications instances and visual applications to Visual Builder instances. You can enable continuous delivery, a method for automatically deploying a build artifact to the target service, by enabling the Auto Start functionality in a pipeline that contains a deployment build step.

> **Note:**
>
> Before you can publish an extension or visual application from the Designer, the pipeline's deployment job must be configured to deploy to the target Oracle Cloud Applications or Visual Builder instance.
>
> - If your deployment job uses Basic authentication, credentials of a user who can connect and deploy to the target instance must be provided. If your project owner hasn't provided these credentials, you'll be prompted for them when you click Publish for the first time. If valid, the credentials will be permanently saved in the deployment job. If you don't know the credentials, you'll need to talk to your project owner or an administrator. Your administrator may choose to enter the credentials directly in the build job instead, so you aren't prompted for them when you try to deploy your extension or visual app.
> - If the deployment job uses OAuth, authorization must be provided for OAuth tokens to be created with the credentials of a user who has permissions to connect and deploy to the target instance. You won't be able to publish without this authorization. It is recommended that you authorize your connection during initial job configuration—though you'll be prompted to authorize missing or expired OAuth tokens when you click Publish.
>
> See Create a Deployment Build Job for extensions or Configure the Deployment Job for visual applications.

Application lifecycle operations for extensions and visual applications can be managed by using the options under their respective build step menus:

- Extensions can be packaged and deployed extensions can be deleted.
- Visual applications can be packaged, data can be imported from and exported to applications, deployed applications can be locked and unlocked, changes can be rolled back to a previous version, and deployed applications can be undeployed.
  See Lock, Unlock, or Roll Back a Deployed Visual Application for information about lock, unlock, and rollback operations.

  See Configure a Build Job to Import or Export Data from a Visual Application for information about importing data from and exporting data to visual applications.

For visual applications, some of these operations can also be managed from the activity menu in the Environments page's **Deployments** tab:

- You can export data from, import data to, and undeploy a visual application that's deployed to your current identity domain's Visual Builder instance.

However, if your visual application is in another identity domain, you'll need to create and use the **Visual Application : Export Data**, **Visual Application : Import Data**, or **Visual Application : Undeploy** build steps to undeploy to perform these operations.

For extensions, some of these operations can be managed on the Manage Extension Lifecycle page:

- You can deploy an already published extension to a new Oracle Cloud Applications instance. See Manage Your Published Extensions in *Extending Oracle Cloud Applications with Visual Builder Studio*.
- You can delete an extension when no longer needed.

# Package, Deploy, and Manage Extensions

From the **Steps** tab on the job's configuration page, you can create an Application Extension Package job that packages an extension build artifact and an Application Extension Deploy job that deploys the build artifact to an Oracle Cloud Applications development instance, production instance, or any other instance. You can then add these jobs to a pipeline and run them in sequence.

Deployed extensions can be viewed from the Manage Extension Lifecycle page and can be deleted manually from there too, if needed. See Manage Your Published Extensions in *Extending Oracle Cloud Applications with Visual Builder Studio*.

## Deploy an Extension to a Oracle Cloud Applications Development Instance

When a project is created for you through the **Edit Page in Visual Builder Studio** workflow in an Oracle Cloud Application, several artifacts are created for you:

- A Git repository that contains the extension's source code
- A Development environment that points to the development instance where your base Oracle Cloud Application is running
- Default build jobs that package and deploy the extension's artifact to your development environment, along with a pipeline to run the build sequence. While these build jobs and pipelines are automatically created for you, whether they are enabled or disabled depends on your extension's **CI/CD Pipeline** setting.
- A private workspace in which you can edit the extension in the Designer.

## Deploy an Extension to an Oracle Cloud Applications Production Instance

If you want to deploy an extension to your Oracle Cloud Applications production instance, or any other instance, you'll need to set up separate packaging and deployment jobs for each—Visual Builder Studio does not create them for you. They're very similar to the default build jobs that are created through the **Edit Page in Visual Builder Studio** workflow in an Oracle Cloud Application. For these jobs, however, you'll also need to create a pipeline to execute the build steps in sequence.

See Create and Configure Build Jobs for information about setting up these jobs for a production environment.

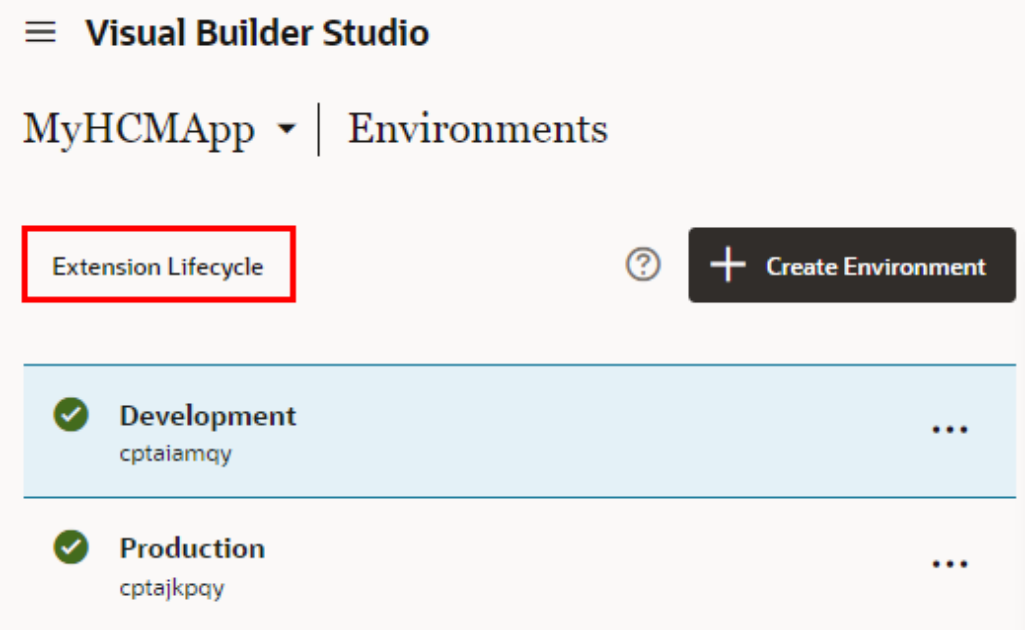
See Create and Configure a Pipeline for more information about setting up a pipeline.

> **Note:**
>
> If working with Oracle Cloud Applications instances where the authentication method is OAuth, you can bypass the CI/CD pipeline and instead use the Manage Extension Lifecycle page to deploy an extension to additional instances. See Manage Your Published Extensions in *Extending Oracle Cloud Applications with Visual Builder Studio*.

## View a Deployed Extension

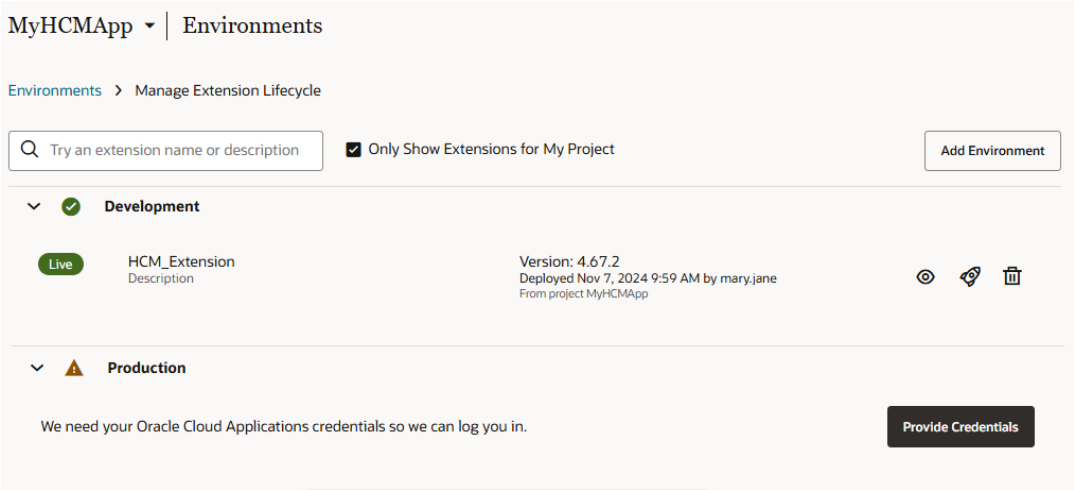
After the deployment job runs successfully, you can view the deployed extension on the Manage Extension Lifecycle page. The Manage Extension Lifecycle page shows all extensions deployed across your environments.

1. In the left navigator, click **Environments**, then either:
   - Click **Extension Lifecycle**.




   - Or, select the Oracle Cloud Applications environment, click the **Deployments** tab, then click the **Extension Lifecycle** link.

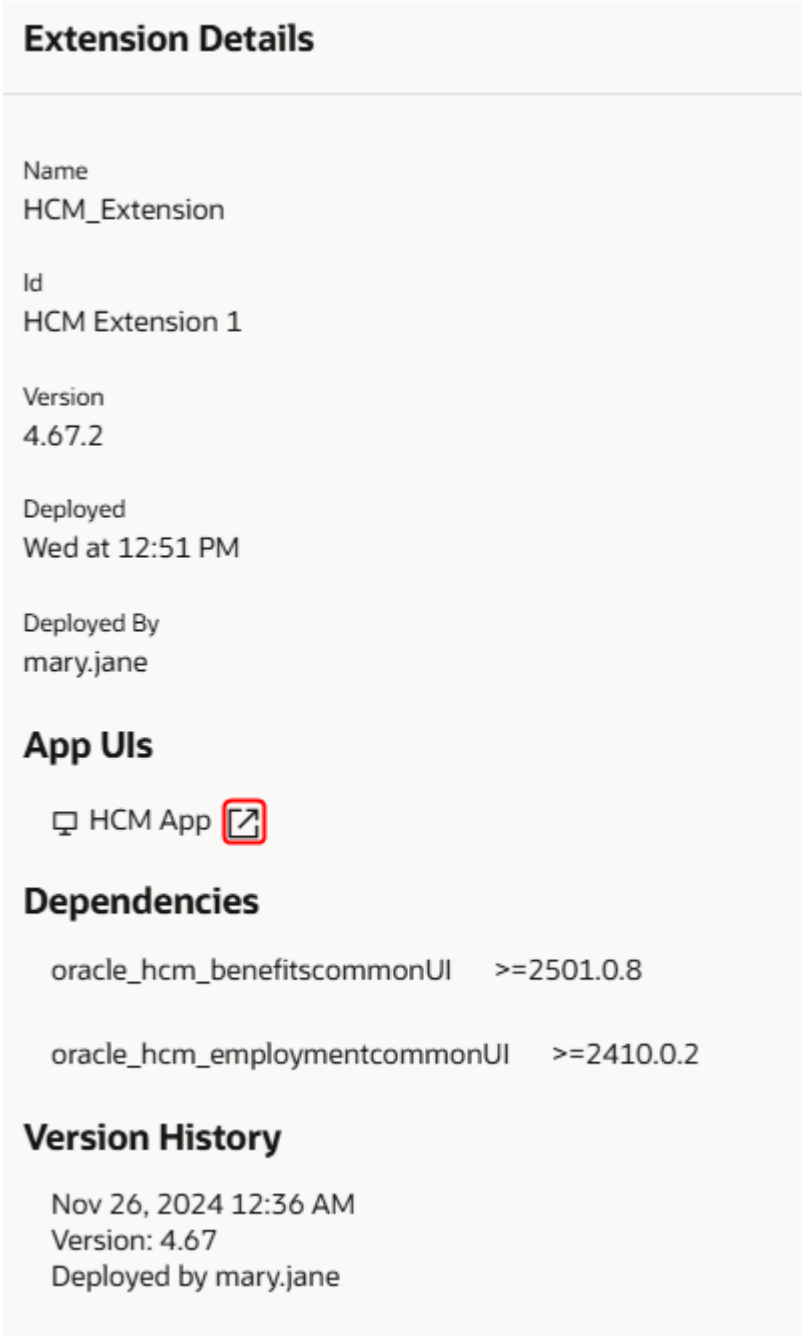
   You should now see the Manage Extension Lifecycle page.




2. Expand each environment to view a list of the extensions already deployed. Each entry includes the extension name plus additional details, such as extension version, deployment date, who deployed the extension, and project name.

   - Select to view even more details, such as the extension's extension ID, App UIs, dependencies, and version history.

     > **Tip:**
     >
     > You can launch an extension's App UI by clicking next to each App UI.




- Use the **Only Show Extensions for My Project** checkbox to toggle between viewing extensions deployed only from the current project, or from all projects.

In addition to *viewing* your deployed extensions, you can also use the Manage Extension Lifecycle page to centrally *manage* them, as well. For example, you can deploy a published extension to a new Oracle Cloud Applications instance, and delete them when no longer needed. To learn more about what you can do on this page, see Manage Your Published Extensions in *Extending Oracle Cloud Applications with Visual Builder Studio*.

## Delete an Extension

Delete unnecessary extensions to restore your application to its original state and clean up your environment.

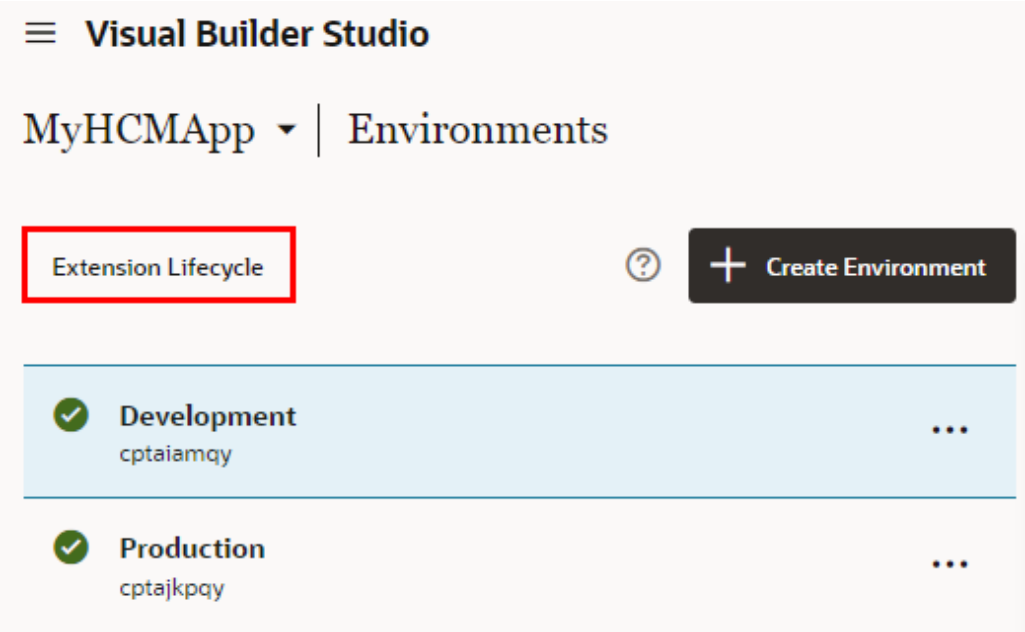
You have two options to delete an extension:

- The best practice is to use the Manage Lifecycle Extension page to delete an extension. When you delete an extension from this page, all versions of that extension are deleted.
- If you need to delete a specific extension *version*, you must configure a build job to delete it.

## Delete an Extension Manually

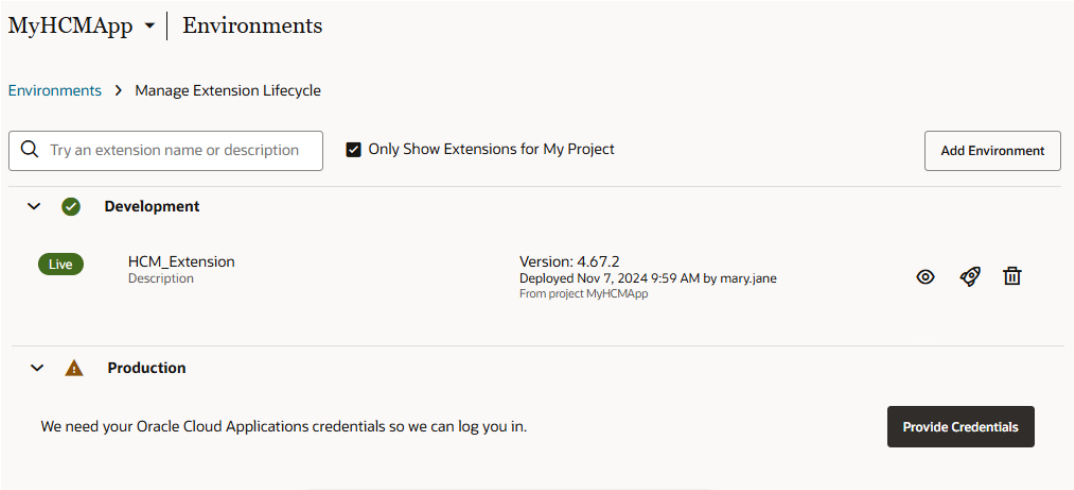
Use the Manage Extension Lifecycle page to delete extensions deployed from any project and to any Oracle Cloud Applications instance.

1. Access the Manage Extension Lifecycle page:
   - From the VB Studio left navigator, click **Environments**, then click **Extension Lifecycle**. You won't see this option if you haven't yet added an Oracle Cloud Applications instance to your environment.




   - Or, from the menu in the Designer's header, click **Extension Lifecycle**.

   You should now see the Manage Extension Lifecycle page, which lists all the environments that are available.




You can work with an environment on this page as long as the authentication method for its Oracle Cloud Applications instance is OAuth. If there's an environment listed whose authentication method isn't OAuth, then you'll first need to add a new environment with the same base URL, but using OAuth as the authentication method, before you can work with it.

> **Note:**
>
> If you see a **Provide Credentials** button next to an environment, you must first sign in before you can delete any extensions deployed to that instance.

Expand each environment to view a list of the extensions already deployed. Use the **Only Show Extensions for My Project** checkbox to toggle between viewing extensions deployed from the current project only, or from all projects.

Each entry includes the extension name plus additional details, such as extension version, deployment date, who deployed the extension, and project name.

2. To delete an extension from an Oracle Cloud Applications instance, find the extension that you want to delete and click .

   All extension versions are deleted. If the extension was originally deployed via a CI/CD pipeline, the extension's build jobs and their artifacts remain; this is helpful in case you ever need to redeploy an earlier extension version.

> **Tip:**
>
> Before deleting an extension from your PROD instance, delete the extension from the DEV instance (or TEST instance, if applicable) and make sure there aren't any adverse effects.

# Configure a Job to Manage a Deployed Extension

VB Studio provides you with build steps that you can add to a job to perform lifecycle operations on extensions and App UIs. VB Studio provides these build options as Application

Extension build steps that you can configure in a packaging job or in separate build jobs that you can add to a CI/CD pipeline.

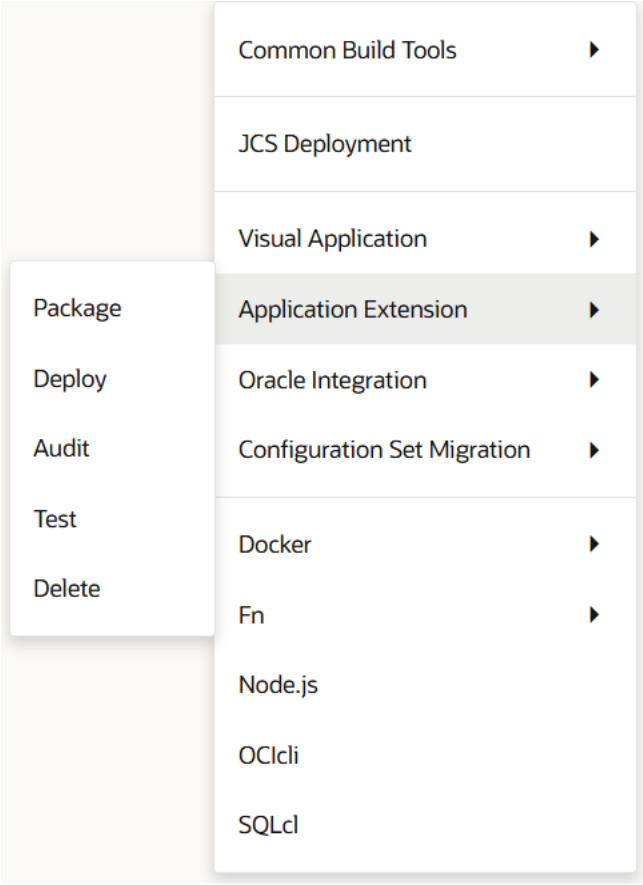
These lifecycle operations include:

- Auditing and testing extensions before deploying them
  The Audit, Test, and Package build steps can be in separate jobs or, for simplicity, you can add the **Test** and **Audit** (in whatever order you want) steps before the **Package** step in the packaging job, as we show next.
- Deleting extensions when they're no longer needed
  Using a Delete build step in a job actually deletes an extension *version*. On the other hand, deleting an extension from the Manage Lifecycle Extension page deletes all versions of an extension, and is considered the recommended best practice.

> **Tip:**
>
> If you create a separate job for each task, after you create and configure the lifecycle management build steps, you may want to add the jobs, in some combination, to the pipeline you created for the packaging and deployment steps for that testing or production instance. By integrating these build steps in your deployment process, you'll ensure a more robust and error-free process when upgrades are done through deployment.

To configure these options in an existing packaging job:

1. In the left navigator, click **Builds** .
2. In the Jobs tab, select the package job and click **Configure**.
   An extension created through the **Edit Page in Visual Builder Studio** workflow in an Oracle Cloud Application includes a default `Application-Extension-Package` build job that packages the application extension's sources—but whether the job is enabled or disabled depends on your extension's **CI/CD Pipeline** setting.
3. On the Job Configuration page, click **Steps**.
4. Click **Add Step**, select **Application Extension** and select the option you want to add to the job:




These options automate CI/CD tasks for you. Each option has its own set of parameters. Some operations should follow a particular order. For example, the **Audit** and **Test** steps should be performed before the **Package** step, and all three steps must precede the **Deploy** step.

See the following for more information about each option:

- See Configure a Job to Audit and Test Your Extension for information about **Audit** and **Test**.
- See Configure a Job to Delete an Extension for information about **Delete**.

5. When you're done, click **Save**.

## Configure a Job to Audit and Test Your Extension

VB Studio provides capabilities for auditing your extensions and running action chain tests you've defined in your Oracle Cloud Application extension. For your convenience, VB Studio provides these operations in Build steps, so you can define all the necessary arguments and option overrides in one place for a build job or to include in a pipeline that simplifies automating the CI/CD lifecycle.

See Debug and Audit Your Code and Test Action Chains for more information about auditing and testing extensions.

### Create a Build Step to Audit Your Extension

1. From the **Git** tab on the Job Configuration page, select **Git** from the **Add Git** dropdown and then select the repository that was created for the extension in **Repository**.

2. In the **Parameters** tab, select **String Parameter** from the **Add Parameter** dropdown list.
   Enter `OUTPUT_FILE` in **Name**.
   The default value is `auditoutput.json`.

   This parameter is used to override the default Grunt options as well as in the artifact archival.
3. In the **Steps** tab, select **Add Step**, **Application Extension**, and **Audit**.
   In the Application Extension Auditing panel:
   a. Enter the extension's identifier in the **Extension ID** field and the extension's version in the **Extension Version** field.
      You can find these values by viewing the extension's details on the Manage Extension Lifecycle page, accessible from the Environments page.
   b. In **Options**, enter `auditoutputfile=$OUTPUT_FILE`, using the parameter you defined in step 2.
4. In the **After Build** tab, select **Artifact Archiver** from the **Add After Build Action** dropdown list.
5. In the Configure Post Build Actions panel, in **Artifacts from files**, enter `$OUTPUT_FILE` in the **Files to archive** field.
6. Click **Save**.

### Create a Build Step to Test Action Chains in Your Extension

1. From the **Git** tab on the Job Configuration page, select **Git** from the **Add Git** dropdown and then select the repository that was created for the extension in **Repository**.
2. In the **Parameters** tab, select **String Parameter** from the **Add Parameter** dropdown list.
   Enter `BUILD_DIR` in **Name**.
   The default value is `build`.

   This parameter is used when the build system executes the Grunt task behind the scenes in the step as well as in the artifact archival.
3. In the **Steps** tab, select **Add Step**, **Visual Applications**, and **Test**.
   The Application Extension Testing dialog displays.
   a. In **Karma Browser**, if you select **FirefoxHeadless**, it requires a Build Executor template that contains the Firefox software package.
      If you select **ChromeHeadless** instead, it requires a custom Docker image with Chrome installed, so, you need to create that custom Docker image and then create a Build Executor template to use from that.
   b. For **Karma Log Level**, select **Info**, **Debug**, **Warn**, **Error**, or **Disable**.
      The different log levels will be generated from the tests. **Debug** is the default level.
   c. In **Mocha Timeout**, enter a number between 0 and 600,000 milliseconds.
4. In the **After Build** tab, select **Artifact Archiver** from the **Add After Build Action** dropdown list.
   The Configure Post Build Actions dialog displays.
5. In the Configure Post Build Actions dialog, in **Artifacts from files**, enter `$BUILD_DIR/build/**/*` in the **Files to archive** field.
6. Click **Save**.

# Configure a Job to Delete an Extension

If you want to delete a specific extension *version* that's deployed to an Oracle Cloud Applications instance, you must configure a build job and run it.

You can also use the Manage Extension Lifecycle page to delete extensions, but keep in mind that deleting an extension from that page deletes the *entire* extension.

Note that there can be only one *active* extension version at a time. The recommended best practice is to avoid deleting the active extension version because, if you delete it, the extension will become inactive as if it wasn't installed at all. You can, however, delete inactive versions without impacting the currently active deployed extension.

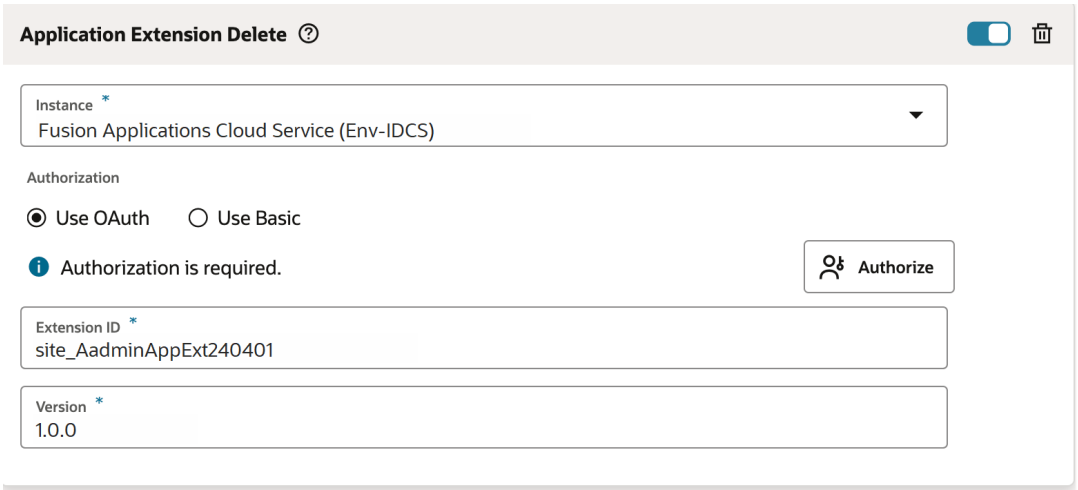
Before you configure and run the job, delete the extension from the DEV instance (or TEST instance, if applicable) and make sure there aren't any adverse effects. For example, let's assume you have an attribute that's hidden in both the extension's business object and the user interface. After you delete the extension, the user interface shows the attribute that's still hidden in the business object. This may cause an error.

To configure the job, make sure you have valid administrator credentials for the Oracle Cloud Applications instance where the extension is deployed.

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select **System Default OL7 for Visual Builder**.
6. Click **Create**.

   The Job Configuration page opens.
7. Click the **Steps** tab.
8. From **Add Step**, select **Application Extension**, and then select **Delete**.

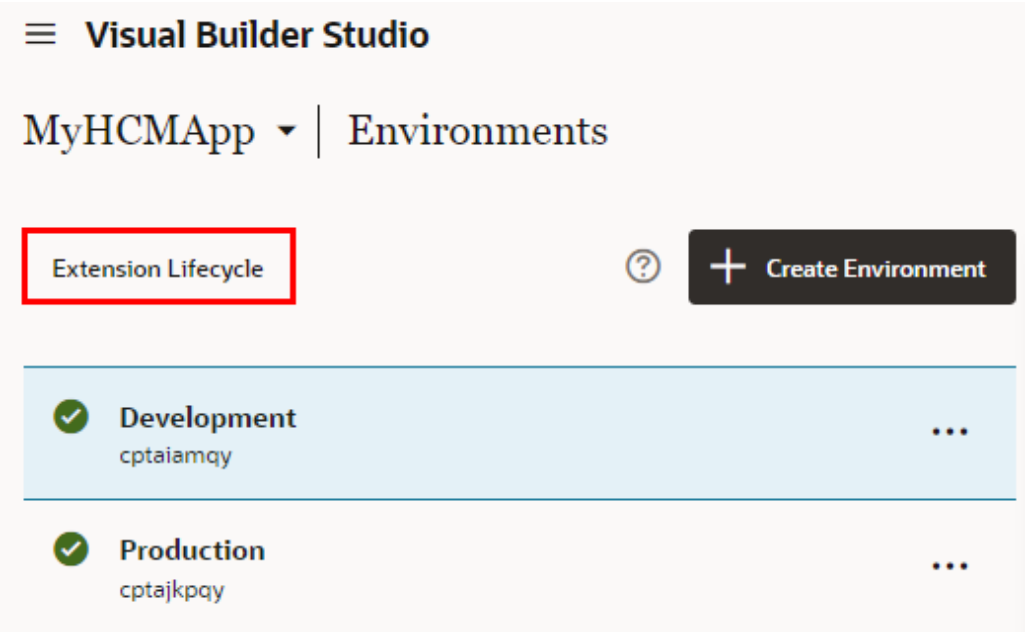
   This image shows the Application Extension Delete build job page that's partially filled in.



9. In **Instance**, select the Oracle Cloud Applications instance where the application is deployed.
10. In the Authorization section, select **Use OAuth** or **Use Basic** to specify the type of authorization used to run this build step.
    - If you selected **Use OAuth**, the `Authorization is required` message displayed before authorization indicates that this build step needs to be authorized, either by clicking the **Authorize** button or by running the build manually and entering the credentials when you are prompted for them at that point. After authorization, the `Authorization has been provided` message and the **Renew Authorization** button are displayed.
    - If you selected **Use Basic**, in **Username** and **Password**, enter the credentials of an IDCS user who is not only an Oracle Cloud Applications user, but one who can access the Oracle Cloud Application's instance and undeploy from it. These credentials must be those of a local user, not a federated identity, and must not require multi-factor authentication.
11. In **Extension ID** and **Version**, enter the extension ID and version of the extension you want to delete.

    You can find these items in the extension's details on the Manage Extension Lifecycle page.

    a. In the left navigator, click **Environments**, then click **Extension Lifecycle**.




    b. Find the extension you want to delete, and select to obtain the extension ID and version.
12. Click **Save**.
13. To run a build, click **Build Now**.

# Package, Deploy, and Manage Visual Applications

From the **Steps** tab on the job's configuration page, you can create a Visual Application Package job that packages a visual application build artifact and a Deploy job that deploys the build artifact to a Visual Builder development instance, production instance, or any other instance. You can then add these jobs to a pipeline and run them in sequence.

You can deploy a visual application to a standalone Visual Builder instance or to a Visual Builder instance that's part of Oracle Integration.

It’s important to keep these things in mind before you deploy a visual application to a Visual Builder instance:

- The Visual Builder instance must be version 19.4.3.1, or later.
- To ensure that business objects work properly, Visual Builder administrator must manually add the VB Studio hostname to the list of domains that are allowed access for **each** Visual Builder instance. See Allow Other Domains Access to Services in *Administering Oracle Visual Builder in Oracle Integration 3*.

Deployed visual applications can be viewed from the **Deployments** tab on the Environments page and can be undeployed manually from there too, if they are deployed to a Visual Builder instance that is in the same identity domain as VB Studio. If a visual application is deployed to a Visual Builder instance that is in a different identity domain than VB Studio, you'll have to create and use a Visual Application build step to undeploy the deployed visual application.

## Deploy a Visual Application to a Development Visual Builder Instance

When you create a project using the Visual Application template, several artifacts are created for you

- A Git repository that contains the visual application's source code.
- A Development environment that points to the Visual Builder development instance.
- Default build jobs that package and deploy the visual application's artifact to the Visual Builder development instance.
- A pipeline to run the build sequence.
- Optionally, a private workspace to edit the visual application in the VB Studio Designer.

You'll need to do some configuration for the build steps before you can use them to deploy the application's build artifact to the Development environment. See Configure the Packaging Job and Configure the Deployment Job for more information.

## Deploy a Visual Application to a Test or Production Visual Builder Instance

If you want to deploy visual applications to your Visual Builder production instance, or any other instance, you'll need to set up separate packaging and deployment jobs for each. They're very similar to the default build jobs that are created from the Visual Application template. For these jobs, however, you'll also need to create a pipeline on your own to execute the build steps in sequence.

See Create and Configure Production Build Jobs for information about setting up these jobs for a production environment.

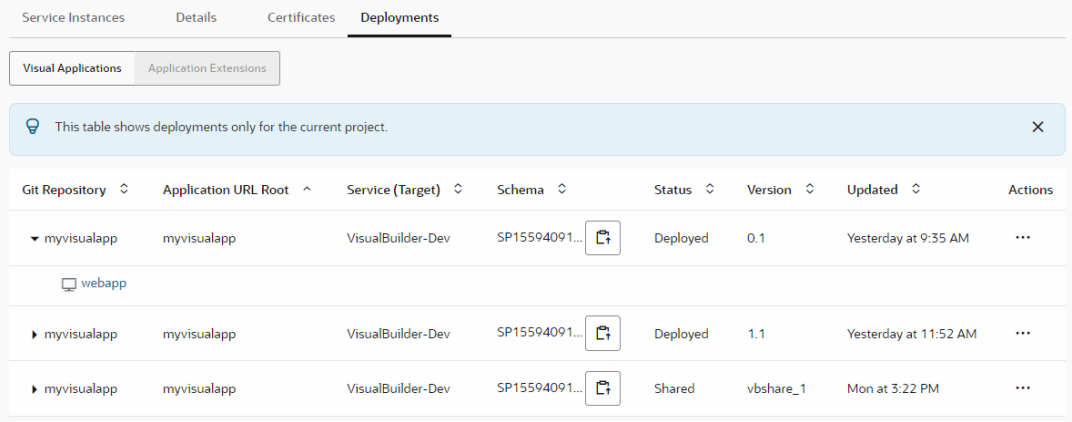
# View a Deployed Visual Application

After the deployment job runs successfully, you can view the deployed application in the **Deployments** tab of the **Environments** page.

1. In the left navigator, click **Environments**.
2. Select the Visual Builder environment.
3. Click the **Deployments** tab.
4. Click the **Visual Applications** toggle button.
5. If the Visual Builder instance is from a different identity domain, provide its access credentials.
6. Expand the app's name to see the deployed app's link.

   The **Deployments** tab displays the applications you've deployed from the current project. It doesn't show applications deployed by other users of the project, or applications deployed from other projects.

   For example:




> **Tip:**
>
> The Schema column lists the database schemas that store the deployed app's business objects. Hover over the text to see the full schema name in a tooltip or, to copy it, select the **Copy to Clipboard** icon, then paste it in a text editor of your choice.
> See View Database Schemas Used During an App's Lifecycle for more information.

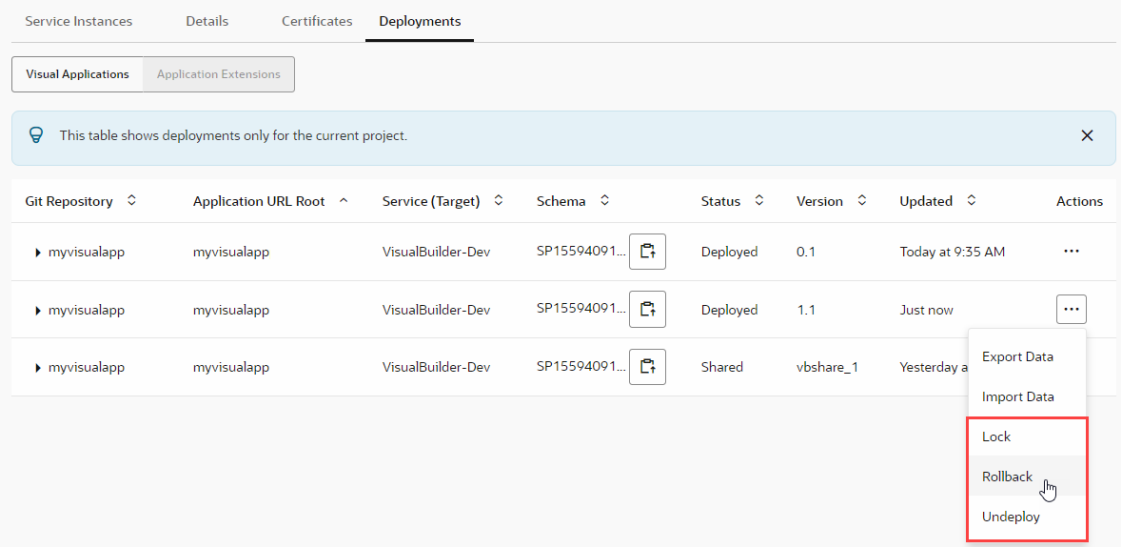
# Lock, Unlock, or Roll Back a Deployed Visual Application

You can lock and unlock deployed visual applications, and the web applications that they contain, or roll back a deployed visual application. You lock and unlock a visual application when you have maintenance tasks to complete and don't want users to access the web application in the deployed visual application during the maintenance period.

You can manage these visual applications lifecycle operations (lock, unlock, roll back) manually through the Environments page's **Deployments** tab or you can manage them with Visual Application build steps (see Configure a Job to Lock, Unlock, or Roll Back a Deployed Visual Application).

You can use the **Rollback** menu option without including the application version in the URL when you've deployed your visual application more than once. That is, `live` appears in the application URL, instead of the application version. For example, if you've deployed two versions of your visual application to `https://host/app-name/live/index.html`, you can roll back to version 1 by using the **Rollback** menu option. You can only roll back one previous version.

If you deploy a visual application to a Visual Builder instance in the same identity domain as your VB Studio instance and you don't include the application version in the URL, you can perform this task from your Environments page's **Deployments** tab.




> **Note:**
>
> If you deployed your visual application to a different identity domain or if the application URL includes the version, you need to add and configure steps in a build job to perform lock, unlock, or roll back operations.

After you create and configure the lifecycle management build steps, you may want to add them, in some combination, to the pipeline you created for the packaging and deployment steps for that testing or production instance. By integrating these build steps in your deployment process, you'll ensure a more robust and error-free process when upgrades are done through deployment.

## Undeploy a Visual App

Deployed visual apps can be undeployed manually or by configuring a build step to undeploy it.

You can manually undeploy a visual app that's deployed to your development Visual Builder instance (or one that's deployed to your current identity domain's Visual Builder instance) from

the **Deployments** tab of its environment, or configure a build job to undeploy it. See Undeploy a Visual App Manually.

To undeploy a visual app that's deployed to your production Visual Builder instance (or one that's deployed to a Visual Builder instance in another identity domain), configure a build job and run it. You can't undeploy it manually. See Configure a Job to Undeploy a Visual Application.

Undeploying an application is a permanent action that completely removes application metadata and any data stored in its database. This action can't be undone. Before you undeploy, consider the impact of removing your application, especially if the version is live, because once a version has been removed from the system, it can't be recovered.

The messaging you see in the dialogs that are displayed after you initiate an undeploy action emphasize the potential for serious unintended consequences:

- For applications that are deployed with the version in the URL (staged applications), the undeploy confirmation dialog warns, "Are you sure you want to undeploy this application? You won't be able to restore it if you do."
- For applications that are deployed without the version in the URL (live applications), the undeploy confirmation dialog warns, "Are you sure you want to undeploy this application that is being used by your customers? You won't be able to restore it if you do."

By default, the **Yes, I'm sure** checkbox is unchecked, **Undeploy** is grayed out (unavailable), and **Cancel** is active and in focus. **Undeploy** will only become active (available) after you opt in by selecting the **Yes, I'm sure** checkbox.

## Undeploy a Visual App Manually

From the **Deployments** tab of its environment, you can manually undeploy a visual app that's deployed to your development Visual Builder instance (or to your current identity domain's Visual Builder instance):

1. In the left navigator, click **Environments** .
2. Select the environment where the visual application is deployed.
3. Click the **Deployments** tab.
4. Expand the application.
5. For the visual application to undeploy, click **Actions** and select **Undeploy**.
6. In the confirmation dialog box, click **Undeploy**.

# Configure a Job to Manage a Deployed Visual Application

If your visual application is deployed to a different identity domain or if your application URL includes the version, you need to add and configure steps in a build job to perform lifecycle operations. VB Studio provides these build options as **Visual Application** build steps that you can configure in a deployment job.

These lifecycle operations include:

- Importing business object data to and exporting data from visual applications
- Locking and unlocking active deployments so they can be upgraded in place
- Rolling back upgrades to a previous version
- Auditing and testing visual apps before deploying them

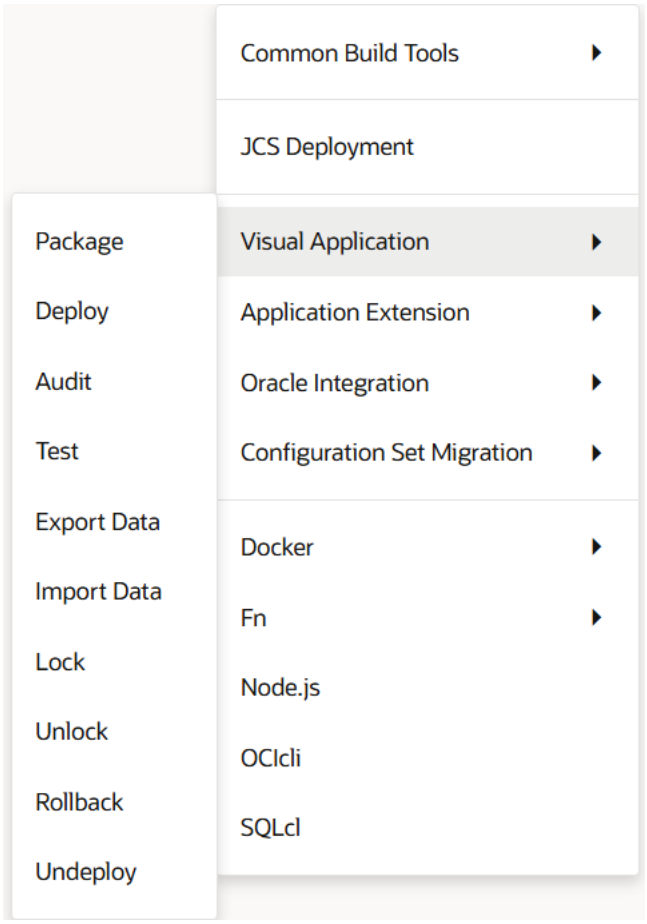
- Undeploying visual apps when they're no longer needed

The Audit, Test, and Package steps can be in separate jobs or, for simplicity, you can add the **Test** and **Audit** steps (in whatever order you want) before the **Package** step in the packaging job, as we show next.

> **Tip:**
>
> If you create a separate job for each task, after you create and configure the lifecycle management build steps, you may want to add the jobs, in some combination, to the pipeline you created for the packaging and deployment steps for that testing or production instance. By integrating these build steps in your deployment process, you'll ensure a more robust and error-free process when upgrades are done through deployment.

To configure these options in a packaging job:

1. In the left navigator, click **Builds** .
2. In the Jobs tab, select the packaging job and click **Configure**.
   A visual app includes a default `Visual-Application-Package` build job that packages the visual application's sources.
3. On the Job Configuration page, click **Steps**.
4. Click **Add Step**, select **Visual Application** and select the option you want to add to the job:




These options are Grunt-based commands that automate CI/CD tasks for you. Each option has its own set of parameters. Some operations should follow a particular order. For example, the **Audit** and **Test** steps should be performed before the **Package** step, and all three must precede the **Deploy** step.

See the following for more information about each option:

- See Configure a Job to Audit and Test Your Visual Application for information about **Audit** and **Test**.
- See Configure a Job to Import Data to or Export Data from a Visual Application for information about **Import Data** and **Export Data**.
- See Configure a Job to Lock, Unlock, or Roll Back a Deployed Visual Application for information about **Lock**, **Unlock**, and **Rollback**.
- See Configure a Job to Undeploy a Visual Application for information about **Undeploy**.

5. When you're done, click **Save**.

## Configure a Job to Audit and Test Your Visual Application

VB Studio provides NPM packages (`grunt-vb-audit`, `grunt-vb-test`) that include the `vb-audit` and `vb-test` Grunt tasks. You can use `vb-audit` to audit your visual applications and use `vb-test` to run action chain tests you've defined in your visual application. For your convenience, VB Studio provides these Grunt tasks in Build steps, that enable you to define all the necessary arguments in one place for a build job or to include in a pipeline that simplifies automating the CI/CD lifecycle.

See Audit Your Application Using the vb-audit Grunt Task and Test Action Chains Using the vb-test Grunt Task for more information about using Grunt tasks to audit application sources and test action chains in visual applications.

### Create a Build Step to Audit Your Visual Application

1. From the **Git** tab on the Job Configuration page, select **Git** from the **Add Git** dropdown and then select the repository that was created for the application in **Repository**.
2. In the **Parameters** tab, select **String Parameter** from the **Add Parameter** dropdown list.
   Enter `OUTPUT_FILE` in **Name**.
   The default value is `auditoutput.json`.

   This parameter is used to override the default Grunt options as well as in the artifact archival.
3. In the **Steps** tab, select **Add Step**, **Visual Application**, and **Audit**.
   In the Visual Application Auditing panel:
   a. In **Target Environment**, select the environment associated with the workspace that was cloned to create the app.
      The values for the Username and Password fields will be populated automatically with the user credentials associated with the environment that was selected.
   b. The **Application URL Root** will automatically be filled in, using the name of the Git repository.
   c. The **Application Version** will automatically be filled in.
   d. In **Options**, enter `auditoutputfile=$OUTPUT_FILE`, using the parameter you defined in step 2.
4. In the **After Build** tab, select **Artifact Archiver** from the **Add After Build Action** dropdown list.
5. In the Configure Post Build Actions panel, in **Artifacts from files**, enter `$OUTPUT_FILE` in the **Files to archive** field.
6. Click **Save**.

### Create a Build Step to Test Action Chains in Your Visual Application

1. From the **Git** tab on the Job Configuration page, select **Git** from the **Add Git** dropdown and then select the repository that was created for the application in **Repository**.
2. In the **Parameters** tab, select **String Parameter** from the **Add Parameter** dropdown list.
   Enter `BUILD_DIR` in **Name**.
   The default value is `build`.

   This parameter is used when the build system executes the Grunt task behind the scenes in the step as well as in the artifact archival.
3. In the **Steps** tab, select **Add Step**, **Visual Applications**, and **Test**.
   The Visual Applications Testing dialog displays.
   a. In **Target**, enter `$BUILD_DIR` that you created in the **Parameters** tab.
   b. In **Karma Browser**, if you select **FirefoxHeadless**, it requires a Build Executor template that contains the Firefox software package.
      If you select **ChromeHeadless** instead, it requires a custom Docker image with Chrome installed, so, you need to create that custom Docker image and then create a Build Executor template to use from that.
   c. For **Karma Log Level**, select **Info**, **Debug**, **Warn**, **Error**, or **Disable**.

The different log levels will be generated from the tests. **Debug** is the default level.

d. In **Mocha Timeout**, enter a number between 0 and 600,000 milliseconds.

4. In the **After Build** tab, select **Artifact Archiver** from the **Add After Build Action** dropdown list.
The Configure Post Build Actions dialog displays.
5. In the Configure Post Build Actions dialog, in **Artifacts from files**, enter `$BUILD_DIR/build/**/*` in the **Files to archive** field.
6. Click **Save**.

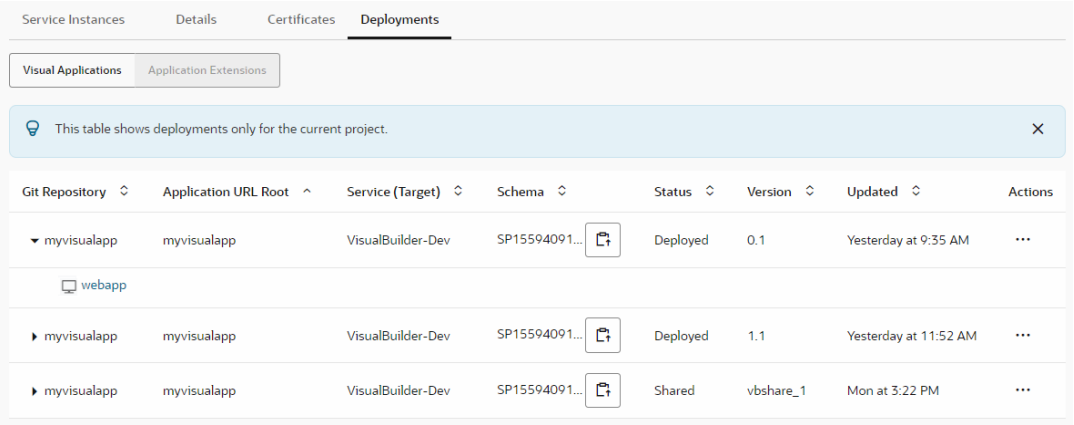
## Configure a Job to Import Data to or Export Data from a Visual Application

To import data to or export data from a visual application that has business object data with a job, you need to add the visual application **Import Data** or **Export Data** steps to a build job, along with the steps for copying or archiving the artifact that contains the data. You also need the credentials of a user who can access the Visual Builder instance where the visual application is deployed.

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The Job Configuration page displays.
7. Click the **Steps** tab.
8. From **Add Step**, select **Visual Application**, and then select **Export Data** or **Import Data**.
9. In **Instance**, select the Visual Builder instance where you want to import or export business object data.
10. In the Authorization section, select **Use OAuth** or **Use Basic** to specify the type of authorization used to run this build step.
    - If you selected **Use OAuth**, the `Authorization is required` message displayed before authorization indicates that this build step needs to be authorized, either by clicking the **Authorize** button or by running the build manually and entering the credentials when you are prompted for them at that point. After authorization, the `Authorization has been provided` message and the **Renew Authorization** button are displayed.
    - If you selected **Use Basic**, enter your Visual Builder instance credentials in **Username** and **Password**.
11. In **Application URL Root** and **Application Version**, enter the visual application's root URL and its version.

    You can find the application's root URL and its version from the **Deployments** tab of the environment where the visual application is deployed.
    Example:




12. In **Artifact**, enter the name of the ZIP file to contain the business object data to import or export. For example, enter `bodata.zip`.
13. Add additional steps to the job to complete the import or export operation.

    For example, to complete exporting data you need to add an after build action that archives the artifact (`bodata.zip`, in our example) while you typically need to copy an artifact from another job for import business object data. See Archive Artifacts and Copy Artifacts from Another Job for more information.
14. Click **Save**.

## Configure a Job to Lock, Unlock, or Roll Back a Deployed Visual Application

If you deployed your visual application to a different identity domain or if the application URL includes the version, you need to add and configure steps in a build job to perform lock, unlock, or rollback operations. After you create and configure the lifecycle management build steps (lock, unlock, rollback), you may want to add them, in some combination, to the pipeline you created for the packaging and deployment steps for that testing or production instance.

> **Tip:**
>
> If you deployed a visual application to a Visual Builder instance that's in the same identity domain as your VB Studio instance and if the URL doesn't include the application version (`live`, rather than the application version, appears in the application URL you're rolling back) , you can roll the application back to a previous version from your Environments page's **Deployments** tab, using the **Rollback** menu option when you've deployed your visual application more than once. You can only roll back one previous version at a time.

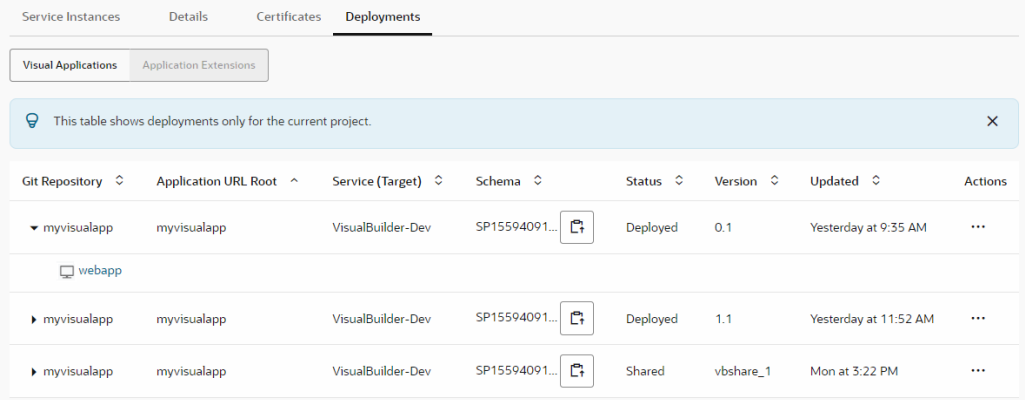
1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The Job Configuration page displays.

7. Click the **Steps** tab.
8. From **Add Step**, select **Visual Application**, and then select one of the following:
   - **Lock** prevents users from accessing the web application in the deployed visual application when you have maintenance tasks to perform.
   - **Unlock** removes the lock on the web application after the maintenance tasks have been completed.
   - **Rollback** restores the previous version of the deployed visual application.
9. Fill out the required fields in the respective dialogs that display:
   a. In **Instance**, select the Visual Builder instance where the application is deployed.
   b. In the Authorization section, select **Use OAuth** or **Use Basic** to specify the type of authorization used to run this build step.
      - If you selected **Use OAuth**, the `Authorization is required` message displayed before authorization indicates that this build step needs to be authorized, either by clicking the **Authorize** button or by running the build manually and entering the credentials when you are prompted for them at that point. After authorization, the `Authorization has been provided` message and the **Renew Authorization** button are displayed.
      - If you selected **Use Basic**, enter your Visual Builder instance credentials in **Username** and **Password**.
   c. In **Application URL Root** and **Application Version**, enter the visual application's root URL and its version.

      You can find the application's root URL and its version from the **Deployments** tab of the environment where the visual application is deployed.

      For example:




> **Note:**
>
> The version that you specify to roll back must be the latest version of the application. If the version isn't the latest, you'll see an HTTPS status 400 error when you run the rollback job. So, if the latest version of the application is version 2.3, that is the version you specify to roll back, not version 2.2 or any other earlier version.

10. Click **Save**.

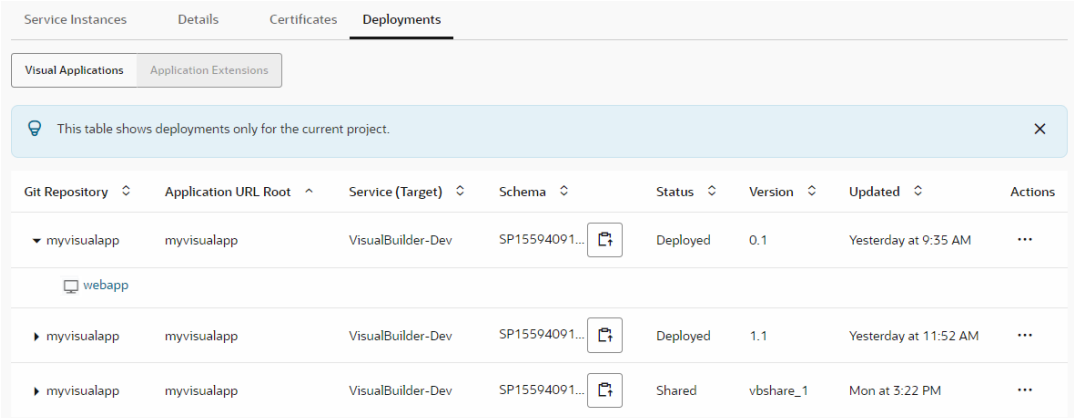
## Configure a Job to Undeploy a Visual Application

You'll need the credentials of a user who can access the Visual Builder instance where the visual application is deployed to undeploy a visual application through a build job:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the **System Default OL7 for Visual Builder** template.
6. Click **Create**.

   The Job Configuration page opens.
7. Click the **Steps** tab.
8. From **Add Step**, select **Visual Application**, and then select **Undeploy**.
9. In **Instance**, select the Visual Builder instance where the application is deployed.
10. In the Authorization section, select **Use OAuth** or **Use Basic** to specify the type of authorization used to run this build step.
    - If you selected **Use OAuth**, the `Authorization is required` message displayed before authorization indicates that this build step needs to be authorized, either by clicking the **Authorize** button or by running the build manually and entering the credentials when you are prompted for them at that point. After authorization, the `Authorization has been provided` message and the **Renew Authorization** button are displayed.
    - If you selected **Use Basic**, enter your Visual Builder instance credentials in **Username** and **Password**.
11. In **Application URL Root** and **Application Version**, enter the visual application's root URL and its version.

    You can find the application's root URL and its version from the **Deployments** tab of the environment where the visual application is deployed.
    For example:



12. Click **Save**.
13. To run a build, click **Build Now**.

# Deploy Build Artifacts to Oracle Cloud Services

You can configure an Oracle Deploy build step to deploy your project's build artifacts like Java and Node.js applications to Oracle Cloud Services, including Oracle Java Cloud Service (JCS).

> **Note:**
>
> With Oracle Java Cloud Service reaching End of Life (EOL) in May 2025, support for JCS deployments in VB Studio is now deprecated. See Deprecated Features in *What's New in Visual Builder Studio?*.

Before you can create a build step for deployment, you must first create an environment and then add the JCS instance to it. This instance will be used for your deployment target. If you don't add it in the Environments page, you won't be able to select it in the build step. See Set Up an Environment for information about creating an environment and adding instances to it.

Today, more and more customers are securing their resources and Oracle Cloud Services, such as JCS, behind private VCNs. If you already run Oracle Cloud services in your private VCN and plan to use VB Studio to deploy to those services, you should configure your private VCN so your services and these VMs are in the same VCN. VM build executors need to access VB Studio and VB Studio needs to access the VM build executors. That access is done through a public subnet that needs to be configured. See Allow VM Build Executors to Access A Private Subnet's Resources.

You can either add a build step that deploys the build artifact(s) to the job that creates and packages the artifact(s) or you can create a separate job for each task. If you use separate jobs, you can create a pipeline that begins with a job that builds and packages the application, followed by a job that deploys the build artifact(s) to the desired target environment. Using pipelines allows you the flexibility to add testing and other tasks to the flow.

## Deploy an Application to JCS

You can create a job that copies build artifacts generated by another job and deploys those artifacts to a JCS target instance:

1. In the left navigator, click **Builds** .
2. In the **Jobs** tab, click **+ Create Job**.
3. In the New Job dialog box, in **Name**, enter a unique name.
4. In **Description**, enter the job's description.
5. In **Template**, select the build executor template.
6. Click **Create**.
7. In the Job Configuration page, in the **Before Build** tab, select **Copy Artifacts** from the **Add Before Build Action** dropdown.
8. Select the job that produces the artifact from the **From job** dropdown and the last successful build from the **Which build** dropdown, then click **Save**.
9. In the **Steps** tab, click the **Add Step** dropdown and select **JCS Deployment**.

A **JCS Deployment** step appears at the bottom of the Steps list.

10. In the **Target Instance** dropdown, select the JCS instance where you want to deploy the application.

    If you don't see the instance that you want to deploy to, you'll need to go to the Environments page and define a new instance. After you do that, it will show up as a target in the dropdown.

11. Enter the HTTPS port number, username, and password in their respective fields.
12. Click **Find Targets** and select the server you want to deploy to from the list of available servers or clusters.
13. Click **OK**.
14. In the **Build Artifact** field, enter the path to the artifact that you want to deploy.
15. In the **Application Name** field, enter the name that will be used by the target JCS service to identify your application.
16. Select the **Deploy as shared library** checkbox to deploy the artifact directly to the JCS server as a shared library or leave it unchecked (default).

    When the artifact is deployed directly to a JCS server, any application can reference the resources in that deployed shared library.

17. Click **Save**.
18. To run the job, from the **Builds** page, click **Build Now** and the job will execute, first copying the artifact, then deploying it to the selected JCS target instance.

At this point, you probably want to create a pipeline that flows a series of jobs that builds and packages the artifact, then retrieves it and deploys it to the desired JCS server in the target instance.

## Access a Deployed Application

You can access an application that has been deployed to an Oracle Cloud service from the target service's console. Here are some ways you can obtain the deployed application's URL. You'll need to enter your identity domain name and your credentials, if you're prompted to do so.

The Deployments tab on the Environments page shows *visual applications* that have been deployed for the current project. The Manage Extension Lifecycle page, also available on the Environments page, lists *extensions* that have been deployed for the current project or across projects.
For an application that's deployed to JCS, meanwhile, here's how you can create the URL:

1. Use the JCS View a Service Instance API to get the Content URL and examine the response body output to find the `content_url`.

   For example:

   ```
   curl -i -X GET -u jdoe@example.com:my_password -H "X-ID-TENANT-
   NAME:exampleidentitydomain" https://jaas.oraclecloud.com/jaas/api/v1.1/
   instances/exampleidentitydomain/exampleservice
   ```

   See REST API for Oracle Java Cloud Service.

   You need to use basic authentication to call the REST API. You can use cURL or a browser REST add-on, such as Postman for Google Chrome, to make this call.

2. Get the context root of the application from the `application.xml` deployment descriptor for EAR deployments or from the `web.xml` deployment descriptor for WAR deployments.

   If there is no such descriptor, get the context root from the WebLogic Console:

   a. Open the WebLogic Console of the JCS instance. You can access the console from the **Java Service** link of the JCS deployment configuration.

   b. Click **Deployments** in the Domain Structure pane.

   c. Click the deployed application name in the Deployments table.

   d. In the Overview tab, copy the value displayed by **Context Root**.

   The `<host>:<port>` referenced in the WebLogic Console is local to the JCS instance. You need to obtain and use the externally-available IP address or the host name of the JCS instance VM to access the deployed application.

3. Join the content URL and the context root of the application to construct the application URL.

   For example, if the content URL is `http://129.130.131.132` and the context root is `/deploy4214351085908057349`, the application's URL would be `http://129.130.131.132/deploy4214351085908057349`.
   See Accessing an Application Deployed to an Oracle Java Cloud Service Instance in *Administering Oracle Java Cloud Service*.

# Manage Oracle Cloud Service Deployments

By using the Oracle Java Cloud Service console, you can start and stop a deployment, redeploy an application, or undeploy a deployment.

| Action | How To |
|---|---|
| Start or stop the application | Open and use the target service's console to start or stop the deployed application on the target service. |
| Redeploy the application | If you've made changes to the source code or the build generated a new artifact, you can rerun the deploy build step to redeploy the application to the target service. |
| View logged deployment information | In the build log, locate and view the deployment section. |
| Undeploy a deployed application | Open and use the target service's console to stop and then undeploy the deployed application on the target service. |

# Part III

# Maximize the Power of Your Project

These are things that a project owner can choose to add and, if added, project members can use. Each chapter combines set-up and usage information.

**Topics:**


- Track and Manage Tasks, Defects, and Features
- Manage Software Releases
- Use Agile Boards to Manage and Update Issues
- Manage Binaries and Dependencies with Maven
- Access External Docker Registries
- Use the Project's NPM Registry
- Send Notifications to External Software Using Webhooks
- Share and Use Code Snippets
- Co-Author Wikis

9

# Track and Manage Tasks, Defects, and Features

Use issues to track new feature requests or enhancements, assign tasks to team members, or file bugs.

You can create, update, and search issues from the **Issues** page or from Agile boards. You can also use REST APIs to create, retrieve, and update issues.

In a project, you can create an issue as a Task, Defect, or a Feature. If your team uses an Agile Scrum board to update issues, Epic and Story types of issues are also available. An Epic is a larger issue typically composed of multiple smaller sub-issues or Story issues. An Epic must have sub-issues and can span multiple sprints.

These are the key steps you'll perform to create and track issues:

1. As the project owner, start by setting up products, components, and releases for your project, which you'll need when you start identifying tasks, defects, and features.

   If the default set of issue fields don't meet your requirements, create custom fields.
2. Create issues and assign them to your team members.
3. Update issues, either from the **Issues** page or from an Agile board.

## Issue Types

An issue can be categorized into task (or action), defect (or bug), and feature (or enhancement).

You can create these types of issues:

| Issue Type | Description |
|---|---|
| **Task** | Indicates a task, which means an action is required.<br>A Task type issue could be an action such as a piece of work that must be completed within a certain time, or a simple function to be performed. |
| **Defect** | Indicates a defect, which means a bug or a fault in the product. This is the default issue type.<br>A Defect type issue could be an error or a flaw that prevents the software from working correctly, or creates an unexpected output. |
| **Feature** | Indicates a new feature or an enhancement request.<br>A Feature type issue could be a request to add a new aspect or characteristic to the product, or enhance an existing feature. |
| **Epic** | Indicates an Epic, a unit of work that cannot be completed in one sprint. It can contain any other types of sub-issues necessary to complete that work.<br>An Epic is a larger issue, typically with multiple smaller sub-issues or stories. An epic can span multiple sprints and must have sub-issues (as stories or other issues). |

| Issue Type | Description |
|---|---|
| **Story** | Indicates a Story, a sub-issue of a parent epic issue.<br>The story issues of an epic issue can be allotted to different sprints in an Agile board. |

# Issue Statuses

An issue moves between one or more of these statuses during its lifetime:

| Status | Description | Statuses You Can Change To |
|---|---|---|
| Unconfirmed | The action required for this issue is unclear. | New, Assigned, Resolved |
| New | New issue, not yet assigned. | Assigned, Awaiting Third Party, Resolved |
| Awaiting Third Party | Awaiting information about a third-party component. | New, Assigned, In Progress, Resolved |
| Assigned | Issue has been assigned to an owner. | New, In Progress, In Code Review, Awaiting Third Party, Resolved |
| In Progress | Issue is being worked on. | Assigned, In Code Review, Awaiting Third Party, Resolved |
| In Code Review | Code changes to resolve issue are under review. | Assigned, In Progress, Resolved |
| Resolved | Issue has been resolved. | Reopened, Verified, Closed |
| Verified | QA has verified that the issue has been properly resolved. | Reopened, Resolved, Closed |
| Closed | Issue has been resolved and verified, and no further action is required. | Reopened, Resolved |
| Reopened | Issue hasn't been properly resolved or verified and needs to be revisited. | New, Assigned, In Progress, In Code Review, Resolved |

# Create Issues

You can create an issue from the Issues page or from the REST API. When you create an issue, it gets assigned a unique ID and is added to the issues list on the Issues page.

When you create an issue, you specify its summary, type, severity and priority, due date, tags, and release. You can assign the issue to a team member or to yourself, or leave the field blank to assign the issue later to a team member.

## Create an Issue from the Issues Page

You can create an issue from the Issues page:

1. In the left navigator, click **Issues** .
2. Click **New Issue**.
3. On the New Issue page, in **Summary** and **Description**, enter the issue's title and description.
4. In **Details**, specify the issue type, its severity and priority, product details, release, ownership, and project tags.
5. In **Time**, specify the due date and estimate (in days).

One day is estimated of 8 hours. To specify 3 hours, enter 0.375. To specify 2 days and 2 hours, enter 2.250.

6. In **Agile**, specify the effort estimate in Agile story points.
7. If there are any custom fields defined in your project, fill in the details, as required.
8. Click **Create Issue**.

> **Note:**
>
> VB Studio doesn't provide an option in the user interface for deleting an issue and doesn't provide the means to perform such a deletion from the service's backend database, even with administrative permissions. Instead, you should resolve and close the issue, using one of the available fields (such as Entered in error, Duplicate, Other) and add a comment explaining the reason.

# Search Issues

You can search for issues using the pre-defined filters under **Standard Searches**, **My Searches**, **Shared Searches**, or **Global Searches**. If you can't find the issue, you can run a basic search or an advanced one.

To run a basic search, use the **Search Issues** box in the upper-right corner of the Issues page. You can search for a term in the summary, description, or comments of issues. To clear the search term, click **Clear Filter** .

To run an advanced search, use the **Advanced Searches** link. You can search through all, not archived, or archived issues. You can also search for issues using various parameters such as sprints, product, version, date, owner, type, and priority.

To save the search query as a filter, click **Save this search**. To see the search query expression, click **Show Search String**. Later, if you want to edit the search query, click **Edit this search**.

## Save a Custom Search

You can save basic or advanced search queries as a custom search filters that can be used later:

1. On the Issues page, run a basic or an advanced search.
2. On the search results page, click **Save this search**.
3. In the Save Search dialog box, enter the search name.

   The custom search filter is available to you only. To share the search filter with project members, in the Save Search dialog box, select the **Shared** check box. In **Share with the following users**, select the users with whom you want to share the search query.

   To share the search filter with all project members, select the **Share with everyone** check box.
4. Click **OK**.

The location where the query filters can be found is determined by the way you configured the search:

- If you didn't select the **Shared** check box, the search query appears as a filter under **My Searches**.
- If you selected the **Shared** check box, the search query appears as a filter under **Shared Searches**.
- If you selected the **Share with everyone** check box, the search query appears as a filter under **Global Searches**.

To edit a custom search query, mouse over the query under **My Searches** and click **Edit**.

To delete a custom search query, mouse over the query under **My Searches** and click **Delete**.

## Share Custom Search Filters

You can share your existing custom search filters with other project members, which they can use to view the issues as you want:

| Element | Description |
|---|---|
| Share a search filter with specific project members | 1. In **My Searches**, mouse over the filter link, and click **Share**.<br>2. In the Start Sharing Search dialog box, select the project member names in **Share with following users**.<br>3. Click **OK**.<br>The filter link moves from **My Searches** to **Shared Searches**. |
| Share a search filter with all project members | 1. In **My Searches**, mouse over the filter link, and click **Share**.<br>2. In the Start Sharing Search dialog box, select the **Share with Everyone** check box.<br>3. Click **OK**.<br>The filter link moves from **My Searches** to **Global Searches**. |
| Stop sharing a search filter | 1. In **Global Searches** or **Shared Searches**, mouse over the filter link and click **Stop Share**.<br>2. In the Stop Sharing Search dialog box, click **OK**.<br>If the search query is being used by other project members, the dialog box shows their list.<br>When you stop sharing a search query, it is removed from the **Shared Searches** or **Global Searches** list for all project users. |

# View and Update Issues

To view or update an issue, click the issue's summary or the ID link on the Issues page. An issue link could also be found in the recent activities feed, wikis, Agile boards, and merge requests.

While updating an issue, you can change its status, properties, reassign it to another member, and change its priority or severity. You can also add comments in the **Comments** tab, upload attachments in the **Attachments** tab, and check the update history of an issue in the **History** tab. Updates made to issues can also be tracked in the recent activities feed of the Project Home page.

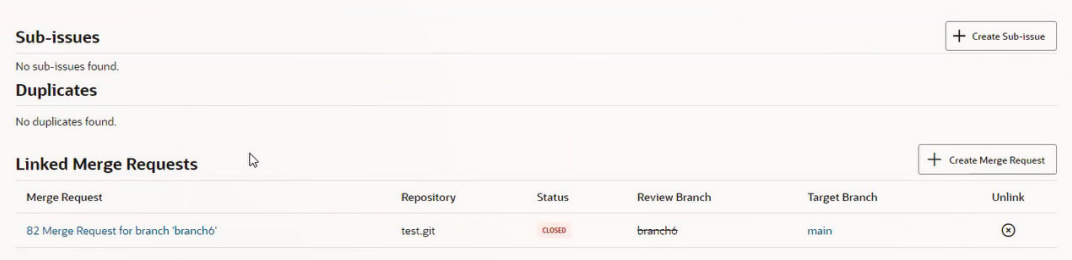
> **Note:**
>
> VB Studio doesn't provide an option in the user interface for deleting an issue and doesn't provide the means to perform such a deletion from the service's backend database, even with administrative permissions. Instead, you should resolve and close the issue, using one of the available fields (such as Entered in error, Duplicate, Other) and add a comment explaining the reason. Another alternative is to archive the issue, which effectively hides it from being viewed in several places in the Issues and Agile pages. See Archive Issues to learn more about this non-destructive process.

## Edit an Issue and Associate a Branch with a Merge Request

You can create a branch of the code directly from an issue and associate the two, clearly indicating which code changes relate to the specific issue.

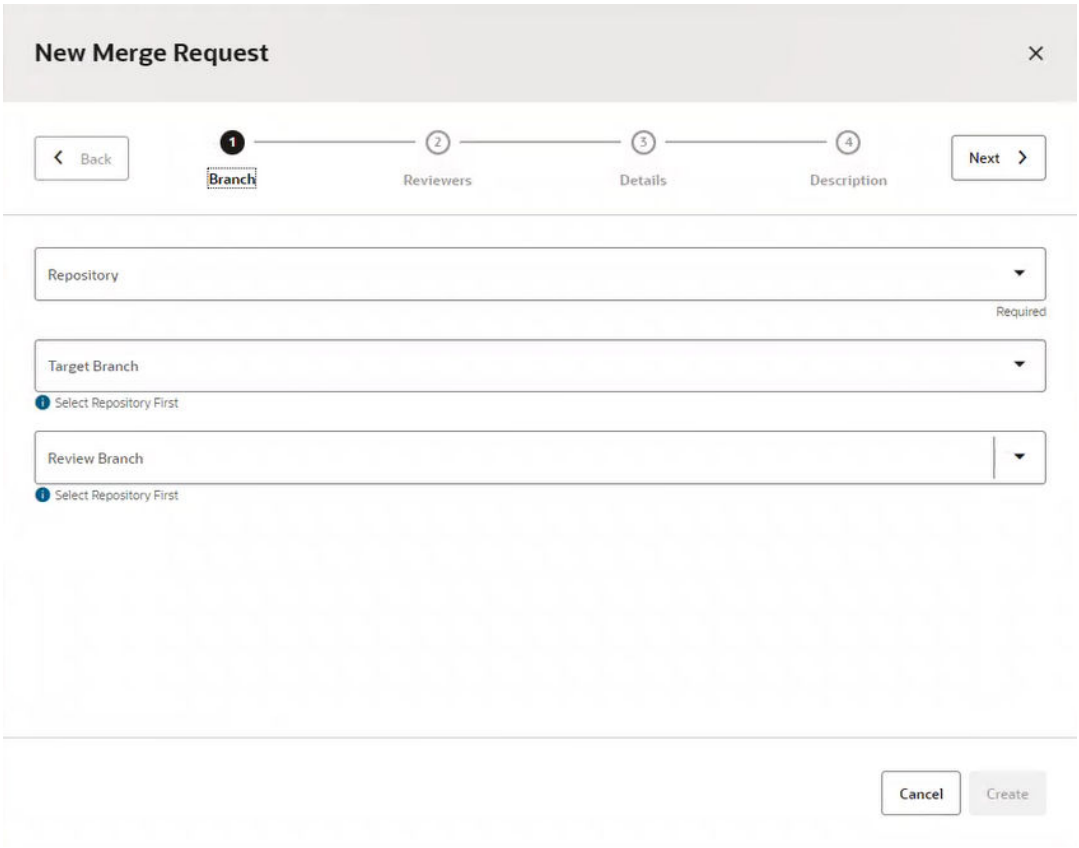
Here's how you create, view, and link/unlink MRs from the **Edit Issue** page:

1. In the **Issues** page, scroll down to the Linked Merge Requests section.




2. Click the **+ Create Merge Request** button and display the New Merge Request wizard.

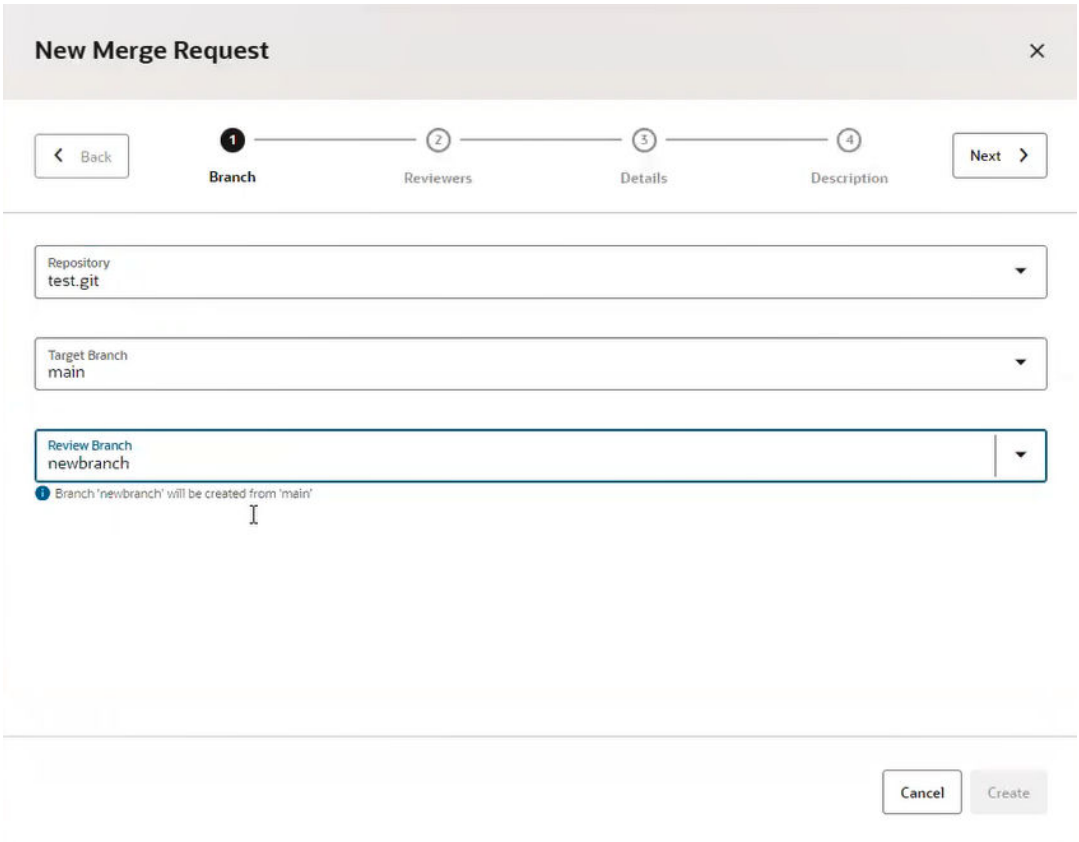
   The Branch page is where you start.




3. Select the repository and the target branch, then click the **Review Branch** dropdown and select an existing branch or create a new review branch. If you need to create a new branch, type the name of this new branch in the field.

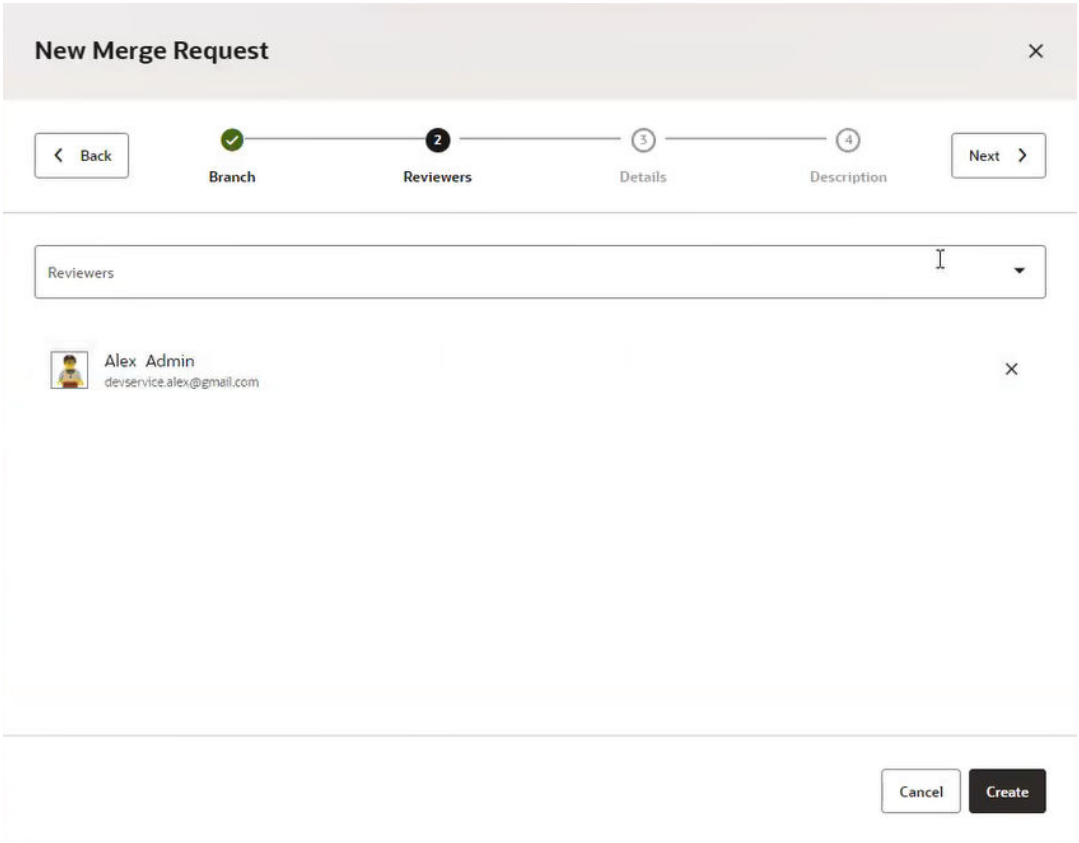
   Let's create a new branch, say newbranch, so we'll type it in the **Review Branch** field. Notice the message that is displayed under the field:

   ```
   Branch 'newbranch' will be created from 'main'
   ```



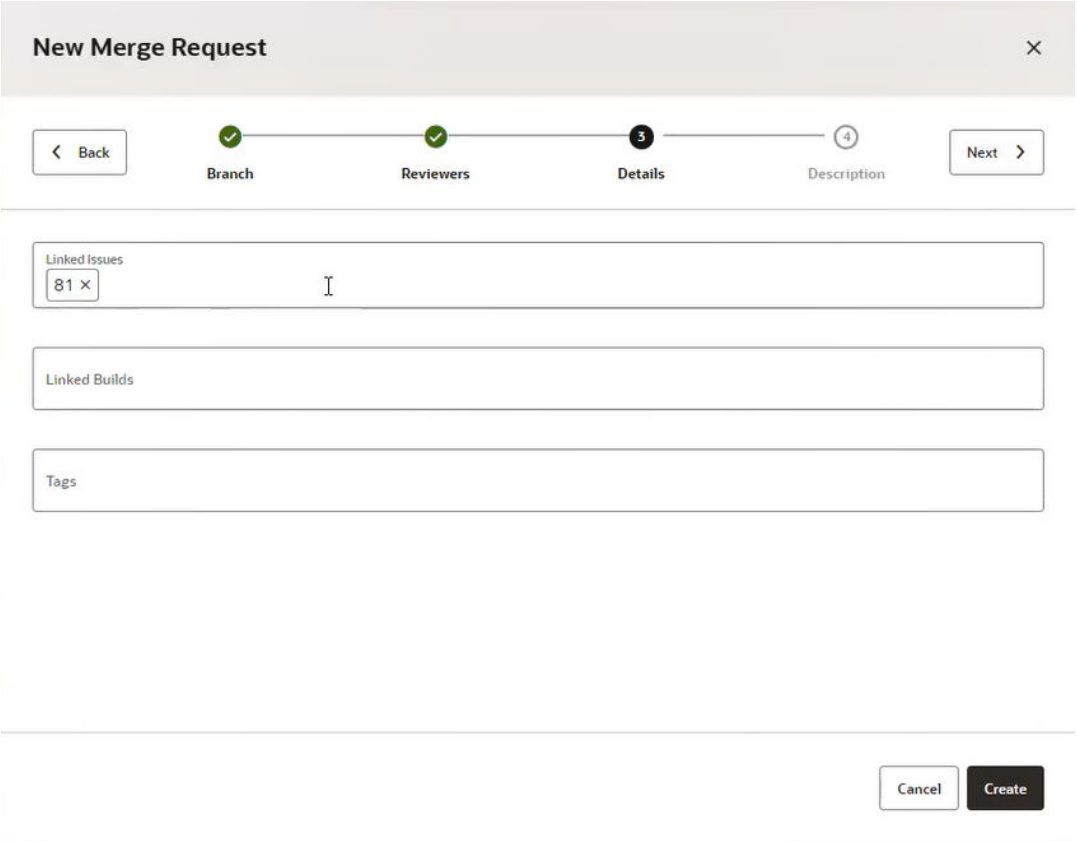

4. Click **Next** to go to the **Reviewers** page.

   Notice that the current user is automatically added as a reviewer.



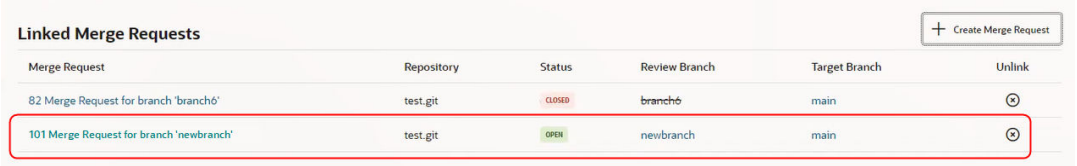
5. Add more reviewers, as needed, and click **Next** to go to the Details page.

   The current issue is automatically linked to the MR and is added to the **Linked Issues** field.

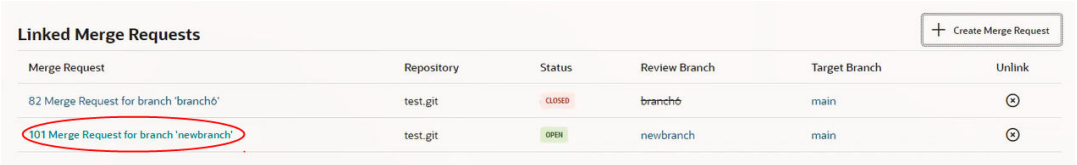



6. Click **Next** to go to the Description page.
7. Enter a summary and optionally a description, then click **Create**.

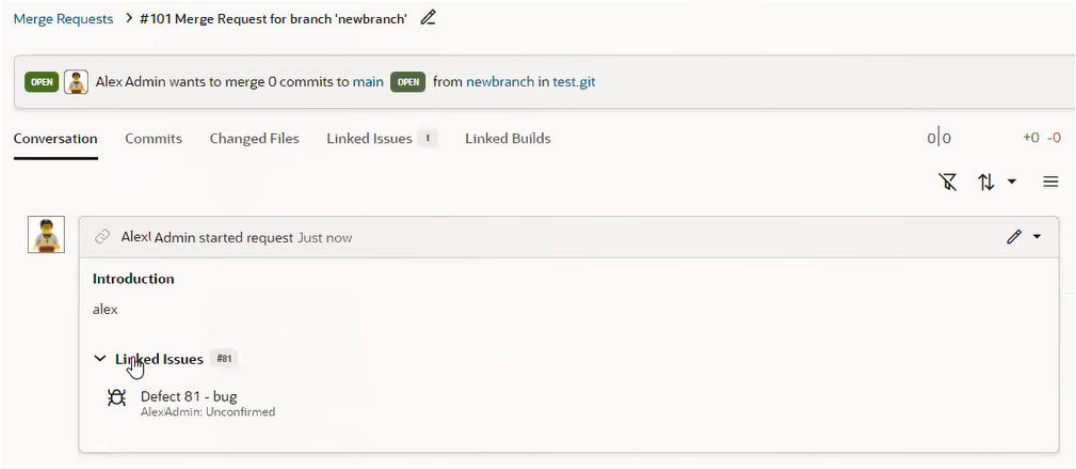
   The linked MR you created appears under the Linked Merge Requests section on the Issues page, where you can see the new MR name (link), repository, status, review and target branches, and the unlink ⊗ action icon.



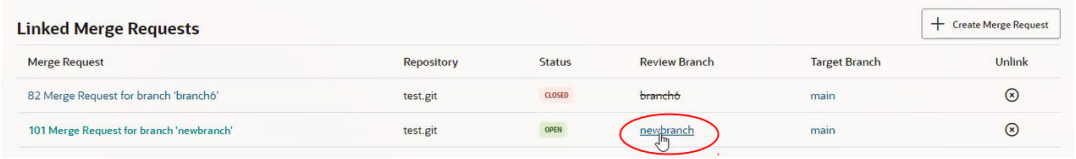

8. Click the MR name link to go directly to the merge request.



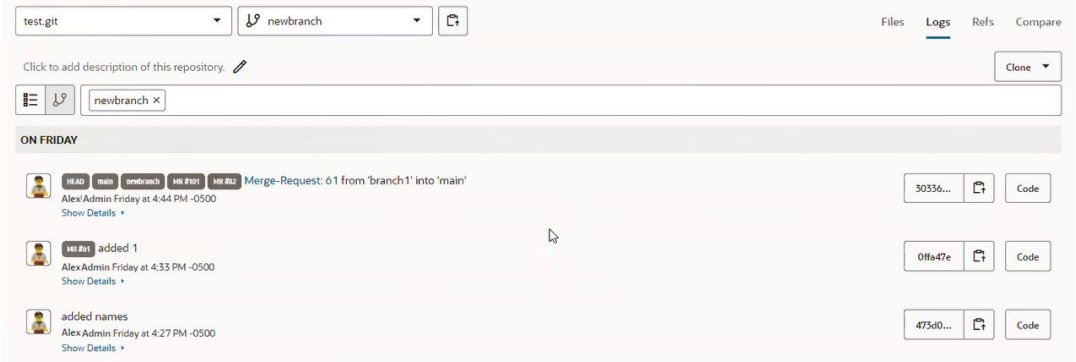

9. Expand "Linked Issues" to return to the Issues page.




10. Click the branch link.




The Branches page displays.




11. Return to the Linked Merge Request section on the Issue page and mouse over the branch6 item in the Review Branch column.

Notice that if a branch is remote or was deleted, you'll see a line through its name and you'll see the message like the one shown here.

12. Click the **Unlink** ⊗ icon and unlink the merge request from the issue.

    Unlinking breaks the association between the MR and the issue, but nothing is deleted. The MR is still there after unlinking.

So, you can now show and create the association from both sides, from a merge request as well as from an issue.

# Resolve an Issue

You can resolve an issue as Fixed, Invalid, Duplicate, Will not fix, Works for me, or Need info:

1. Click the issue link to open it in the Issues page.
2. From the **Status** drop-down list, select `Resolved`.
3. From the **Resolution** drop-down list, select the resolution.

| Sub-status | Indicates ... |
|---|---|
| **Fixed** | The issue has been fixed and is awaiting feedback from the QA team.<br>After verifying the fix, the QA team sets the issue's status to **Verified** or **Closed**. |
| **Invalid** | The issue isn't a valid issue. |
| **Will not fix** | The issue won't be fixed. |
| **Duplicate** | The issue is a duplicate of an existing issue.<br>Enter the issue ID of the existing issue in **Duplicate Of**. |
| **Works for me** | The issue cannot be reproduced. |
| **Need info** | The current issue description isn't sufficient to reproduce the issue; more information is required. |

## Mark an Issue as Duplicate

If you find a duplicate issue, mark it as a duplicate and specify the original issue:

1. Click the issue link to open it in the Issues page.
2. From the **Status** drop-down list, select **Resolved**.
3. From the **Resolution** drop-down list, select **Duplicate**.
4. In **Duplicate Of**, enter the original issue identifier or the summary text, and select the original issue.
5. Click **Save**.

## Update Time Spent on an Issue

When you work on an issue, create a time spent entry each time you update the issue:

1. Click the issue link to open it in the Issues page.
2. In the **Time** section, click **Add Time Spent**.
3. In the Add Time Spent dialog box, in **Time Spent**, specify the number of days you've spent on the issue.
4. To subtract the value specified in **Time Spent** from the existing value of **Remaining** (if set), use the default **Reduce remaining ... days by entered Time Spent** option.

   If **Remaining** isn't set, then the value specified in **Time Spent** is subtracted from **Estimate**. The option is disabled if the **Estimated** field isn't set.

   To specify the remaining days manually, select the **Set to** option and specify the remaining estimate.
5. In **Comment**, add a comment.
6. Click **OK**.

The **Time Spent Log** section shows the time spent entry of the time spent and updates the graph.

- To edit the time spent entry, click **Edit** and update the fields in the Edit Time Spent dialog.
- To remove a time spent entry, click **Remove** and update the fields in the Update Time Spent dialog box.

The remaining time is adjusted automatically.

## Associate an Issue with a Sprint

You can associate an issue with a sprint from the Edit Issue page:

1. Click the issue link to open it in the Issues page.
2. In the **Agile** section, from the **Sprint** drop-down list, click the search box, and select the sprint from the list.
3. Click **Save**.

Note that you can associate only one sprint with an issue.

## Create Parent-Child Relationships Between Issues

A parent-child relationship can be established between two or more issues:

| Action | How To |
|---|---|
| Create a child issue for an issue | You can create multiple child issues for an issue:<br>1. Click the issue link to open it in the **Issues** page.<br>2. Click **+ New Sub-issue**.<br>3. Enter details for the new issue and click **Create Issue**.<br>4. Click the parent issue's ID in the header to open it and, in the **Associations** section, verify the child issue's ID. |
| Add a parent issue to an issue | You can only add one parent to an issue:<br>1. Click the issue link to open it in the **Issues** page.<br>2. In the **Associations** section, in **Parent Issue**, enter the parent issue's ID or summary text, and select it.<br>3. Click **Save**. |

## Link One Issue to Another One (or to Multiple Issues)

Visual Builder Studio provides a way to link issues to other issues, using a predefined set of bidirectional relations. You can use linked issues to connect work across teams and projects in VB Studio. Linked issues can also help your team uncover, understand, and manage dependencies and related work.

Issues can have the following link types, all but the first one being used in pairs:

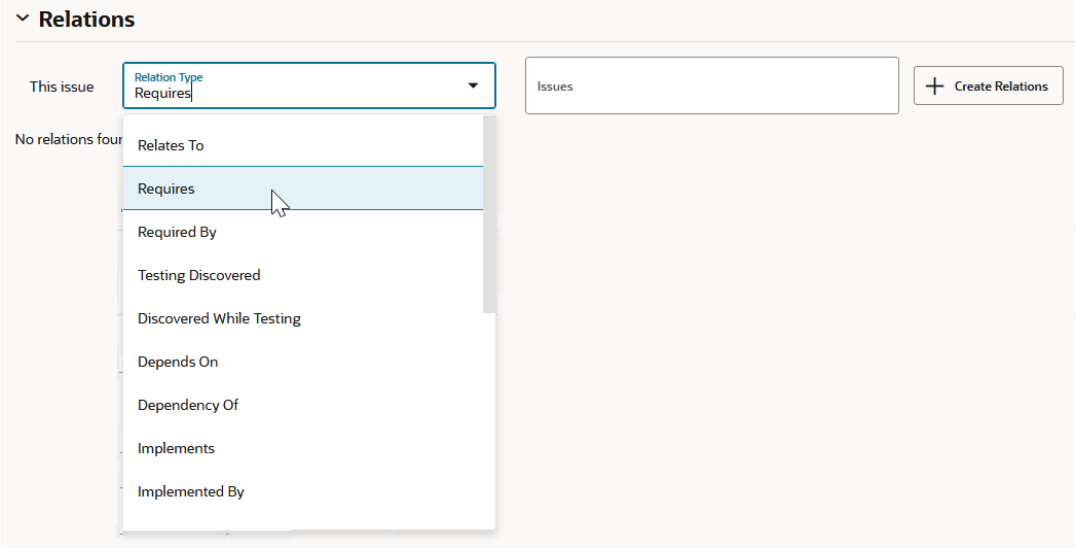
- Relates to
- Requires
- Required By
- Testing Discovered

- Discovered While Testing
- Depends On
- Dependency Of
- Implements
- Implemented By
- Fixes
- Will Be Fixed By
- Split To
- Split From
- Contains
- Contributes To

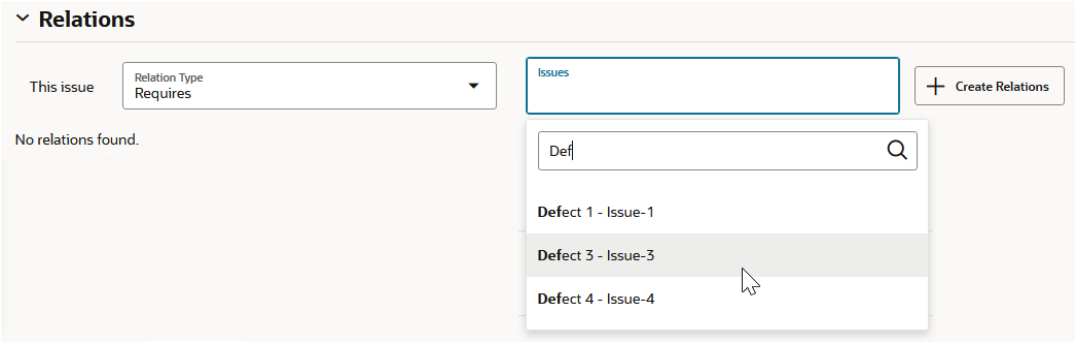
For example, if you set the relation type to Requires for the first issue, the issue you link it to will automatically be set to Required By. The linked issue is required by the original issue.

Set up a relation between linked issues from the **Edit Issue** page:

1. Click an issue's link to open it in the **Issues** page.
2. Scroll down to the Relations section and expand it.
3. Expand the **Relation Type** drop-down list and select the relation type to set up.




4. Click the **Issues** search box, enter your search criteria and, from the list, select the issue to link.




5. Click the **+ Create Relations** button.

   The row below the selector and search box displays the type of relation you selected, a link to the linked issue, and the linked-to issue's description and status.

6. Click the issue link to display the **Issues** page for the linked-to issue.

   Below the selector and search box, the Relations section displays a row showing the type of relation that was automatically selected for the linked-to issue.

   For example, if you selected "Requires" for the relation type when you created the relationship, "Required By" will be the relation type that is shown for the linked-to issue.

7. Click the linked issue's link and return to the issue from which the linked-to issue was linked.

## Update Multiple Issues

On the Issues page, you can update multiple issues in a batch to apply the same update:

1. In the issues list, press the Ctrl key or the Shift key while clicking to select the rows of issues.

   You can also use the Space bar and Up-Down arrow keys to select the issues. To select all issues, click **Select All**.

2. Click **Update Selected**.

3. On the Mass update page, select the check boxes of fields to update and specify their values.

   For example, at the bottom of the list, select the checkbox to the left of the **Archive or unarchive issue** field selector, then choose **Archive** to hide issues or **Unarchive** to redisplay issues that are currently archived.

   Note that the **Component** check box is enabled when its **Product** is selected.

The contents of **Found In** and **Release** are determined by **Product**. If **Product** isn't specified, the intersection of all known products is used. For example, if product `P1` has **Found In** set to `1.0, 2.0` and product `P2` has **Found In** set to `1.0, 1.5`, then with no product specified, the **Found In** is set to `1.0`. The same logic is applied for **Release** too.

4. Click **Next**.
5. On the Issues will be Updated page, verify the summary, and click **Save**.

Issues that fail the update are listed with a description of the error. To resolve errors for multiple issues, select the error issues and click **Update Selected**. You'll be taken to the Issues Selected page where the previous changes you made are shown.

If all issues are successfully updated, you'll be returned to the Issues page.

## Archive Issues

VB Studio does not support deleting issues, but does provide a way to hide (archive) them as a way to clean up obsolete, out of scope, and decommissioned issues so they don't pile up unnecessarily in the backlog. There are no special permissions needed to mark an issue as one that is obsolete. Issues that were archived (hidden) can be unarchived (unhidden) at any time. Unlike deletion, archiving issues isn't a destructive operation.

VB Studio supports query filtering based whether an issue was archived or not. Multiple issues can be archived in a single operation as well. Archived issues only affect relevant queries on the Issues tab, not issues referenced in MRs, builds, links, or elsewhere. Archived issues don't show up in the default predefined queries so, if you want to display a default query that shows all archived issues, you'll have to add a new default query for that.

**Global Search** (in the upper right corner of the Issues page) doesn't filter issues when you look for specific issues, whether they've been archived or not. Typeahead search in the issue's "Duplicate Of" and "Associations" fields doesn't exclude archived issues either, and neither do sub-issue queries in the "Sub-issues" table or duplicate queries in the "Duplicates" table. The **Advanced Search** function does expose archived issues.

The issue owner and CC list members receive notifications after an issue has been archived (or unarchived). This notification excludes mass updates, effectively limiting email notifications to a per-issue basis. Archiving/unarchiving activity is automatically logged in the issue's History as well as in the project's Recent Activities feed. Archived issues don't appear in the Agile backlog.

Once an issue has been archived, it won't appear again in most user queries. Archived issues can be unarchived at any time, after which they will reappear wherever they had been hidden.

An issue cannot be archived while it is being created. You can only archive an existing issue.

1. In the Issues page, select an issue, and click its **Summary** link to open it.
2. Click the **Archive** checkbox and then click **Save**.

   VB Studio displays a message that says the issue has been updated.
   When you return to the Issues page, notice that the archived issue no longer appears in any of the views under **Standard Searches** (**Assigned to me**, **Open issues**, **Recently changed**, or **Related to me**). It only shows up in the list view under Archived issues.

   Also notice that the archiving/unarchiving action you performed shows up in the **History** tab under **Activity** and under Recent Activities on the Project Home page.

> **Tip:**
>
> You can archive or unarchive multiple issues in a single operation:
>
> a. Click the checkboxes to the left of each issue you want to update.
> b. Click **Update Selected**.
> c. Under **Details**, scroll to the bottom of the list, select the **Archive** checkbox, and use the selector to pick **Archive** (or **Unarchive**).
> d. Click **Next** to display the **Summary** window.
> e. Review your choices, and then click **Save**.
> VB Studio displays a message that says the selected issues have been updated.

3. Return to the Issues page, select an issue, and click its **Summary** link to open it.
4. Click the **Unarchive** checkbox and then click **Save**.

   VB Studio displays a message that says the issue has been updated.
   When you return to the Issues page, notice that the archived issue reappears in all the views under **Standard Searches** (**Assigned to me**, **Open issues**, **Recently changed**, and **Related to me**). It no longer appears in the list view under **Archived issues** any more though.

Archived issues, indicated by the Archived badge just to the left of the summary text, are displayed in some Agile boards, in the backlog, in active sprints, and in reports:

- A board cannot be queried using the predefined **Archived Issues** query.
- Archived issues aren't shown the Backlog page's Backlog section.
- If you send an issue (or drag-and-drop it) from a sprint to the backlog in the Backlog page, the issue will be removed from the sprint but it won't be added into the Backlog section.
- Archived issues can be added to or removed from a sprint.

# Watch an Issue

You can set up a watch on an issue and get email notifications when a project user updates an issue, adds a comment, or adds or removes an attachment:

| Action | How To |
|---|---|
| Issues assigned to you | By default, you get email notifications of issues assigned to you. If you aren't getting the email notifications, select the **Issue updates, attachments and comments** check box in your user preferences page:<br>1. In the branding bar, click the user avatar, and select **Preferences**.<br>2. Click the **Notifications** tab.<br>3. Select the **Issue updates, attachments and comments** check box, if not selected.<br>4. To the left of the **User Preferences** title, click **Close** ‹ to return to the last opened page. |

| Action | How To |
| --- | --- |
| Issue created by another user | 1. In the branding bar, click the user avatar, and select **Preferences**.<br>2. Click the **Notifications** tab.<br>3. Select the **Issue updates, attachments and comments** check box, if not selected.<br>4. To the left of the **User Preferences** title, click **Close** < to return to the last opened page.<br>5. Open the issue in the Issues page.<br>6. In the **Details** section, in **CC**, enter and select your name.<br>You may also enter other names of other users if you want to notify them too.<br>7. Click **Save**.<br>To stop watching, remove your name from the **CC** field. |
| Issues you created but are assigned to another user | By default, you get email notifications of issues created by you. When you create an issue and assign it to another user, your name is set in the **CC** field of the issue. Open the issue in the Issues page and verify your name in the **CC** field of the **Details** section.<br>To stop watching, remove your name from the **CC** field. |

10

# Manage Software Releases

A Release enables you to provide a stable code and artifacts of your applications that project users can download. For a release, you can specify tags or branches of Git repositories with stable code, artifacts of project Maven repository, build artifacts of stable builds, and binary files.

For example, you can you create a release titled *V18-Q1* to mark stable code files, artifacts, and binaries of your application for the first quarter release of 2018 release. Project users then won't have to look around or ask which Git repository or branch has the stable code. They can then download Git repository archives and other artifacts of the *V18-Q1* release from the Release page itself.

You can access and manage releases from the **Releases** page. When a project user opens a release, the user can download source snapshots of a specified branch or tag of the project Git repository, artifacts from the project Maven repository, specified binaries, and archived build artifacts.

## What Are Release States?

A release can be in one of these states: Draft, Pre-Release, or Public.

| State | Description |
|---|---|
| Draft | Indicates that the features of the release are under development.<br>When you create a release, you specify the Maven artifacts and the Git repository tags. While adding a Git repository to a release, you may want to specify a branch that has the stable code at the time of release. Usually, it's the `main` branch, but you can specify any branch name. You may also want to specify the Git repository tag that indicates the stable state of the branch. Usually, the tag is created before the release when the code in the branch is stable. While creating a Release, if you specify a tag name that doesn't exist, it's automatically created when you change the status of the Release to **Public**. |
| Pre-Release | Indicates that the release is stable, but might need some fixes before it becomes **Public**.<br>You usually set the release's status to this state after your team has completed all features, staged the software, and are awaiting approvals to release the software. If the Maven artifacts, Git repositories, tags, or branch names have changed since the release was in the **Draft** state, edit the release and update the artifacts. |
| Public | Indicates that the release is public or is ready to go public.<br>While creating a release, if you specify a tag name that doesn't exist, it will be automatically created for the specified branch when the release is set to **Public**. If you specified an existing tag name, it will be used. This might be useful when you create a release that is already public.<br>You might want to edit the release and update the Maven artifacts, Git repositories, tags, or branch names if they have changed while the release was in the **Pre-Release** or the **Draft** state. |

# Create a Release

When you create a release, you specify the build artifacts, Git repositories and branches, and Maven artifacts. You can create a release or clone an existing release.

| Action | How To |
|---|---|
| Create a release | 1. In the left navigator, click **Releases** .<br>2. Click **+ Create Release**.<br>3. In **Name** and **Description**, enter a release name and description.<br>4. In **Status**, specify the status of the release.<br>5. Add the artifacts.<br>6. In **Notes**, enter the release notes in the **Page Text** tab. Preview the notes in the **Preview** tab.<br>You can use the project's wiki markup language to format the notes.<br>7. Scroll to the top of the page and click **Save**. |
| Clone a release | 1. In the left navigator, click **Releases** .<br>2. Select the release that you want to edit or clone, click **Actions** and then select **Clone**.<br>3. In **Name** and **Description**, enter a release name and description.<br>4. In **Status**, specify the status of the release.<br>5. Add, update, or remove the artifacts.<br>6. In **Notes**, enter the release notes in the **Page Text** tab. Preview the notes in the **Preview** tab.<br>You can use the project's wiki markup language to format the notes.<br>7. Scroll to the top of the page and click **Save**. |

# Change a Release's Status

You can change the status of a release from the Edit Release page.

1. In the left navigator, click **Releases** .
2. In the Release list, select the release with a status you want to change.
3. On the right, click **Actions** and select **Edit**.
4. In the Edit Release page, change the selected **Status** option to the desired state.
5. Click **Save**.

The Activity Feed on the Project Home page will display a notification about the changed release state.

# Specify a Release's Artifacts

You can specify a release's artifacts when you create it or edit it.

| Action | How To |
|---|---|
| Add Build artifacts | Expand **Builds** and specify the job, build number, and its artifact. Click **Add to Release** ✓ to add the artifact. You can specify multiple artifacts.<br>To use the last build of the specified job when the release's status changes to Public, in **Build**, select **Last Build**. When the release's status changes to Public, make sure that the last build is successful and has generated desired artifacts.<br>To store the artifact in the project's storage system, in case the specified job or build is removed, select the **Store** check box. |
| Add Maven artifacts | Expand **Maven Artifacts** and specify the group ID, artifact ID, and version of artifacts. Click **Add to Release** ✓ to add the artifact. You can specify multiple artifacts. |
| Add Git repositories | Expand **Repositories**, and specify Git repositories and branches (or tags).<br>Click **Add to Release** ✓ to add the artifact. You can specify multiple artifacts.<br>If you enter a tag name that doesn't exist, a Git repository tag of the same name will be created when the release is marked as **Public**. |
| Add binary artifacts | The only time you can add binary artifacts is when you *edit* a release. You can't add binary artifacts when you *create* a release.<br>Expand **Binaries** and upload the binary files. |

# Manage Releases

After creating a release, you can edit its artifacts and properties, change its status, or delete it.

| Action | How To |
|---|---|
| Edit a release | On the Releases page, select the release that you want to edit. Click **Actions** ☰ and then select **Edit**. On the Edit Release page, update its name, description, artifacts, and click **Save**. |
| Change a release's state | Edit a release. On the Edit Release page, in **Status**, change the state. The name of the release at the top of the page shows the selected release state. |
| Delete a release | On the Releases page, select the release that you want to edit. Click **Actions** ☰ and then select **Delete**. |

# Download a Release's Artifacts

You can download these artifacts: binary files, build artifacts, Maven artifacts, and a Git repository tag's archive.

Here's where you get those artifacts:

- Binary files from the Binaries section
- Build artifacts of successful builds from the Builds section
- Maven artifacts from the Maven Artifacts section
- An archive of a tag of a Git repository in the Repositories section

To download an artifact, expand its section, click the file name, and save the file at the desired location on your computer.

11

# Use Agile Boards to Manage and Update Issues

The Agile methodology is a type of software development that's based on an incremental model focusing on process adaptability and customer satisfaction. In Oracle Visual Builder Studio (VB Studio), you use the Agile methodology to manage issues in Scrum and Kanban boards.

If you're new to Agile, see `http://agilemethodology.org/` for more information.

Before creating a board, appoint a team member as the Agile board's leader. This leader is responsible for managing and updating the board's issues, setting up team meetings to discuss the progress with these issues, and updating them in the board.

Here are the key steps the board's leader performs when creating and managing issues:

1. If required, create an issue query that returns a list of issues from which you'll select and add to the Agile board.
2. Create an Agile board (Scrum or Kanban).
3. Configure the working days, progress states, and other board properties.
4. Manage sprints or active issues.
5. Update the issues' progress states.
6. Review reports and adjust the sprints, issues, or the board accordingly.

## Agile Boards Concepts and Terms

Before you start using the Agile boards, it's important that you know about key components and concepts of the Boards page.

| Component | Description |
|---|---|
| **Board** | A Board is used to display and update issues of the project using the Agile methodology. There are two types of boards available: Scrum and Kanban. When you create a board, you associate it with an issue query and the issues returned by the query are added to the board. You can create your own board or use a board created by a team member.<br>In a board, you update issues by moving them to different progress states of the board. Each progress state has some pre-defined conditions that specify which issues can be assigned to a progress state. |
| **Scrum** | In a Scrum board, tasks are broken small actions to be completed in fixed duration cycles, called as Sprints. |
| **Kanban** | In a Kanban board, tasks are managed with a focus on continuous delivery. |

| Component | Description |
|---|---|
| **Sprint** | A Sprint is a short duration (usually, a week or two) during which your team members try to implement a product component.<br>You add the product component related issues to a sprint. When you start working on a product component, you start (or activate) its related sprints. To update issues of a sprint, you must first activate the sprint and add the sprint to the Active Sprints view. |
| **Story Points** | A Story Point is a metric that defines the relative effort of work and helps to understand how complex the issue is. |
| **Backlog** view | In a Scrum board, the Backlog view displays issues of the board, active and inactive sprints of the board, and the sprints from other boards that contain issues matching the board's query.<br>Each sprint lists issues added to it. The Backlog section (the last section of the Backlog page) lists all open issues that aren't part of any sprint yet. The Backlog view doesn't show the resolved and closed issues.<br>In a Kanban board, the Backlog view displays active issues (issues being actively worked on) in the Active Issues section and a backlog list of issues (issues aren't being actively worked on) in the Backlog section. The Epic issues don't appear in the Backlog view. |
| **Active Sprints** view | Available in a Scrum board, the Active Sprints view lists all active sprints of the board and enables you to update an issue status by dragging and dropping it to the respective status columns. |
| **Active Issues** view | Available in a Kanban board, the Active Issues view enables you to manage the progress of active issues. |
| **Reports** view | Displays various reports and charts that summarize the progress of issues. |

# Create and Configure Agile Boards

An Agile board contains issues that are returned by an issue filter. If none of the pre-defined or shared issue filters meet your requirements, you can create a custom search query and save it as a filter.

## Create a Board

When you create a board, you specify the board type, an issue search query, and the estimation criteria.

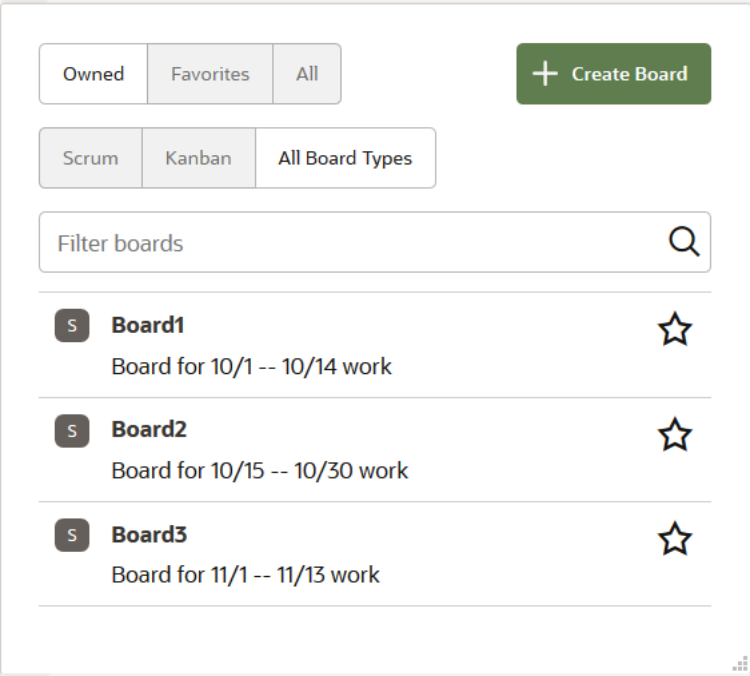
Any project member can create a board from the **Boards** page:

1. In the left navigator, click **Boards**.
2. Click **+ Create Board**.
3. In the Create Board dialog, enter a name and select the board type.
4. In **Search**, select the standard or custom issue search query. By default, **All Issues** is selected.
5. In **Estimation**, select the estimation type as **Story Points** or **Estimated Days**.
6. Click **Create**.

A board is created, issues that match the search query are added to the board, and you're brought to the Backlog view. The board's owner role is also granted to you.

Note that **Resolved**, **Verified**, and **Closed** issues aren't added to the board. To add new issues to a board, edit the issue search query to reflect the issues in its search result. The issues are automatically reflected in the Backlog list of the board.

You can also create a board from the Switch Board menu. From the board name menu, click **+ New Board**.




After creating a board, you can configure its working days, progress states, and conditions.

## Add and Manage Progress States of a Board

A progress state defines the progress of issues in a board. By default, each board has three progress states (To Do, In Progress, and Completed), but you can add more. Each progress state has some pre-defined conditions. A condition defines an issue's state. You cannot add, edit, or delete a condition.

You can add and manage progress states from the Configure Board page of the board.

1. Open the board.
2. From the **Board** drop-down list, select **Configure**.
3. Click the **Progress States** tab.

| Action | How To |
|---|---|
| Edit a progress state | 1. From the progress states list, select the progress state.<br>2. To edit the name and description, in **Name** and **Description**, enter a new name and description.<br>3. To update the capacity (number of issues in the progress state), update the value in **Suggested Issue Capacity**.<br>If the number of issues exceeds the suggested capacity, a warning icon and a message appears in the Active Sprints or the Active Issues view.<br>4. To remove a condition, select the condition from the **Conditions** list, click > and move it to the **Unassigned Conditions** list.<br>To add a condition, select the condition from the **Unassigned Conditions** list, click < and move it to the **Conditions** list.<br>For example, if you remove the **Resolved - WorksForMe** from the **Completed** progress state, issues in the **Resolved - WorksForMe** state don't appear in the board. |
| Add a progress state | A condition can be associated with one progress state only. Before you add a progress state, remove the conditions that you want to apply to the new progress state from their existing progress states.<br>1. Click **+ Add Progress State**.<br>2. In **Name** and **Description**, enter a name and description.<br>3. In **Suggested Issue Capacity**, specify the number of issues to be allowed.<br>If the number of issues exceeds the suggested capacity, a warning icon and a message appears in the Active Sprints or the Active Issues view.<br>4. To mark the state as the Completed state, select the **Completed State** check box.<br>The check box is disabled if a **Completed** state exists or the current state isn't the last state in the list.<br>5. To add a condition, select the condition from the **Unassigned Conditions** list, click **Move to Conditions** < and move it to the **Conditions** list.<br>6. To remove a condition, select the condition from the **Conditions** list, click **Move to Unassigned Conditions** > and move it to the **Unassigned Conditions** list. |
| Reorder progress states | In the progress states list, use the Up and Down order buttons to change the orders of the states. The buttons appear when you mouse over the state name. The order of states in the list is reflected in the **Swimlanes** and **Columns** views.<br>The **Completed** state must be the last state in the list. If **Completed** isn't the last state in the list, a **Warning** icon appear next to the state name. Any changes made to the page aren't saved until the **Completed** state is the last state in the list. |
| Delete a progress state | In the progress states list, mouse over the progress state, and click **Delete**. All conditions of the deleted progress state move to the **Unassigned Conditions** list and are available to new progress states.<br>You can't delete the **Completed** state, but you can delete other states. |

Click **Save** when you're finished.

# Configure Working Days of a Board

You can configure the working days and non-working of a week by modifying the board's calendar.

The working and non-working days that you specify affect the output of the Sprint Report, Issues Report, and the Burndown Chart.

1. Open the board.
2. From the **Board** drop-down list, select **Configure**.
3. Click the **Working Days** tab.
4. On the Configure board page, specify standard working and non-working days.
   - In **Standard Working Days**, select or deselect the check boxes of the working weekdays.
   - In **Non-Working Days**, click **+ Add** to add a non-working date (such as a holiday). From the calendar, select the date.

     To edit a non-working day, select it from the list, and select the new date in the calendar.
   - Select (or deselect) the **Show Non-Working Days in Sprint Report Chart** check box to show (or hide) the non-working days in the sprint reports.

     If selected, the non-working days appear in gray at the top in the burndown charts.
5. Click **Save**.

# Configure and Manage a Board

From the Board menu, you can select options to configure, duplicate, and delete the board. From the Configure Board page, you can edit and update the name, description, associated issue search query, and estimation criterion of a board.

| Action | How To |
|---|---|
| Edit a board's name and description | 1. From the **Board** drop-down list, select **Configure**.<br>2. In the **General** tab, in **Name** and **Description**, update the values.<br>3. Click **Save**. |
| Edit the board's search query and estimation | 1. From the **Board** drop-down list, select **Configure**.<br>2. In the **General** tab, in **Search** select the search query. In **Estimation**, enter the new estimation value.<br>3. Click **Save**. |

| Action | How To |
|---|---|
| Enable or disable time tracking | If the time tracking is enabled, then the **Active Sprints** page shows the estimation in **Remaining Days** only. The **Backlog** page shows the estimation in **Remaining Days** if estimation metric is **Estimated Days**.<br>If the time tracking is disabled, then the **Backlog**, **Active Sprints** and **Reports** pages show the chosen metric for **Estimation** (estimated days or story points) instead of **Remaining Days** and **Time Spent**.<br>1. From the **Board** drop-down list, select **Configure**.<br>2. In the **General** tab, in **Time Tracking**, select **On** or **Off** to enable or disable time tracking.<br>3. Click **Save**. |
| Create a copy of a board | To create another board with similar properties of an existing board that you can access, instead of creating a new board and manually copying properties, you can create a copy of the board. The copied board has properties include time tracking, progress states, and working days.<br>1. From the **Board** drop-down list, select **Copy Board**.<br>2. In the Copy Board dialog box, click **Copy**. |
| Delete a board | You can't delete a Scrum board with active sprints. You must complete the active sprints before you delete the board. You can delete a Kanban board with active issues or archives. All issues of the deleted board are returned to the backlog.<br>1. From the **Board** drop-down list, select **Delete Board**.<br>2. In the Delete Agile Board dialog box, select the **I understand that my agile board will be permanently deleted** check box and click **Delete**. |

# Use Scrum Boards

Using a Scrum board, you manage and update issues using sprints.

A Scrum board has three views:

- **Backlog**: Lists all of the board's active and inactive sprints, as well as a backlog list of issues.
- **Active Sprints**: Manages work on issues in an active sprint.
- **Reports**: Displays several different types of issue reports.

## Create and Manage Sprints

You can create and manage sprints from the Backlog view of a Scrum board. You must be a project owner or the board owner to create, edit, update, or delete a sprint.

When an issue is assigned to a sprint, the sprint is displayed in all boards whose issue query returns any issue of the sprint in its result. You might find such sprints in your board. Note that you can't edit or start sprints that you didn't create, or sprints that weren't created in the current board.

If there are no issues assigned to a sprint, the sprint is available only in the board in which it was created.

| Action | How To |
|---|---|
| Create a sprint | 1. From the toggle buttons, click **Backlog**.<br>2. In the Backlog view, click **+ Add Sprint**.<br>3. In the Add Sprint dialog box, enter the sprint name. If the Scrum board uses story points, then specify the sprint's capacity.<br>4. Click **OK**. |
| Edit a sprint | 1. From the toggle buttons, click **Backlog**.<br>2. In the Backlog view, for an inactive sprint, click **...** and select **Edit sprint**.<br>If the sprint is active (or started), click **Edit Sprint**.<br>3. In the Edit Sprint dialog box, you can update the sprint's name, board, start and end dates, and its capacity in story points.<br>The **Story Points** field is available if story points were selected as the estimation for the board.<br>4. Click **OK**. |
| Start a sprint | When you or your team begin work on a sprint and want to update the issues of the sprint, you must first start (or activate) it.<br>1. From the toggle buttons, click **Backlog**.<br>2. In the Backlog view, for the sprint that you want to start, click **Start Sprint**.<br>3. In the Start Sprint dialog box, specify the start and end dates of the sprint.<br>If necessary, update the sprint's name and its estimate.<br>4. Click **Start**.<br>The started sprint is now available in the Active Sprints view. |
| Reorder a sprint | In the Backlog view, by default, the inactive sprints (also called as future sprints) in the order they were created. You can change their display order manually.<br>For the inactive sprint that you want to move up or down, click **...** and select **Move sprint up** or **Move sprint down**.<br>You cannot change the order of active sprints. The active sprints are ordered by Start Date in the Backlog view. If two sprints have the same Start Date, then they are ordered by name. |
| Move a sprint to another board | 1. From the toggle buttons, click **Backlog**.<br>2. In the Backlog view, for an inactive sprint, click **...** and select **Edit sprint**.<br>If the sprint is active (or started), click **Edit Sprint**.<br>3. In the Edit Sprint dialog box, from the **Board** drop-down list, select the target board.<br>4. Click **OK**. |
| Delete a sprint | You can delete an inactive sprint from the Backlog view of the board. You can't delete an active sprint.<br>For the inactive sprint that you want to delete, click the **...** and select **Delete Sprint**. The sprint is deleted and all issues of the sprint are moved to the Backlog list. |

# Add and Manage Issues in a Sprint

From the Backlog view, you can drag-and-drop issues to add or remove them from a sprint.

When you add or remove issues from a sprint, keep a close eye on the sprint's capacity . If the sprint's total story points for issues exceed the sprint's story points capacity, you'll see a warning message. If this happens\, you can either increase the sprint's capacity or move some issues to another sprint.

Here's how to manage issues in a sprint:

| Action | How To |
|---|---|
| Add an issue to a sprint | 1. From the Backlog list or from the sprint that contains the issue, drag the issue to the target sprint.<br>2. In the blue dotted rectangle that appears when you drag the issue to the target sprint, drop the issue in the blue rectangle.<br>You can also right-click the issue and select **Send to** > *target sprint name* to move the issue to the target sprint. A Sprint field is also added to the issue indicating the sprint it's associated with.<br>If you're unable to use the drag-and-drop action, click the issue link to open it in the Issues page. Navigate to the **Agile** section. From the **Added To** drop-down list, select the sprint. |
| Create an issue from the sprint | In the sprint, below the issues table, click **New Issue**. On the New Issue page, enter the issue's details, and click **Create Issue**.<br>The new issue is automatically associated with the current sprint. |
| Remove an issue from a sprint | 1. From the sprint that contains the issue, drag the issue to the **Backlog** list.<br>2. In the blue dotted rectangle that appears when you drag the issue to the Backlog list, drop the issue in the blue rectangle.<br>You can also right-click the issue and select **Send to** > **Backlog** to remove the issue from the sprint.<br>If you're unable to use the drag-and-drop action, click the issue link to open it in the Issues page. Navigate to the **Agile** section. In the **Added To** field, click . |

# Update Issues in an Active Sprint

You can use the Active Sprints view to manage the progress of issues in an active sprint.

You can use either the **Swimlanes** view or the **Columns** view to see the issues in an active sprint. The *Swimlanes* view displays issues categorized by issue owner (the member to whom the issue is assigned). For each issue owner, issues are grouped in vertical progress (or status) columns.

The *Columns* view displays issues grouped in vertical progress columns.

By default, each board contains three columns: To Do, In Progress, and Completed. If you need to, you can use the Configure Board page to add more progress columns to the board.

## Update an Issue's Progress in an Active Sprint

You can update an issue's progress in the Active Sprints view by dragging it from one progress column to another.

If you're unable to use the drag-and-drop action, click the issue ID to update its progress from the Edit Issue page.

1. Open the board that owns the active sprint.
2. Click **Active Sprints**.
3. Select the issue list view: **Swimlanes** or **Columns**.
4. To update an issue's progress, drag and drop it from one column to another.

   For example, when a team member starts work on an issue, drop the issue to the **In Progress** column (if exists).
5. In the Change Progress wizard, from the **To** drop-down list, select the new status of the issue. If necessary, enter a comment in the **Comment** field.

   If you want to update the time spent on the issue, click **Next**.
6. Click **OK**.

If the board uses story points, the number to the right of the column name is updated. An activity is also added to the **History** tab of the issue's **Activity** section.

**Update Time Spent on an Issue**

When you move an issue from one state to another, you can also update the time spent on the issue in the Change Progress wizard.

In the Add Time Spent page, in **Time Spent**, specify the number of days you've spent on the issue. In **Remaining**, use the default **Reduce remaining ... days by entered Time Spent** option to automatically subtract the value specified in **Time Spent** from the existing value of **Remaining**, if **Remaining** was set previously.

If **Remaining** was not set, then the value specified in **Time Spent** is subtracted from **Estimate**. The option is disabled if the **Estimated** field isn't set.

To specify the remaining days manually, select the **Set to** option and specify the remaining days.

## Reschedule a Sprint

While updating issues of a sprint, you can change the start or end date of the sprint, or update its capacity from the Edit Sprint dialog box.

1. In the Backlog view, for the sprint you want to reschedule, click **Edit sprint**.
2. In the Edit Sprint dialog box, change the start and end dates.

   To update the sprint's capacity, update **Story Points** . The field is available if story points were selected as the estimation of the board.
3. Click **OK**.

## Complete a Sprint

You can complete a sprint from the Active Sprints view of the board.

You must be a project owner or the board owner to mark the sprint as completed.

1. Open the board the sprint belongs to.
2. Click **Active Sprints**.
3. In the sprint drop-down list on the left, select the sprint.
4. Click **Complete Sprint**.
5. In the Complete Sprint dialog box, select the **I understand that it will be removed from the Active Sprint view** check box, and click **Complete Sprint**.

After a sprint is complete, it's removed from the Active Sprints view. A warning displays if there are any incomplete issues in the sprint. All incomplete issues go back to the next inactive sprint, or to the Backlog section if there are no inactive sprints. The Sprint Report page opens showing the day-by-day progress of the sprint issues.

## Review Issue Reports for a Scrum Board

The Reports view in a scrum board provides several different kinds of reports:

| This Kind of Report | Displays ... |
|---|---|
| Burndown chart | The amount of unfinished work in a sprint or in an epic |
| Cumulative Flow chart | The total number of issues in each of the board's progress states, over time |
| Control chart | Information about an issue's progress state change events on the timeline |
| Sprint report | A sprint's complete, incomplete, and open issues |
| Epic report | An epic's complete, incomplete, and open stories |
| Velocity report | A velocity chart for completed sprints |

See Review Agile Reports and Charts.

# Use Kanban Boards

Using a Kanban board, you manage issues using Active issues.

A Kanban board has three views: Backlog, Active Issues, and Reports. The Backlog view lists active issues (issues that're being actively worked on) and a backlog list of issues (issues aren't being actively worked on). The Active Issues view enables you to manage the progress of active issues. The Reports view displays various issue reports.

## Add and Manage Active Issues

From the Backlog view, you can add issues to or remove issues from the Active Issues list using drag-and-drop actions.

When you add or remove issues from the Active Issues list, keep a watch on its capacity. If the total story points of active issues are more than the story points capacity of the board, a warning message appears. In such a case, you can either increase the capacity of the board or remove some issues from the Active Issues list.

| Action | How To |
|---|---|
| Activate an issue | 1. From the Backlog list, drag the issue to the **Active Issues** section.<br>2. In the blue dotted rectangle that appears when you drag the issue to the **Active Issues** section, drop the issue in the blue rectangle.<br>In the Backlog list, you can also right-click the issue and select **Send to** >**Active Issues** to move the issue to the Active Issues list. Issues already added to sprints of Scrum boards aren't available in the Kanban board Backlog list.<br>If you're unable to use the drag-and-drop action, click the issue link to open it in the Issues page. Navigate to the **Agile** section. From the **Added To** drop-down list, select the Active Issues option under the board name. |
| Create an issue from the Active Issues section | In the Active Issues section, below the issues table, click **New Issue**. In the New Issue page, enter the issue's details, and click **Create Issue**.<br>The new issue is automatically activated and associated with the Kanban board. |
| Remove an issue from the Active Issues list | 1. From the Active Issues section, drag the issue to the **Backlog** list.<br>2. In the blue dotted rectangle that appears when you drag the issue to the Backlog list, drop the issue in the blue rectangle.<br>You can also right-click the issue and select **Send to** > **Backlog** to remove the issue.<br>If you're unable to use the drag-and-drop action, click the issue link to open it in the Issues page. Navigate to the **Agile** section. In the **Added To** field, click **Remove** ⊗. |

# Update Active Issues

The Active Issues view enables you to manage the progress of active issues.

You can use either the **Swimlanes** sub-view or the **Columns** sub-view to view the active issues. The *Swimlanes* sub-view displays issues categorized into issue owners (member whom the issue is assigned to). For each issue owner, the issues are categorized into vertical progress (or status) columns. The *Columns* sub-view displays the issues categorized into vertical progress columns.

By default, each board contains three columns: To Do, In Progress, and Completed. If required, you can add more progress columns to the board from the Configure Board page.

From the Active Issues view, you can update an active issue's progress state and archive the completed issues.

## Update the Progress State for an Active Issue

You can update an active issue's progress in the Active Issues view by dragging it from one progress column to another.

If you're unable to drag-and-drop the issue, click the issue ID and update its progress from the Edit Issue page.

1. Open the board that owns the active issues.
2. Click **Active Issues**.

3. Select the desired issue list view: **Swimlanes** or **Columns**.

   If necessary, use the sort list boxes to sort the active issues.

4. To update an issue's progress, move it from one column to another.

   For example, when a team member starts work on an issue, move the issue to the **In Progress** column (if exists).

5. In the Change Progress wizard, from the **To** drop-down list, select the new status of the issue. If necessary, enter a comment in the **Comment** field.

   If you want to update the time spent on the issue, click **Next**.

6. Click **OK**.

An activity is added to the **History** tab of the issue's **Activity** section.

**Update Time Spent on an Issue**

When you move an issue from one state to another, you can update the time spent on the issue in the Change Progress wizard.

In the Add Time Spent page, in **Time Spent**, specify the number of days you've spent on the issue. In **Remaining**, use the default **Reduce remaining ... days by entered Time Spent** option to automatically subtract the value specified in **Time Spent** from the existing value of **Remaining**, if **Remaining** was set previously.

If **Remaining** was not set, then the value specified in **Time Spent** is subtracted from **Estimate**. The option is disabled if the **Estimated** field isn't set.

To specify the remaining days manually, select the **Set to** option and specify the remaining days estimate.

## Complete a Sprint

You can complete a sprint from the Active Sprints view of the board.

You must be a project owner or the board owner to mark the sprint as completed.

1. Open the board the sprint belongs to.
2. Click **Active Sprints**.
3. In the sprint drop-down list on the left, select the sprint.
4. Click **Complete Sprint**.
5. In the Complete Sprint dialog box, select the **I understand that it will be removed from the Active Sprint view** check box, and click **Complete Sprint**.

After a sprint is complete, it's removed from the Active Sprints view. A warning displays if there are any incomplete issues in the sprint. All incomplete issues go back to the next inactive sprint, or to the Backlog section if there are no inactive sprints. The Sprint Report page opens showing the day-by-day progress of the sprint issues.

# Review Issue Reports for a Kanban Board

Several kinds of reports are available in the Reports view for a Kanban board:

| This Kind of Report | Displays ... |
|---|---|
| Burndown chart | The amount of unfinished work in issues or in an epic |
| Cumulative Flow chart | The total number of issues in each of the board's progress states, over time |

| This Kind of Report | Displays ... |
|---|---|
| Control chart | Information about issue’s progress state change event on the timeline |
| Issues report | Active and archived issues. |
| Epic report | An epic's complete, incomplete, and open stories |
| Velocity report | A velocity chart of completed issues |

See Review Agile Reports and Charts.

# Review Agile Reports and Charts

Several kinds of reports and charts cab be used with your Scrum and Kanban boards:

- Burndown charts
- Sprint reports
- Issues reports
- Epic reports
- Velocity reports
- Cumulative flow charts
- Control charts

## Burndown Charts

Burndown charts show how much work still needs to be completed in issues or epics.

The burndown chart for issues and epics is available in Scrum and Kanban boards:

1. Open the board.
2. Click **Reports**.
3. Click **Issues** or **Epic** .
4. Click the **Burndown Chart** tab.

### Scrum Boards

For a Scrum board, the chart for your active sprint is displayed.

- To use a different sprint, click the **Sprint** drop-down.
- To use a different estimate criterion, click the **Burndown** drop-down and select from **Estimated Days**, **Story Points**, or **Number of Issues**. The Y-axis in the chart reflects this setting.

The burndown chart displays the configured tracking statistic for the active sprint, start and end dates, the sprint capacity, and a guideline for completing the statistic you’re tracking in the sprint. The horizontal axis tracks time; the vertical axis represents your configured tracking statistic: story points, estimation days, or number of issues. Use burndown charts to see the total work remaining and increase the accuracy for predicting the likelihood that you'll achieve the sprint's goal. By tracking the remaining work throughout the iteration, your team can manage its progress and respond appropriately, especially if things don’t go as planned. If time tracking is enabled for the board, the burndown chart always shows the number of remaining days and the amount of time spent.

A burndown chart includes all of the sprint's issues, those that've been completed as well as those that are still being worked on. Mapping the status for these issues to your board determines when an issue is considered completed or not completed.

The bottom of the page shows a history, a table of events associated with the issues, including issues that still haven't been completed.

**Kanban Boards**

For a Kanban board, the chart displays your active issues. To see issues for an archived version, click the **Issues** list and select the desired one.

The burndown chart displays the configured tracking statistic for the active issues. The horizontal axis tracks time; the vertical axis represents your configured tracking statistic, either story points, estimation days, or the number of issues. Use burndown charts to see how much work remains to be done. This will increase your ability to accurately predict the likelihood that you'll achieve the goal. By tracking the remaining work throughout the iteration, your team can manage its progress and respond appropriately if things don't go as planned. If you enable time tracking for the board, the burndown chart always shows the number of days remaining and the amount of time spent thus far.

The burndown chart includes all issues, those that have been completed and those that are still pending. Mapping these statuses to your board determines when an issue is considered completed or not completed.

The bottom of the page shows a history, a table of events associated with the issues, including issues that still haven't been completed.

## Sprint Reports

Sprint Reports show completed and open (not yet completed) issues in a sprint.

Sprint Reports are available for sprints in Scrum boards only:

1. Open the board.
2. Click **Reports**.
3. If necessary, click .
4. Click the **Sprint Report** tab. The Sprint Report chart for your active sprint is displayed.
   - To select a different sprint, select it from the **Sprint** drop-down list.
   - To select a different estimate criterion, select **Estimated Days**, **Story Points**, or **Number of Issues** from the **Burndown** drop-down list. The Y-axis in the chart reflects this setting.

A Sprint Report provides a day-by-day progress report, with much of the same information that's in the burndown chart, although in a slightly different format. The Sprint Report shows the list of issues in each sprint. It provides useful information for your Sprint Retrospective meeting and for mid-sprint progress checks. Mapping the statuses to your board determines when an issue is considered completed or not Completed. If you enabled time tracking for the board, the Sprint Report chart shows the number of days remaining in the sprint and the amount of time spent to date.

At the bottom of the page, the Sprint Report displays tables that show completed, open, and removed issues.

# Issues Reports

Issues Reports show active and archived issues.

The Issues Report is available for issues in Kanban boards only:

1. Open the board.
2. Click **Reports**.
3. If necessary, click **Issues** .
4. Click the **Issue Report** tab. The Issue Report chart for active issues is displayed.

   To select a version that shows archived issues, click the **Issues** drop-down list and select the desired option.

An Issues Report provides a day-by-day progress report with much of the same information found in a burndown chart, although in a slightly different format. An Issues Report lists active and completed issues. If you enabled time tracking for the board, the Issue Report chart shows the number of days remaining in the sprint and the amount of time spent to date.

At the bottom of the page, the Issues Report displays a table of issues. If you select **Active Issues** (default) in the **Issues** drop-down list, the table lists completed and non-completed issues. If you selected an archive instead, the table lists only completed issues.

For each issue, the report shows the original estimate value and modified values in Estimated Days (or Story Points).

# Epic Reports

Epic Reports show which of the epic's stories have been completed and which are still open (not completed).

The Epic Report is available in Scrum boards and Kanban boards for epics:

1. Open the board.
2. Click **Reports**.
3. Click **Epic** .
4. Click the **Epic Report** tab.

**For Scrum Boards**

The Epic Report Chart for your active sprint is displayed.

- To select a different epic, select it from the **Epic** drop-down list.
- To select a different estimate criterion, select **Estimated Days**, **Story Points**, or **Number of Issues** from the **Burndown** drop-down list. The Y-axis in the chart reflects this setting.

An Epic Report provides a day-by-day progress report with much of the same information that's in the burndown chart, although in a slightly different format. An Epic Report lists the stories (or sub-issues) in each epic. It provides useful information for your Epic Retrospective meeting and for mid-sprint progress checks. Mapping statuses to your board determines when a story is considered completed or not. If you enable time tracking for the board, the Epic Report chart shows the number of days remaining and the amount of time spent to date.

At the bottom of the page, the Epic Report displays tables of completed, open, and removed stories.

**For Kanban Boards**

The Epic Report Chart for your active issues is displayed. To select an archived issue version, click the **Issues** list and select the desired version.

An Epic Report provides a day-by-day progress report with much of the same information that's in the burndown chart, although in a slightly different format. An Epic Report lists the stories (or sub-issues) in each epic. It provides useful information for your Epic Retrospective meeting and for progress checks. Mapping statuses to your board determines when a story is considered completed or not. If you enable time tracking for the board, the Epic Report chart shows the number of days left in the epic and the amount of time spent to date.

At the bottom of the page, the Epic Report displays tables of completed, open, and removed stories.

## Velocity Reports

Velocity Reports show velocity charts for completed sprints.

Velocity Reports are available for Scrum boards only:

1. Open the board.
2. Click **Reports**.
3. Click **Velocity** .
4. From the **Estimation** drop-down list, select **Estimated Days**, **Story Points**, or **Number of Issues**.
5. Depending on the value selected in **Estimation**, a Velocity Chart is displayed for committed and completed values.

   If you selected Story Points in the **Estimation** drop-down list, the chart will also display the **Suggested Capacity** for each sprint as a dashed horizontal line.

A Velocity Report includes a Velocity chart, showing a graph of the last seven completed sprints for the selected estimation. It also shows a table that lists completed sprints, the number of issues in each sprint, estimated values committed, and estimated values completed. Active sprints aren't shown or listed.

You can use the Velocity Report to plan the amount of work that can be committed to future sprints. Managers can see whether the team met the original estimation and can plan the effort required for new or future sprints.

At the bottom of the page, there is a sprint table, whose columns are determined by the value you selected in the **Estimation** drop-down list.

## Cumulative Flow Charts

Cumulative Flow Charts display the total number of issues for each of the board's progress states over time.

Cumulative Flow Charts are available in Scrum boards and Kanban boards for issues only:

1. Open the board.
2. Click **Reports**.

3. If you are using a Scrum board, click **Sprint**. If you're using a Kanban board, click **Issues**.
4. Click the **Cumulative Flow Chart** tab.

### Scrum Boards

A Cumulative Flow Chart displays the total number of issues in each of the board's progress states over time for the active sprint. The issues that are listed in the chart are the same issues that the Sprint Report displays.

These events can change the number of issues in a progress state:

- An issue is added to the sprint
- An issue is removed from the sprint
- An issue's progress state in the sprint changes because its status or resolution changes

The Configure Board page shows the progress states that correspond to the board's current list. The chart is a stacked area chart that enables you to see the number of issues in each progress state and the total number of issues in the sprint at any given point on the timeline. The color for each progress state is randomly assigned. By clicking the progress state names in the chart legend, you can show or hide them. The events table has a column for each progress state and shows the number of issues for each progress state that's affected by the event.

### Kanban Boards

A Cumulative Flow Chart displays the total number of issues in each of the board's progress states over time for the Active Issues or an archive. The issues listed in the chart are the same as those that are displayed in the Issues Report.

These events can change the number of issues in a progress state:

- An issue is added to active issues or an archive
- An issue is removed from active issues or an archive
- An issue's progress state in active issues or an archive changes because its status or resolution was changed

The Configure Board page shows the progress states that correspond to the board's current list. The chart is a stacked area chart that enables you to see the number of issues in each progress state and the total number of issues in the sprint at any given point on the timeline. The color for each progress state is randomly assigned. By clicking the progress state names in the chart legend, you can show or hide them. The events table has a column for each progress state and shows the number of issues for each progress state that's affected by the event.

## Control Charts

Control charts show progress state changes to issues on a timeline.

Control charts are available in Scrum boards and Kanban boards for both Sprint reports and Issues reports. A Control chart is a scatter chart with each point representing an event where a progress state changed on the timeline. This event occurs when an issue moves from one progress state to another.

Here's how to create a Control chart

1. Open the board.
2. Click **Reports**.
3. If you are using a Scrum board, click **Sprint** . If you are using a Kanban board, click **Issues** .
4. Click the **Control Chart** tab.

The Y-axis of the chart shows how many days an issue spent in its previous progress state (the progress state from which the issue was moved). For example, if an issue's status was changed on Jan 10 from **To Do** to **In Progress**, the chart displays a point on Jan 10 and the Y-axis shows the number of days the issue was in the **To Do** progress state.

The colors for the progress states are randomly assigned. You can show or hide the progress state points by clicking on the names of the progress states in the chart legend. Click the chart legend to display a line on the chart that shows the average number of days spent in a progress state.

The events table has a column for each progress state in the board, with values representing the number of days spent in the progress state at the time of each event. The table displays the average number of days a little differently than the chart does.

12

# Manage Binaries and Dependencies with Maven

View, upload, and search artifacts in the project's Maven repository. When a project is created, VB Studio creates a hosted Maven repository in the project. You can use the repository to store binary files and dependencies. If you're developing Maven applications, you can use the Maven repository to store and access build artifacts.

If you see the **Connect Your OCI Account** message, contact your organization administrator to connect an OCI account.

## Maven Concepts and Terms

Apache Maven is a software project management tool that uses the Project Object Model (POM) concept to manage a project's build.

If you're new to Maven, see `https://maven.apache.org` to learn about Maven basics such as POM files and Maven repositories.

In VB Studio, you use the project's Maven repository to store build artifacts and dependencies for your project's applications. Usually, you store the dependencies on the project's Maven repository that aren't available on a public Maven repository, such as Maven Central Repository or Oracle Maven Repository.

Here are the terms that this documentation uses to describe the Maven terms and components of a project.

| Term | Description |
|---|---|
| POM file | An XML file that contains configuration about how to build the application. Usually, the file is saved as `pom.xml`. For more information, see `https://maven.apache.org/guides/introduction/introduction-to-the-pom.html`. |
| **Browse** view | Displays and allows you to browse artifacts of the project's Maven repository. |
| **Upload** view | Allows you to upload artifacts manually to the project's Maven repository. |
| **Artifact Search** view | Enables you to search artifacts in the project's Maven repository. |

To upload and access the files of the repository programmatically, configure the POM file of your application. You can use the project Maven repository among other projects of the organization for local builds as well as project builds.

## Set Up and Populate Your `settings.xml` File

The Maven page provides you with the settings you need to put in your `settings.xml` file for basic or token-based authentication, used for establishing a connection with your VB Studio Maven repository, as well as profile-specific settings with repository details that the Maven build process uses during dependency resolution. You can easily copy that information to the

clipboard and paste it into your file. The only thing you need to modify is the password or encrypted password.

Authentication is configured in Maven using `<server>` elements in the `settings.xml` file. Each `<repository>` element specified in the `settings.xml` file must have a corresponding `<server>` element with a matching `<id>` that specifies the username and password.

VB Studio generates a snippet with these elements for you to copy and paste into your local `settings.xml` file.

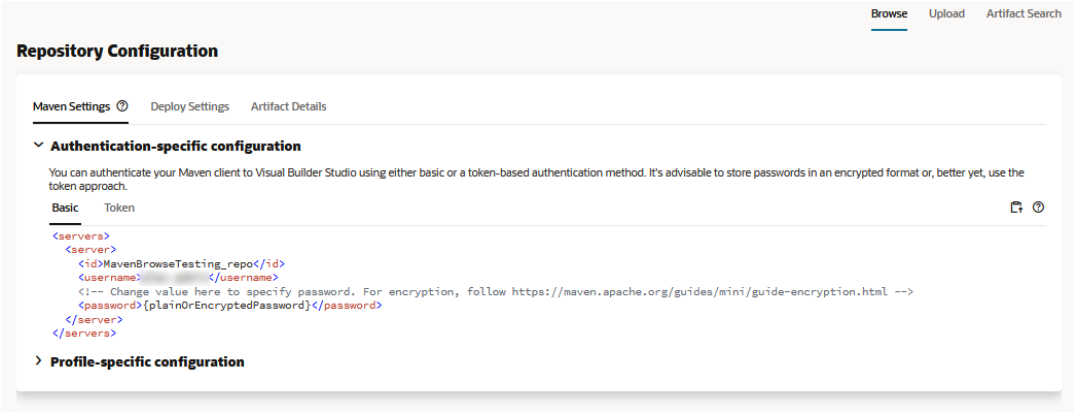
**Perform the Authentication-Specific Configuration:**

You can authenticate your Maven client to Visual Builder Studio using either basic or a token-based authentication. If you decide to use basic authentication, you can use a plain text password, but this approach is never recommended. Passwords really need to be stored in an encrypted format, at a minimum. A much better alternative would be to use token-based authentication instead of basic authentication.

**Copy and Use the Basic Authentication Snippet**

To configure basic authentication, copy the snippet and paste it into your local `settings.xml` file then modify the password:

1. In the Browse view, in the Repository Configuration section, select the **Maven Settings** tab.
2. Click ❯ to expand the **Authentication-specific Configuration** section, if it's collapsed.
3. Click the **Basic** tab, if it isn't already selected.




4. Click the **Copy** icon to copy the settings.
5. Open your Maven `settings.xml` file in a text editor.
6. Paste the settings you copied directly into the file, in the `<settings>` element.
7. Modify the password element.
   This is a multi-step process that's described in the page linked to from the snippet: https://maven.apache.org/guides/mini/guide-encryption.html. The summarized steps follow.
   a. Create a master password that's used to encrypt all the other Maven passwords. Open a terminal window and type:

      ```
      $ mvn --encrypt-master-password
      ```

      When you're prompted for a master password, enter it and Maven will spit out a long string, something like this:

      ```
      {w5+NYEttGTAHV3FanFoel4N5uUmbcvtzRoWZHI5N97jtssbo0O/93W/XLlm0caeM}
      ```

Keep your terminal window open while you do the next step.

b. Store the master password.
Create a file called `settings-security.xml` in the `~/.m2` directory and copy/paste the following block into it:

```
<settingsSecurity>
  <master></master>
</settingsSecurity>
```

Copy/paste the long encrypted string that Maven spit out in the previous steps in between the `<master>` tags and save the file.

c. Encrypt your password.
Open another terminal window and type:

```
$ mvn --encrypt-password
```

Enter the password you want to encrypt and Maven will return an encrypted string that looks similar to the one it returned for the master password. Copy it and open your `settings.xml` file. Replace the placeholder password between the `<password>` tags with the new encrypted version.

d. Save the file.

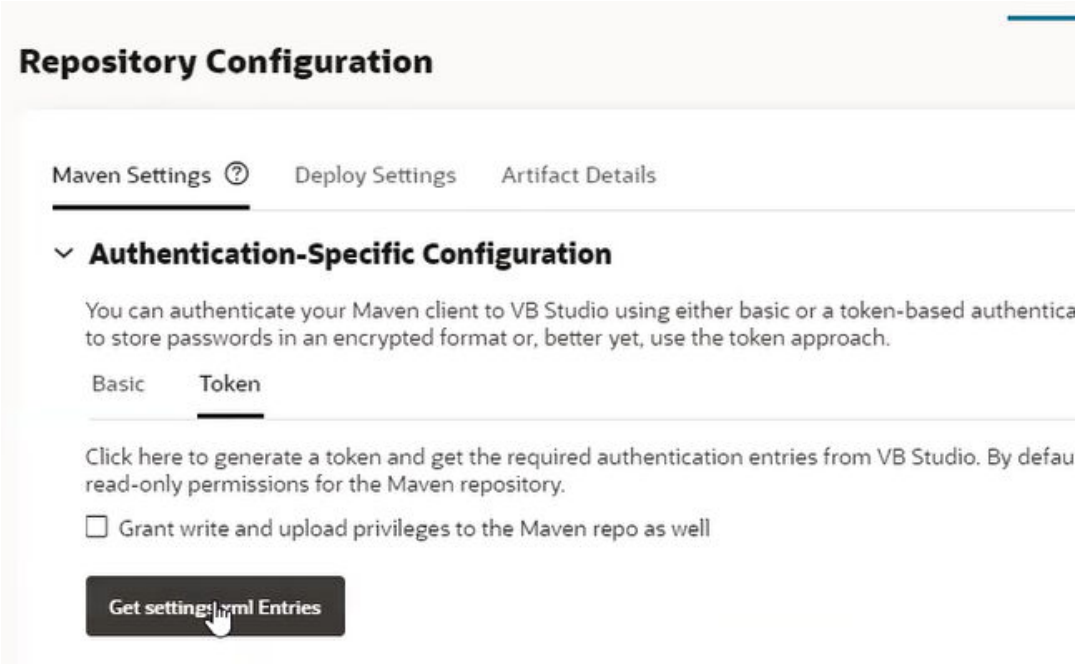
Encrypted passwords are certainly safer to use than unencrypted ones, but they can still be decrypted by someone that has the master password and settings security file, so you need to keep it secure or store it separately if you expect that there's any chance that the `settings.xml` file may be retrieved. Given the limitations and inherent risks of using basic authentication, you may be better off using token-based authentication instead.

**Copy and Use the Token-Based Authentication Snippet**

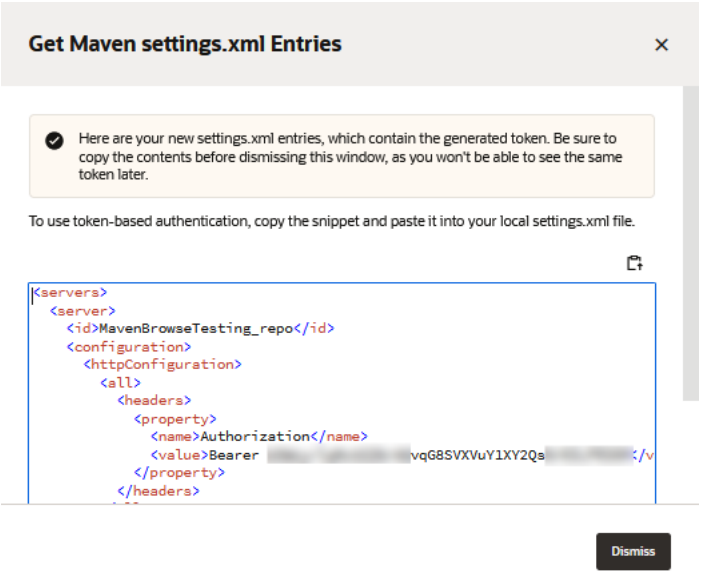
To use token-based authentication, copy the snippet provided and paste it into your local `settings.xml` file. The new `settings.xml` entries contain the generated token. Make sure you copy the contents before dismissing the window, since you won't be able to see the same token later.

To configure token-based authentication, copy the snippet containing the generated token and the required authentication entries from VB Studio and paste it into your local `settings.xml` file:

1. In the Browse view, in the Repository Configuration section, select the **Maven Settings** tab.
2. Click › to expand the **Authentication-specific Configuration** section, if it's collapsed.
3. Click the **Token** tab, if it isn't already selected.

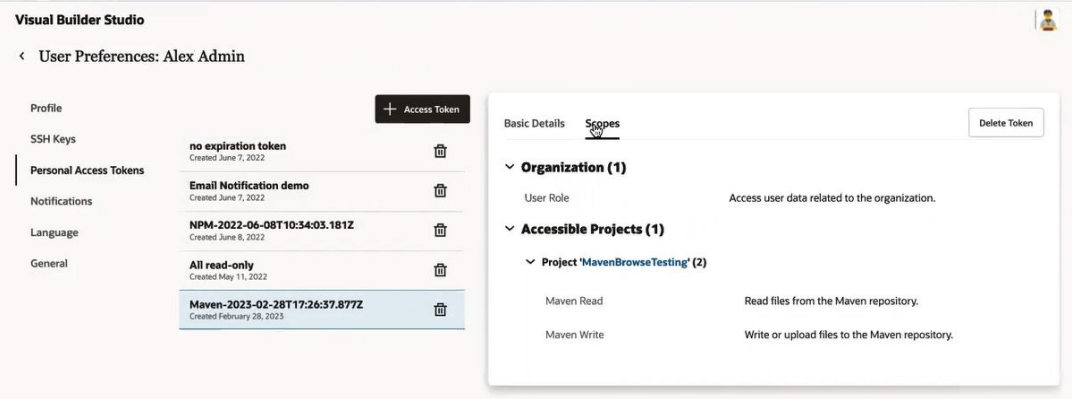



4. By default, the token that is generated grants you read-only privileges. If you want write/upload privileges as well, click **Grant write and upload privileges to the Maven repo as well.**
5. Click the **Get settings.xml Entries** button.
The **Get Maven settings.xml Entries** window displays.



> **Note:**
>
> If you are an org administrator and, after clicking **Get settings.xml Entries**, see a warning message that you reached the maximum number of tokens allowed per user, to create another token to use with Maven, you can either click **Dismiss** and then delete the existing tokens by going to the **Personal Access Tokens** tab on your **Preferences** page and using the **Delete** icon to delete some of your existing tokens or you can click **Open Access Token Settings** to go to the **Organization** page's **Properties** tab and change the maximum value for the entire organization.
>
> If you are not an org administrator and, after clicking **Get settings.xml Entries**, see the message about having too many tokens, to create another token to use with Maven, you can either click **Dismiss** and then delete the existing tokens by going to the **Personal Access Tokens** tab on your **Preferences** page and using the **Delete** icon to delete some of your existing tokens or you can ask one of the org administrators shown in the dialog to change the maximum number of tokens allowed per user.
>
> See Set Up Token-Based Authentication for more information about creating personal access tokens.

The token was retrieved from a previously-defined personal access token for the project. The token has sufficient permissions to perform Maven read and write access.




6. Click the **Copy** icon to copy the settings.
7. Click the **Dismiss** button to close the window.
8. Open your Maven `settings.xml` file in a text editor.
9. Paste the settings you copied directly into the file, in `<settings>` element.
10. Save the file.

**Perform the Profile-Specific Configuration:**

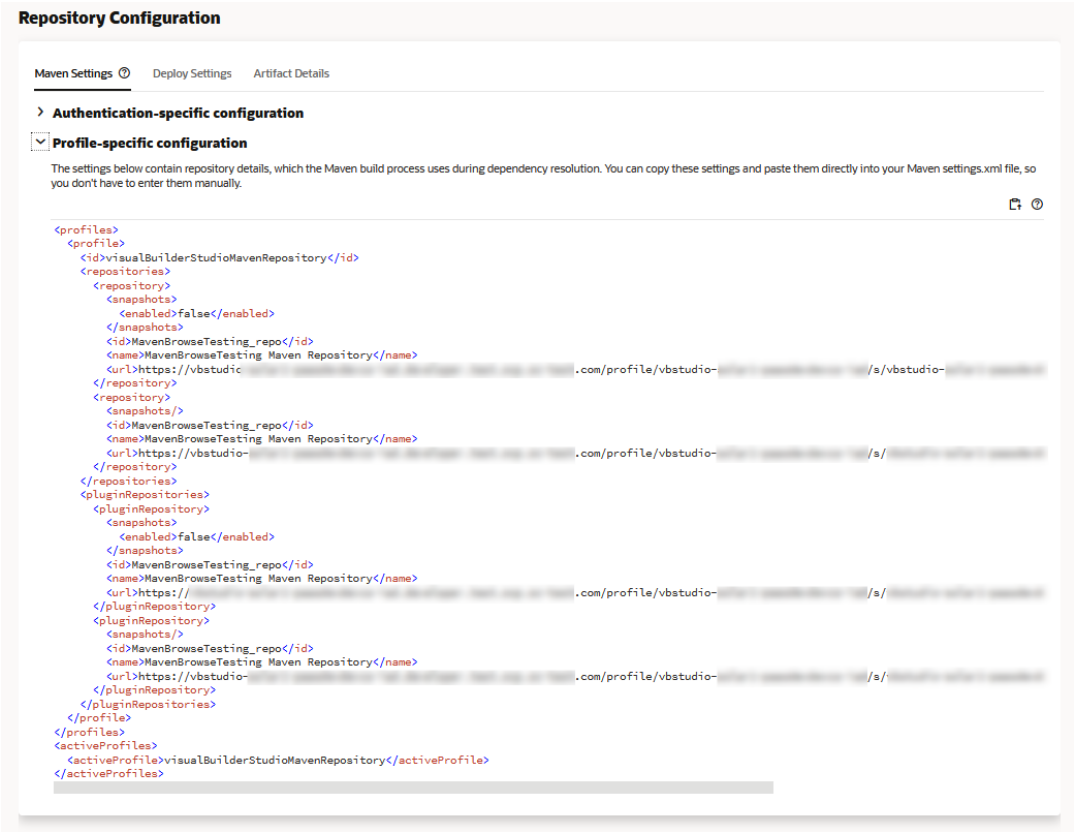
The profile-specific settings contain repository details that the Maven build process uses during dependency resolution. When you install an artifact using `mvn install`, Maven automatically tries to install any dependencies required by the artifact. Your VB Studio Maven repository will be checked first and, if the dependency isn't found there, the dependency will be resolved from the official Maven Central Repository.

The `<profile>` element in the `settings.xml` file is essentially a truncated version of the `pom.xml` file's profile element, primarily consisting of the `<repositories>` and `<pluginRepositories>` elements. The `settings.xml` file includes these elements because they're concerned with the build system as a whole, not about individual project object model settings.

> **Note:**
>
> If a profile is active from settings, its values override any equivalently identified profiles in the POM file.

You can copy the provided settings to the clipboard and paste them directly into your Maven `settings.xml` file, so you don't have to enter them manually:

1. In the Browse view, in the Repository Configuration section, select the **Maven Settings** tab.
2. Click ❯ to expand the **Profile-specific Configuration** section, if it's collapsed.




3. Click the **Copy** icon to copy the settings with the repository details the Maven build process uses to resolve dependencies.
4. Open your Maven `settings.xml` file in a text editor.
5. Paste the settings you copied directly into the file.
6. Save the file.

# Copy Distribution Management Snippets

To upload or download dependencies while running a build, add the Distribution Management snippet or the Dependency Declaration snippet to the POM file.

| Action | How To |
|---|---|
| Copy the Distribution Management snippet | 1. In the Browse view, navigate to the root directory.<br>2. In the Respository Configuration section, click the **Deploy Settings** tab and expand **Distribution Management**.<br>3. In the **Maven** tab, click **Copy** to copy the `<distributionManagement>` code snippet to the clipboard.<br>4. Open your project's `pom.xml` file in a code editor (or text editor) and paste the contents you copied under the `<project>` element. |
| Copy the Dependency Declaration snippet | 1. Browse the Maven repository and select the artifact.<br>You can also click the **Artifact Search** tab to search for and locate the artifact, then click its name to open it in the **Browse** view.<br>2. In the **Artifact Details** section, expand **Dependency Declaration**.<br>3. In the **Maven** tab, click **Copy** to copy the `<dependency>` code snippet to the clipboard.<br>4. Open the `pom.xml` file of your project in a code editor (or text editor) and paste the contents you copied under the `<dependencies>` element. |

**Tip:**

You can copy the Maven repository's URL from the distribution snippet or you can copy it from the **Project Home** page.
On the **Project Home** page, click the **Repositories** tab. In the Maven section, from the project repository's **URLs** menu, select **HTTPS** or **Webdav**, and click **Copy** to copy either the HTTPS or the WebDAV URL to the clipboard.

Use the **HTTPS** URL to connect to the Maven repository using the HTTP protocol.
Use the **Webdav** URL to connect to the repository using the Webdav protocol.

# Upload an Artifact Manually

From the Upload view, you can upload artifacts manually to the project's Maven repository without installing Maven on your computer. You must be a project owner or member to upload an artifact to the project Maven repository.

1. In the left navigator, click **Maven**.
2. On the right side of the page, click **Upload**.
3. In the **Upload Artifacts** section, use the drag-and-drop operation to drop files to the drop area, or click the select artifact files link, browse, and select the files.

4. In the table below the Upload Artifacts section, if necessary, update the **Classifier** field of the selected artifact.

   If you're uploading only one artifact, you can leave the classifier field empty. If you're uploading multiple artifacts, provide the classifier value for each artifact. The classifier helps to distinguish artifacts that were built from the same POM file but differ in their content. The classifier string is appended to the artifact name, after the version number.

   For example, if you're uploading artifacts with identical names but different extensions (such as `fileX-1.0.jar` and `fileX-1.0.pdf`), you may provide classifiers, such as `main` and `documentation` for these files. After the files are uploaded, they are renamed to `fileX-1.0-main.jar` and `fileX-1.0-documentation.pdf`.

5. After you add the artifacts, you must specify their Maven coordinates manually or from a POM file. These coordinates are used when uploading artifacts to the project's Maven repository.

   If you want to specify the artifacts' Maven coordinates manually, note the following:

   - The auto-suggest list of **GroupId**, **Version**, and **ArtifactId** are based on Maven indexes. If no index data is available, the auto-suggest list isn't displayed.
   - By default, the **Generate POM** check box is selected. The upload process deploys the artifact and generates the default POM file, `maven-metadata.xml`, and associated `sha1/md5` checksum files. If `maven-metadata.xml` already exists, it's updated.

     If you deselect the check box, the upload process deploys the artifact to the target deployment path based on Maven attributes. The POM file and the `maven-metadata.xml` file aren't generated.

6. Click **Start Upload**.

   You can track the transfer status and its progress in the drop area of the Upload Artifacts section. To cancel the upload process, click **Cancel Upload**. The upload process will also be cancelled if the page is refreshed or closed.

## Upload Artifacts Using the Maven Command-Line Interface

You can also use the Maven command-line interface to upload artifacts to the project's Maven repository.

The Maven repository URL is available on the **Project Home** page of your project. Use the `dav:` URL to upload files and the `http://` URL to view them in the browser.

Note that the credentials in `settings.xml` aren't required to access the project Maven repository when running a build. The build job has full access to the project Maven repository for uploads and downloads.

1. Download and install Maven on your local computer.

   You can download Maven from `http://maven.apache.org/download.cgi`.

2. Open `MVN_HOME/conf/settings.xml` in a text editor and make the following changes.

   a. Specify the proxy server, if necessary.

      Example:

```
<proxies>
    <proxy>
        <active>true</active>
        <protocol>http</protocol>
```

```
        <host>PROXY_URL</host>
         <port>80</port>
         <nonProxyHosts>www.anything.com|*.somewhere.com</nonProxyHosts>
      </proxy>
</proxies>
```

b. Specify a unique ID and your VB Studio user name and password to access the project Maven repository.

   Example:

```
<servers>
    <server>
        <id>remoteRepository</id>
        <username>USERNAME</username>
        <password>PASSWORD_IN_PLAINTEXT</password>
    </server>
</servers>
```

c. Specify a unique ID, name, and URL for the project Maven repository. You can copy the Maven repository URL from the **Repositories** tab of the **Project Home** page.

   Example:

```
<profiles>
    <profile>
        <id>default</id>
        <repositories>
            <repository>
                <id>remoteRepository</id>
                <name>My Remote Repository</name>
                <url>dav:https://developer.us2.oraclecloud.com/……/
maven/</url>
                <layout>default</layout>
            </repository>
        </repositories>
    </profile>
</profiles>
```

3. Open the command-line and follow these commands to upload files to the hosted Maven repository. Ensure that the `MVN_HOME/bin` path is available in the `PATH` variable.

   a. Navigate to the directory that contains the files that you want to upload.

   b. Create the `pom.xml` file, if it hasn't been created already.

      For more information about `pom.xml`, see `http://maven.apache.org/guides/introduction/introduction-to-the-pom.html`.

   c. Run the `mvn deploy` command to upload files.

      Example: `mvn deploy:deploy-file -DpomFile=c:\myproject\pom.xml -Dfile=c:\myproject\myfile.jar -DrepositoryId=remoteRepository -X -Durl=dav:https://developer.us2.oraclecloud.com/……/maven/`

# Download an Artifact Manually

You can download an artifact manually from the Maven page's **Browse** tab:

1. In the left navigator, click **Maven** .
2. If necessary, on the right side of the page, click the **Browse** tab.
3. Browse and select the artifact that you want to download.

   You can also click the **Artifact Search** tab, specify the name of artifact you're searching for, then click its name to open it in the **Browse** tab.
4. With the artifact selected, in Artifact Details, to the left of the artifact's file name, click **Download** .

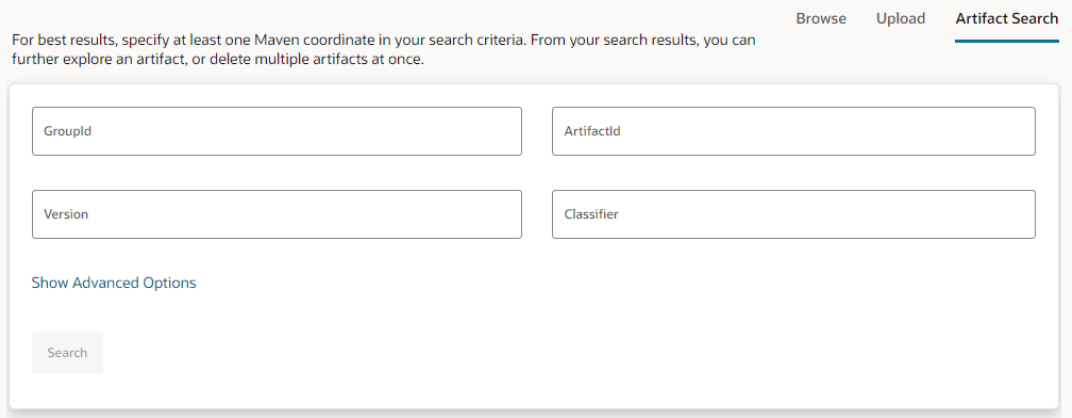
The browser will download the artifact and save it to your computer.

# Search Artifacts

You can go to the Maven page's **Artifact Search** tab and search for artifacts using the following criteria: GroupID, ArtifactID, Version, and Classifier. You can also search by size, date last updated, or build property, which you access by clicking **Show Advanced Options**.

To search for Maven artifacts:

1. In the left navigator, click **Maven** .
2. Click the **Artifact Search** tab to specify criteria to search the Maven repository.




3. Fill out one or more of the following fields:
   - Specify the group identifier in **GroupID**, for example com.companyname.project.
   - Specify the artifact identifier in **ArtifactID**, for example projectName.
   - Specify the version in **Version**, for example 1.0 or 1.0-SNAPSHOT.
   - Specify the classifier in **Classifier**, for example sources or javadoc.

   > **Note:**
   >
   > You should always specify at least one Maven coordinate to ensure precise search results.

4. Optionally, you can click **Show Advanced Options** and specify additional criteria to further narrow your search results.

- Under Size, use the up and down arrows to set the minimum and/or maximum file sizes and units (KB, MB, or GB). Set both the minimum and maximum sizes to specify a range in which to search.
- Under Updated, click **Select Date** and select the updated Since date and/or the updated Before date.
- Under Search By Build Property, click **Select Property** and select a build property, such as job name, pipeline name, result, or Git repository.
  Click **+ Build Property** to specify additional build details to narrow your search results even further.

5. Click **Search**.

The search result is grouped by Maven coordinates in the **Artifacts** tab (default setting) and by files in the **Files** tab. Use the **Sort Ascending** and **Sort Descending** arrows to sort search results by Maven coordinate, or by size or last updated date. You can optionally select and delete searched artifacts from the Artifact Search page.

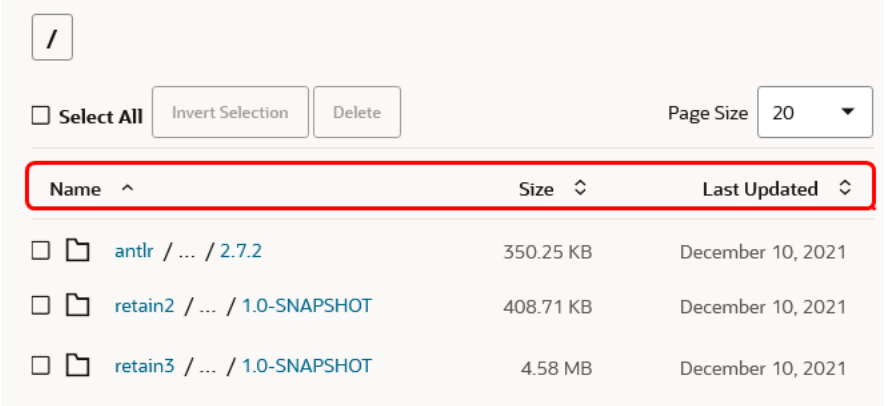
# Sort Maven Artifacts and Snapshots

From the Maven **Browse** tab, you can also sort artifacts based on name, size, and last updated date. You can also use the sorting facility to order snapshots using the same criteria in your Maven project's Snapshots folder.

To sort Maven artifacts:

1. In the left navigator, click **Maven**.
2. If necessary, on the right side of the page, click the **Browse** tab.

   The **Browse** window is displayed, showing artifacts.




3. Use the **Sort Ascending** and **Sort Descending** arrows to sort the Name, Size, and Last Updated columns in ascending or descending alphabetic, numeric, or oldest/newest order.

   The sort settings are remembered for the duration of the session, but revert back to the default setting after you open a new browser window. Being able to sort this way can be very useful when you want to analyze artifacts for retention, based on their size and how frequently they have been used, which you can see from the last updated date. The default setting is Last Updated, in descending order. With this default, the most recently modified folders/files are shown at the top.

If the list still has too many items to go through, you may want to do a basic or advanced search for a particular artifact or group of artifacts. See Search Artifacts.

# Maven Repository Administration

You can configure the Maven repository to limit the number of snapshots and overwrite an artifact if another with same groupID, artifactID, and version value is uploaded.

## Configure Auto-Cleanup for Snapshots

By default, when you upload a new snapshot of an artifact to the project's Maven repository manually or through job builds, the repository retains the old versions of the snapshots. The project owner can configure auto-cleanup so the project will automatically remove old versions when a new version is uploaded:

1. In the left navigator, click **Project Administration** .
2. Click **Repositories**.
3. In **Maven Repository**, if necessary, expand **Configure auto cleanup for Snapshot versions**.
4. Select the **Purge** check box.
5. In **Default Max Snapshots**, enter a number between 2 and 500 to specify the maximum versions to retain. By default, 2 versions of the snapshots are retained.

All changes are saved automatically when you navigate to another field. After the rule is enabled, any new upload of a snapshot will remove its older versions if the number of snapshots exceed the value in **Default Max Snapshots**.

You can also add exceptions to the auto-cleanup and **Default Max Snapshots rule** and customize the snapshot counts. The project Maven repository retains the snapshot counts of group IDs and artifacts defined in the **Customized Snapshot Counts** section and uses the default value specified in **Default Max Snapshots** for artifacts that don't have exceptions defined.

To define exceptions and customize snapshot count:

1. In the **Maven Repository** section of the Repositories page, select the **Purge** check box and configure the default auto-cleanup as described above.
2. To add an artifact group or an artifact name as an exception, click **+ Add** in the **Customized Snapshot Counts** section.
3. Specify these details:
   - **Group Id** (Required): Enter or select the Group ID of the artifact. You can select the ID from the list or start typing and then select the ID from the list of suggestions. The auto-suggest list is based on the Maven indexes. If no index data is available, the auto-suggest list doesn't display.
   - **Artifact Id** (Optional): Enter or select the Artifact ID of the artifact. You can select the ID from the list or start typing and then select the ID from the list of suggestions. The auto-suggest list is based on the Maven indexes. If no index data is available, the auto-suggest list doesn't display.
   - **Snapshot Count**: Select the number of snapshots to retain in the project Maven repository. By default, 2 snapshots are retained.

To remove an exception, on the right, click **Remove** . For a long list of exceptions, you can use the **Filter** field and enter a search coordinates criteria to see the exceptions matching the

criteria. If you enter an exception with duplicate coordinates, an error message `Unable to save. Coordinates already exists. Enter unique coordinates` appears.

## Configure Overwriting for Release Artifacts

By default, if a user or build tries to upload a Maven release artifact that has the same groupID, artifactID, and version values as an existing artifact, the upload will fail. The project owner can configure the Maven repository to allow uploaded duplicate release artifacts to overwrite existing ones.

Here's how the project owner can configure overwriting:

1. In the left navigator, click **Project Administration** .
2. Click **Repositories**.
3. In **Maven Repository**, if necessary, expand **Configure Overwrite Property for Release Artifacts**.
4. Select the **Allow** check box.

13

# Access External Docker Registries

If you use an external Docker registry, such as DockerHub or Oracle Cloud Infrastructure Registry (OCIR), you can link the registry to your project and browse its repositories and images from Oracle Visual Builder Studio (VB Studio) .

A Docker Registry is a server-side application that stores and helps you distribute Docker images. To find more about Docker images, see `https://docs.docker.com/registry/`.

## Link an External Docker Registry to Your Project

As a project owner, you can link an external Docker registry to your project:

1. In the left navigator, click **Project Administration** .
2. Click **Repositories**.
3. In **Docker Registries**, click **+ Link External Registry**.

   To view all linked Docker registries, expand **Linked External Docker Registries**.
4. In **Registry Name** and **Short Description**, enter a unique Docker registry name and a description.
5. In **Registry URL**, enter the URL of the Docker (or Docker Hub) registry. For example, you could enter `https://registry-1.docker.io`.

   The registry you specify should support the v2 API version and catalog endpoints.

   To link to OCIR, enter the registry path in the `https://<ocir_region_code>.ocir.io` format. For example: If your region is Ashburn, enter `iad.ocir.io`. See Availability by Region Name and Region Code to locate the region codes.
6. In **Authentication**, select the authentication type
   - **Basic** (default): Select and enter the basic username and password details in **Username** and **Password**.
     To authenticate and access the external Docker Hub registry, specify your Docker ID.

     To connect to OCIR, enter the username of the OCI user who can access OCIR in the`<tenancy_name>/<oci_user_name>` format. For example, `mytenancy/myociuser`. In password, enter the OCI user's auth token. See Getting an Auth Token.
   - **OAuth2**: Select and enter the long-lived access token for authentication in **Auth Token**. Don’t enter a short-lived access token.
     For more information about OAuth2 in Docker, see `https://docs.docker.com/registry/spec/auth/oauth/`.
   - **Anonymous**: Select if the registry can be accessed anonymously and doesn’t require authentication.
7. Click **Create**.

After verifying the credentials and the URL, the registry will be added to the **Linked External Docker Registries** section. You can browse its repositories and images from the **Docker** page.

After linking a Docker registry, you can also configure a job to use it while running a build. If you update the linked Docker registry's details, the updated information will be passed to any build jobs that use those details.

# Browse a Linked Docker Registry

As you browse a registry, you can see its repositories and images, download its image manifest file, copy push and pull commands, and delete an image tag.

To browse a linked Docker registry's repository, select it from the Docker Registry drop-down list.

| Action | How To |
|---|---|
| View images of a repository | Open the Docker registry repository and click the repository name. To view tags of an image, click an image tag. |
| Download the repository's image manifest | 1. Open the Docker registry repository.<br>2. In **Image Tags**, click the image tag.<br>3. On the right side, in the **Info** section of **Tag Details**, click **Download**.<br>4. Save the file to your computer. |
| Copy pull and push commands | You use the `docker pull` command to get images of a repository. To upload images to the repository, you use the `docker push` command. You can use these commands to configure a build job that connects to the Docker registry.<br>To copy the push and pull commands:<br>1. Open the Docker registry repository.<br>2. In **Image Tags**, click the image tag.<br>3. On the right side, in the **Docker Command** section, pull and push commands are displayed.<br>In the **Pull** tab, click **Copy** to copy the `docker pull` commands to the clipboard.<br>In the **Push** tab, click **Copy** to copy the `docker push` commands to the clipboard. |
| Delete an image tag | 1. Open the Docker registry repository.<br>2. In **Image Tags**, mouse over the tag you want to delete, and click **Delete**.<br>3. In the Confirm Delete dialog, select the **I understand that my selected tag will be permanently deleted** check box, and click **Yes**. |

14

# Use the Project's NPM Registry

Whenever a project is created, VB Studio provisions a private Node Package Manager (NPM) registry for the project.

You can use the built-in private NPM registry in VB Studio to do these things:

- Download, publish, and audit private Javascript packages using Node.js/NPM command line tools.
- Configure a build job in a Node.js project to download a JS package from and/or publish a JS package to the project's NPM registry, and audit the JavaScript packages using Node.js/NPM command line tools.
- Browse and search for the available NPM package in the project's NPM registry.

When you install a package using the `npm install` command, NPM automatically attempts to install any dependencies required by the package. Your project's private VB Studio NPM registry will be checked first and, if a dependency isn't found there, the request is automatically forwarded to the default remote NPM registry. You just upload your project-specific package(s), and let NPM take care of resolving any dependencies at build time.

This feature is provided for customers who have developed their own Node.js packages that they want to use in their projects at their company but they don't want to publish to npmjs.org. This gives them a way to share those private Node.js packages with the rest of their company. The project's private NPM registry is a registry of your own internal packages.

Any package, scoped or unscoped, which has been published to the VB Studio NPM registry can be downloaded. You can configure the NPM client to download these packages or you could also use the NPM page's browse feature to download the packages. If a required package isn't found in the VB Studio NPM registry, it'll be downloaded from the external NPM registry. The package found in the remote registry won't be pulled down to the VB Studio NPM registry automatically, but nothing prevents you from doing so manually.

You can access the NPM page by clicking **NPM** on the left navigator or by clicking the Project Registry link under the NPM category in the Repositories panel on the Project Home page.

## Configure Your Connection to the Project's NPM Registry

To simplify the NPM configuration process, VB Studio generates a snippet for you to use in your local `.npmrc` file. The NPM command line client needs this information to establish a connection with your project's NPM registry and the default external NPM registry. There are two ways to establish your connection: with basic authentication or with token-based authentication.

### Configure Basic Authentication

To set up your connections with basic authentication:

1. In the **npm Configuration** section on the NPM page, in the **Basic** tab, enter your VB Studio password, which VB Studio encodes and inserts into the NPM configuration snippet located below it.

Your password isn't stored. It's only used to generate a base64-encoded string for the NPM configuration snippet.

2. Copy the contents from the NPM Configuration snippet and manually paste it into your local `.npmrc` file:
   - On a Windows system, the `.npmrc` file is in `%USERHOME/.npmrc`.
   - On a UNIX system, the file location is `~/.npmrc`.

> **Note:**
>
> Don't change the `always-auth` property value to "true" in your local `.npmrc` file. The default value for the property is "false" and must not be changed. If you set the `always-auth` property to "true", the NPM client sends an Authorization header to the remote proxied public NPM Registry.

The contents of the **Default** tab configure your connection to the project's private NPM registry to push both scoped (any scoped package) and unscoped packages. The contents of the **Scoped** tab configure the connection to the project's private NPM registry for a particular scoped value only.

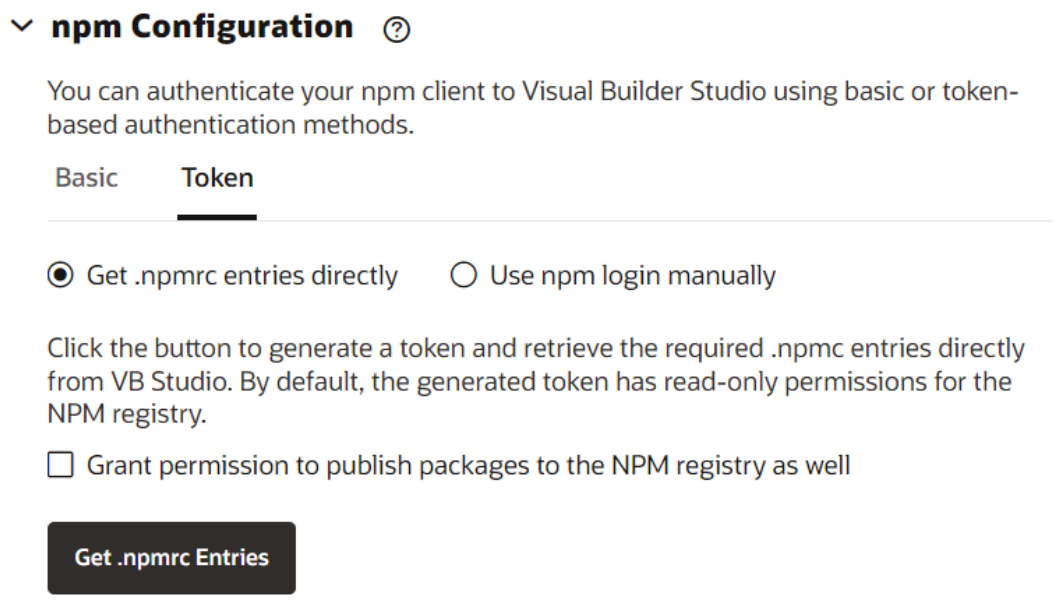
3. Save your local `.npmrc` file.

### Configure Token-Based Authentication

> **Note:**
>
> The `npm login` command won't generate a token for an NPM client if your username contains non-URL-safe special characters, such as the `@` character. This limitation means that your username cannot be an email address if you want to use the `npm login` command for token generation (and use the generated token to perform NPM operations such as `npm install`, `npm publish`, and `npm search`).
>
> To configure token-based authentication, you can simply use the **Get .npmrc entries directly** token generation option that doesn't have this limitation. VB Studio generates the necessary snippet to copy and paste into your .npmrc file so you can use the access token with your current username. The alternative is having your organization administrator generate an IDCS user (has no non-safe characters in the username) for you.

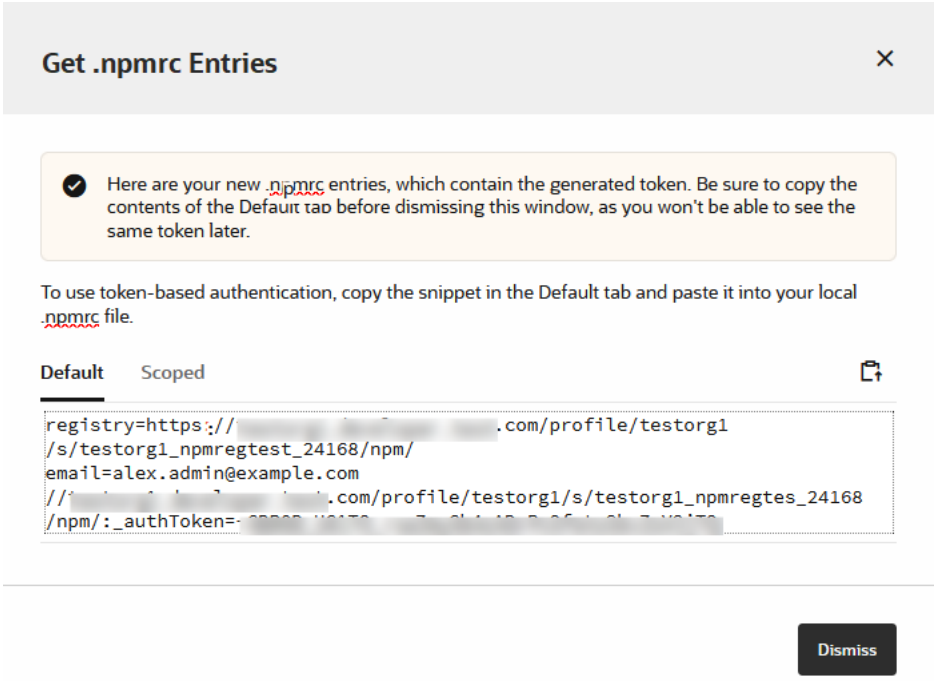
To set up your connections with token-based authentication:

1. In the NPM page's **npm Configuration** section, select the **Token** tab:




2. Choose a configuration method for token-based authentication:
   - Select the **Get .npmrc entries directly** option to generate the token and retrieve the required .npmrc entries directly from VB Studio. See step 3.
   - Select the **Use npm login manually** option only if your username contains no non-URL-safe characters. You could then supply your VB Studio login credentials when prompted after entering `npm login` on the command line. See step 4.
3. By default, the **Get .npmrc Entries** button generates a read-only token. If you want write/upload access also, click **Grant permission to publish packages to NPM registry as well**.
4. After selecting **Get .npmrc entries directly**, click **Get .npmrc Entries**.

The **Get .npmrc Entries** dialog displays.



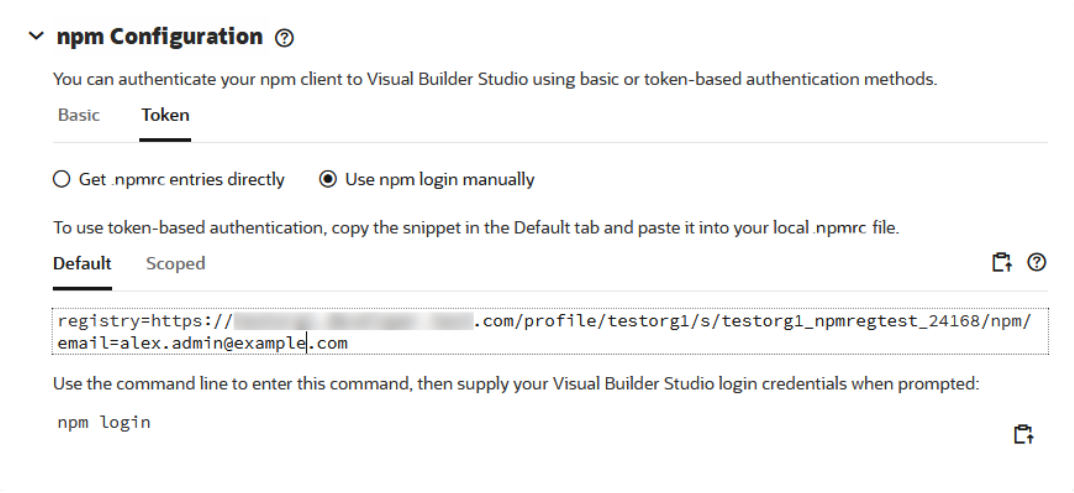

a. Click to copy the generated .npmrc entries in the **Default** tab (to push both scoped and unscoped packages) or the **Scoped** tab (to push a particular scoped package only).

b. Paste the content you just copied into your local .npmrc file and save it.

c. Click **Dismiss** to close the dialog.

> **Note:**
>
> If you are an org administrator and, after clicking **Get .npmrc Entries**, see a warning message that you reached the maximum number of tokens allowed per user, to create another token to use with NPM, you can either click **Dismiss** and then delete the existing tokens by going to the **Personal Access Tokens** tab on your **Preferences** page and using the **Delete** icon to delete some of your existing tokens or you can click **Open Access Token Settings** to go to the **Organization** page's **Properties** tab and change the maximum value for the entire organization.
>
> If you are not an org administrator and, after clicking **Get .npmrc Entries**, see the message about having too many tokens, to create another token to use with NPM, you can either click **Dismiss** and then delete the existing tokens by going to the **Personal Access Tokens** tab on your **Preferences** page and using the **Delete** icon to delete some of your existing tokens or you can ask one of the org administrators shown in the dialog to change the maximum number of tokens allowed per user.
>
> See Set Up Token-Based Authentication for more information about creating personal access tokens.

5. After selecting **Use npm login manually**, the **npm Configuration** section in the **Browse** tab displays this information.




   a. Click to copy the snippet in the **Default** tab (to push both scoped and unscoped packages) or the **Scoped** tab (to push a particular scoped package only).

   b. Paste the contents you just copied into your local `.npmrc` file and save it:

      - On a Windows system, the `.npmrc` file is in `%USERHOME/.npmrc`.
      - On a UNIX system, the file location is `~/.npmrc`.

   c. Click to copy the provided `npm login` command (for all scoped and unscoped packages) or `npm login --scope=@scope<scope>` command (for scoped packages only).

   d. Paste the command you just copied to the command line and run it. Modify the scope, as needed and supply your VB Studio login credentials when you're prompted for them.
   VB Studio generates a personal access token and, once the NPM client receives the response that contains the generated token, it adds that to your `.npmrc` file. This process generates a non-expirable token.

      You can examine the details for the token in the **Personal Access Tokens** tab under the user **Preferences**. Select the token and view the information in the **Basic Details** and **Scopes** tabs.

The token remains valid until the you change your password. You can also invalidate a single token by logging out on a machine where you're logged in with that token.

# Create and Manage a Project's Remote NPM Registry Connection

If your project users access the remote default NPM registry frequently, you can create a pre-defined connection for them. Project users can then configure a job and use the connection to access as well as publish scoped and unscoped packages in the default NPM registry while running builds.

To create a connection, you'll need the credentials of a user who can access the remote NPM registry.

You must be a project owner to add and manage the remote NPM registry:

| Action | How To |
|---|---|
| Add a remote NPM registry connection | 1. In the left navigator, click **Project Administration** .<br>2. Click **Builds**.<br>3. Click the **NPM Connection** tab.<br>4. Click **Add NPM Connection**.<br>5. In the Create NPM Connection dialog box, in **Connection Name**, enter a unique name.<br>6. In **Username** and **Password**, enter the credentials of a user who has access to the remote NPM registry.<br>7. In **Registry URL**, enter the Registry URL for the default remote NPM registry, for example https://registry.npmjs.org.<br>8. Click **Create**. |
| Edit a connection and change the connection's user credentials or provide another Registry URL | 1. In the left navigator, click **Project Administration** .<br>2. Click **Builds**.<br>3. Click the **NPM Connection** tab.<br>4. Click **Edit** for the NPM connection you want to edit.<br>5. In the Edit NPM Connection dialog box, if necessary, enter the credentials of a user who has access to the remote NPM registry.<br>In **Registry URL**, if necessary, enter the URL for the default remote NPM registry.<br>6. Click **Update**. |
| Delete the connection | 1. In the left navigator, click **Project Administration** .<br>2. Click **Builds**.<br>3. Click the **NPM Connection** tab.<br>4. Click to delete the NPM connection.<br>5. In the Delete NPM Connection dialog box, click **Delete**. |

# Configure a Job to Connect to the NPM Registry

You can use a predefined connection to set up a job to connect to the project's NPM registry or to an external NPM registry:

1. Open the job's configuration page.
2. In the **Configure** view, click the **Before Build** tab.
3. Click **Add Before Build Action**, select **NPM Registry Connection**, and make sure that the build action is **Enabled**.
4. Select the registry to use:
   - To access scoped or unscoped packages in your project's private NPM registry, select the **Use project's NPM registry** checkbox and the URL will be populated automatically.

If you want the build to use the project's custom configuration file (`.npmrc`), specify its path (for example, *`path/to/my/project/`*`.npmrc`) in the **Custom .npmrc** field.

- To access packages in an external NPM registry, select the **Use External NPM registry** checkbox, set the toggle to **Use existing connection**, and select a pre-defined connection from the dropdown, if your project owner has created a connection. (If a pre-defined connection isn't available, you won't be able to set one up from this page so you'll need to contact your project owner to define a connection for you to use.) Once the registry has been selected, the configured registry URL and the credentials are displayed in the read-only fields.
  If you want the build to use the project's custom configuration file (`.npmrc`), specify its path (for example, *`path/to/my/project/`*`.npmrc`) in the **Custom .npmrc** field.

5. To add NPM commands that use the NPM registry connection and run when the job is executed:
   a. Select the **Steps** tab in the job configuration page.
   b. Click the **Add Step** dropdown.
   c. Select **Common Build Tools**, and then select **Unix Shell**.
   d. Specify the NPM command to run, for example `npm install --verbose`, in that step.
6. Click **Save**.

# Use the NPM Command Line with the Project's NPM Registry

You can use the following NPM commands either on the NPM command line or as a command or series of commands in a UNIX shell build step:

| Command | Description |
|---|---|
| `npm install` | Installs a package, and any packages that it depends on, in the project's NPM registry.<br>There are several ways to use the `npm install` command, so refer to npm-install to learn about the different variations. When this command is used with the project's NPM registry, packages are always searched for in the project's NPM registry first. |
| `npm publish` | Publishes a package to the project's NPM registry, so it can be installed by name. By default, the standard `npm publish` command publishes packages to the public registry. The default behavior can be overridden by specifying a different default registry or using a scoped package name.<br>See npm-publish. |
| `npm unpublish` | Removes a package version from the project's NPM registry, deleting its entry and removing the tarball. If you want to encourage users to upgrade, consider using the `npm deprecate` command instead.<br>See npm-unpublish. |

| Command | Description |
|---|---|
| `npm audit` | Generates an audit report that summarizes all known security vulnerabilities in your NPM packages and dependencies, provides the necessary NPM commands and recommendations that will fix these vulnerabilities once they are applied, and, when used with the `npm audit fix` command, will try to apply these recommendations automatically, wherever possible.<br>See npm-audit and Check for Security Vulnerabilities in your Project's NPM Packages and Dependencies. |
| `npm deprecate` | Updates the project's NPM registry entry for a specific version or a range of versions of a package and provides a deprecation warning to everyone who attempts to install the deprecated package. You must be the package owner to use this command.<br>The format of the command is:<br>`npm deprecate <pkg>[@<version>] <message>`<br>See npm-deprecate. |
| `npm dist-tag` | Adds, removes, and enumerates distribution tags on a package.<br>See npm-dist-tag. |
| `npm search` | Searches the project's NPM registry for packages that match the search terms. The command searches through package metadata for all files in the project's NPM registry. Note that the search API performs local searches only (searches the private VB Studio NPM registry only). It doesn't search and return results from the remote proxied public NPM registry.<br>See npm-search. |
| `npm login` | Allows you to log in to NPM, creating/modifying entries in the `~/.npmrc` file for authentication. Note that at least one of the two ways to log in must be configured, that is, you must either provide the required fields (username, password and email) using environment variables or pass them as command line arguments, or both.<br>See npm-cli-login. |

| Command | Description |
|---|---|
| `npm logout` | Logs out of the registry.<br>If you're logged into a registry that supports token-based authentication, this command directs the server to end this token's session, invalidating the token everywhere you're using it, not just for the current environment.<br>If you're logged into a registry that uses basic authentication (username and password), this command clears the credentials in your user configuration. In this case, it *only* affects the current environment.<br>If `--scope` is provided, the command finds the credentials for the registry that's connected to that scope, if one was set.<br>See npm-logout. |

# Publish JS Packages to VB Studio's NPM Registry

JS packages can be uploaded manually to the project's NPM registry from the **Publish** tab on the NPM page. Alternatively, JS packages can be uploaded to the registry by executing a build job or pipeline.

From the **Publish** tab, you can drag and drop or select a JS package (a gzipped tarball, in other words a .tar, .tar.gz, or .tgz file) that you want to publish and initiate the publishing operation. Multiple versions of the same JS package or multiple different JS packages can be uploaded in one operation as well. You can monitor the status of ongoing as well as completed operations.

As part of the publishing operation, the package metadata document will be generated, if one doesn't already exist, or it will be updated, if one already exists. The version metadata document will be generated. After a successful upload operation, the generated or updated metadata documents can be viewed in the NPM page.

See Publish Packages with a Job or Pipeline for more information about using a build job or pipeline instead of the UI to upload packages to the NPM registry.

## Select and Publish Packages from the NPM Page

Here's how you can use the NPM page to select and publish a JS package (or multiple packages):

1. Open the project's NPM page.
2. Open the **Publish** tab.
3. Drag and drop the Node.js package(s) you want to publish to the upload area or click in the area to bring up a file browser for locating and selecting the package(s) to upload.

   The upload infrastructure expects a tarball (or multiple tarballs) created using Gzip. In other words, it expects a file (or files) with a .tar, .tar.gz, or .tgz file extension. The names, sizes, and upload statuses of the file(s) to be uploaded are displayed in an area below the upload area.
4. Under the **Specify NPM identifiers** section, select the package layout used for the upload operation.

The options are:

- Let VB Studio derive the internal package file(s) from the internal package name(s) of the selected tarball(s).

  This is the default option for selections with multiple tarballs. The tarball's (or tarballs') internal package file(s) are used to determine the target upload path.

- Specify the NPM identifiers that will be used to rename the packages manually and VB Studio will derive the layout from the **Scope**, package **Name** (required), **Version** (Required), and **Tags** fields.

  With this option, the target upload path is determined from the specified fields' values. This option is useful for publishing a downloaded tarball under a custom folder layout created with the values of the specified NPM identifiers.

If you selected a single file to upload, you can override the default automatic option and manually specify the identifiers, but if you selected multiple files, the manual option isn't available.

5. **Generate or update metadata files** is selected by default, but you can deselect it.
6. If you selected to manually define the fields, **Target path where the selected packages will be published** will display the package's new path using the specified scope and name NPM identifiers.
   Verify that it is what you expected (and wanted).
7. Click **Publish to NPM**.

## Monitor the Upload and View the Published Package(s)

VB Studio initiates the upload process and progressively shows the status as it occurs.

1. After you click **Publish to NPM**, the tarball files are scanned for viruses.
   If any tarball files are determined to be unsafe, the **Antivirus scanning of attachment failed** error message is displayed.
2. If the tarball files are determined to be safe, the tarball content is examined, looking for the `package.json` file, which is parsed if found.
   If the package.json file isn't found in the tarball file, or if the internal `package.json` file cannot be parsed, if the option to **Use Selected Tarball(s)'s internal package file** was selected for the package layout option, the **Unable to find or parse correctly the package archive (package.json) within the tarball content** error message will be displayed.
3. The project's NPM registry is checked to see whether it already has same version of package.
   If it does, the **Package already exists and cannot be overwritten** error message will be displayed.
4. If the **Generate or update metadata files** option was selected, the NPM registry is checked for an existing package metadata file (or files).
   If package metadata is already available, that metadata is retrieved and the package metadata is updated with the new version. Otherwise, a new package metadata is created. Creating and updating this metadata is based on the tarball file's internal package file and the user-specifed value for the custom layout option.

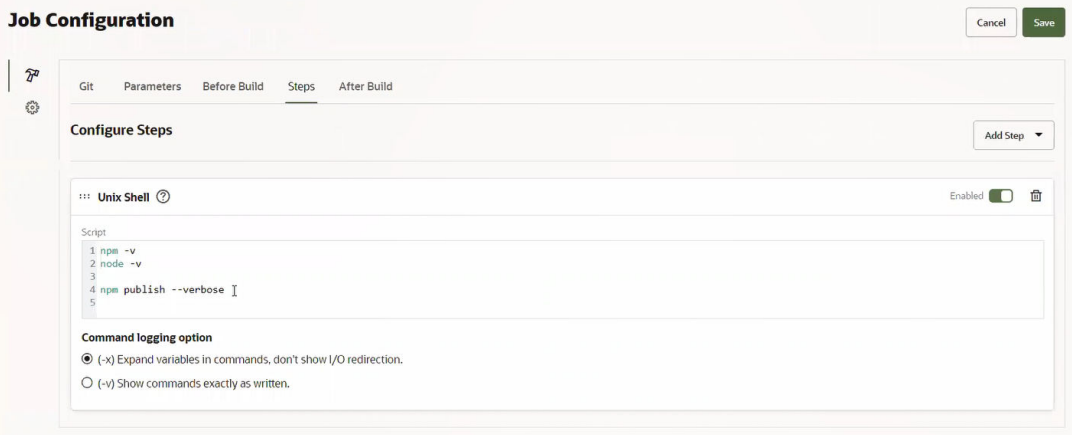
   The new version of the metadata file is created, based on the tarball file's internal package file and the user-specifed value for the custom layout option.
5. The tarball is put into the project's NPM registry.

6. The aggregated package metadata is put into the project's NPM registry and the aggregated package metadata is uploaded.
7. The version metadata is added to the project's NPM registry.
   The status(es) of successful, partial, and failed upload(s) are displayed.

## Publish Packages with a Job or Pipeline

You can create a build job or pipeline to publish a JS package (or multiple packages) to the project's NPM registry: Here's how to configure a build job that does that:

1. From the **Jobs Overview** page, click **Configure** to bring up the **Job Configuration** page.
2. In the **Configure** view, select the **Git** tab, and configure Git. Select the repository and the branch or tag.
3. In the **Before Build** tab, click **Add Before Build Action** and select **NPM Registry Connection**. Select **Use project's NPM registry**. Notice that the Registry URL is displayed, but is grayed out.
4. In the Steps tab, add a Unix Shell step with a script, similar to this:




> **Tip:**
>
> You may want to configure an additional Unix Shell build step that audits the package's dependencies, sets a minimum audit threshold level for failure, and fixes vulnerabilities that were discovered as a result of the audit. See Check for Security Vulnerabilities in your Project's NPM Packages and Dependencies for more information about implementing these options.

5. In the **Settings** view, select the **Software** tab.
   Make sure that the selected Software template lists one of these Node.js versions in the **Available Software** section:
   - Node.js 18 (Version 18.18.1)
   - Node.js 20 (Version 20.8.0)

   If the job doesn't use one of these Node.js versions, the build information won't be displayed in **Build Details** on the NPM page's **Package Details** section.
6. Click **Save**.

A build job that has been configured in this manner can be included in a pipeline and, after it has been executed successfully, the pipeline build information will be displayed in the **Build Details** section in the NPM page's **Package Details** section.

# Check for Security Vulnerabilities in your Project's NPM Packages and Dependencies

A security audit is a process that assesses package dependencies for security vulnerabilities. Security audits help you protect those who use your packages by helping you find and fix known vulnerabilities in dependencies that could cause data loss, service outages, unauthorized access to sensitive information, or other issues.

The `npm audit` command submits a description of the dependencies configured in your project's package(s) to your project's built-in NPM registry and asks for a report of known vulnerabilities. If any vulnerabilities are found, the impact and appropriate remediation are calculated. The audit report includes the affected package name, vulnerability severity and description, path, and other information, and, if available, commands to apply patches to resolve vulnerabilities.

If updates for identified security vulnerabilities are available, you can either:

- Run `npm audit fix` to apply remediations to the package tree automatically.
- Run the recommended commands individually to manually install updates to vulnerable dependencies.

If there are no patches available for the identified vulnerabilities, the audit report will provide information about the vulnerability to help you investigate further.

If no security vulnerabilities were found, this means that packages with known vulnerabilities were not found in your package dependency tree. However, since the advisory database can be updated at any time, you should regularly run `npm audit` manually (see Run npm audit Manually), or add a build step with `npm audit` to your continuous integration process.

It's also worthwhile to note that by default `npm audit` automatically runs whenever you install a package with `npm install` but, if you prefer, you can turn off `npm audit` on package installation:

- To turn off `npm audit` when installing a single package, use the `--no-audit` flag:

  ```
  npm install <package-name> --no-audit
  ```

- To turn off `npm audit` when installing all packages, set the `audit` setting to `false` in your user and global npmrc config files:

  ```
  npm set audit false
  ```

### Run npm audit Manually

Here's how to manually run `npm audit`:

1. On the command line, type `cd path/to/your-package-name` and navigate to your package directory, then press **Enter**.
2. Make sure that your package contains `package.json` and `package-lock.json` files.
3. Type `npm audit` and press **Enter**.

4. Review the audit report and run the recommended commands or investigate further, if needed.

### Understand npm audit Exit Codes

The `npm audit` command exits with a 0 exit code when no vulnerabilities are found or a non-zero code when any vulnerability is found. The `npm audit fix` command exit with a 0 exit code if no vulnerabilities are found or if the remediation is able to successfully fix all vulnerabilities. If vulnerabilities are found, the exit code depends on the audit-level configuration setting. In CI environments, you may want to include the `--audit-level` argument to specify the minimum vulnerability level that will cause the command to fail. This option doesn't filter the report output, it simply changes the command's failure threshold.

### Examples

Scan your project for vulnerabilities and just show the details, without fixing anything:

```
$ npm audit
```

Do a dry run to get an idea of what audit fix will do, and also output install information in JSON format:

```
$ npm audit fix --dry-run --json
```

The `dry-run` option indicates that you don't want NPM to make any changes and that it should only report what it would have done. This can be passed into any of the commands that modify your local installation, such as install, update, uninstall, pack, and publish. The `json` option indicates whether or not to output JSON data, rather than the normal output.

Scan your project for vulnerabilities and automatically install any compatible updates to vulnerable dependencies:

```
$ npm audit fix
```

Fail an audit only if the results include a vulnerability with a level of moderate or higher:

```
$ npm audit --audit-level=moderate
```

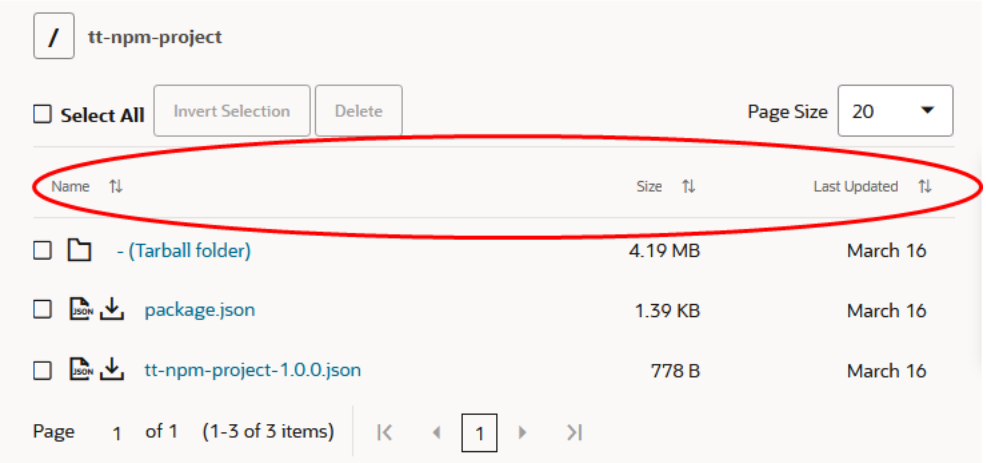
The `audit-level` option indicates the minimum level of vulnerability ("info", "low", "moderate", "high", "critical", or "none") for `npm audit` to exit with a non-zero exit code.

# Browse and Search Packages in Your Project's NPM Registry

From the NPM page, you can browse the project's NPM registry or search for and locate packages in the registry.

With **Browse** selected, you can display and select packages, then view details about the packages in the **Info** section, under **Registry Details**. You can select just one package, or use **Select All** to select all packages. You can also select **Invert Selection** or **Delete**.

When the list of items is long, you can sort by **Name**, **Size**, and **Last Updated**, in ascending or descending order:

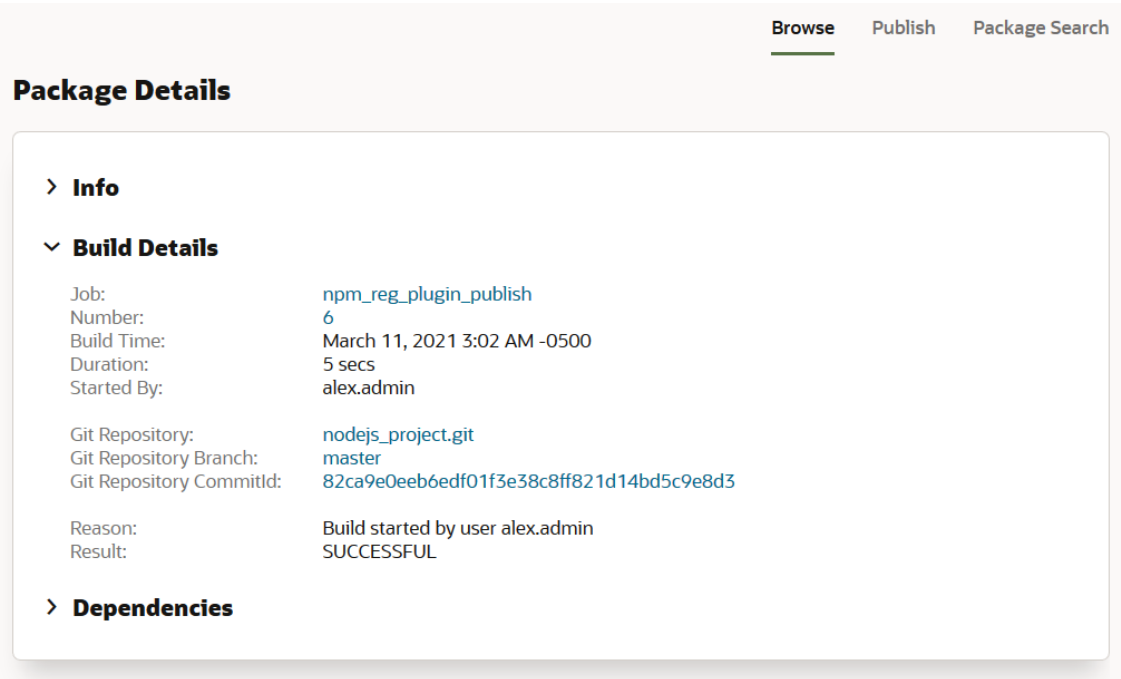



The sort settings will be remembered for the duration of the session, but will revert back to the default setting after a new browser window is opened. Being able to sort this way can be very useful when you want to analyze artifacts for retention, based on their size and how frequently they have been used, which you can see from the last updated date.

You can select a file and see details in the panel on the right. The panel can display up to three categories of details, including **Info** at the top and **Dependencies** at the bottom, with tabs for **Default**, **Optional**, **Dev**, and **Peer** views. If there are too many dependencies to list in the display, click the **Download**, then open and examine the file.

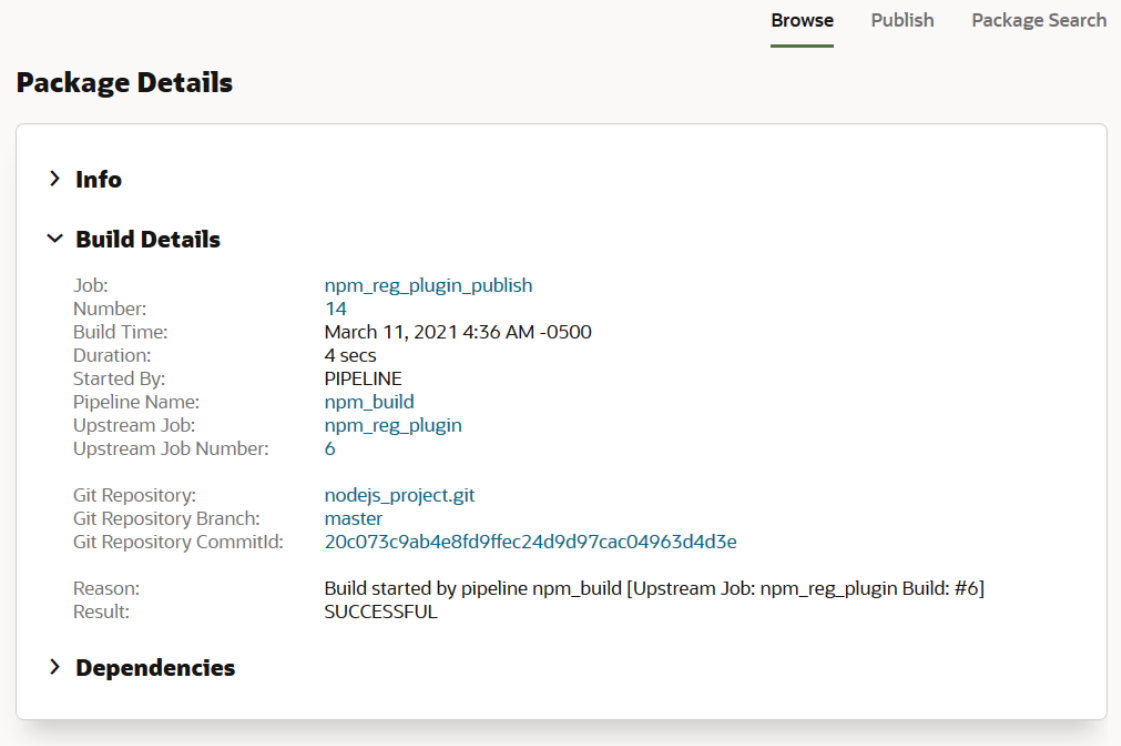
If you uploaded the package by executing a build job or a pipeline, a third category, **Build Details**, will be displayed between the other two.

Here's an example showing build details for a package that was uploaded by a user-initiated job:



> **Note:**
>
> You can only view and examine build details for published Node.js packages in the NPM page if you select a build executor template that contains one of the following Node.js versions from the software catalog:
>
> - Node.js 20 (Version 20.8.0)
> - Node.js 18 (Version 18.18.1)
> - Node.js 17 (Version 17.9.1)
> - Node.js 16 (Version 16.17.0)
> - Node.js 14 (Version 14.20.0)

Here's an example showing build details for a package that was uploaded by a pipeline:




If the selected package has been deprecated, its details will be displayed under Deprecation Details, in the Package Details panel. These details could include a message provided by the author when the package is being deprecated, such as "Version no longer supported because it used outdated code".

You can use the page's basic search capability by entering a search string, then using **All** or **Files** to filter the results.

You can use Package Search to use the **Name**, **Keywords**, **Version**, or **Author** fields to search for a package in the registry. The asterisk (*) can be used as a wildcard character. To improve performance, specify your search criteria as precisely as possible.

To narrow your search results, select **Show Advanced Options**, specify additional search criteria, then click **Search**. These are the options you can specify:

- **Size**: Specify the **Min** and/or **Max** package sizes to search. The sizes could range from KB to GB.
- **Updated**: Use **Since** and/or **Before** by entering the date in mm/dd/yy format or by selecting it using the calendar.
- **Search By Build Property**: Click **Select Property** and specify a build property, such as job name, pipeline name, result, or Git repository.
  Click **+ Build Property** to specify additional build details to narrow your search results even further.

Search results can be returned under two views, packages or files. Use the **Sort Ascending** and **Sort Descending** ⌃⌄ arrows to sort search results by NPM identifier, or by size or last updated date.

To delete a file or package that was returned by the search, select the checkbox next to the package or file you want to delete, then click **Delete**. When you're asked to confirm, press **Yes, Delete All**.

15

# Send Notifications to External Software Using Webhooks

A project owner can create and configure a webhook to send notifications to remote services and applications about Oracle Visual Builder Studio (VB Studio) events such as a Git push, an issue update, a merge request update, or a build completion.

When you create a webhook, you specify a webhook provider. When an event occurs and the webhook triggers, the webhook provider processes the event, sets the properties used to generate the HTTP request, and dispatches the HTTP request to the target service.

## Slack

Slack is cloud-based team collaboration software. Using a Slack Webhook, you can configure VB Studio to send events and activities notifications to a Slack channel. To find more about Slack, see `https://slack.com/`.

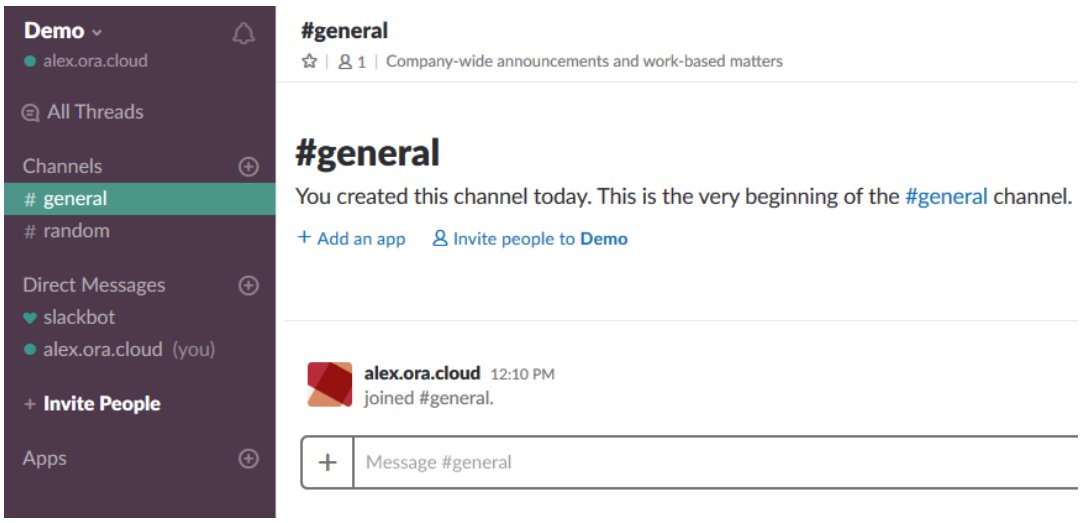
To send notification to a Slack channel, get its incoming webhook URL. Then, create a VB Studio webhook and add the incoming webhook URL to the webhook.

### Get the Slack Channel's Incoming Webhook URL

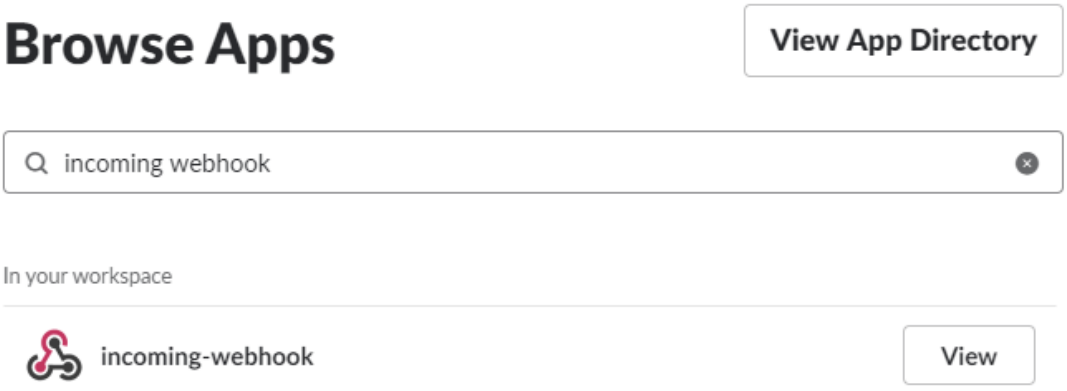
You must be the workspace owner to get the incoming webhook URL.

1. Open the Slack workspace in a web browser or the Slack app.

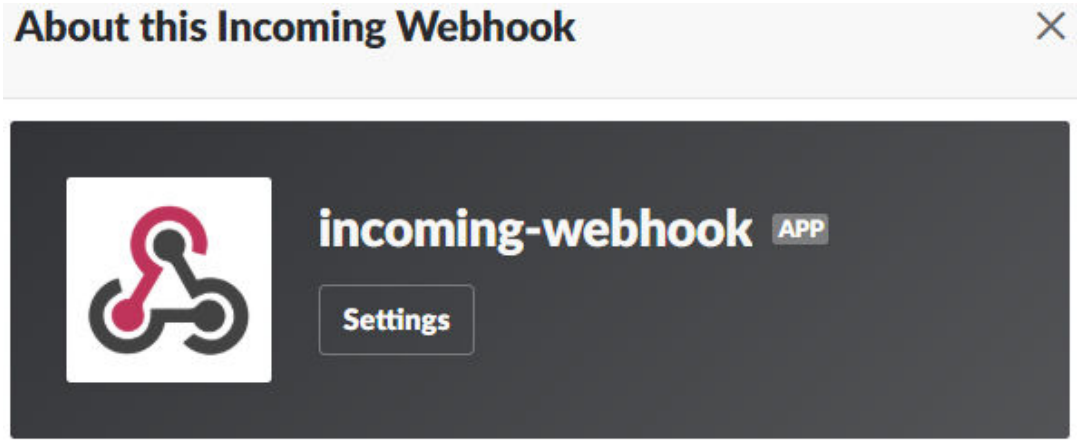
   For example, this image shows a Slack workspace called Demo.




2. In the left navigator, click **Apps**.
3. In the search box on the Browse Apps page, enter `incoming webhook`.




4. If **incoming-webhook** is pre-installed, click **View**, and then click **Settings**.

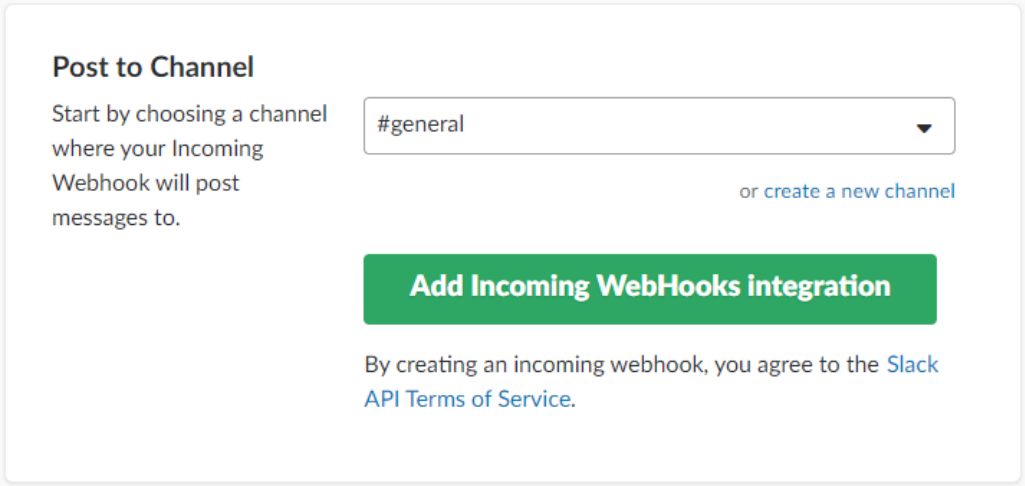



If it isn't installed, then install it and configure it.

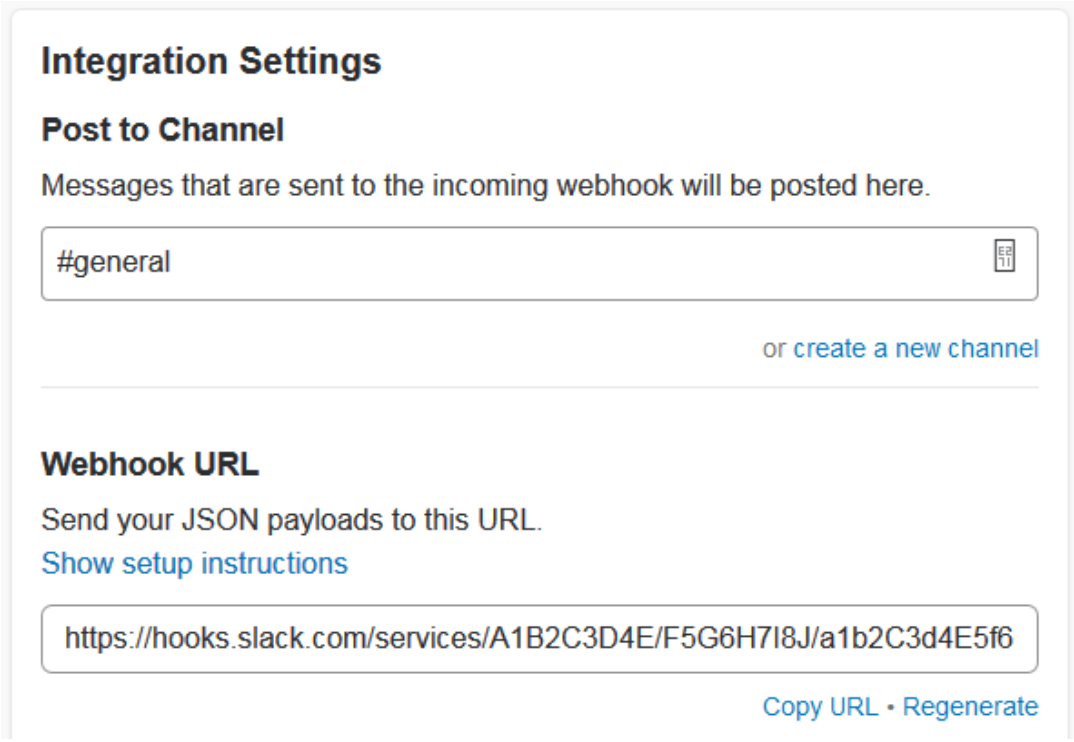
a. Click **Install**.

b. On the Incoming WebHooks page, click **Add Configuration**.

c. From the **Post to Channel** list, select the channel, and click **Add Incoming WebHooks integration**.




5. In **Integration Settings**, from the **Post to Channel** drop-down list, select the channel. In **Webhook URL**, click **Copy URL**.




6. Scroll down to the bottom of the page and click **Save Settings**.

## Configure a Slack Webhook in VB Studio to Send Event Notifications

The Slack webhook is a outgoing webhook that sends VB Studio event notifications to a Slack channel.

The project owner can create and configure a webhook:

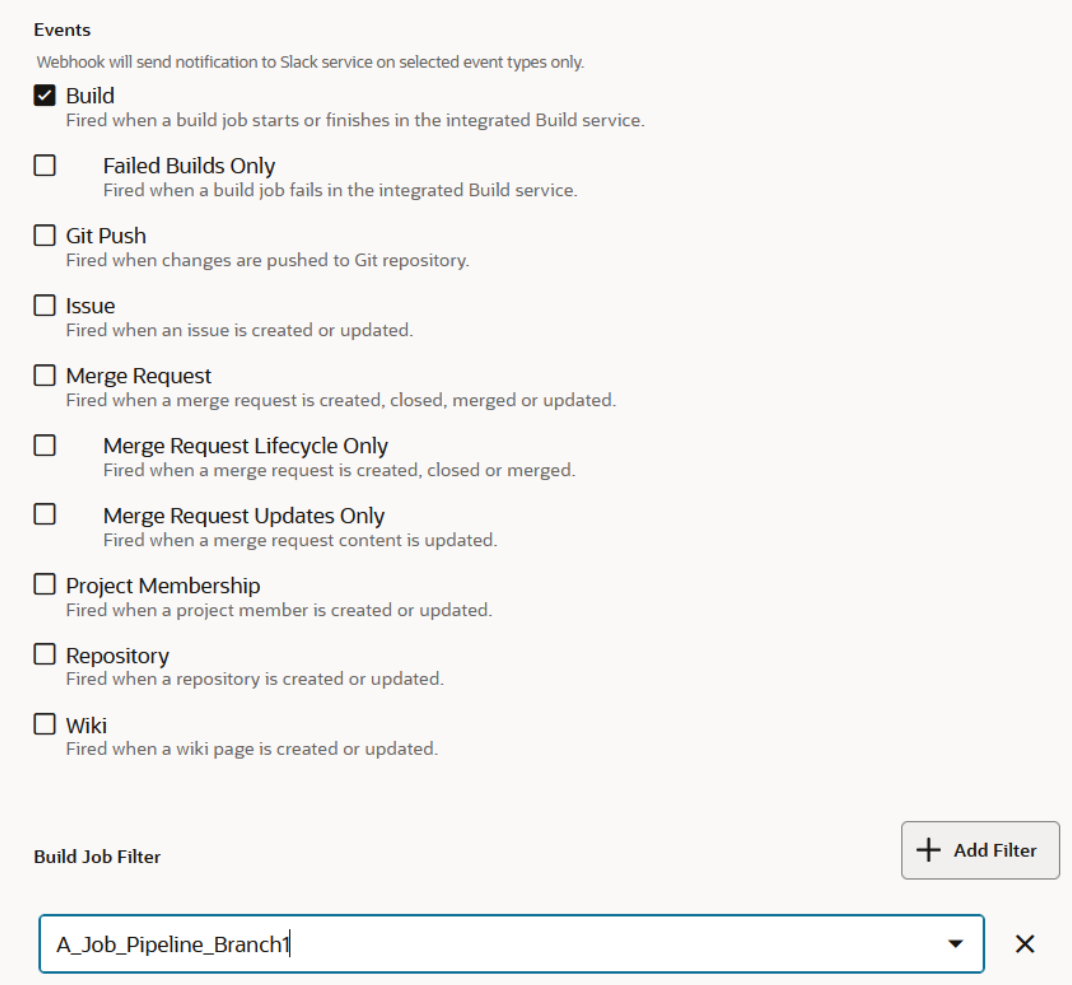
1. In the left navigator, click **Project Administration** ⚙.
2. Click **Webhooks**.
3. Click **+ Create Webhook**.

4. From **Type**, select **Slack**.
5. In **Name**, enter a unique name.
6. In **URL**, enter or paste the Slack channel's incoming Webhook URL.

   Make sure it's in the `https://hooks.slack.com/services/...` format.
7. In **Events**, select the type(s) of events that trigger the webhook and then, under the filter, select from the item(s) displayed or use a regular expression to make a more complex selection (a range or ranges, perhaps) that involves multiple items that match a pattern. See Glob Pattern Reference for Matching Branch, Job, and Pipeline Names for detailed information about the syntax for regular expressions.

> **Note:**
>
> The Glob Pattern is only available for Repository/Branch filtering with Git Push and Merge Request event types. The filter for Build event types doesn't currently support using the Glob pattern.

Here's what you can choose from.




Let's say you want to add event notifications from a build job that is triggered by an SCM commit to a selected repository (configured in the job configuration page's Git tab). To do this, you'd select **Build** then, under Build Job Filter at the bottom of the page, click **+ Add**

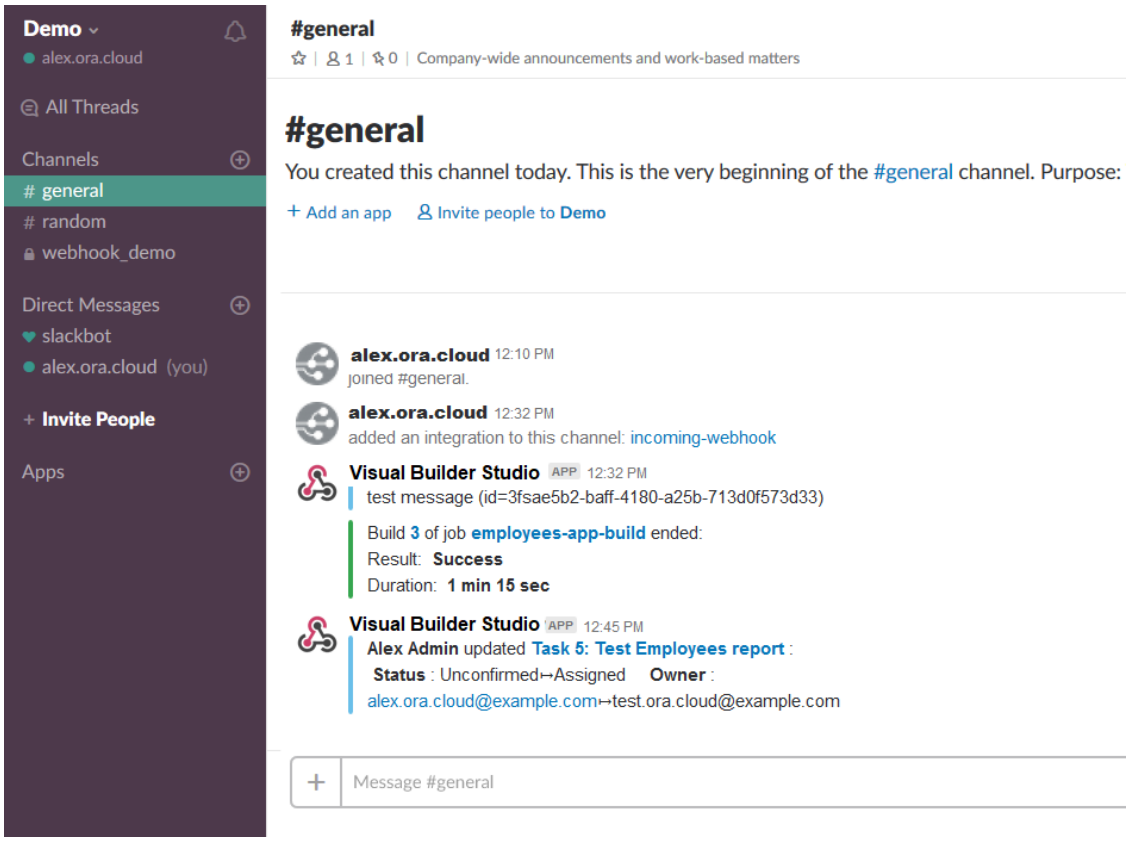
**Filter** and select the job whose event messages will be streamed to the channel you specified in step 6.

Since the build job was triggered by an SCM commit, the Slack message will include information such as commit message(s), the author's name, the duration, and other details. This information helps identify what was being processed and who was responsible for it.

If you select **Git Push** or **Merge Request**, click **+ Add Filter** in **Repository/Branch Filter**, and use the menu to select the repository and branch to track.

8. Click **Create**.
9. To test the webhook, click **Test**.
10. Click **Done** when you're satisfied that it works.

When VB Studio events happen, notifications are sent to the Slack channel.




# PagerDuty

PagerDuty is an incident management platform that enables you to send notifications via email, push, SMS, and phone. Using the PagerDuty webhook, you can send notifications to your PagerDuty service about events in VB Studio. When the PagerDuty service receives notifications from VB Studio, it can redirect those notifications via email, push, SMS, and phone. To find more about PagerDuty, see `https://www.pagerduty.com/`.

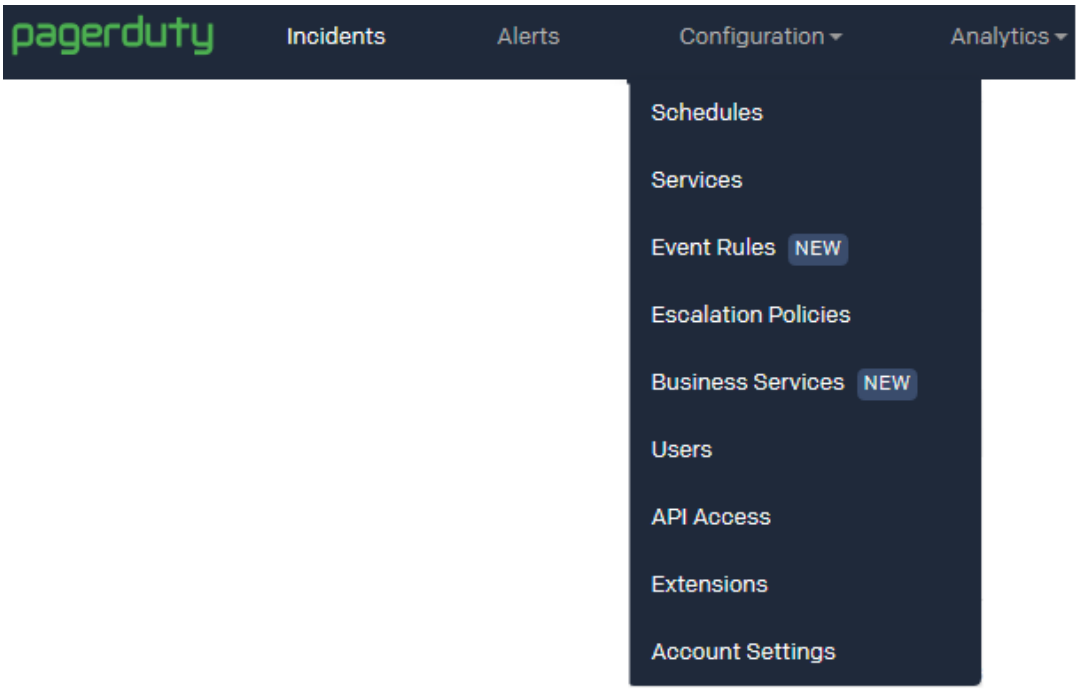
To send notifications to PagerDuty, set up your PagerDuty account to receive notifications and create a VB Studio webhook.

# Set Up the PagerDuty Account

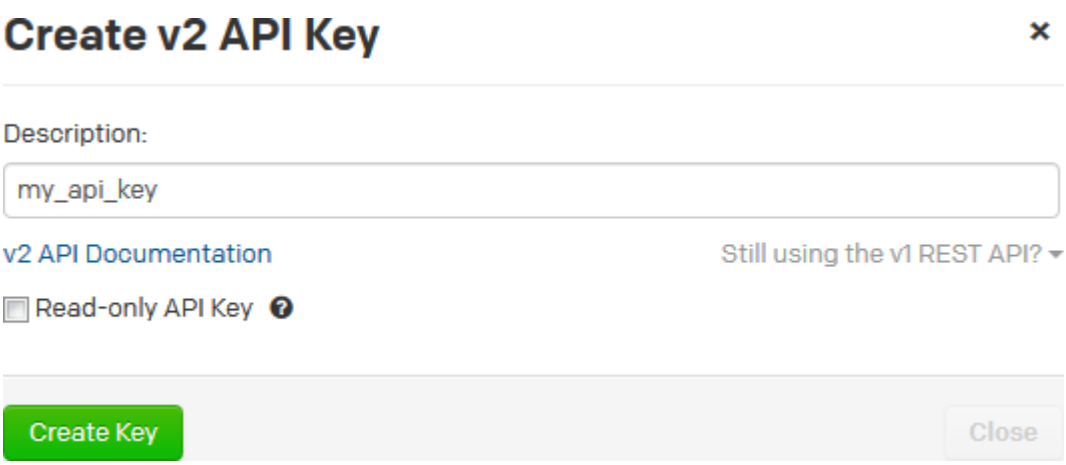
To set up PagerDuty, create an API key, add services, and add users who would receive PagerDuty notifications .

You must be the account owner or assigned the PagerDuty Admin role to set up the PagerDuty account.

1. Log in to PagerDuty as the account owner or administrator.
2. To set up the API key, from the **Configuration** menu, select **API Access**.

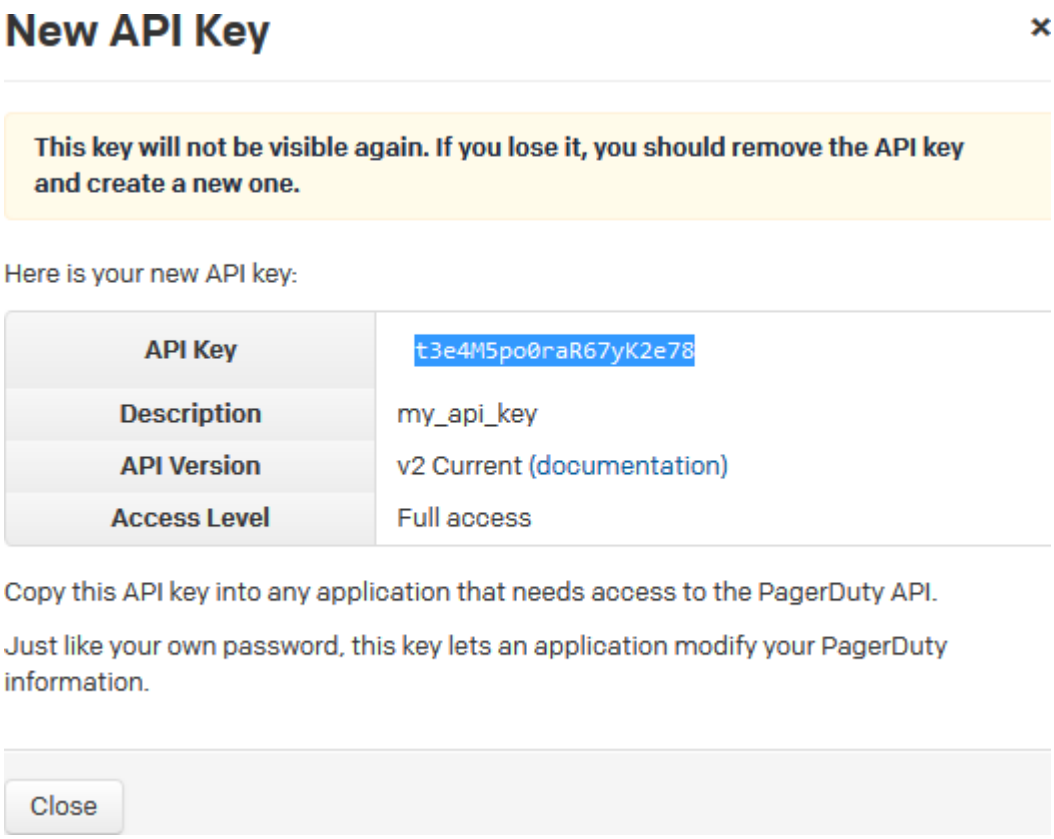



3. Click **Create New API Key**.
4. In the Create API Key dialog box, enter a name for the key and click **Create Key**.

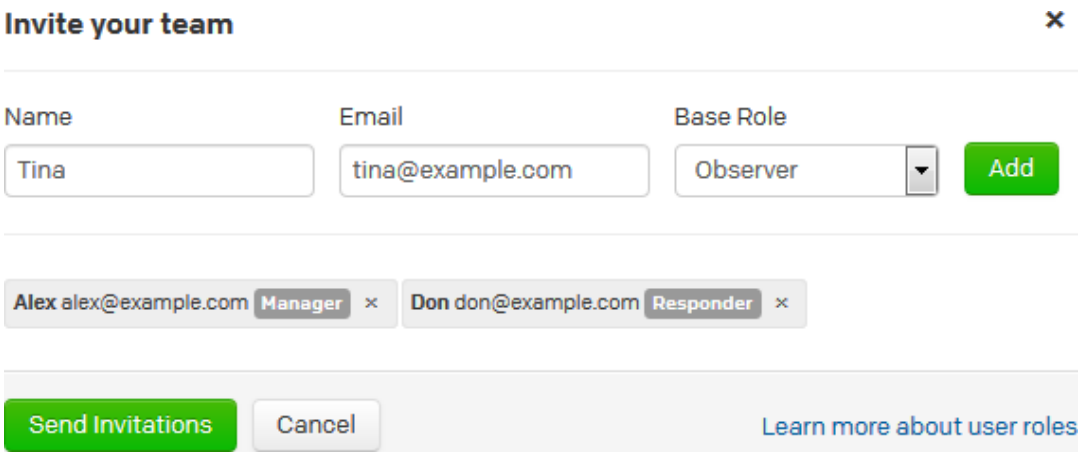



5. From the New API Key dialog box, copy the **API Key** value and keep it safe.

   You can't view or copy the key after closing the dialog box.




6. Click **Close**.
7. If not configured, set up services (such as applications or components) you wish to open incidents against. From the **Configuration** menu, select **Services**.
8. Click **New Service**.
9. Fill in the details and click **Add Service**.
10. If not configured, add users who'd receive notifications. From the **Configuration** menu, select **Users**.
11. Click **Add Users**.
12. In the Invite your team dialog box, add the details of users you want to invite, and click **Add**.



13. When you're finished adding users, click **Send Invitations**.
14. Return to the dashboard page of PagerDuty.

## Configure a PagerDuty Webhook in VB Studio to Send Event Notifications

The PagerDuty webhook is a outgoing webhook used to send VB Studio event notifications to a PagerDuty account.

The project owner can create and configure a webhook:

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From **Type**, select **PagerDuty**.
5. In **Name**, enter a unique name.
6. In **API Key**, enter the API key of the PagerDuty service.
7. In **Service**, select the desired PagerDuty service from the list. The webhook sends event notifications to the selected service.
8. In **Sender**, select the PagerDuty registered user whose name will be attached to the events sent by the webhook.
9. In **Event Groups**, select the events that trigger the webhook.

   If you selected the **Select specific events** option, in **Events**, select the check boxes of events that trigger the webhook.
10. Click **Done**.

# Jenkins

Jenkins is an open-source continuous integration software used to build and test your software applications. Using the various Jenkins webhooks, you can integrate your Jenkins with VB Studio to run builds. Jenkins must be available on the public Internet to accept webhook notifications.

You can use these webhooks to integrate Jenkins with VB Studio:

| To do this ... | Use this webhook |
|---|---|
| Trigger a Jenkins job on SCM polling of the job's Git repository | Hudson/Jenkins Git Plugin |
| Trigger a Jenkins job on a project's Git repository update | Hudson/Jenkins Build Trigger |
| Link a Jenkins job with a merge request | Jenkins Merge Requests |
| Receive notifications in VB Studio project's activity feed from Jenkins when a job's build runs or completes | Jenkins Notification Plugin |

# Use SCM Polling to Trigger a Jenkins Job

Use the Hudson/Jenkins - Git Plugin Webhook to trigger a Jenkins job when SCM polling indicates source file changes in a VB Studio Git repository.

To trigger the Jenkins job:

1. If not installed, install the Git plugin.
2. Create or configure the Jenkins job to use the Git repository in a VB Studio project as source.
3. Enable SCM polling in the Jenkins job.
4. Create or configure a webhook to send a notification to Jenkins when the job's Git repository (or any project Git repository) is updated.

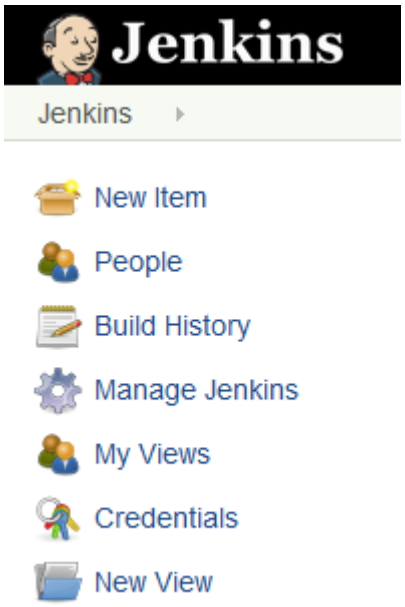
When the Jenkins Git plugin receives notification of a change in a rep[ository, it goes through all Jenkins jobs that have SCM polling enabled and matches the provided notification parameters, such as Git repositories and branches. For all matching jobs, it starts a build. A build won't run when polling indicates that no changes were found.

For more information about the Jenkins Git plugin, see `https://wiki.jenkins-ci.org/display/JENKINS/Git+Plugin#GitPlugin-Pushnotificationfromrepository`.

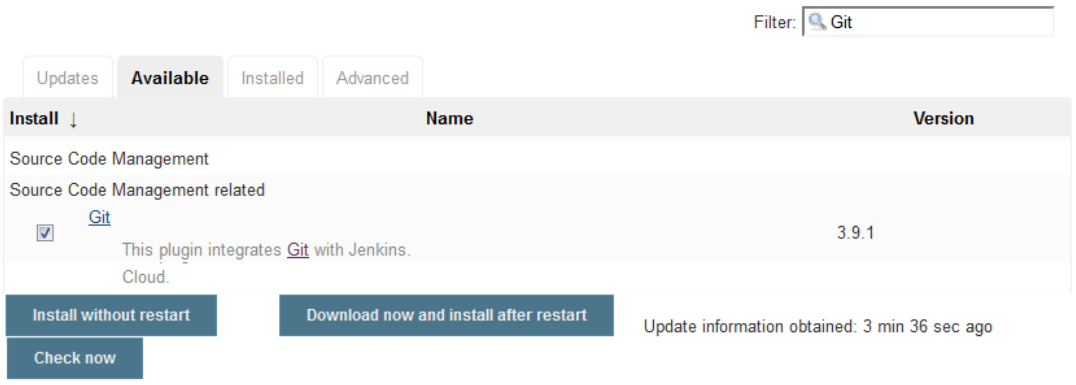
## Set Up Git on Jenkins

To set up Git on Jenkins, you must be assigned the Jenkins Admin role. Git must also be installed on the computer running Jenkins. If the plugin is already installed and configured, ignore this section.

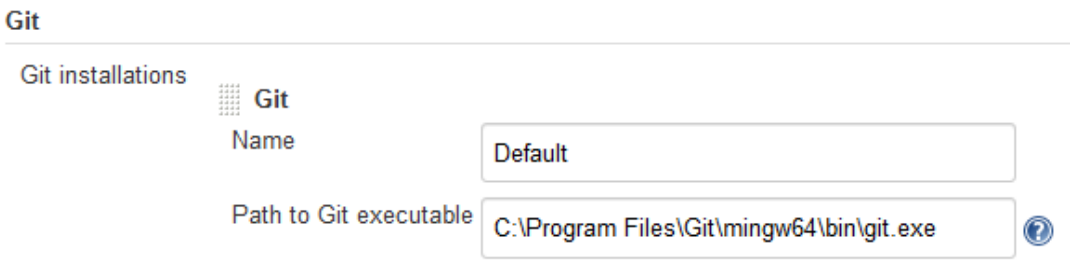
1. Log on to Jenkins using the administrator credentials.
2. From the links on the left side of the page, click **Manage Jenkins**.




3. To install the Git plugin, click **Manage Plugins**.
4. In the **Available** tab, search for `Git`. Under **Source Code Management**, select the plugin's check box and click **Download now and install after restart** or **Install without restart**.




5. Wait for the plugin to install.
6. Restart Jenkins.
7. From the links on the left side of the page, click **Manage Jenkins**.
8. Click **Global Tool Configuration**.
9. In **Git**, enter the local path of the Git executable.



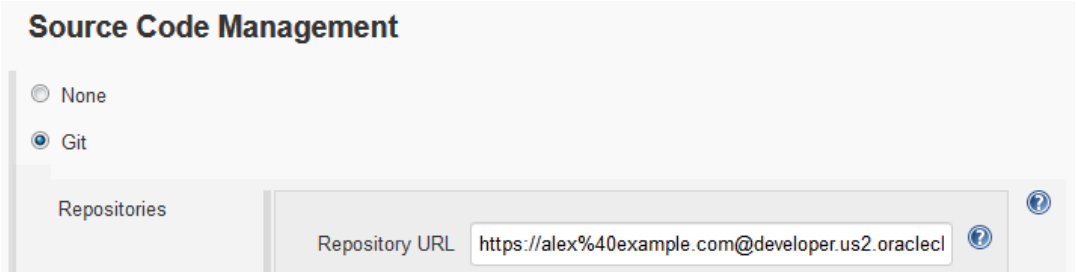

10. Click **Save**.

## Configure the Jenkins Job to Use the VB Studio Git Repository and Enable SCM Polling

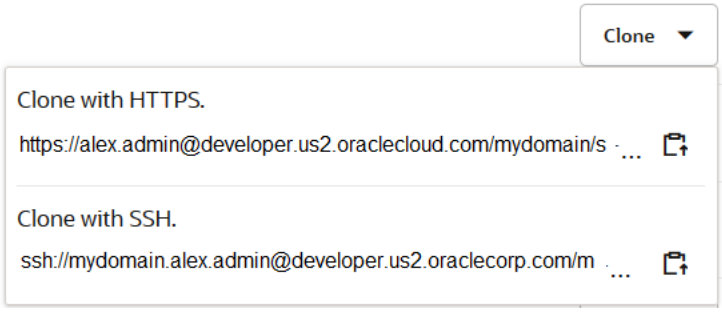
Configure the job to use the VB Studio Git repository and enable SCM polling.

1. Log on to Jenkins.
2. Create or open a job.
3. From the links on the left side of the page, click **Configure**.
4. Click the **Source Code Management** tab.
5. Select **Git**.
6. In **Repository URL**, enter the VB Studio project's Git repository URL.

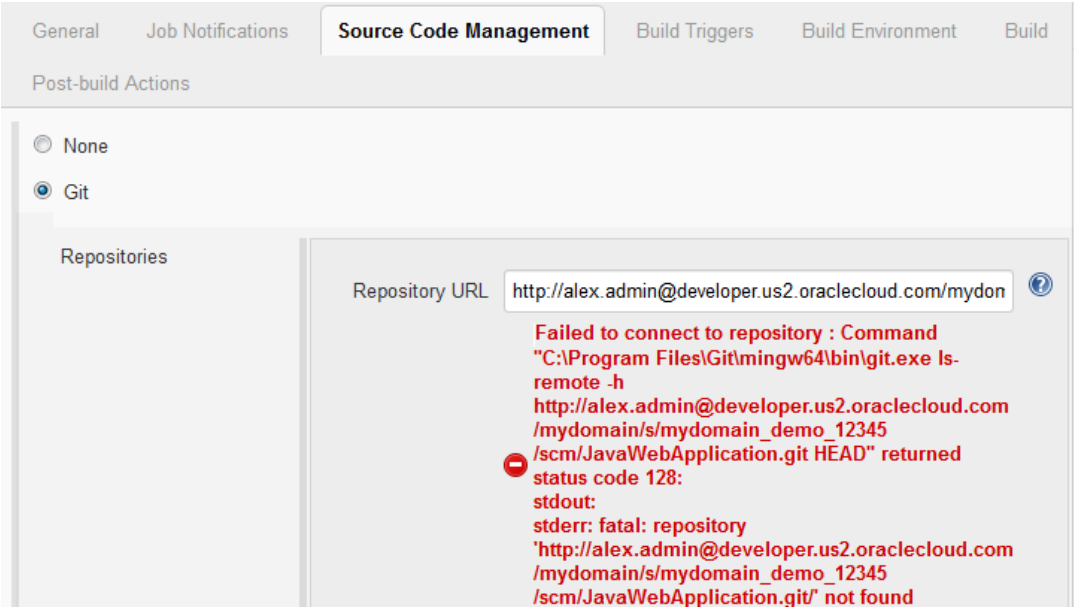
   Remember the URL's protocol as you'd need to specify it when you create the webhook.




You can copy the URL from the **Clone** menu of the VB Studio **Git** page.

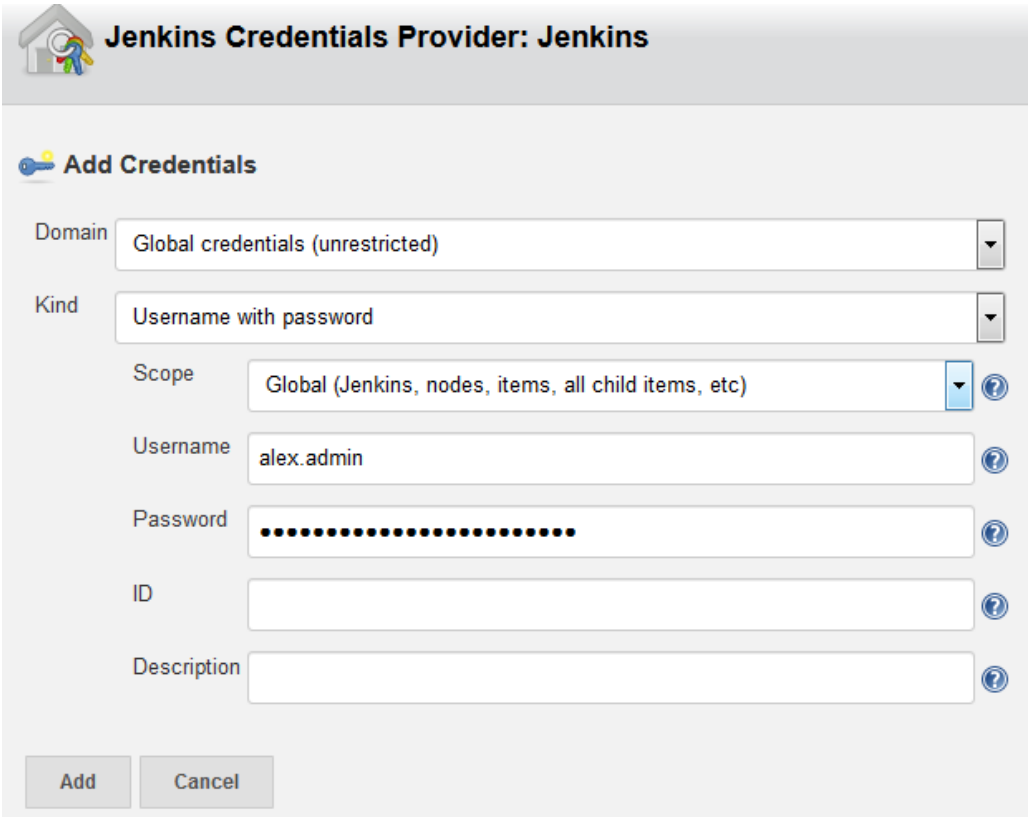



After entering the URL, you might see a `Failed to connect to repository …` error message. It appears because you haven't provided the VB Studio access credentials to Jenkins.




a. Next to the **Credentials** list, click **Add** and then select **Jenkins**.

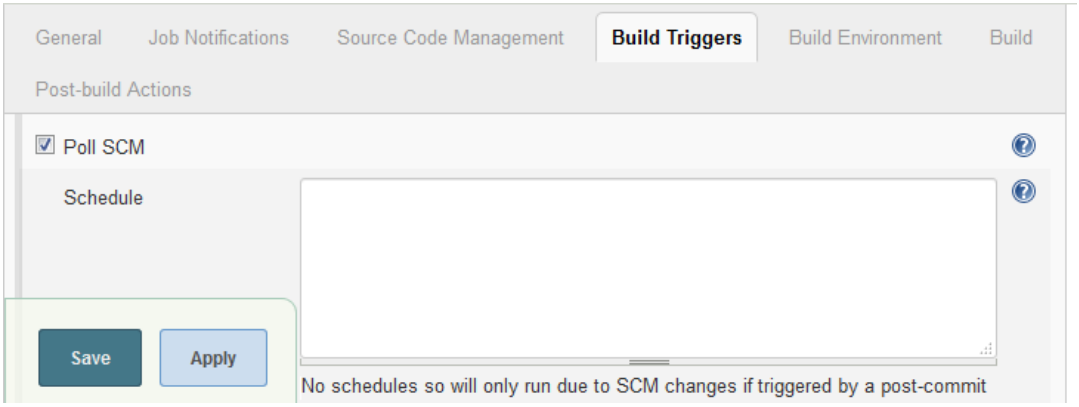
b. In the Jenkins Credentials Provider dialog box, enter the VB Studio username and password in **Username** and **Password**. Leave other fields with their default values.




c. Click **Add**.

The error message should disappear. If you still see the error message, configure the proxy settings of Jenkins. See the Jenkins documentation to see how to do that..

7. Click the **Build Triggers** tab.
8. Select the **Poll SCM** check box.



9. Continue to configure the job.
10. When you're finished, click **Save**.

## Configure a Hudson/Jenkins Git Plugin Webhook

After configuring the Jenkins job, create the VB Studio webhook to trigger the job when a Git repository gets updated.

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From **Type**, select **Hudson/Jenkins - Git Plugin**.
5. In **Name**, enter a unique name.
6. In **Notification URL**, enter the URL of the target Jenkins server.

   The URL must be in the `http://your_server/.../git/notifyCommit` format. Example: `http://my_jenkins.com:8080/git/notifyCommit`
7. To ignore SSL errors, select the **Ignore SSL Errors** check box.
8. In **Notification Parameters**, specify the URL type.
   - In **Repository URL Type**, select **HTTP Repository Address** to send the HTTP URL of the selected Git repository in the webhook notification. Select **SSH Repository Address** to send the SSH URL of the selected Git repository in the webhook notification.

     You must specify the same protocol that's used in the Jenkins job configuration.
   - In **Append**, to append the SHA-1 Checksum hash of the last commit in the webhook notification, select the **sha1 (Jenkins only)** check box.
   - To append branch information of the last commit in the webhook notification, select the **branches** check box. This enables jobs to poll the specified branches only.
9. In **Repository** and **Branches**, specify the Git repository and branches that trigger the webhook.

   In **Repository**, select **All Repositories** to trigger Jenkins jobs that use any of the project's Git repositories.

   In the filter, you select from the item(s) displayed or use a regular expression to make a more complex selection (a range or ranges, perhaps) that involves multiple items that match a pattern. See Glob Pattern Reference for Matching Branch, Job, and Pipeline Names for detailed information about the syntax for regular expressions.
10. Click **Done**.

# Create a Hudson/Jenkins - Build Trigger Webhook to Trigger a Jenkins Job on an Update to a Git Repository

You can use the Hudson/Jenkins - Build Trigger webhook to trigger a Jenkins job when a Git repository gets updated. It's not necessary for the Jenkins job to use a VB Studio project's Git repository as source.

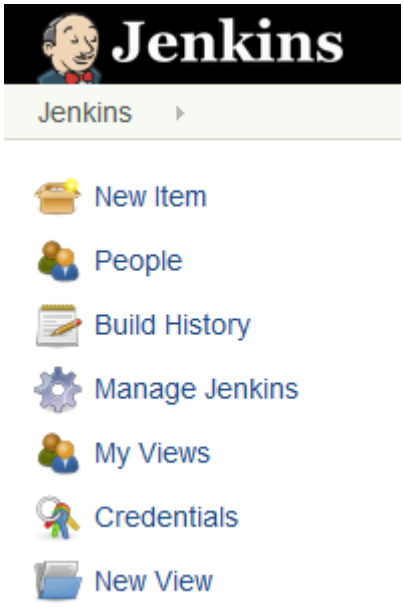
You need to specify the Jenkins security settings to enable the webhook to connect to Jenkins.

| If ... | Do this: |
|---|---|
| Jenkins allows anonymous user to trigger a build | 1. Create an authentication token in the Jenkins job.<br>2. Configure the webhook to connect to the Jenkins job using the authentication token. |
| Jenkins allows only authenticated users to trigger a build | 1. Get an authenticated user's API Access token.<br>2. Create an authentication token in the Jenkins job.<br>3. Configure the webhook to connect to the Jenkins job using the API Access and the authentication token. |
| You want to trigger the job without an authenticated user's credentials but anonymous access on Jenkins is disabled or lacks read permissions<br>or<br>Jenkins uses a build token root to trigger builds | 1. Install the Build Authorization Token Root Plugin on Jenkins if it has not already been installed.<br>2. Create an authentication token in the Jenkins job.<br>3. Configure the webhook to connect to Jenkins job using the authentication token. |
| Security is completely disabled on Jenkins. | Configure the webhook to connect to the Jenkins job. No Jenkins configuration is required. |

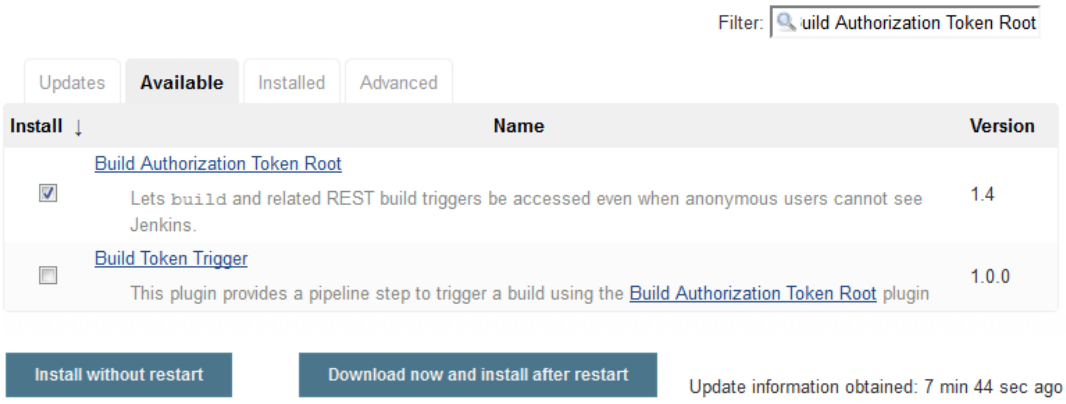
## Install the Build Authorization Token Root Plugin on Jenkins

If anonymous access is disabled on Jenkins and if you want to trigger Jenkins jobs without an authenticated user's credentials, install the Build Authorization Token Root plugin on Jenkins. You must be assigned the Jenkins Admin role to install the plugin. The plugin is required. To find out more about the plugin, see `https://wiki.jenkins-ci.org/display/JENKINS/Build+Token+Root+Plugin`.

1. Log on to Jenkins using the administrator credentials.
2. From the links on the left side of the page, click **Manage Jenkins**.




3. Click **Manage Plugins**.
4. In the **Available** tab, search for `Build Authorization Token Root`, select its check box, and click **Download now and install after restart** or **Install without restart**.




5. Wait for the plugin to install.
6. Restart Jenkins.

## Get the Jenkins API Access Token

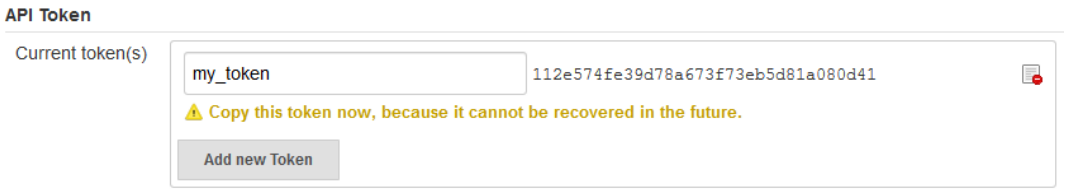
If Jenkins allows authenticated users only to trigger builds, use the API Access token of an authenticated user as the user's credentials in the VB Studio webhook.

To use the API Access token in a VB Studio webhook, provide the username and the token of an authenticated user. If you don't want to provide a user's details, create a separate username to trigger builds and assign the user the **Overall/Read**, **Job/Read** and **Job/Build** permissions. Then, use this user's details in the webhook.

1. Log on to Jenkins using the user's credentials whose API Access Token you want to use in the webhook.
2. In the upper-right corner, mouse over the user name, click ▾ and select **Configure**.

3. In the **API Token** section, add a new token or use the legacy token.

   To view the legacy token, click **Show Legacy API Token** and then copy the token. Keep the legacy token value someplace safe, because you'll need to enter it in the VB Studio webhook.
   To create a token, click **Add new token** and immediately copy the token value. You won't be able to see the token value later and you'll need to generate another token. Keep the new token value someplace safe because you'll need to enter it in the VB Studio webhook.




4. Click **Save**.

## Configure the Jenkins Job to Set an Authentication Token

You need to set the authentication token if Jenkins allows anonymous access, access to authenticated users only, or uses the build token root plugin. Use the same token name when you configure the webhook.

1. Log on to Jenkins.
2. Click the job name.
3. From the links on the left side of the page, click **Configure**.
4. Click the **Build Triggers** tab.
5. Select the **Trigger builds remotely (e.g., from scripts)** check box.

6. In **Authentication Token**, enter a unique string as a token. You can enter any string value. Example: `my_auth_token`

   Make sure that the authentication token isn't used in any other job.

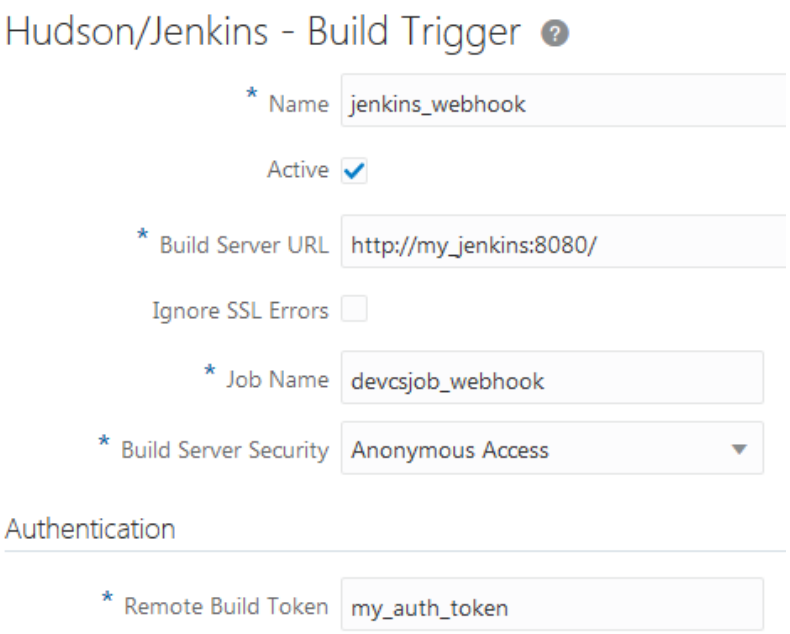
7. Continue to configure the job.
8. When you're finished, click **Save**.

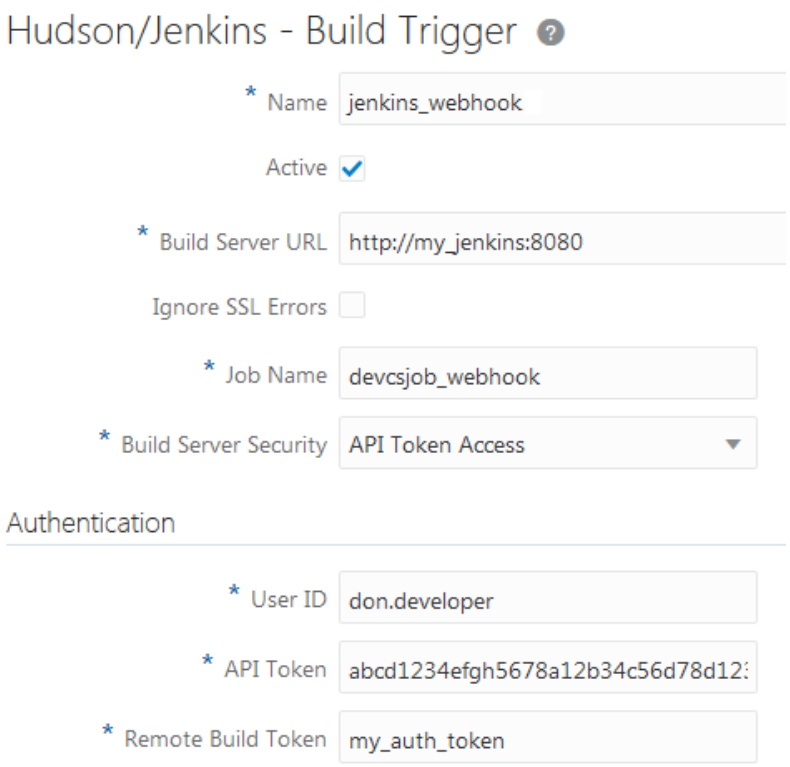
## Configure a Hudson/Jenkins Build Trigger Webhook

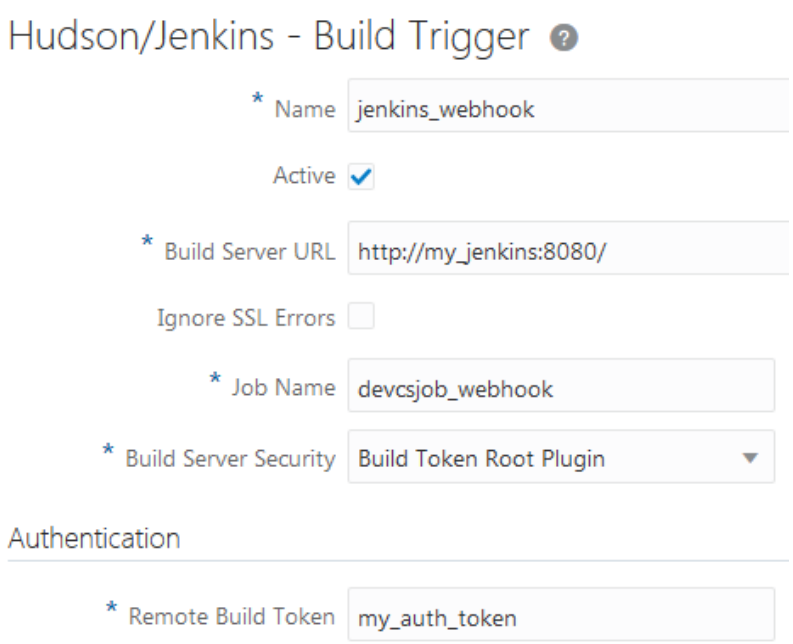
Before you create the webhook, make sure that you've installed the required plugins and have the token required to access Jenkins through the webhook.

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From **Type**, select **Hudson/Jenkins - Build Trigger**.
5. In **Name**, enter a unique name.
6. In **Build Server URL**, enter the Jenkins base URL.

   If the Jenkins job URL is `http://my_jenkins/path/job/my_job`, then enter `http://my_jenkins/path/`.
7. If you provided an HTTPS URL in **Build Server URL**, select the **Ignore SSL Errors** check box to ignore SSL errors if Jenkins uses a self-signed certificate (or an invalid one).
8. In **Job Name**, enter the case sensitive name of the job on the target build server.
9. From **Build Server Security**, select the Jenkins security schema and enter the required details:

| Security Option | Fill in these fields |
|---|---|
| Anonymous Access | Under **Authentication**, in **Remote Build Token**, enter the Jenkins authentication token, similar to what is shown in this example:   |

| Security Option | Fill in these fields |
| --- | --- |
| API Token Access | Under **Authentication**, enter the authenticated user's details:<br>• In **User ID**, enter the username of the Jenkins user.<br>• In **API Token**, enter the API token of the Jenkins user.<br>• In **Remote Build Token**, enter the Jenkins authentication token, similar to what is shown in this example:<br>  |

| Security Option | Fill in these fields |
|---|---|
| Build Token Root Plugin | Under **Authentication**, in **Remote Build Token**, enter the Jenkins authentication token, similar to what is shown in this example:   |
| No Security | NA |

10. In **Trigger Event: Git Push**, complete the following:
    a. Select the Git repository.
    b. Specify what to follow: **All branches/tags** (pushes to any branch or tag), **Branch** (pushes to one specific branch), or **Tag** (forced updates of one specific tag)
    c. If you specified **Branch** or **Tag** in the previous step, select the branch or tag to be followed.

       In the filter, you can either select from the branch(es) or tag(s) displayed or use a regular expression to make a more complex selection (a range or ranges, perhaps) that involves multiple items that match a pattern. See Glob Pattern Reference for Matching Branch, Job, and Pipeline Names for detailed information about the syntax for regular expressions.
    d. Set up the directory filter.

       Click **Add directory** to specify a directory or list of directories to watch for changes.

       The directory or list of repository directories act as an additional filter for triggering the webhook only when changes are detected in at least one directory or in at least one of its subdirectories. Each directory path needs to be relative to the repository root and be specified using Unix-like slash ("/") separators.
    e. Enable the **Parametrized Build** toggle if the build job on the target server accepts parameters. (The target URL is different for parametrized and non-parametrized builds.)

If **Parametrized Build** is enabled, you can add build parameters using **Add Parameter**. For each parameter, set the name that must match the parameter name defined on the build server side.

f. Verify the URL displayed in **Target URL**.

Use cURL with HTTPS GET and the Target URL to check your configuration:

```
curl -X GET '<Target_URL>'
```

11. Click **Done**.

## Use the Jenkins - Merge Requests Webhook to Link a Jenkins Job to a Merge Request

You can use the Jenkins - Merge Requests webhook to link a Jenkins job to a merge request. When a commit is pushed to the review branch of the merge request, the webhook sends a notification to Jenkins and triggers a build of the linked job. When the build completes, it sends a notification back to VB Studio. The linked build will approve or reject the merge request, based on the status of the build.

The Jenkins Merge Request is an outgoing as well as an incoming webhook. The Jenkins job and the webhook must use the merge request's Git repository with parameters to define the branch. The Notification plugin must also be installed on Jenkins.
You need to specify the Jenkins security settings to allow the webhook to connect to Jenkins.

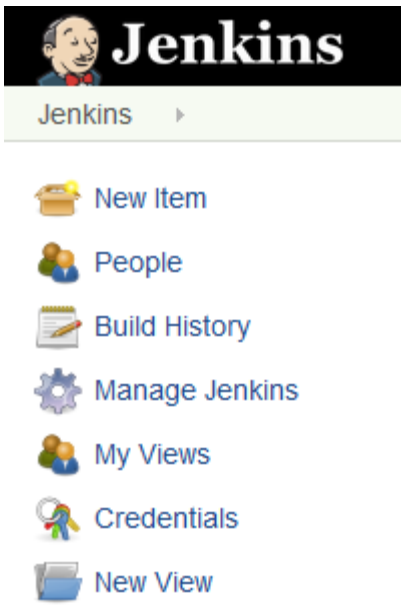
| If ... | Do this: |
|---|---|
| Jenkins allows an anonymous user to trigger a build on Jenkins | 1. Create an authentication token in the Jenkins job.<br>2. Use the authentication token to configure the webhook to connect to the Jenkins job. |
| Jenkins allows only authenticated users to trigger a build | 1. Get an authenticated user's API Access token.<br>2. Create an authentication token in the Jenkins job.<br>3. Use the API Access and the authentication token to configure the webhook to connect to the Jenkins job. |
| Anonymous access on Jenkins is disabled or lacks read permissions and you want to trigger the job without an authenticated user's credentials<br>or<br>Jenkins uses a build token root to trigger builds | 1. Install the Build Authorization Token Root Plugin on Jenkins if it has not already been installed.<br>2. Create an authentication token in the Jenkins job.<br>3. Configure the webhook to connect to Jenkins job using the authentication token. |
| Security is completely disabled on Jenkins | Configure the webhook to connect to Jenkins job. No Jenkins configuration is required. |

## Install the Notification Plugin on Jenkins

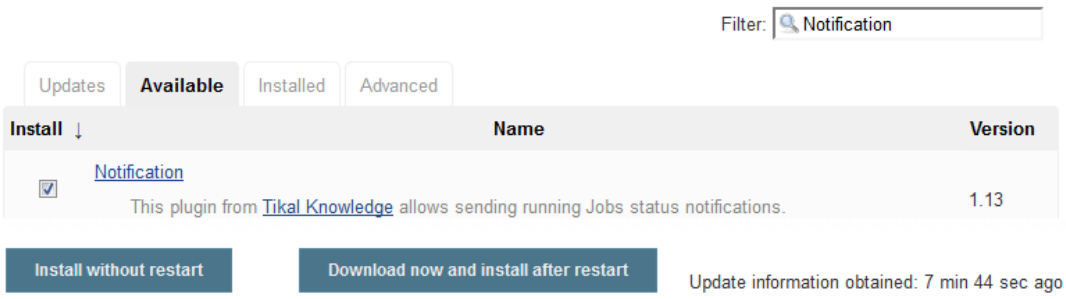
Install the Notification plugin to send notifications from Jenkins.

You must be assigned the Admin role of the Jenkins server to install plugins.

1. Use the administrator credentials to log on to Jenkins.
2. From the links on the left side of the page, click **Manage Jenkins**.




3. Click **Manage Plugins**.
4. In the **Available** tab, search for `Notification`, select its check box, and click **Download now and install after restart** or **Install without restart**.

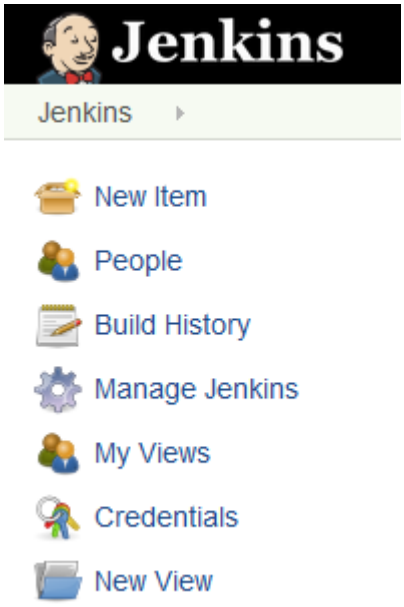



5. Wait for the plugin to install.
6. Restart Jenkins.

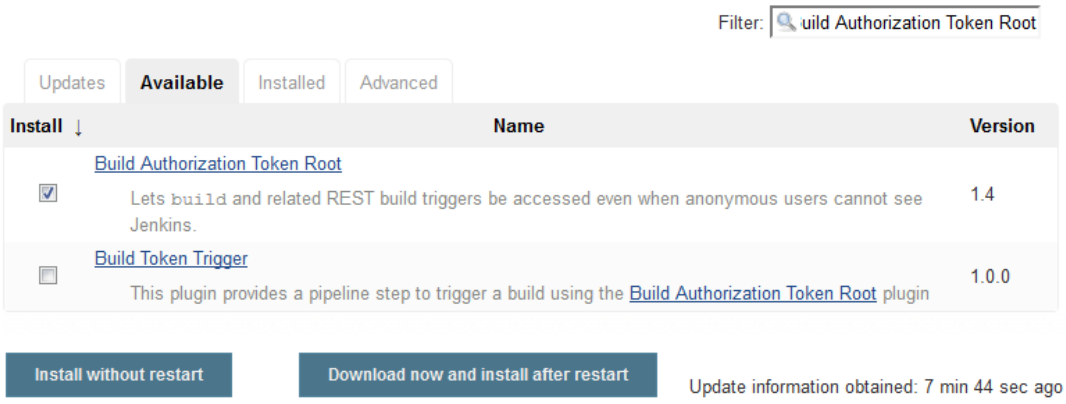
## Install the Build Authorization Token Root Plugin on Jenkins

If anonymous access is disabled on Jenkins and if you want to trigger Jenkins jobs without an authenticated user's credentials, install the Build Authorization Token Root plugin on Jenkins. You must be assigned the Jenkins Admin role to install the plugin. The plugin is required. To find out more about the plugin, see `https://wiki.jenkins-ci.org/display/JENKINS/Build+Token+Root+Plugin`.

1. Log on to Jenkins using the administrator credentials.
2. From the links on the left side of the page, click **Manage Jenkins**.




3. Click **Manage Plugins**.
4. In the **Available** tab, search for `Build Authorization Token Root`, select its check box, and click **Download now and install after restart** or **Install without restart**.




5. Wait for the plugin to install.
6. Restart Jenkins.

## Get the Jenkins API Access Token

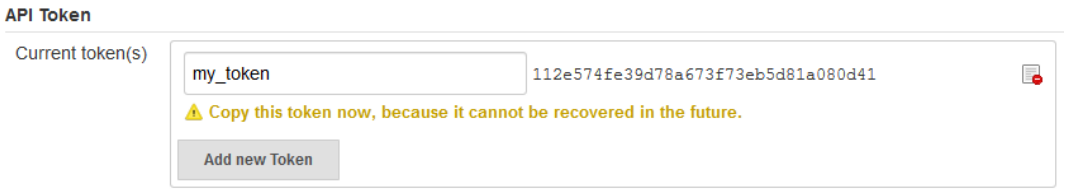
If Jenkins allows authenticated users only to trigger builds, use the API Access token of an authenticated user as the user's credentials in the VB Studio webhook.

To use the API Access token in a VB Studio webhook, provide the username and the token of an authenticated user. If you don't want to provide a user's details, create a separate username to trigger builds and assign the user the **Overall/Read**, **Job/Read** and **Job/Build** permissions. Then, use this user's details in the webhook.

1. Log on to Jenkins using the user's credentials whose API Access Token you want to use in the webhook.
2. In the upper-right corner, mouse over the user name, click ▾ and select **Configure**.

3. In the **API Token** section, add a new token or use the legacy token.

   To view the legacy token, click **Show Legacy API Token** and then copy the token. Keep the legacy token value someplace safe, because you'll need to enter it in the VB Studio webhook.
   To create a token, click **Add new token** and immediately copy the token value. You won't be able to see the token value later and you'll need to generate another token. Keep the new token value someplace safe because you'll need to enter it in the VB Studio webhook.

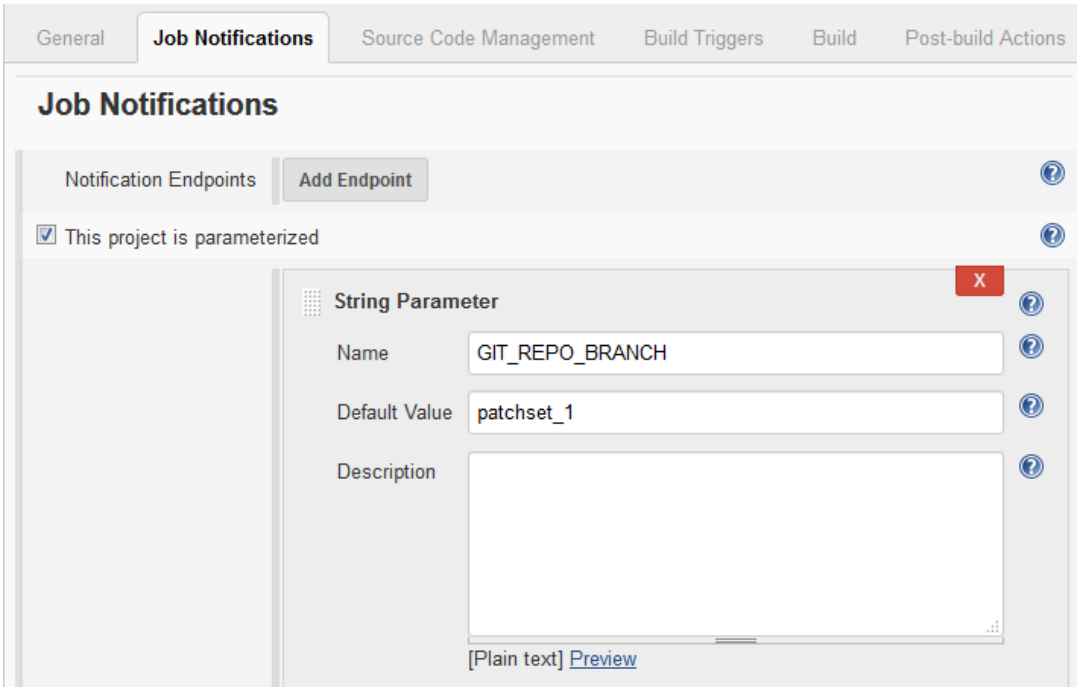



4. Click **Save**.

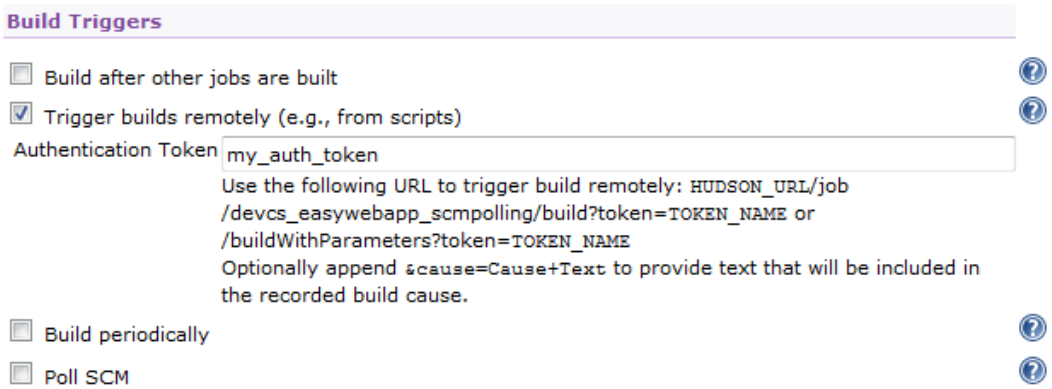
## Configure the Jenkins Job to Set an Authentication Token and Accept Build Parameters

To trigger the Jenkins job when it receives a notification from VB Studio, configure it to accept the Git repository's branch name as a parameter and set an authentication token.

1. Log on to Jenkins.
2. Create or open the job.
3. On the left side of the page, click **Configure**.
4. Click the **Job Notifications** tab.
5. Select the **This project is parameterized** check box.
6. From **Add Parameter**, select **String Parameter**.
7. In **Name**, enter `GIT_REPO_BRANCH`.
8. In **Default Value**, enter the review branch name. Example: `patchset_1`




9. Click the **Build Triggers** tab.
10. Select the **Trigger builds remotely (e.g., from scripts)** check box.




11. Enter a unique string as a token. You can enter any string value. Example: `my_auth_token`

    Make sure that the authentication token isn't used in any other job.
12. Continue to configure the job.
13. When you're finished, click **Save**.

## Configure a Webhook in VB Studio to Trigger a Jenkins Job on a Merge Request Update

After installing the required plugins and configuring the Jenkins job, create the webhook.

1. In the left navigator, click **Project Administration** ⚙.
2. Click **Webhooks**.

3. Click **+ Create Webhook**.
4. From **Type**, select **Jenkins - Merge Requests**.
5. In **Name**, enter a unique name.
6. In **Build Server URL**, enter the Jenkins base URL.

   If the Jenkins job URL is `http://my_jenkins/path/job/my_job`, then enter `http://my_jenkins/path/`.
7. Select the **Ignore SSL Errors** check box to ignore SSL errors if Jenkins uses a self-signed certificate (or an invalid one) and you provided an HTTPS URL in **Build Server URL**.
8. In **Job Name**, enter the case-sensitive name of the job on the target build server.
9. In **Repository**, select the merge request's Git repository.
10. From **Build Server Security**, select the security schema of Jenkins and enter the required details.

| Security Option | Fill in these fields |
|---|---|
| Anonymous Access | Under **Authentication**, in **Remote Build Token**, enter the Jenkins authentication token. |
| API Token Access | Under **Authentication**, enter the authenticated user's details:<br>• In **User ID**, enter the username of the Jenkins user.<br>• In **API Token**, enter the API token of the Jenkins user.<br>• In **Remote Build Token**, enter the Jenkins authentication token. |
| Build Token Root Plugin | Under **Authentication**, in **Remote Build Token**, enter the Jenkins authentication token. |
| No Security | NA |

11. Click **Done**.

## Link the Jenkins Job with the Merge Request

1. In the left navigator, click **Merge Request**.
2. Open the merge request.
3. Click the **Linked Builds** tab.

   The tab displays linked jobs, if any.
4. In **Search and Link Build Jobs**, enter the Jenkins job name and select it from the list.
5. Click **Save** ✓.

When a commit is pushed to the merge request's review branch, the webhook triggers a build of the specified job on the remote Jenkins server and a notification is posted to the project's Recent Activity Feed. If the build succeeds, it'll be added to the Approve section of the Review Status list in the Merge Request page. If the build fails, it'll be added to the Reject section of the Review Status list.

# Create a Jenkins - Notification Plugin Webhook So VB Studio Accepts Build Notifications from a Jenkins Job

Use the Jenkins - Notification Plugin Webhook to configure VB Studio to accept build notifications from Jenkins and show build notifications in the **Project Home** page's recent activities feed.

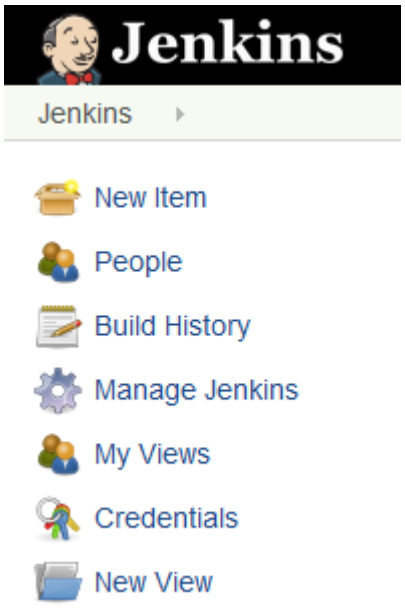
Jenkins - Notification Plugin Webhook is an incoming Webhook and accepts build notifications only. Don’t use this webhook to pass information to any external server or accept information of any other type. To use the webhook, install the Notifications plugin on Jenkins, configure the VB Studio webhook to connect to Jenkins, and then configure the Jenkins job to send build notifications.

## Install the Notification Plugin on Jenkins

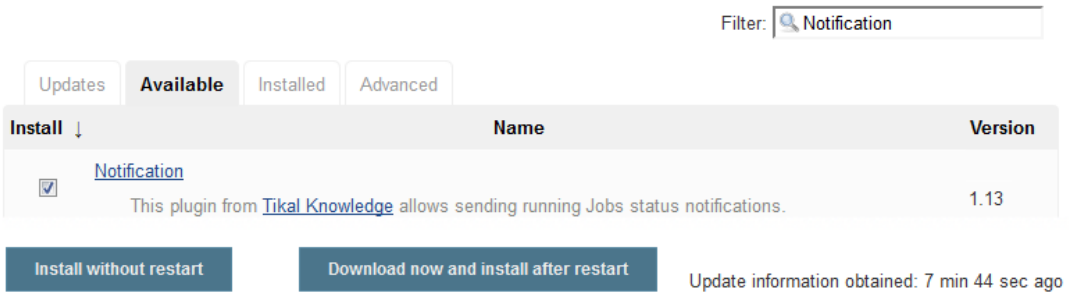
Install the Notification plugin to send notifications from Jenkins.

You must be assigned the Admin role of the Jenkins server to install plugins.

1. Use the administrator credentials to log on to Jenkins.
2. From the links on the left side of the page, click **Manage Jenkins**.




3. Click **Manage Plugins**.
4. In the **Available** tab, search for `Notification`, select its check box, and click **Download now and install after restart** or **Install without restart**.

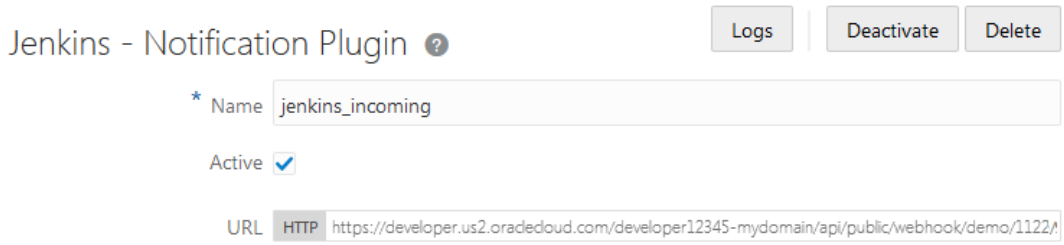



5. Wait for the plugin to install.
6. Restart Jenkins.

## Configure a Webhook in VB Studio to Accept Notifications from Jenkins

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From **Type**, select **Jenkins - Notification Plugin**.
5. In **Name**, enter a unique name.
6. In **Base URL**, enter the base URL of Jenkins.

   If the Jenkins job URL is `http://my_jenkins/path/job/my_job`, then enter `http://my_jenkins/path/`.
7. In **Track**, select the check boxes for the build job actions to be listed in the Recent Activities Feed of the **Project Home** page.
   - To display activities that occur after the build server job finishes, select the **Build Results** check box.
   - To display activities associated with running builds, select the **Ongoing Builds** check box.
8. Click **Done**.
9. On the Webhooks page, from the webhooks list, select the webhook. From the details displayed on the right, copy the value of **URL**.

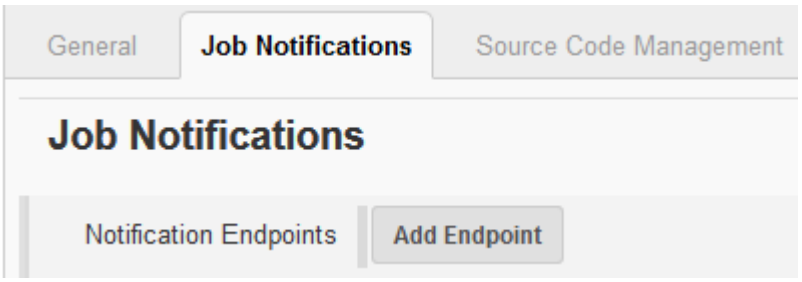



## Configure the Jenkins Job to Send Build Notifications

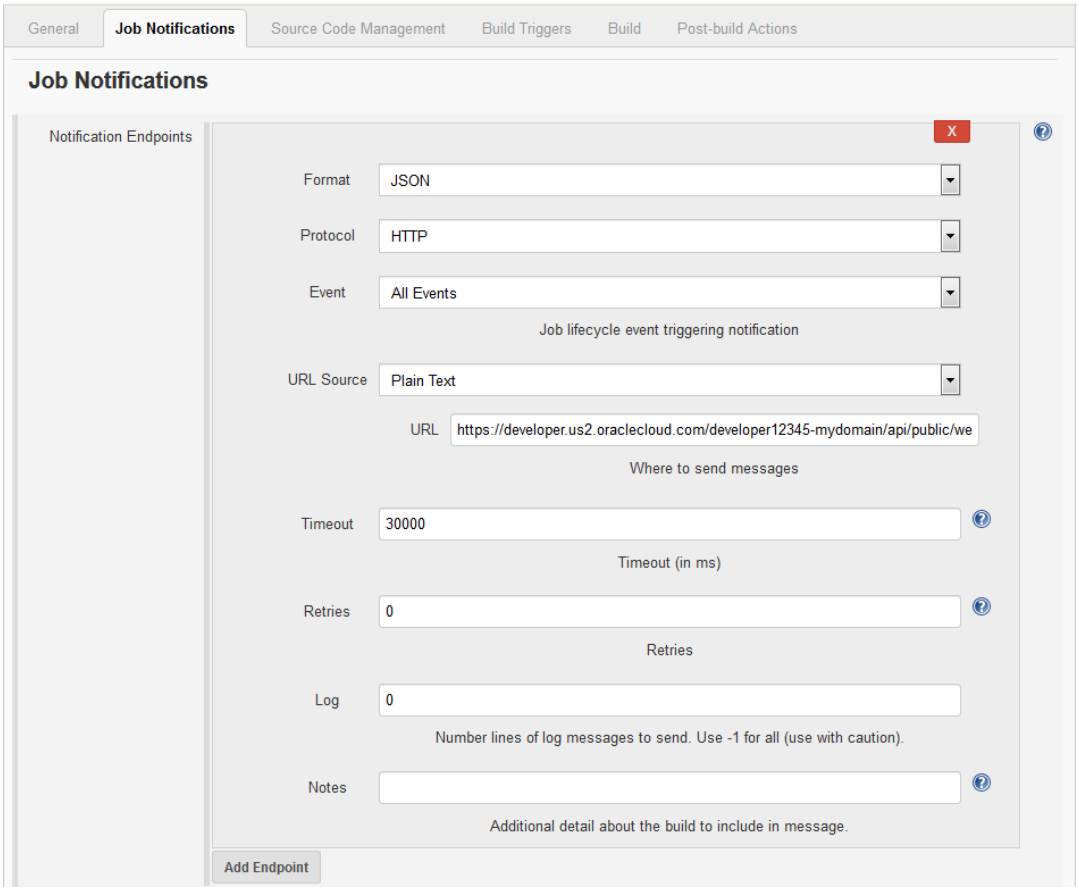
Add the VB Studio webhook's URL as a notification endpoint URL to configure the Jenkins job to send build notifications.

1. Log on to Jenkins.

2. Click the job name.
3. From the links on the left side of the page, click **Configure**.
4. Click the **Job Notifications** tab.
5. In **Notification Endpoints**, click **Add Endpoint**.




6. In the **URL** field, paste the URL that you copied from the VB Studio webhook. Leave the default values in the other fields.




7. Click **Save**.

# Hudson

Hudson is an open-source extensible continuous integration software used to build and test your software applications. Using webhooks, you can integrate your Hudson server with VB

Studio to run builds. Hudson must be available on the public Internet to accept webhook notifications.

You can use these webhooks to integrate Hudson with VB Studio:

| To do this ... | Use this webhook |
|---|---|
| Trigger a Hudson job on SCM polling of the job's Git repository | Hudson/Jenkins Git Plugin |
| Trigger a Hudson job on a project's Git repository update | Hudson/Jenkins Build Trigger |

# Trigger a Hudson Job on SCM Polling

Using the Hudson/Jenkins - Git Plugin Webhook, you can trigger a Hudson job that uses a VB Studio Git repository as source on SCM polling.

To trigger the job:

- If not installed, install the Git plugin
- Create or configure the Hudson job to use the VB Studio project Git repository as source
- Enable SCM polling in the Hudson job
- Create or configure a webhook to send a notification to Hudson when the job's Git repository (or any project Git repository) is updated

When the Git plugin of Hudson receives a notification, it goes through all Hudson jobs that have SCM polling enabled and match the provided notification parameters (such as Git repositories and branches). For all matching jobs, it starts a build. The build won't run if no changes are found by polling.

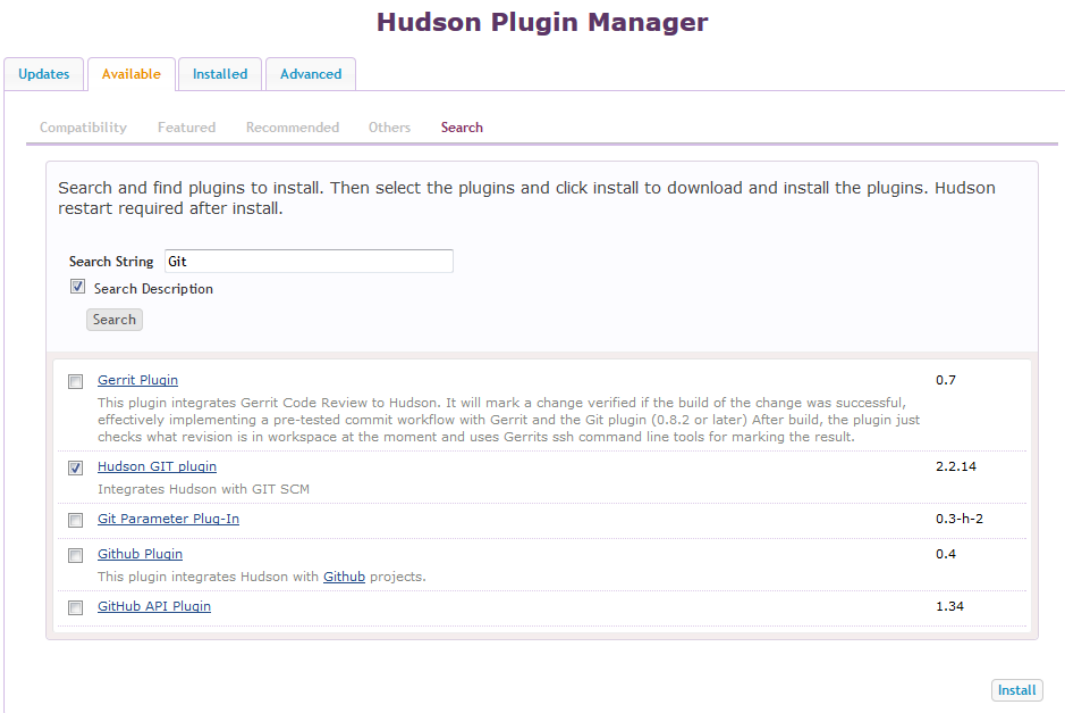
For more information about the Hudson Git plugin, see `http://wiki.hudson-ci.org/display/HUDSON/Git+Plugin#GitPlugin-PostCommitHook`.

## Set Up Git on Hudson

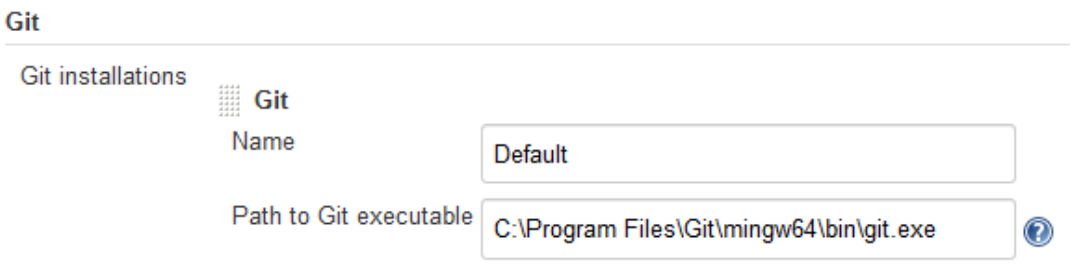
You must be assigned the Admin role of Hudson to set up Git on it. Git must also be installed on the computer running Jenkins. If the plugin is already installed and configured, ignore this section.

1. Log on to Hudson using the administrator credentials.
2. From the links on the left side of the page, click **Manage Hudson**.

3. To install the Git plugin, click **Manage Plugins**.
4. In the **Available** tab, click the **Search** subtab, search for `Git`, select the **Hudson GIT plugin** check box, and click **Install**.




5. Wait for the plugin to install.
6. Restart Hudson.
7. From the links on the left side of the page, click **Manage Hudson**.
8. Click **Configure System**.
9. In **Git**, enter the local path of the Git executable.



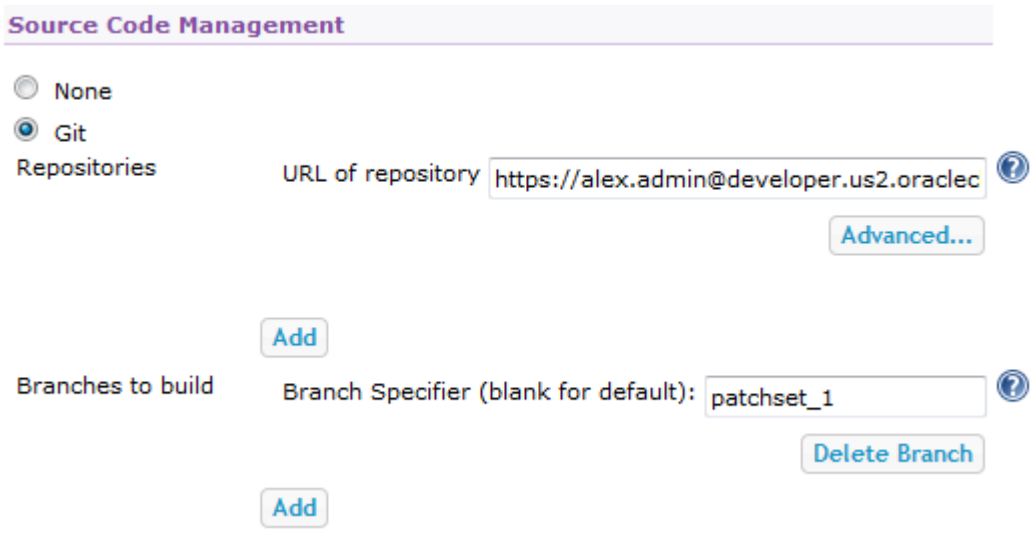

10. Click **Save**.

## Configure the Hudson Job to Use VB Studio Git Repository and Enable SCM Polling

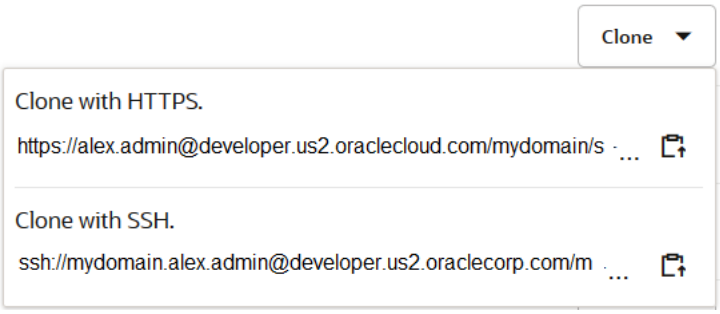
Configure the job to access the VB Studio Git repository and enable SCM polling.

1. Log in to Hudson.
2. Create or open a job.
3. From the links on the left side of the page, click **Configure**.
4. In **Source Code Management**, select **Git**.
5. In **URL of repository**, enter the VB Studio project's Git repository URL.

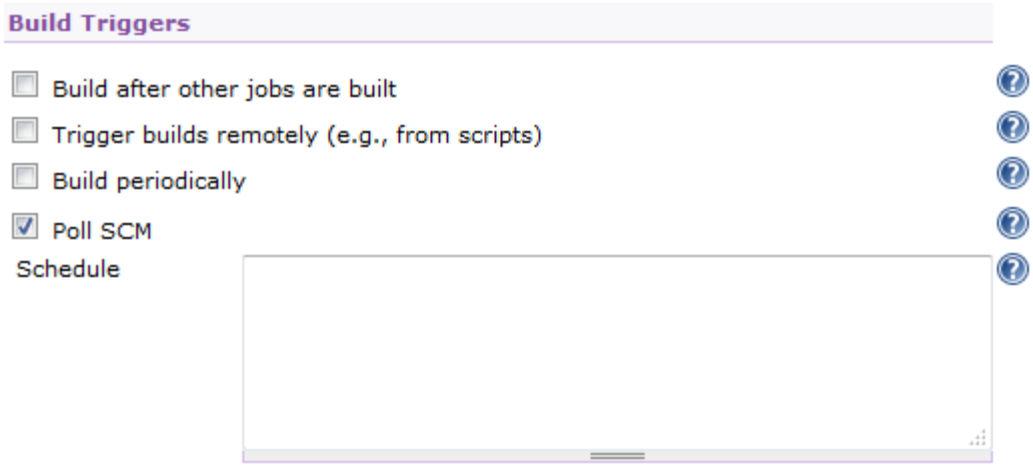
   Remember the URL's protocol as you'd need to specify it when you create the webhook.




You can copy the URL from the **Clone** menu of the VB Studio Git page.



6. In **Branches to build**, specify the branch name.
7. In the **Build Triggers** section, select the **Poll SCM** check box.




8. Continue to configure the job.
9. Click **Save**.

## Configure a Webhook in VB Studio to Trigger a Hudson Job When the Git Repository Gets Updated

After configuring the Hudson job, create the VB Studio webhook to trigger the job when the Git repository is updated.

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From **Type**, select **Hudson/Jenkins - Git Plugin**.
5. In **Name**, enter a unique name.
6. In **Notification URL**, enter the Hudson URL.

   The URL must be in the `http://your_server/.../git/notifyCommit` format. Example: `http://my_hudson.com:8080/git/notifyCommit`
7. To ignore SSL errors, select the **Ignore SSL Errors** check box.
8. In **Notification Parameters**, specify the URL type.

   In **Repository URL Type**, select **HTTP Repository Address** to send the HTTP URL of the selected Git repository in the webhook notification. Select **SSH Repository Address** to send the SSH URL of the selected Git repository in the webhook notification.

   You must specify the same protocol that's used in your the job configuration to access the Git repository.

   To append branch information of the last commit in the webhook notification, select the **branches** check box. This enables jobs to poll the specified branches only.
9. In **Repository** and **Branches**, specify the Git repository and branches that trigger the webhook.

In **Repository**, select **All Repositories** to trigger all Hudson jobs that uses a Git repository of the project.

10. Click **Done**.

# Trigger a Hudson Job on a Git Repository Update

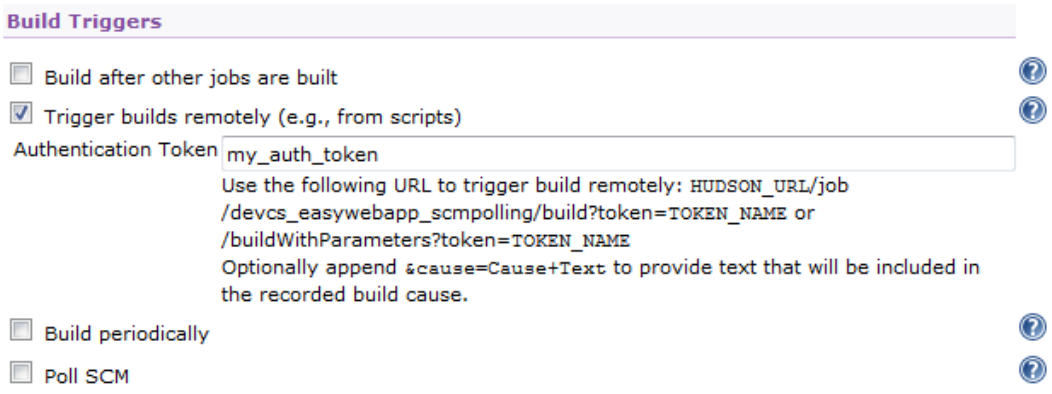
You can use the Hudson/Jenkins - Build Trigger webhook to trigger a Hudson job when a project's Git repository is updated. The Hudson job doesn't have to use a VB Studio project's Git repository as source.

To allow the webhook to connect to Hudson, you need to specify the Hudson security settings:

| If ... | Do this: |
|---|---|
| Hudson allows anonymous user to trigger a build | 1. Create an authentication token in the Hudson job.<br>2. Configure the webhook to connect to the Hudson job using the authentication token. |
| Hudson allows only authenticated users to trigger a build | 1. Get an authenticated user's credentials.<br>2. Create an authentication token in the Hudson job.<br>3. Configure the webhook to connect to the Hudson job using the credentials and the authentication token. |
| Security is completely disabled on Hudson | Configure the webhook to connect to Hudson job. No Hudson configuration required. |

## Configure the Hudson Job

1. Log in to the Hudson server.
2. Click the job name.
3. From the links on the left side of the page, click **Configure**.
4. In the **Build Triggers** section, select the **Trigger builds remotely (e.g., from scripts)** check box.




5. In **Authentication Token**, enter a unique string as a token. You can enter any string value. Example: `my_auth_token`

Make sure that the authentication token name is not used in any other job.

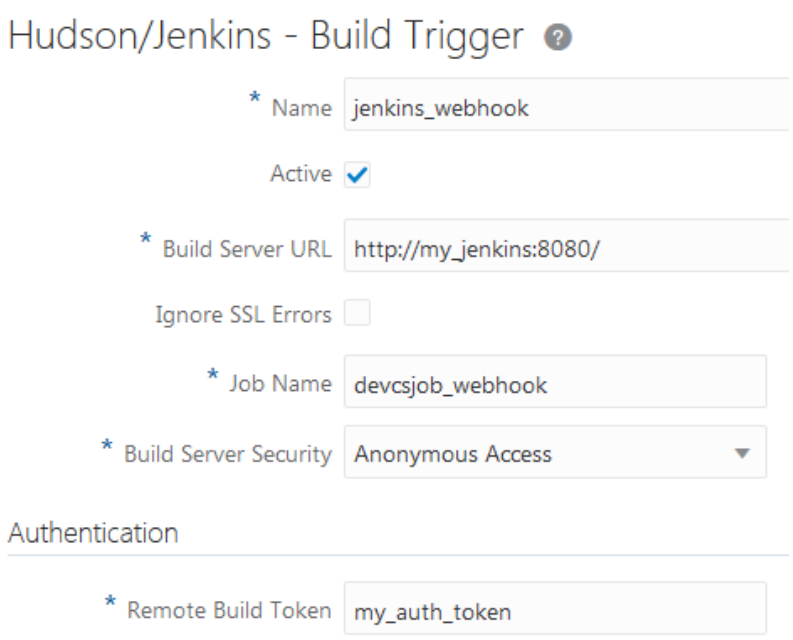
6. Click **Save**.

## Configure a Webhook in VB Studio to Trigger a Hudson Job on a Git Repository Update

Before you create the webhook, make sure you've installed the required plugins and have the required token to access Hudson through the webhook.

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From **Type**, select **Hudson/Jenkins - Build Trigger**.
5. In **Name**, enter a unique name.
6. In **Build Server URL**, enter the Hudson URL.

   If the target build job has address `http://my_server/path/job/my_job`, then enter `http://my_hudson/path/`.
7. To ignore SSL errors if the target build server uses self-signed (or an invalid) certificate and you've provided an HTTPS URL in **Build Server URL**, select the **Ignore SSL Errors** check box.
8. In **Job Name**, enter the case sensitive name of the Hudson job.
9. From **Build Server Security**, select the job's security schema configured on the target server.

| Security Option | Fill in these fields |
|---|---|
| Anonymous Access | Under **Authentication**, in **Remote Build Token**, enter the Jenkins authentication token.<br>Example:<br>  |
| API Token Access | Under **Authentication**, enter the authenticated user's details.<br>• In **User ID**, enter the username of the Jenkins user.<br>• In **API Token**, enter the password of the user.<br>• In **Remote Build Token**, enter the Hudson authentication token. |
| No Security | NA |

10. In **Trigger Event: Git Push**, specify the Git repository and the branch or tag.

    Select the **Parameterized Build** check box if the build job on target server accepts parameters. The target URL differs for parameterized and non-parameterized builds.

    If the **Parameterized Build** is enabled, you can add build parameters using **Add Parameter**. For each parameter, set the name that must match the parameter name defined on build server side.

11. Verify the URL displayed in **Target URL**.

    You can use the URL to check your configuration (for example using `curl -X GET '<Target_URL>'`).

12. Click **Done**.

# GitHub Apps

If you're using apps that accept incoming webhook connections from GitHub, you can use the GitHub-compatible webhook to send VB Studio event notifications to those apps. The payload

will be sent in a format similar to that used by GitHub, so you don't need to make any changes to your GitHub apps.

See `https://developer.github.com/webhooks/` to learn more about GitHub webhooks.

To create a GitHub-compatible webhook that sends VB Studio event notifications to an app that accepts incoming webhook connections from GitHub:

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From the **Type** drop-down list, select **GitHub Compatible**.
5. In **Name**, enter a unique name.
6. In **URL**, enter the GitHub app's URL.
7. In **Secret**, enter a secret phrase that's passed as a string with the HTTP request as a signature header.
8. From the **Payload Type** drop-down list, select the media type for the payload. You can select either `form-urlencoded` (default) or `json`.
9. To ignore the host's SSL certificate verification when delivering the HTTP request, select the **Ignore SSL Errors** check box.
10. In **Event Groups**, select the events that trigger the webhook.

    If you selected the **Select specific events** option, in **Events**, select the check boxes of events that trigger the webhook.
11. Click **Done**.

When you're finished, use the left navigator to switch to another page.

# Send Event Notifications to Any Application

Using the VB Studio Generic Webhook, you can send event notifications to any application that accepts webhook requests and can parse payload-specific content. The webhook payload format depends on the type of the event.

The generic webhook supports all VB Studio events, including Git pushes, issue updates, merge request updates, and project builds. It sends a POST request to the remote service in the JSON format with details of the subscribed events.

1. In the left navigator, click **Project Administration** .
2. Click **Webhooks**.
3. Click **+ Create Webhook**.
4. From the **Type** drop-down list, select **Generic**.
5. In **Name**, enter a unique name.
6. In **URL**, enter the URL of the remote service where you want to deliver the HTTP request.
7. In **Secret**, enter a secret phrase that's passed as a string with the HTTP request as a signature header.
8. To ignore the host's SSL certificate verification when delivering the HTTP request, select the **Ignore SSL Errors** check box.

9. In **Event Groups**, select the events that triggers the webhook.

   If you selected the **Select specific events** option, in **Events**, select the check boxes of the events to trigger the webhook.

10. Click **Done**.

The newly created webhook appears in the webhooks table.
To find more about the data structure of a generic webhook, see What Is a Generic Webhook's Data Structure?.
When you're finished, use the left navigator to switch to another page.

# What Is a Generic Webhook's Data Structure?

Information sent by a generic webhook is delivered with a POST request that has the `application/json` content-type, and the UTF-8 character set, in a `Message` object.

These are the `Message` object's fields:

| Field | Description |
|---|---|
| `apiVersion` | Version of the API that changes when the payload format of the request changes |
| `messageId` | Unique identifier of the message |
| `timestamp` | Timestamp of the message when it was generated |
| `testEvent` | Set to `true` if this event is generated by the Test button |
| `projectId` | Unique identifier of the project |
| `events` | List of events delivered by the message |

Each `event` delivered by the message follows a common structure. These are its fields:

| Field | Description |
|---|---|
| `eventId` | Type of the event (`ISSUE`/`PUSH`/`BUILD`/`REVIEW`/`ACTIVITY`) |
| `projectId` | Unique identifier of the project |
| `timestamp` | Timestamp of the event |
| `data` | Data specific to the type of the event |

The structure of `data` of each event type is described in the following sections.

### ISSUE Event

The `ISSUE` event contains these fields:

| Field | Description |
|---|---|
| `type` | Type of the activity (`CREATED` - issue is created, `COMMENTED` - comment added, `UPDATED` - fields changed) |
| `date` | Timestamp of the activity |
| `description` | Description of the change |
| `task` | Description of the issue after the change |
| `id` | Issue ID |

| Field | Description |
|---|---|
| `version` | Change version |
| `url` | URL of the issue |
| `title` | Title of the issue |
| `type` | Type of the issue (`Defect`, `Feature`, or `Task`) |
| `resolution` | Resolution of the issue. The value is `null` if the issue isn't resolved, otherwise, it's set to one of the issue resolution values such as `FIXED`, `DUPLICATE`, and `WORKSFORME`. |
| `reporter` | User who reported the issue |
| `asignee` | User to whom the issue is assigned |
| `comment` | Content of the added comment, available if the activity type is `COMMENTED` |
| `fieldUpdates` | List of changed fields, available if the activity type is `UPDATED` |
| `name` | Field name |
| `oldValue` | Value before the change |
| `newValue` | Value after the change |

Here's a JSON payload example for an issue create event:

```
   {
    "apiVersion": "1.0",
    "messageId": "04abc282-a44e-4c23-ba53-15b519d30066",
    "projectId": "qa-dev_example-project",
    "testEvent": false,
    "timestamp": 1417810876408,
    "events": [
        {
            "eventId": "ISSUE",
            "projectId": "example-project",
            "timestamp": 1417810876,
            "data": {
                "activities": [
                    {
                        "type": "CREATED",
                        "date": 1417810875820,
                        "description": "",
                        "author": {
                            "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                            "loginName": "alex.admin",
                            "realName": "Alex Admin"
                        },
                        "issue": {
                            "id": 2,
                            "resolution": null,
                            "title": "Test Issue",
                            "type": "Feature",
                            "url": "http://test-server/#projects/example-
project/task/2",
                            "version": "1417810875834",
```

```
                            "reporter": {
                                "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                                "loginName": "alex.admin",
                                "realName": "Alex Admin"
                            }
                        }
                    }
                ]
            }
        }
    ]
   }
```

Here's a JSON payload example for an issue update event:

```
   {
    "apiVersion": "1.0",
    "messageId": "ccce183e-097d-4668-a07b-cf762108716e",
    "projectId": "qa-dev_example-project",
    "testEvent": false,
    "timestamp": 1417811058243,
    "events": [
        {
            "eventId": "ISSUE",
            "projectId": "example-project",
            "timestamp": 1417811058
            "data": {
                "activities": [
                    {
                        "type": "UPDATED"
                        "date": 1417811057698,
                        "description": "Assign to alex.admin\nset Resolution
to FIXED\nset Status to RESOLVED\n",
                        "author": {
                            "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                            "loginName": "alex.admin",
                            "realName": "Alex Admin"
                        },
                        "issue": {
                            "id": 2,
                            "resolution": "FIXED",
                            "title": "Test Issue",
                            "type": "Feature",
                            "url": "http://test-server/#projects/example-
project/task/2",
                            "version": "1417811057698",
                            "asignee": {
                                "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                                "loginName": "alex.admin",
                                "realName": "Alex Admin"
                            },
                            "reporter": {
```

```
                                "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                                "loginName": "alex.admin",
                                "realName": "Alex Admin"
                            }
                        },
                        "fieldUpdates": [
                            {
                                "name": "assigned_to",
                                "newValue": "alex.admin",
                                "oldValue": ""
                            },
                            {
                                "name": "resolution",
                                "newValue": "FIXED",
                                "oldValue": ""
                            },
                            {
                                "name": "bug_status",
                                "newValue": "RESOLVED",
                                "oldValue": "UNCONFIRMED"
                            }
                        ]
                    },
                    {
                        "type": "COMMENTED"
                        "date": 1417811057929,
                        "description": "Feature is implemented",
                        "author": {
                            "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                            "loginName": "alex.admin",
                            "realName": "Alex Admin"
                        },
                        "comment": {
                            "author": {
                                "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                                "loginName": "alex.admin",
                                "realName": "Alex Admin"
                            },
                            "date": 1417811057929,
                            "text": "Feature is implemented",
                            "type": "UNKNOWN"
                        },
                        "task": {
                            "id": 2,
                            "resolution": "FIXED",
                            "title": "Test Issue",
                            "type": "Feature",
                            "url": "http://test-server/#projects/qa-
dev_example-project/task/2",
                            "version": "1417811057698",
                            "asignee": {
                                "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
```

```
                                "loginName": "alex.admin",
                                "realName": "Alex Alex Admin"
                            },
                            "reporter": {
                                "gravatarHash":
"8940829abebbc5d8d84e37af7161fd31",
                                "loginName": "alex.admin",
                                "realName": "Alex Admin"
                            }
                        }
                    }
                ]
            }
        }
    ]
  }
```

### PUSH Event

The `PUSH` event contains these fields:

| Field | Description |
|---|---|
| `refName` | Updated references |
| `commits` | Commits of the Push event |
| `sha` | Commit identifier |
| `comment` | Comment in the commit |
| `author` | Author of the commit |
| `date` | Timestamp of the commit |
| `parents` | List of commit parent identifiers |
| `repository` | Name of the repository to which the commit was pushed |

Here's a JSON payload example for a Git Push event:

```
  {
    "apiVersion": "1.0",
    "messageId": "c3378be6-6be5-4191-9b20-1fb5d429bfce",
    "projectId": "example-project",
    "testEvent": false,
    "timestamp": 1417810424512,
    "events": [
        {
            "eventId": "GIT_PUSH",
            "projectId": "example-project",
            "timestamp": 1417810424,
            "data": {
                "refName": "refs/heads/main",
                "commits": [
                    {
                        "sha": "32e03bc46a3a42eeab5dd25144a90c5b4f0b2e11",
                        "repository": "example-project.git",
                        "date": 1417810387000,
                        "comment": "file1.txt deleted, file3.txt created\n",
```

```
                        "author": {
                            "email": "alex.admin@example.com",
                            "firstName": "Alex",
                            "lastName": "Admin",
                            "username": "alex.admin"
                        },
                        "parents": [
                            "1106e8c81cb49e71024e9017235f89dc3983d4ee"
                        ]
                    },
                    {
                        "sha": "1106e8c81cb49e71024e9017235f89dc3983d4ee",
                        "repository": "example-project.git",
                        "date": 1417810290000,
                        "comment": "file2.txt updated\n",
                        "author": {
                            "email": "alex.admin@example.com",
                            "firstName": "Alex",
                            "lastName": "Admin",
                            "username": "alex.admin"
                        },
                        "parents": [
                            "8dab56fb6ba6dd0fc0d0aa8c7ce4f01d77fa0835"
                        ]
                    }
                ]
            }
        }
    ]
}
```


### BUILD Event

The `BUILD` event contains these fields:

| Field | Description |
|---|---|
| `jobName` | Name of the job |
| `timestamp` | Build timestamp |
| `number` | Build number |
| `url` | Build URL |
| `result` | Build result (`SUCCESS`/`UNSTABLE`/`FAILURE`/`NOT_BUILT`/`ABORTED`) |
| `duration` | Build duration |
| `fileName` | Name of the artifact |
| `relativePath` | Path relative to the job workspace |
| `url` | URL of the artifact |

Here's a JSON payload example for a Build event:


```
{
 "apiVersion":"1.0",
 "messageId":"4a253425-4598-4838-a4b5-aac30d0b9710",
```

```
   "timestamp":1417795613257,
   "testEvent":true,
   "projectId":"test-project",
   "events":[
      {
         "eventId":"BUILD",
         "projectId":"test-project",
         "timestamp":1417795613256,
         "data":{
            "jobName":"example-job",
            "details":{
               "timestamp":1417795590256,
               "number":16,
               "url":"http://server/test-dev/s2/test-project/hudson/job/test-
project.example-job/16/",
               "result":"SUCCESS",
               "duration":36905,
               "artifacts":[
                  {
                     "fileName":"sample-1.0-SNAPSHOT.jar",
                     "relativePath":"sample-project/target/sample-1.0-
SNAPSHOT.jar",
                     "url":"http://server/test-dev/s2/test-project/hudson/job/
test-project.example-job/16/artifact/sample-project/target/sample-1.0-
SNAPSHOT.jar"
                  }
               ]
            }
         }
      }
   ]
  }
```


### REVIEW Event

The `REVIEW` event represents changes in merge requests and contains these fields:

| Field | Description |
|---|---|
| `review` | Description of the merge request |
| `id` | Unique ID of the merge request |
| `title` | Title of the merge request |
| `created` | Timestamp of the merge request creation |
| `modified` | Timestamp of the merge request last modification |
| `reporter` | Profile of the user who created the merge request |
| `repository` | Name of the Git repository |
| `reviewBranch` | Name of the review branch |
| `targetBranch` | Name of the target branch |
| `user` | Profile of the user who performed the action |

| Field | Description |
|---|---|
| `action` | Merge request action<br>These are the merge request actions:<br>• `CREATED`: Merge request is created<br>• `COMMIT`: New commits are pushed to the review branch<br>• `MERGED`: Review branch is merged into the target branch<br>The `MERGED` action is created if the review branch is merged via the Merge button in the web user interface. If the review branch is merged from a Git client (such as the Git command line interface), no action is generated.<br>• `REVIEWED`: Reviewer approves or rejects a merge request<br>• `COMMENTED`: A comment is added to the merge request<br>• `CLOSED`: Merge request is closed |
| `commits` | List of commits added to the merge request<br>The `commits` field is generated for the `COMMIT` action. These fields are also generated for the `commits` action:<br>• `author`: Author of the commit<br>• `message`: Commit message<br>• `sha`: SHA-1 checksum hash of the commit |
| `text` | Text of the comment<br>The `text` field is generated for the `COMMENTED` action. |
| `comment` | Comment of the rejected or approved review action<br>The `comment` field is generated for the `REVIEWED` action. |
| `result` | Result of the merge (`FAST_FORWARD`, `FAST_FORWARD_SQUASHED`, `ALREADY_UP_TO_DATE`, `FAILED`, `MERGED`, `MERGED_SQUASHED`, `MERGED_SQUASHED_NOT_COMMITTED`, `CONFLICTING`, `ABORTED`, `MERGED_NOT_COMMITTED`, `NOT_SUPPORTED`, `CHECKOUT_CONFLICT`)<br>The `result` field is generated for the `MERGED` action. |
| `status` | Status of the merge request (`APPROVED`, `REJECTED`, `COMPLETED`, `CANCELLED`)<br>The `status` field is generated for the `REVIEWED` and the `CLOSED` action. |

Here's a JSON payload example for a `REVIEW` event.

```
{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "action": "CREATED",
                "review": {
                    "created": 1431944319181,
                    "id": 6,
                    "modified": 1431944319635,
                    "reporter": {
                        "email": "alex.admin@example.com",
                        "firstName": "Alex",
                        "lastName": "Alex Admin",
                        "username": "alex.admin"
                    },
                    "repository": "example-project.git",
```

```
                    "reviewBranch": "bug_branch",
                    "targetBranch": "main",
                    "title": "Bug Fix"
                },
                "user": {
                    "email": "alex.admin@example.com",
                    "firstName": "Alex",
                    "lastName": "Alex Admin",
                    "username": "alex.admin"
                }
            },
            "eventId": "REVIEW",
            "projectId": "example-project",
            "timestamp": 1431944327
        }
    ],
    "messageId": "08758261-e4e7-4c8f-b9fe-7b74f715803f",
    "projectId": "example-project",
    "testEvent": false,
    "timestamp": 1431944329923
}


{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "action": "COMMIT",
                "commits": [
                    {
                        "author": "alex.admin",
                        "message": "fix version #3\n",
                        "sha": "8fd1d2a53a181aa7015e7535b6f64295c432eca7"
                    },
                    {
                        "author": "alex.admin",
                        "message": "fix version #2\n",
                        "sha": "ff2bdf91d0fb6fb664315879ec38acc0931beeb6"
                    }
                ],
                "review": {
                    "created": 1431944319181,
                    "id": 6,
                    "modified": 1431944340209,
                    "reporter": {
                        "email": "alex.admin@example.com",
                        "firstName": "Alex",
                        "lastName": "Alex Admin",
                        "username": "alex.admin"
                    },
                    "repository": "example-project.git",
                    "reviewBranch": "bug_branch",
                    "targetBranch": "main",
                    "title": "Bug Fix"
                },
```

```
                "user": {
                    "email": "alex.admin@example.com",
                    "firstName": "Alex",
                    "lastName": "Alex Admin",
                    "username": "alex.admin"
                }
            },
            "eventId": "REVIEW",
            "projectId": "example-project",
            "timestamp": 1431944353
        }
    ],
    "messageId": "5de98d08-49cd-4a19-86b5-d89757f75a1d",
    "projectId": "example-project",
    "testEvent": false,
    "timestamp": 1431944355646
}

{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "user": {
                    "email": "clara.coder@example.com",
                    "firstName": "Clara",
                    "lastName": "Coder",
                    "username": "clara"
                },
                "review": {
                    "created": 1436521285722,
                    "id": 23,
                    "modified": 1438246154916,
                    "reporter": {
                        "email": "alex.admin@example.com",
                        "firstName": "Alex",
                        "lastName": "Admin",
                        "username": "alex"
                    },
                    "repository": "example-project.git",
                    "reviewBranch": "bug_branch",
                    "targetBranch": "main",
                    "title": "Some Review"
                },
                "action": "REVIEWED",
                "status": "REJECTED",
                "comment": "rejected the request because ...",
            },
            "eventId": "REVIEW",
            "projectId": "example-project",
            "timestamp": 1438246163
        }
    ],
    "messageId": "f0a75815-3470-4dc4-be82-975935152ed3",
    "projectId": "example-project",
    "testEvent": false,
```

```
    "timestamp": 1438246165924
}

{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "action": "COMMENTED",
                "review": {
                    "created": 1431944319181,
                    "id": 6,
                    "modified": 1431944478701,
                    "reporter": {
                        "email": "alex.admin@example.com",
                        "firstName": "Alex",
                        "lastName": "Alex Admin",
                        "username": "alex.admin"
                    },
                    "repository": "example-project.git",
                    "reviewBranch": "bug_branch",
                    "targetBranch": "main",
                    "title": "Bug Fix"
                },
                "text": "General comment",
                "user": {
                    "email": "alex.admin@example.com",
                    "firstName": "Alex",
                    "lastName": "Alex Admin",
                    "username": "alex.admin"
                }
            },
            "eventId": "REVIEW",
            "projectId": "example-project",
            "timestamp": 1431945965
        }
    ],
    "messageId": "d2a36692-dae6-44d4-a112-7a615b524cc3",
    "projectId": "example-project",
    "testEvent": false,
    "timestamp": 1431945967166
}

{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "action": "MERGED",
                "result": "FAST_FORWARD",
                "review": {
                    "created": 1431944319181,
                    "id": 6,
                    "modified": 1431944478701,
                    "reporter": {
                        "email": "alex.admin@example.com",
```

```
                        "firstName": "Alex",
                        "lastName": "Alex Admin",
                        "username": "alex.admin"
                    },
                    "repository": "example-project.git",
                    "reviewBranch": "bug_branch",
                    "targetBranch": "main",
                    "title": "Bug Fix"
                },
                "user": {
                    "email": "alex.admin@example.com",
                    "firstName": "Alex",
                    "lastName": "Alex Admin",
                    "username": "alex.admin"
                }
            },
            "eventId": "REVIEW",
            "projectId": "example-project",
            "timestamp": 1431945438
        }
    ],
    "messageId": "b06d5581-d38a-4972-9c80-dc1455547776",
    "projectId": "example-project",
    "testEvent": false,
    "timestamp": 1431945440287
}

{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "action": "CLOSED",
                "review": {
                    "created": 1431944319181,
                    "id": 6,
                    "modified": 1431945453967,
                    "reporter": {
                        "email": "alex.admin@example.com",
                        "firstName": "Alex",
                        "lastName": "Alex Admin",
                        "username": "alex.admin"
                    },
                    "repository": "example-project.git",
                    "reviewBranch": "bug_branch",
                    "targetBranch": "main",
                    "title": "Bug Fix"
                },
                "status": "COMPLETED",
                "user": {
                    "email": "alex.admin@example.com",
                    "firstName": "Alex",
                    "lastName": "Alex Admin",
                    "username": "alex.admin"
                }
            },
```

```
            "eventId": "REVIEW",
            "projectId": "example-project",
            "timestamp": 1431945459
        }
    ],
    "messageId": "b434f6d2-b5c7-4c0a-bab2-3e6614025865",
    "projectId": "example-project",
    "testEvent": false,
    "timestamp": 1431945453967
}
```

### ACTIVITY Event

The `ACTIVITY` event contains these fields:

| Field | Description |
|---|---|
| `author` | Profile of the user whose action produced the activity<br>The value is `null` for system activities. |
| `name` | Name of the activity |
| `properties` | Description of the activity, or the object whose fields depends on the `name` field.<br>These activities are supported:<br>• `BUILD`: Triggered when a build in the integrated Hudson server ends.<br>• `DEPLOYMENT`: Triggered when the application is deployed, undeployed, started, or stopped using Deploy page in the web user interface.<br>• `MEMBER`: Triggered when a user is added, removed, or role is updated.<br>• `REVIEW`: Triggered when a merge request is created, closed, or updated.<br>• `RSS`: Triggered when a new article is acquired from a registered feed.<br>• `SCM_COMMIT`: Triggered when a commit is pushed to a project repository.<br>• `SCM_REPO`: Triggered when a project repository is added or removed.<br>• `TASK`: Triggered when an issue is created or updated.<br>• `WIKI`: Triggered when a wiki page is created or updated. |

Here's a JSON payload example for an Activity event:

```
{
    "apiVersion": "1.0",
    "events": [
        {
            "data": {
                "author": {
                    "email": "alex.admin@example.com",
                    "firstName": "Alex",
                    "lastName": "Alex Admin",
                    "username": "alex.admin"
                },
                "name": "WIKI",
                "properties": {
                    "page": "New Page Title",
                    "type": "CREATED"
                }
            },
```

```
            "eventId": "ACTIVITY",
            "projectId": "example-project",
            "timestamp": 1432035029
        }
    ],
    "messageId": "45066d85-5a5c-4647-9a6c-43fc8e99481a",
    "projectId": "qa-dev_test-rss",
    "testEvent": false,
    "timestamp": 1432035031418
}
```

16

# Share and Use Code Snippets

Snippets host reusable code in files that can be used in the project and shared with other project members. A snippet could include a small block of reusable source code or text that could be incorporated into larger modules.

Content in snippet files doesn't have to be code, but it must be useful. This content could be notes that you want to share with project members, or something you want to keep private, such as a reminder to yourself. If a snippet is shared, project members can copy or download the snippet files and then use them in their own applications.

A snippet can contain several files. When you create a snippet, you can add only one file but, after creating the snippet, you can add additional files.

VB Studio sets these defaults for the maximum size limits for snippets:

- Maximum object size: 1MB
- Maximum repository size: 1GB

## Create and Manage Snippets

You can create a snippet from the Snippets page or from a text selection in a code editor. You can only add one file when you create the snippet, but you can add more later.

This is how to create and manage a snippet:

| Action | How To |
|---|---|
| Create a snippet | 1. In the left navigator, click **Snippets** <>.<br>2. Click **+ Create Snippet**.<br>3. On the Snippet Details page of the New Snippet wizard, enter the snippet name and description.<br>4. In **Visibility**, select **Private** if you don't want to share the snippet's files and keep them for personal use. Select **Shared** to share the snippet's files with project members.<br>5. To edit the snippet and add more files immediately after creating the snippet, select **Edit snippet when finished**.<br>6. Click **Next**.<br>7. In the Snippet Content page of the New Snippet wizard, enter content for the default file of the snippet.<br>If you don't enter content, an empty file `snippet1.txt` will be added to the snippet.<br>8. Click **Finish**. |

| Action | How To |
|---|---|
| Create a snippet from a selection | You can create a snippet from a file that's open in the code editor.<br>1. In the open file, select the text.<br>2. Right-click and select **New Snippet from Selection**.<br>3. On the Snippet Details page of the New Snippet wizard, enter the snippet name and description.<br>4. In **Visibility**, select **Private** if you don't want to share it and keep for personal use. Select **Shared** to share the snippet with project members.<br>5. To edit the snippet and add more files immediately after creating the snippet, select **Edit snippet when finished**.<br>6. In the Snippet Content page of the New Snippet wizard, enter content for the default file of the snippet.<br>If you don't enter content, an empty file is added to the snippet.<br>7. Click **Finish**. |
| Share or stop sharing a snippet | After you create a snippet, you may want to share it with your team members, or you may want to stop sharing it if it is already shared. You can set the share status for any snippet that you own.<br>To share a snippet, in the **My Snippets** view of the Snippets page, click **Share**. The icon changes to **Shared**. To stop sharing a snippet, click **Shared**. The icon changes to **Private**. |
| Edit a snippet's title | You can edit the title of a snippet that you own. On the Snippets page, click the snippet name, and then click **Edit**. |
| Delete a snippet | You can delete a snippet that you own. On the Snippets page, click the snippet name, and then click **Delete**. In the Delete Snippet dialog box, click **Yes** to confirm. |

# Add and Manage Files in a Snippet

You must be the snippet's creator to add or manage its files:

| Action | How To |
|---|---|
| Add a file | 1. Open the snippet.<br>2. Scroll down and after the snippet's files, click **Add File**.<br>3. In the header, enter the file name with extension. In the editor, enter the file's content.<br>The editor supports various code editing features such as autocomplete, indentation, syntax highlighting, code folding, and bracket matching.<br>4. After adding the content, scroll up and at the top of the page, click **Save**.<br>To save the updates and stay on the Edit Snippet page, select **Save** To save the updates and exit, select **Save and Exit**. |

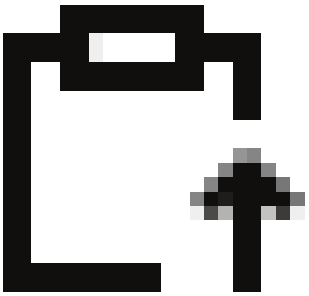
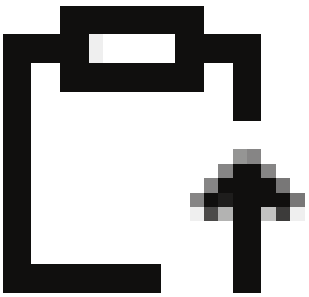
| Action | How To |
| --- | --- |
| Edit a file | 1. Open the snippet.<br>2. If necessary, rename the file and configure its properties.<br>3. In the editor, update the file contents.<br>4. At the top of the page, click **Save**.<br>To save the updates and stay on the Edit Snippet page, select **Save** To save the updates and exit, select **Save and Exit**. |
| Delete a file | 1. Open the snippet.<br>2. For the file that you want to delete, on the right side of the file header, click **Remove File** ⊗.<br>3. In the Delete Snippet File dialog box, click **Yes** to confirm.<br>4. At the top of the page, click **Save**.<br>To save the updates and stay on the Edit Snippet page, select **Save** To save the updates and exit, select **Save and Exit**. |

# Copy a Snippet File's Contents

There are two ways you can copy the contents of a snippet file, You can copy the contents manually from the Snippets page or you can insert them from the context menu in the code editor.

You'll be using the code editor when you edit a variety of pages and input fields: in the Edit Wiki page, when you're editing the readme file, using the snippet file editor, entering text in the merge request comment box, and filling out the shell command box on the Configure Build page.

To get the text you want to use from the snippet file:

| Action | How To |
|---|---|
| Copy from the Snippets page | 1. Open the snippet.<br>2. For the file whose contents you want to copy, click **Copy**  .<br>In some browsers, you must press Ctrl+C to copy the content to the clipboard after clicking **Copy**  .<br>3. Paste the contents into the text field. |
| Insert from the context menu | 1. In the code editor, right-click, and select **Insert from Snippet**.<br>2. In the Select Snippet page of the Insert From Snippet wizard, select the snippet, and click **Next**.<br>3. In the Select File page, click the file name whose contents you want to insert.<br>4. Click **Finish**. |

# Add a Comment to a Snippet

Add a comment to a snippet to share information with other developers.

To add a comment to a snippet:

1. Open the snippet.
2. Scroll down to the **Comments** section.
3. Enter the comment in the comment box.

   Use the project's wiki markup language to format the comment.
4. Click **Add**.

# Use Git with Snippets

You can use Git to clone a snippet repository and manage its files. After you clone the snippet's repository, you can view the file history, and update and commit files locally. However, you can only push updates to repositories for snippets that you own.

1. Open the snippet.
2. On the Snippet Details page, at the top, click **Clone**, and then click **Copy** to copy the **HTTP** or the **SSH** URL of the snippet repository.
3. Use Git commands to clone the repository, update files, and push the commits to the project.

   If the directory into which you want to clone the repository isn't empty, you'll need to create a new subdirectory and clone the repository into it. You can only perform a cloning operation into an empty directory.

# Download an Archive of the Snippet

You can download a zip or a tgz file of the snippet to your computer. You may want to do this if you back up the files of the snippet before deleting it.

1. Open the snippet.
2. On the Snippet Details page, at the top, click **Clone**, and the select **Download ZIP** or the **Download TGZ** option.

The downloaded file is an archive file that contains the latest content of all the files in the snippet. To view previous versions of the files, you must clone the snippet repository, and then use Git commands to show the history of the files.

17

# Co-Author Wikis

You can use Wikis to collaborate on your projects' documentation.

VB Studio supports these Wiki markup languages:

- Confluence

  See `http://www.atlassian.com/software/confluence/`.

- Textile

  See `http://textilewiki.com`.

- Markdown

  See `http://daringfireball.net/projects/markdown/`.

Project users can use the project's Wiki markup language to format content in wiki pages and in issue and merge request comments.

## Working with Markup Languages in VB Studio

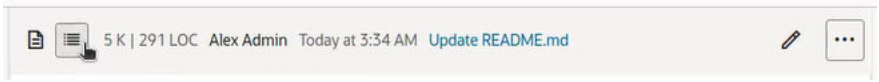
VB Studio's markdown parser/renderer provides an easy-to-use viewing and editing experience with markup, and uses markdown specifications that other sites, such as GitHub, support. The server generates a JSON model from markup and then the client side processes it, rendering the HTML in the VB Studio user interface. This results in markup rendering that's similar to and consistent with what other open-source software companies support. You can use markup in the Wiki, SCM, MR, Announcement, and Environment Detail pages.

You can read more about using these markup languages with VB Studio:

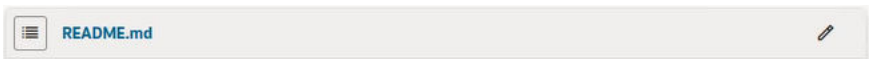
- Textile Syntax Reference
- Confluence Syntax Reference
- Markdown Syntax Reference

As you work with these markup languages in VB Studio, here are some helpful features to keep in mind:

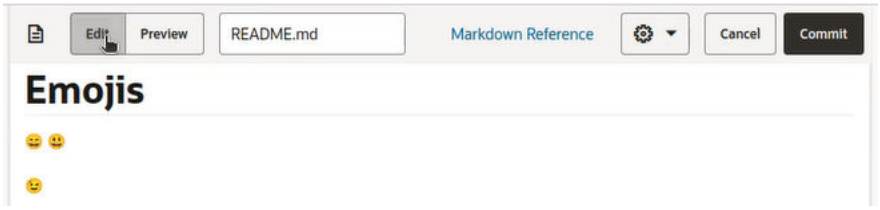
- The Table of Contents is always available on wiki pages (and in the Git page's File view). When the icon is clicked, the contents are displayed. No special coding is needed for this.




- An **Edit** button is available on the SCM `README.md` page to facilitate quick direct edits.

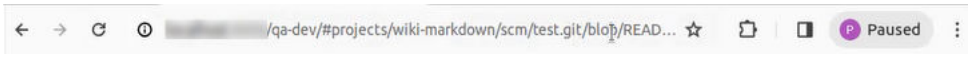


- A markup toolbar provides common formatting actions so that you don't have to remember or search for the markup language syntax you need. The toolbar is available in the Wiki, SCM, and MR pages.
- After making an edit, you can go to the **Preview** tab to look at it. Your edit is displayed immediately. You don't have to scroll down through the page you were editing to find your change. In the Preview tab, it should auto scroll to the line or section being edited.
- The header bar is sticky, so when you go from the **Edit** tab to the **Preview** tab to look at your changes, the header bar stays at the top of the page as well.




- Headers have link accessors, which are shown when you hover over the header.

When you click the header, the browser page's URL is updated. You can copy the URL and embed it in your pages, to provide a direct link to the header.




- Code in markup is shown with syntax coloring when the code language is specified in code blocks.

For example, for this Java code:

```
```java
public class Main {
  public static void main(String[] args) {
    if (20 > 18) {
      System.out.println("20 is greater than 18"); // obviously
    }
  }
}
```
```

The Java syntax highlighting in Confluence markup would be:

```
{code:language=java}
public class Main {
  public static void main(String[] args) {
    if (20 > 18) {
      System.out.println("20 is greater than 18"); // obviously
    }
  }
}
{code}
```

In Textile markup it would be:

```
bc[java].
public class Main {
  public static void main(String[] args) {
    if (20 > 18) {
      System.out.println("20 is greater than 18"); // obviously
    }
  }
}
```

- Empty lists in markdown are shown as empty lists.
- Line breaks for Markdown are standardized. The backslash (\) at the end of a line forces a line break. To create a paragraph, you don't have to enter the entire paragraph on one long line. Instead, enter it using many lines and then just toggle line wrap on or off.
  Line breaks in existing text blocks may not be displayed properly. This is a known issue.
- The use of images in Wiki comments is supported.

# Textile Syntax Reference

Textile is a lightweight markup language that uses a text formatting syntax to convert plain text into structured HTML markup. The syntax, a shorthand version of HTML, is designed to be easy to read and write.

**Phrase Modifiers**

| Markup | Output |
|---|---|
| *strong* | **strong** |

| Markup | Output |
|---|---|
| __italic__ | *italic* |
| **bold** | **bold** |
| ??citation?? | *citation* |
| -deleted text- | ~~deleted text~~ |
| +inserted text+ | inserted text |
| ^superscript^ | superscript |
| ~subscript~ | subscript |
| %span% | span |
| @code@ | `code` |

### Block Modifiers

| Markup | Output |
|---|---|
| hn. | Heading |
| bq. | Block quote |
| fnn. | Footnote |
| p. | Paragraph (optional) |
| bc. | Block code |
| pre. | Pre-formatted |
| # | Numeric list |
| * | Bulleted list |

### Links

```
"title":http://www.example.com
```

```
!images/logo.png!:http://www.example.com
```

### Images

```
!images/logo.png!
```

### Punctuation

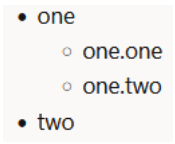
| Markup | Output |
|---|---|
| "quotes" | “quotes” |
| 'quotes' | ‘quotes’ |
| it's | it’s |
| em -- dash | — symbol (emdash) |
| en - dash | – symbol (endash) |
| 2 x 4 | 2 × 4 |

| Markup | Output |
|---|---|
| foo(tm) | foo™ |
| foo(r) | foo® |

### Lists

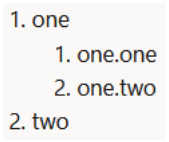
- **Bulleted Lists**

```
* one
** one.one
** one.two
* two
```




- **Numeric Lists**

```
# one
## one.one
## one.two
# two
```




### Tables

```
|_. heading 1|_. heading 2|_. heading 3|
|col A1|col A2|col A3|
|col B1|col B2|col B3|
```

### Alignment and Padding

| Markup | Output | Example |
|---|---|---|
| < | left text alignment | p<. |
| > | right text alignment | p>. |
| <> | justify text | p<>. |
| = | center text | p=. |
| ( | pad left | p(. |
| ) | pad right | p). |
| () | pad left and right | |

### Attributes

| Markup | Example |
|---|---|
| (class) | h1(foo). |
| (#id) | h1(foo). |
| {style} | h1{color:red}. |
| [language] | h1[en]. |

For example:

```
h1(main-heading){color: red}. header text
```

### Footnotes

```
reference[1]
fn1. footnote text
```

### Extended Blocks

```
bc..
bq..
pre..
```

### Acronyms

CSS (Cascading Style Sheets) generates CSS.

### Generated Content

| Markup | Description | Example |
|---|---|---|
| {toc} | Generates a table of contents. | {toc} or {toc:style=disc\|maxLevel=3} |
| {glossary} | Generates a glossary based on acronyms in the document. | {glossary} or {glossary:style=disc} |

## Confluence Syntax Reference

Confluence pages are where you capture all your important (and unimportant) information, starting with a blank page and adding rich text, tasks, images, links, and more.

### Phrase Modifiers

| Markup | Output |
|---|---|
| _emphasis_ | *emphasis* |
| *strong* | **strong** |
| ??citation?? | *citation* |
| -deleted text- | ~~deleted text~~ |

| Markup | Output |
| --- | --- |
| +inserted text+ | inserted text |
| ^superscript^ | superscript |
| ~subscript~ | subscript |
| {{monospace}} | `monospace` |

### Block Modifiers

| Markup | Description |
| --- | --- |
| hn. | Heading (where n is some digit, for example h1) |
| bq. | Block quote (single paragraph) |
| {quote} | Block quote (multiple paragraphs) |
| {code} | Block code |
| {note:title=Note} | A simple note |
| {warning:title=Caution} | A warning note |
| {info:title=Useful Information} | Informational note |
| {tip:title=Tip} | A help tip |
| {noformat} | Pre-formatted block |
| # | Numeric list |
| * | Bulleted list |

### Links

```
[Title|http://www.example.com]
[Title|http://www.example.com|tip]
```

### Anchors

| Markup | Description |
| --- | --- |
| {anchor:anchorname} | Creates anchor 'anchorname' |
| [Title|#anchorname] | Links to anchor 'anchorname' |

### Tables

```
||heading 1||heading 2||heading 3||
|col A1|col A2|col A3|
|col B1|col B2|col B3|
```

### Text Breaks

| Markup | Description |
| --- | --- |
| (empty line) | Produces a new paragraph |
| ---- | horizontal rule |
| --- | — symbol (emdash) |

| Markup | Description |
|---|---|
| -- | – symbol (endash) |

# Markdown Syntax Reference

Markdown uses fewer elements than other kinds of wiki markup. For elements not included in the syntax (such as underline, superscript, subscript, tables, notes), you can use actual HTML tags.

If you use special characters, such as & and <, Markdown automatically escapes them and produces the correct HTML code.

### Phrase Modifiers

| Markup | Output |
|---|---|
| *italic* or _italic_ | *italic* (em element) |
| **bold** or __bold__ | **bold** (strong element) |
| ~~deleted text~~ | ~~deleted text~~ |
| `code` | `code` |

### Tables

```
| Left alignment | Centre alignment | Right alignment |
| :---- | :----: | ----: |
| col A1 | col A2 | col A3 |
| col B1 | col B2 | col B3 |
```

The rendered output looks like this:

| Left Alignment | Center Alignment | Right Alignment |
|---|---|---|
| col A1 | col A2 | col A3 |
| col B1 | col B2 | col B3 |

### Block Modifiers

| Markup | Output |
|---|---|
| #, ##, ###, and so on | Heading (levels 1 through 6) |
| === (equals signs) under title | Level 1 heading |
| --- (hyphens) under title | Level 2 heading |
| > | Block quote (before the quoted line or lines; long lines will wrap) |
| ``` before and after code block, or indented 4 spaces or a tab | Block code |

### Lists

| Markup | Output |
|---|---|
| 1. (any digit) | Numeric list |

| Markup | Output |
|---|---|
| *, -, + | Bulleted list |

Blank lines between list items cause the list item text to be surrounded with paragraph elements.

To nest list items, indent the nested items two spaces.

To place paragraphs within list items, put a blank line above, and indent the text 4 spaces or one tab.

**Links**

Inline links consist of bracketed link text immediately followed by text in parenthesis. For example:

```
[Link text](http://www.example.com)
[Link text with title](http://www.example.com "Example.com's home page")
```

Reference links consist of bracketed link text followed by a reference (a number or other text in brackets). The reference is defined elsewhere in the document. For example:

```
[Link text][1]
[Link text][any text]
...
[1]: http://www.example.com
[any text]: http://www.example.com "Example.com's home page"
```

**Images**

Like links, images can be inline or reference. Inline images look like this (titles are allowed):

```
![link text](images/logo.png)
![link text](images/logo.png "My logo")
```

Reference images look like this:

```
![link text][id]
...
[id]: images/logo.png "My logo"
```

# Use the Markup Toolbar

The markup toolbar provides commonly used formatting actions so that you don't have to remember or search for the right markup language syntax you need (whether Textile, Confluence, or Markdown). The toolbar also provides options to customize the content editor, and is available when editing wiki pages, merge request comments, and the Git repository's readme file.

Use the markup toolbar to:

- Format text - add headings, and make text bold or italicized, for example
- Add elements like links, images, lists, and tables

- Add phrase or block modifiers when quoting text or adding code snippets
- Customize the editor to improve efficiency during content creation. You can:
  - Toggle line numbers on and off
  - Change line wrapping
  - Show (or hide) the content editor's minimap

When using the toolbar to add phrase or block modifiers, note that the and actions behave differently depending on what's selected.

For example:

- If a single line or text phrase is selected when clicking the action, a *phrase modifier* is wrapped around the selected text. In Confluence markup, for example, you'd see:
  ```
  {{something selected}}
  ```
- If multiple lines are selected, a *block modifier* is wrapped around the selected text. The Confluence markup would be:

  ```
  {code}
  multiline selected
  text
  {code}
  ```

- If no text is selected, a *template* is inserted:

  ```
  {code:language=java}

  Code goes here...
  more code...

  {code}
  ```

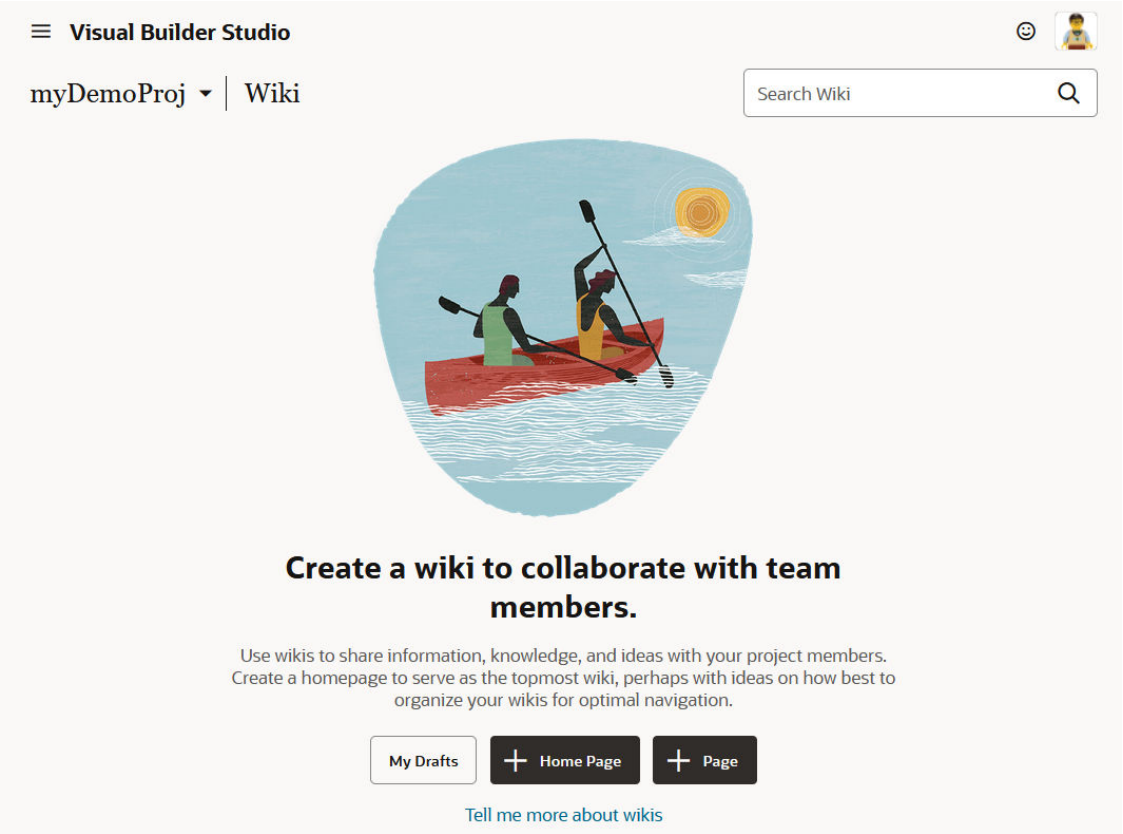
  Here's an example of a template inserted for a table, in Confluence markup:

  ```
  ||heading 1||heading 2||heading 3||
  |col A1|col A2|col A3|
  |col B1|col B2|col B3|
  ```

# Create and Manage Wiki Pages

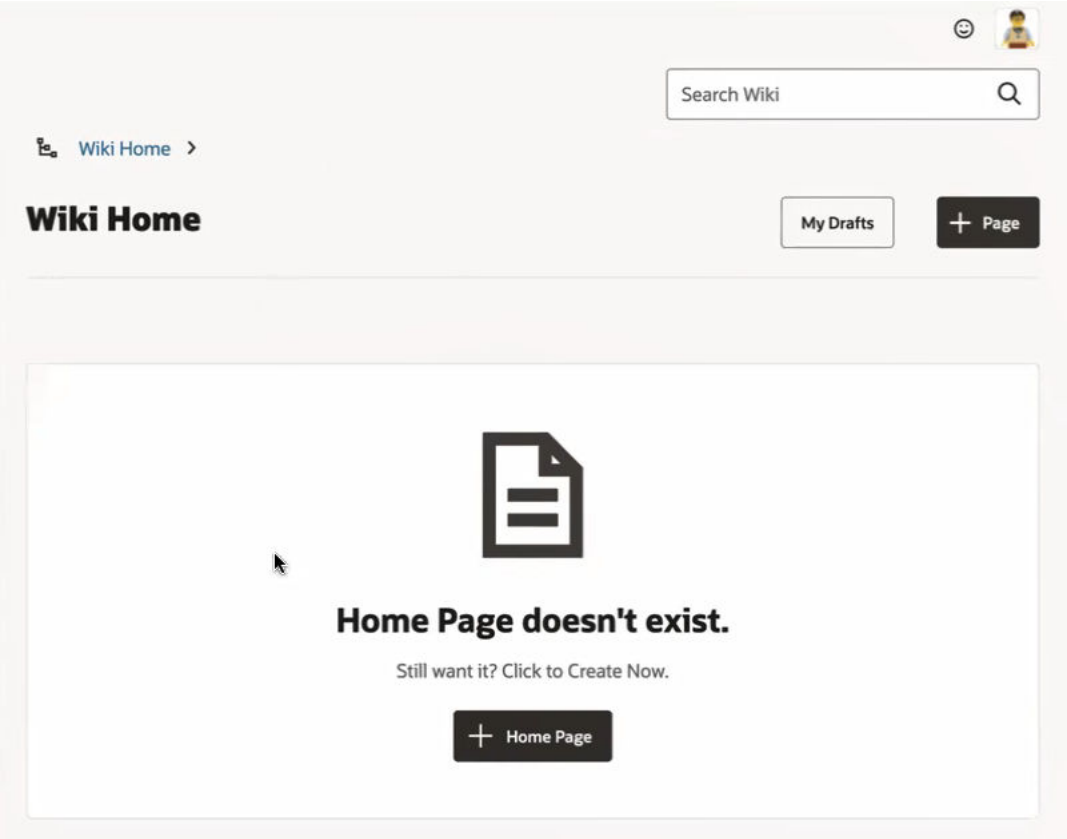
From the Wiki Home page, you can create and open wiki pages, add child wiki pages, add attachments, show or hide the tree view that displays the structure for the project's wiki pages, and delete or restore deleted wiki pages. From the Wiki Draft page, you can open and edit saved drafts, publish drafts, and delete any drafts that are no longer needed.

In the left navigator, click **Wiki** to display the top-level **Wiki** page.




From there, you can create a Wiki Home page, as well as create and manage your project's wiki pages and drafts. Any project user can create a wiki page. For a page that you didn't create, depending on the edit and delete access set by the creator of the wiki page, you may or may not be able to edit or delete it. However, if you're a project owner, you can always do that.

If a collection of wiki pages already exists, but has no Wiki Home page, you can click **+ Home Page** to create one. Before creating a new **Wiki Home** page for the collection, notice that the **Refresh**, **Edit**, and **Delete** buttons are not available.




There is no button for creating a child page, since all pages under the Wiki Home page are child pages. That's why you just see the **+ Page** button, to the right.

Here's what you need to know about creating and editing wiki pages:

| Action | How To |
|---|---|
| Create a wiki home page | 1. On the **Wiki** page, click **+ Home Page**.<br>2. The path field, where you see /, denotes the path to the Wiki Home page you want to create.<br>The default path cannot be modified.<br>3. In the **Page Text** tab, enter the content.<br>Use the project's wiki markup language to format the contents. To open the markup language's cheat sheet, click the reference link above the text area.<br>You can also use the markup toolbar for quick formatting. The toolbar provides frequently used actions so you can quickly and easily format text, or add elements like links, images, lists, tables, and more.<br>The toolbar also provides editor options so you can turn off line numbers, change line wrapping, and show (or hide) the wiki editor's minimap.<br>4. Click **Publish** to publish the new Wiki Home page.<br>If you click **Close**, you are prompted to retain a draft of the page you were working on or to delete the draft and discard your work.<br>If you click **Discard Draft**, the draft you were working on won't be saved. |

| Action | How To |
| --- | --- |
| Create a wiki page | 1. On the **Wiki** page, click **+ Create Page**.<br>2. In the field where you see the `Enter path` text, enter the full path to the page you want to create.<br>The path should consist of URL-safe characters, without any spaces or periods. Don't add a trailing slash at the end. There's no need to create folders. Just enter the full path you want to create and folders will be created automatically.<br>3. In the **Page Text** tab, enter the content.<br>Use the project's wiki markup language or the markup toolbar to format the contents. To open the markup language's cheat sheet, click the reference link above the text area.<br>4. Click **Save** to publish the new page.<br>Click **Close** and you'll be allowed to retain a draft of the page you were working on or delete the draft and discard your work. |
| Open a wiki page | To open a wiki page, click the wiki page title on the Wiki Home page or in the tree view.<br>**Tip:**<br>Click the **Show/Hide Wiki Tree View** icon to show or hide the wiki navigation tree that's displayed on the page's left side.<br>If a wiki page has child wiki pages, you can expand the parent (and any child pages that have child wiki pages) in the Wiki Home page or in the tree view by clicking the ▶ icon to the left of the wiki page title and select the child wiki page name to open it.<br>**Tip:**<br>Click the **Pages** link in the wiki tree to navigate back to the Wiki Home page.<br>You can collapse any expanded wiki page by clicking the ▼ icon to the left of its page title, in the page or tree view. |

| Action | How To |
| --- | --- |
| Edit a wiki page | To edit a page, click **Edit**. If a saved draft for the page exists, select **Resume Editing** to display the draft with the last saved changes or select **New Edit** to start editing the published page. Use the wiki's markup language or the markup toolbar to format the contents.<br><br>**Tip:**<br>To open the markup language's cheat sheet, click the reference link above the text area.<br><br>If the project is using the Markdown markup language:<br>• Use # followed by the issue ID to add a reference to the issue. For example, #12 will create a link to the issue ID 12.<br>• Use ! followed by the merge request ID to add a reference to the merge request. For example, !34 will create a link to the merge request ID 34.<br>• See `http://www.emoji-cheat-sheet.com/` for the complete list of supported emoticons and text patterns. |
| Edit a draft of a wiki page | 1. Above the wiki navigation tree that's displayed on the page's left side, click **Wiki Draft**.<br>You can also click the separator › to the right of the wiki tree view icon in the page's breadcrumbs.<br> <br><br>Select **Wiki Draft** and the **Wiki Draft** page will open.<br>2. Locate the draft of the wiki page you want to edit and click **Edit** under **Actions**. |
| Create a child wiki page | 1. Open the parent wiki page.<br>2. Click **+ Child**.<br>You can also create a child wiki page without opening or creating its parent wiki. On the **Wiki** page, click **+ Page**. In **Page Title**, enter the path of the child wiki page. For example, to create a `HelloWorld` child page of the `Welcome` page, enter `Welcome/HelloWorld` in the title. If the `Welcome` page doesn't exist, VB Studio will create both the `Welcome` and `HelloWorld` pages. Open the empty `Welcome` wiki page, click **Publish** to create the page, then add content to it. |
| Add an attachment | You can attach files of any type to a wiki page, including images, videos, docs, spreadsheets, and archived files.<br>1. Open the wiki page or the draft in edit mode.<br>2. Click the **Attachments** tab.<br>3. In the **Files to Attach or Update** section, click **Select**. Browse and select files to attach.<br>You can also drag and drop files to the drop area.<br>4. Click **Attach**.<br>5. Click **Save**. |

| Action | How To |
|---|---|
| Publish a draft of a wiki page | 1. On the Wiki Drafts page, locate the draft of the wiki page you want to publish.<br>2. Click **Publish** under **Actions**.<br>A draft of a new page can only be published if it contains a path. A draft of an existing page can only be published if the draft contains changes from the underlying page. |
| Delete a wiki page | 1. Open the wiki page<br>2. Click **Delete**.<br>3. In the Delete Wiki Page dialog box, select **I understand that my wiki page will be permanently deleted** and click **Delete**.<br>The wiki page along with its comments is deleted. An activity with a link to restore the wiki page (if necessary) is added to the recent activities feed of the **Project Home** page.<br>If you delete a parent wiki page, its child wiki pages aren't deleted. The parent wiki page name continues to appear in the breadcrumb path.<br>To delete a draft of a wiki page, from the Wiki Drafts page, click **Delete** under **Actions** . In the Delete Wiki Page dialog box, select **I understand that my wiki draft will be permanently deleted**, and click **Delete draft**. |
| Restore a wiki page | 1. Click the deleted wiki name in the recent activity feed or any other wiki page where it was referenced.<br>2. Click **Restore**.<br>The wiki page, along with all comments and attachments, will be restored to its original path.<br>To restore a parent wiki page, open it and click **Restore**. |

# Add Comments to a Wiki Page

Only project members can add comments to a private project. In a shared project, however, any user in organization can add comments.

This is how to add general or block comments to a wiki page:

| Action | How To |
|---|---|
| Add a generic comment | 1. Open the wiki page.<br>2. Scroll down to the Comments section. In the **Write** tab, enter your comment, and click **Comment**.<br>You can use the project's wiki markup language to format your comment.<br>The comment appears in a conversation box along with icons to **Reply**, **Edit**, and **Delete**.<br>You can't edit comments entered by other users. You can't delete a parent comment if there are any child comments, or delete comments entered by other users. |

| Action | How To |
|---|---|
| Add a block comment | A block comment is a comment that you add to a content block, such as a paragraph or a table.<br>1. Open the wiki page.<br>2. Move the mouse pointer to the right edge of the block, and click **Add Comment** .<br>If you see a number on the right edge of the block, it indicates the number of existing comments of the block. Click the number to view the comments and then click **Leave a comment**.<br>3. In the popup, add your comment, and click **Post**.<br>When you add a block comment, a watch is set on the wiki page and you automatically get email notifications of future comments and updates.<br>To hide the comment, click the comment number or **Hide** . To edit or delete the comment, click the number and then click **Edit** or **Delete**.<br>If you edit, format, or move a content block, its comments move automatically. If you remove a content block, its comments are deleted too and can't be restored. If you merge a block with another, the comments of the source block will be hidden, but the comments of the target block remain visible. When you the split the source block from the target block and move it back its original position, the hidden comments reappear. |

# Watch a Wiki Page

You can set a watch on a wiki page and receive email notifications whenever someone updates the page or adds a comment.

To receive email notifications, in the User Preferences page, click the **Notifications** tab, and then select **Wiki page updates and comments**.

| Action | How To |
|---|---|
| Watch a wiki page | 1. Open the wiki page.<br>2. Click **Watch Wiki**.<br>3. In the menu, select the **Watch Page** check box.<br>The button's label changes to **Watching**. To stop watching the page, click **Watching**, and deselect the **Watch Page** check box. |
| Watch all pages of the project | On the Wiki page, click **Watch Wiki**.<br>The button label changes to **Watching**. To stop watching all pages, click **Watching**. |

# View a Wiki Page's History and Compare Versions

Each time you save a wiki page, VB Studio creates a version of it.

You can view the contents of a previous version of a wiki page, compare any two versions, restore the wiki page to a particular version, or delete a particular version:

| Action | How To |
|---|---|
| View the history of a wiki page, including all of its versions | 1. Open the wiki page.<br>2. Click the **Page History** tab. |
| View the contents of a previous version | In the **Page History** tab, in the version number row, click **View**. |
| Compare two versions | In the **Page History** tab, select the check boxes of versions, and click **Compare selected versions**. |
| Restore the wiki to a version | In the **Page History** tab, in the version number row, click **Restore this version**. In the Revert Wiki Page dialog box, select the **I understand that my wiki page will be updated to this revisions' content** check box and click **Yes**.<br>Restoring a page creates a new version of the page with the contents from the version you want to restore. You can't restore a wiki page if you don't have its edit access. |

# Wiki Administration

Project owners and wiki page creators can make some administrative updates for wikis. These updates include changing edit and delete access rights for a wiki page and changing a project's wiki markup language.

> **Note:**
>
> Only an organization administrator can change an organization's *default* wiki markup language (see Set the Organization's Default Wiki Markup Language).

## Configure Edit and Delete Rights for Wiki Pages

You can set up edit permissions so users can edit the wiki content, create child pages, or restore deleted versions. You can set up delete permissions so users can delete a wiki page or remove a version from the **Page History** tab.

Access rights are granted to a project role, not to a particular user. A project owners can always edit or delete any wiki page and can grant edit and delete rights to other roles. If you created a wiki page, you can assign its edit and delete access to other roles.

As a project member, you could lose your edit or delete rights for the page you create if you allow other users to edit its access rights and then another user changes the edit or delete rights to **Owners** only.

To change edit and delete permissions for a wiki page:

1. Open the wiki page.
2. Click **Edit**.
3. Click the **Access Rights** tab.
4. From the **Edit Access** list or the **Delete Access** list, select the project role. By default, it's set to **Members and Owners**.

| Select this option ... | To assign the access to ... |
|---|---|
| **All** | All organization users. Use this option to enable all users in your organization to edit or delete the page. |
| **Members and Owners** | All project users. In a shared project, users in your organization can view the wiki page, but they can't edit or delete it. |
| **Owners** | Project owners only. Project members can view the page, but they can't edit or delete it. In a shared project, organization users can also view the wiki page, but can't edit or delete it. |

5. Click **Save**.

## Change a Project's Wiki Markup Language

A project's wiki markup language is defined when the project is created. If you're a project owner, you can change the markup language for new pages and comments, but existing wiki pages and comments will continue to use the original wiki markup language.

Here's how to change a project's wiki markup language:

1. In the left navigator, click **Project Administration** .
2. Click **Properties**.
3. In the Properties page, from **Markup Language**, select the wiki markup language.

# Part IV
# Troubleshooting

The information in this part should help you troubleshoot problems you encounter when using Oracle Visual Builder Studio.

**Topics:**


- Troubleshooting VB Studio Issues


If you're looking for troubleshooting topics for building extensions in the Designer, see Troubleshooting and FAQs; when building responsive apps in the Designer, see Troubleshooting.

18

# Troubleshooting VB Studio Issues

Look up how to troubleshoot and resolve issues you may encounter as you use Visual Builder Studio.

## Fix OCI Storage Issues

If you notice any storage issues while archiving artifacts, they could be caused by problems with the connection to OCI Object Storage. Contact your organization administrator to verify the connection. To find your organization administrator, click Contacts under your user profile. Your administrator, or a list of administrators, will be displayed.

If you see an error message that indicates your storage is full, you should contact the OCI Administrator to increase the storage limit. Limits are described in Service Limits. You could also try to remove unnecessary artifacts to free up storage, forgoing any need to increase current storage limits.

## Fix “E-mail address of PersonIdent must not be null" Error

If you see the "E-mail address of PersonIdent must not be null" error message when you create or update a merge request, it may mean that you haven't specified your email address in your profile. Access your profile, verify your user email address, then retry the merge request.

For information about how to provide and verify your email address, see Update Your Email Address. You'll also need to ensure that you've subscribed to merge request notifications. For details on how to do this, see Configure Your Global Email Notifications.