Oracle® Communications Service Catalog and Design

Solution Designer Installation Guide

Release 8.2
G18850-01
March 2025

ORACLE

Oracle Communications Service Catalog and Design Solution Designer Installation Guide, Release 8.2

G18850-01

Contents

Preface

This document describes how to install Oracle Communications Service Catalog and Design - Solution Designer.

Audience

This document is intended for DevOps administrators and those involved in installing and maintaining Oracle Communications Service Catalog and Design - Solution Designer. The person installing the software must be familiar with the following topics:

- Operating system commands
- Kubernetes
- Podman

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

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.

1

Overview of the Solution Designer Installation

Get an overview of Oracle Communications Service Catalog and Design - Solution Designer cloud native deployment, architecture, and the Solution Designer cloud native toolkit.

This chapter provides an overview of Solution Designer deployed in a cloud native environment using container images and a Kubernetes cluster.

About Solution Designer Cloud Native Deployment

You can deploy Oracle Communications Service Catalog and Design - Solution Designer in a Kubernetes-based shared cloud (cluster) while implementing modern DevOps “Configuration as Code” principles to manage system configuration in a consistent manner. You can automate system lifecycle management. You set up your own cloud native environment and can then use the Solution Designer cloud native toolkit to automate the deployment of Solution Designer instances. By leveraging the pre-configured Helm charts, you can deploy Service Catalog and Design - Solution Designer instances quickly ensuring your services are up and running in far less time.

Service Catalog and Design - Solution Designer supports the following deployment models:

- **On Private Kubernetes Cluster**: Solution Designer cloud native is certified for a general deployment of Kubernetes.
- **On Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE)**: Solution Designer cloud native is certified to run on Oracle's hosted Kubernetes OKE service.

The Solution Designer deployment has the microservices architecture using Helidon. Each service is self-contained and implements a single business capability within Solution Designer. The Solution Designer includes the following services:

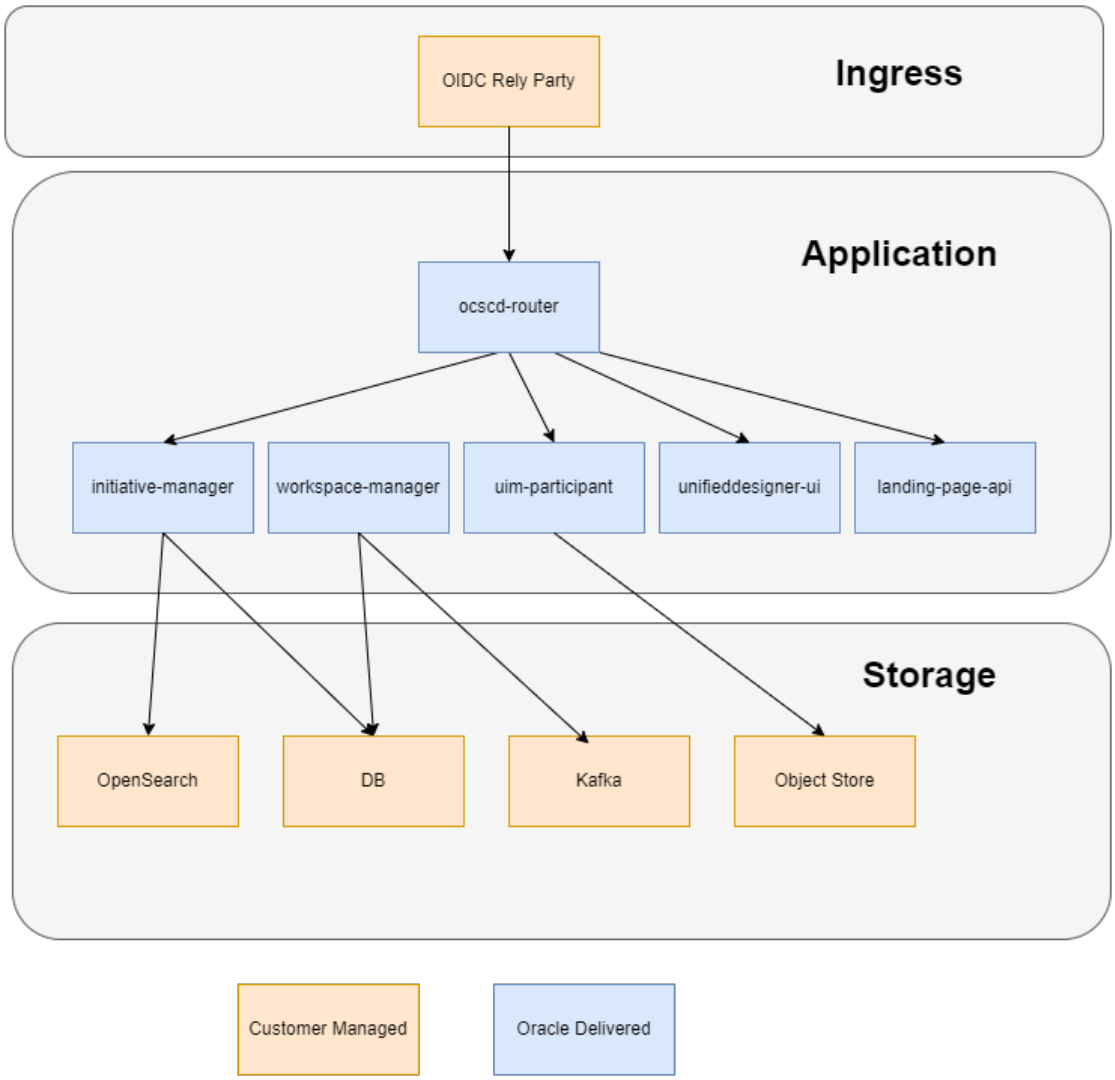
- Initiative Manager
- Workspace Manager
- Landing Page API
- UI
- Kafka
- OpenSearch
- UIM Participant
- Oracle PDB or ADB
- Headless Design Studio

Solution Designer Architecture

This section describes and illustrates the Solution Designer cloud native architecture and the deployment environment.

The following diagram illustrates the Solution Designer cloud native architecture.

Figure 1-1 Solution Designer Architecture



The Solution Designer cloud native architecture requires components such as the Kubernetes cluster. The Solution Designer cloud native artifacts include the container images built using Podman and the Solution Designer cloud native toolkit.

About the Solution Designer Container Image Toolkit

The Solution Designer image builder toolkit (**ocscd-image-builder.tgz**) creates an image with a base image as Linux 8 and installs prerequisite packages, Java, and the microservices of Solution Designer. The toolkit contains scripts to install the required packages and installers. The builder toolkit includes manifest files and scripts to build the images.

About Instances

An instance is a function of Solution Designer. Examples of those functions include Initiative manager, workspace manager, landing page, and so on. Each function contains the corresponding scripts and jar files. For example, the initiative manager contains the scripts and jar files to build the images of the initiative manager.

About the Solution Designer Cloud Native Toolkit

The Solution Designer toolkit (**ocscd-cntk.tgz**) is an archive file that includes utility scripts and samples to deploy Solution Designer in a cloud native environment.

Contents of the Solution Designer Toolkit

The Solution Designer toolkit contains the following artifacts:

- Helm charts for Solution Designer:
  The Helm chart for Solution Designer is located in the **ocscd-cntk** directory.
- Scripts to manage the lifecycle of microservices of Services Catalog and Design - Solution Designer.

About Helm Overrides

The specifications of the Solution Designer deployment are consumed from the **manifest.yaml** file. The values defined in the **manifest.yaml** file are used by deployment, service, and ingress yaml files.

About Samples for Solution Designer

The samples file (**ServiceModelSamples.zip**) contains the example service models that you can import into the Solution Designer application. After you import, you can use it as is or modify the models based on your business requirements. The sample models are available in JSON file format that can be imported into the Solution Designer application. See "Importing PSR Models" in the *Solution Designer User's Guide* on how to import the samples files into Solution Designer.

2

Planning and Validating Your Cloud Environment

In preparation for Oracle Communications Service and Catalog Design - Solution Designer cloud native deployment, you must set up and validate prerequisite software. This chapter provides information about planning, setting up, and validating the environment for Solution Designer cloud native deployment.

See the following topics:

- Required Components for Solution Designer Deployment
- Planning Your Cloud Native Environment
- Planning Your Container Engine for Kubernetes (OKE) Cloud Environment

Required Components for Solution Designer Deployment

In order to run, manage, and monitor the Solution Designer deployment, the following components and capabilities are required. These must be configured in the cloud environment:

- Kubernetes Cluster
- Oracle Multitenant Container Database (CDB)
- Container Image Management
- Helm
- Load Balancer
- Domain Name System (DNS)
- Persistent Volumes
- Authentication
- Secrets Management
- Object Store

For details about the required versions of these components, see *Service Catalog and Design Compatibility Matrix*.

In order to utilize the full flexibility, reliability and value of the deployment, the following aspects must also be set up:

- Continuous Integration (CI) pipelines for custom images
- Continuous Delivery (CD) pipelines for creating, scaling, updating, and deleting instances of the cloud native deployment

Planning Your Cloud Native Environment

This section provides information about planning and setting up Solution Designer cloud native environment. As part of preparing your environment for Solution Designer cloud native, you

choose, install, and set up various components and services in ways that are best suited for your cloud native environment. The following sections provide information about each of those required components and services, the available options that you can choose from, and the way you must set them up for your cloud native environment.

Setting Up Your Kubernetes Cluster

For Solution Designer, Kubernetes worker nodes must be capable of running Linux 8.x pods with software compiled for Intel 64-bit cores. A reliable cluster must have multiple worker nodes spread over separate physical infrastructure and a very reliable cluster must have multiple master nodes spread over separate physical infrastructure.

Solution Designer requires Kubernetes.

To check the Kubernetes version, run the following command:

```
kubectl version
```

Typically, Kubernetes nodes are not used directly to run or monitor Kubernetes workloads. You must reserve worker node resources for the execution of Kubernetes workload. However, multiple users (manual and automated) of the cluster require a point from which to access the cluster and operate on it. This can be achieved by using kubectl commands (either directly on command line and shell scripts or through Helm) or Kubernetes APIs. For this purpose, set aside a separate host or set of hosts. Operational and administrative access to the Kubernetes cluster can be restricted to these hosts and specific users can be given named accounts on these hosts to reduce cluster exposure and promote traceability of actions.

Typically, the Continuous Delivery pipeline automation deploys directly on a set of such operations hosts (as in the case of Jenkins) or leverage runners deployed on such operations hosts (as in the case of GitLab CI). These hosts must run Linux, with all interactive-use packages installed to support tools such as Bash, Wget, cURL, Hostname, Sed, AWK, cut, and grep. An example of this is the Oracle Linux 8.7 image (Oracle-Linux-8.7) on Oracle Cloud Infrastructure.

In addition, you need the appropriate tools to connect to your overall environment, including the Kubernetes cluster. For instance, for a Container Engine for Kubernetes (OKE) cluster, you must install and configure the Oracle Cloud Infrastructure Command Line Interface.

Additional integrations may need to include Authentication software such as Keycloak for users to be able to login to this host, appropriate NFS mounts for home directories, security lists and firewall configuration for access to the overall environment, and so on.

Synchronizing Time Across Servers

It is important that you synchronize the date and time across all machines that are involved in testing, including client test drivers and Kubernetes worker nodes. Oracle recommends that you do this using Network Time Protocol (NTP), rather than manual synchronization, and strongly recommends it for Production environments. Synchronization is important in inter-component communications and in capturing accurate run-time statistics.

Provisioning Oracle Multitenant Container Database (CDB)

Solution Designer deployment architecture is best supported by the multitenant architecture that enables an Oracle database to function as a multitenant container database (CDB).

A container database is either a Pluggable Database (PDB), Autonomous Database (ADB) or the root container. The root container is a collection of schemas, schema objects, and non-schema objects to which all PDBs or ADB's belong. A PDB container contains the Solution Designer schema. Solution Designer requires access to PDBs in an Oracle 19*c* Multitenant database. For more information about the benefits of Oracle Multitenant Architecture for database consolidation, see *Oracle Database Concepts Guide*.

Provisioning an Empty PDB

To create an empty PDB:

1. Run the following SQL commands using the sysdba account for the CDB:

```
CREATE PLUGGABLE DATABASE _replace_this_text_with_db_service_name_ ADMIN
USER _replace_this_text_with_admin_name_ IDENTIFIED BY
"database_administrator_password" DEFAULT TABLESPACE "USERS" DATAFILE
'+DATA' SIZE 5M REUSE AUTOEXTEND ON;
ALTER PLUGGABLE DATABASE _replace_this_text_with_db_service_name_ open
instances = all;
ALTER PLUGGABLE DATABASE _replace_this_text_with_db_service_name_ save
state instances = all;
alter session set container=_replace_this_text_with_db_service_name;
GRANT CREATE ANY CONTEXT TO SYS WITH ADMIN OPTION;
GRANT CREATE ANY CONTEXT TO _replace_this_text_with_admin_name_ WITH ADMIN
OPTION;
GRANT CREATE ANY VIEW TO _replace_this_text_with_admin_name_ WITH ADMIN
OPTION;
GRANT CREATE SNAPSHOT TO _replace_this_text_with_admin_name_ WITH ADMIN
OPTION;
GRANT CREATE SYNONYM TO _replace_this_text_with_admin_name_ WITH ADMIN
OPTION;
GRANT CREATE TABLE TO _replace_this_text_with_admin_name_ WITH ADMIN
OPTION;
GRANT CREATE USER TO _replace_this_text_with_admin_name_ WITH ADMIN OPTION;
GRANT CREATE VIEW TO _replace_this_text_with_admin_name_ WITH ADMIN OPTION;
GRANT CREATE materialized view to _replace_this_text_with_admin_name_;
GRANT GRANT ANY PRIVILEGE TO _replace_this_text_with_admin_name_ WITH
ADMIN OPTION;
GRANT QUERY REWRITE TO _replace_this_text_with_admin_name_ WITH ADMIN
OPTION;
GRANT UNLIMITED TABLESPACE TO _replace_this_text_with_admin_name_ WITH
ADMIN OPTION;
GRANT SELECT ON SYS.DBA_TABLESPACES TO _replace_this_text_with_admin_name_
WITH GRANT OPTION;
GRANT SELECT ON SYS.V_$PARAMETER TO _replace_this_text_with_admin_name_
WITH GRANT OPTION;
GRANT SELECT on SYS.dba_jobs to _replace_this_text_with_admin_name_ with
grant option;
GRANT "CONNECT" TO _replace_this_text_with_admin_name_ WITH ADMIN OPTION;
GRANT "DBA" TO _replace_this_text_with_admin_name_ WITH ADMIN OPTION;
GRANT "EXP_FULL_DATABASE" TO _replace_this_text_with_admin_name_ WITH
ADMIN OPTION;
GRANT "IMP_FULL_DATABASE" TO _replace_this_text_with_admin_name_ WITH
ADMIN OPTION;
GRANT "RESOURCE" TO _replace_this_text_with_admin_name_ WITH ADMIN OPTION;
GRANT EXECUTE ON SYS.DBMS_LOCK TO _replace_this_text_with_admin_name_ WITH
```

```
GRANT OPTION;
grant execute on utl_file to _replace_this_text_with_admin_name_ with
grant option;
grant sysdba to _replace_this_text_with_admin_name_;
ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'tag' FORCE KEYSTORE
IDENTIFIED BY "_replace_this_text_with_sys_password_" WITH BACKUP USING
'_replace_this_text_with_db_service_name_backup_';
```

2. Log into the PDB as the sysdba account for the PDB (defined by the "_replace_this_text_with_admin_name_" parameter in the above commands) and adjust the PDB tablespace by running the following command:

> **Note:**
>
> In the command, replace DATA with the proper name from v$asm_diskgroup.

```
create tablespace USERS datafile '+DATA' size 1024m reuse autoextend on next 64m;
```

Service Catalog and Design uses the USERS tablespace by default. You can choose to name the tablespace as per your requirements. If securing the data is a requirement, the recommended approach is to use transparent data encryption (TDE) to encrypt the tablespaces used to store data.

Provisioning an ADB

To use Autonomous Database (ADB) for Service Catalog and Design database, you can create an Autonomous Database of type **Autonomous Transaction Processor** that is using database version 19c. After you create the ADB, download the database Wallet.

To create ADB and download the Database wallet:

1. Create an Autonomous Database. See "Provision Autonomous Database" for instructions about creating an Autonomous Database.

> **Note:**
>
> While creating the ADB, choose a workload type as **Transaction Processing**.

2. Download the database wallet.

   To download the database wallet:

   a. In Oracle Cloud Infrastructure console, go to the ADB page. Click **Database Connections**.

      From the Download client credentials (Wallet) section, download an Instance Wallet.

   b. In the **Connection Strings** section, note the TNS Name that ends in _tp. For example, jkddddd9sfeq1ug4_tp. This is needed during the Solution Designer provisioning.

   c. Select the **Wallet Type** as Instance Wallet.

   d. Click **Download wallet**.

      Choose a Wallet password and download the wallet. Remember the password you chose.

> **Note:**
>
> This downloads a zip file which is required while setting up the Database connection details in Service Catalog and Design.

e. Save and unpack the zip file on the system you will be running the CNTK.

About Container Image Management

A Solution Designer cloud native deployment generates container images for the microservices and its database installer. Additionally, images are downloaded for the ingress (for example, OIDC Relying Party).

Oracle highly recommends that you create a private container repository and ensure that all nodes have access to that repository. Images are saved in this repository and all nodes would then have access to the repository. This may require networking changes (such as routes and proxy) and include authentication for logging in to the repository. Oracle recommends that you choose a repository that provides centralized storage and management of not just container images, but also other artifacts.

Failing to ensure that all nodes have access to a centralized repository will mean that images have to be synchronized to the hosts manually or through custom mechanisms (for example, using scripts), which are error-prone operations as worker nodes are commissioned, decommissioned or even rebooted. When an image on a particular worker node is not available, then the pods using that image are either not scheduled to that node, wasting resources, or failing on that node. If image names and tags are kept constant (such as myapp:latest), the pod may pick up a pre-existing image of the same name and tag, leading to unexpected and hard to debug behaviors.

Installing Helm

Solution Designer cloud native requires Helm, which delivers reliability, productivity, consistency, and ease of use.

In the cloud native environment, using Helm enables you to achieve the following:

- You can apply custom domain configuration by using a single and consistent mechanism, which leads to an increase in productivity.
- You can change the Solution Designer domain configuration with Helm using the following features:
  - Helm Lint allows pre-validation of syntax issues before changes are applied.
  - Multiple changes can be pushed to the running instance with a single upgrade command.
  - Configuration changes may map to updates across multiple Kubernetes resources (such as deployments, config maps and so on). With Helm, you merely update the Helm release and its responsibility to determine which Kubernetes resources are affected.
- Including configuration in Helm charts allows the content to be managed as code, through source control, which is a fundamental principle of modern DevOps practices.

In order to co-exist with older Helm versions in production environments, Solution Designer requires Helm 3.7 or later saved as **helm** in PATH.

> **Note:**
>
> The Helm version mentioned in the commands is used as an example only. See *Service Catalog and Design Compatibility Matrix* for the recommended versions.

The following text shows sample commands for installing and validating Helm:

```
$ cd some-tmp-dir
$ wget https://get.helm.sh/helm-v3.7.0-linux-amd64.tar.gz
$ tar -zxvf helm-v3.7.0-linux-amd64.tar.gz
# Find the helm binary in the unpacked directory and move it to its desired
destination. You need root user.
$ sudo mv linux-amd64/helm /usr/local/bin/helm

# verify Helm version
$ helm version
version.BuildInfo{Version:"v3.7.0",
GitCommit:"eeac83883cb4014fe60267ec6373570374ce770b",GitTreeState:"clean",
GoVersion:"go1.16.8"}
```

Helm leverages **kubeconfig** for users running the `helm` command to access the Kubernetes cluster. By default, this is $HOME/.**kube/config**. Helm inherits the permissions set up for this access into the cluster. You must ensure that sufficient cluster permissions are granted to users running Helm.

About Load Balancing

Each Solution Designer cloud native instance is a Kubernetes cluster. To access application endpoints, you must enable HTTP/S connectivity to the cluster through an appropriate mechanism. This mechanism must be able to route traffic to the appropriate Solution Designer cloud native instance in the Kubernetes cluster.

For Solution Designer cloud native, ingress endpoint is required to expose appropriate services from the Solution Designer cluster and direct traffic appropriately to the cluster members. An external load balancer is an optional add-on.

An external load balancer serves to provide a highly reliable singe-point access into the services exposed by the Kubernetes cluster. In this case, this would be the NodePort services exposed by the Relying Party on behalf of the Solution Designer cloud native instance. Using a load balancer removes the need to expose Kubernetes node IPs to the larger user base, and insulates the users from changes (in terms of nodes appearing or being decommissioned) to the Kubernetes cluster. It also serves to enforce access policies. In the case of HTTPS, the load balance mechanism must enable TLS and handle it appropriately.

Using Domain Name System (DNS)

A Kubernetes cluster can have many routable entrypoints. Common choices are:

- External load balancer (IP and port)
- Ingress service (master node IPs and ingress port)
- Ingress service (worker node IPs and ingress port)

You must identify the proper entrypoint for your Kubernetes cluster.

Solution Designer cloud native requires hostnames to be mapped to routable entrypoints into the Kubernetes cluster. Regardless of the actual entrypoints (external load balancer, Kubernetes master node, or worker nodes), users who need to communicate with the Solution Designer cloud native instances require name resolution. The Solution Designer cloud native instances must have the DNS entry and SSL certificate to secure it.

Configuring Kubernetes Persistent Volumes

Typically, runtime artifacts in Solution Designer cloud native are created within the respective pod filesystems. As a result, they are lost when the pod is deleted.

While this impermanence may be acceptable for highly transient environments, it is typically desirable to have access to these artifacts outside of the lifecycle of the Solution Designer cloud native instance. It is also highly recommended to deploy a tool chain for logs to provide a centralized view with a dashboard. To allow for artifacts to survive independent of the pod, Solution Designer cloud native allows for them to be maintained on Kubernetes Persistent Volumes.

Solution Designer cloud native does not dictate the technology that supports Persistent Volumes, but provides samples for NFS-based persistence. Additionally, for Solution Designer cloud native on an Oracle OKE cloud, you can use persistence based on File Storage Service (FSS). It is recommended to have at least 25 GB of free space.

About NFS-based Persistence

For use with Solution Designer cloud native, one or more NFS servers must be designated.

In general, ensure that the instances have dedicated NFS support, so that they do not compete for disk space or network IOPS with others.

The exported filesystems must have enough capacity to support the intended workload. Given the dynamic nature of the Solution Designer cloud native instance, it is prudent to put in place a set of operational mechanisms to:

- Monitor disk usage and warn when the usage crosses a threshold
- Clean out the artifacts that are no longer needed

About Authentication

You must configure an identity provider to access the Solution Designer application. For authenticating users, Identity and Access Management (IAM) such as KeyCloak can be used. These users can perform the Solution Designer application administrative tasks. Solution Designer uses OpenID Connect (OIDC) as the identity provider protocol.

When Solution Designer cloud instances use IAM, ensure that you create separate users and groups for each environment (or class of environments) in the external authenticator service. The specifications of the users and groups depend on the authenticator.

Solution Designer cloud native toolkit provides a sample configuration that uses KeyCloak. Solution Designer uses OpenID Connect for authentication and authorization. The Solution Designer client that provides access to Solution Designer functions is used for OIDC client applications. Each OIDC client application must have the following properties:

- Client type: OpenID Connect
- Client authentication: enabled
- Authorization: enabled

- Standard flow: enabled
- Direct access grants: enabled
- OAuth 2.0 Device Authorization Grant: enabled

> **Note:**
>
> The names or the properties may differ depending on the IAM used.

Scopes

The following are the Scopes to define for Solution Designer:

Table 2-1 Scopes for Solution Designer

Scope	Client Application	Description
/lcm	The Solution Designer Client application	Used for user access and any external API access such as TMF interface
/lcmOperation	The Solution Designer Client application	Used for internal operations between microservices

Authentication in REST API:

For REST APIs, you must create a token for REST APIs and pass these tokens while calling the REST APIs.

Audience

Each application requires an audience for the scopes used.

Roles

Solution Designer provides various roles to access user interface. The roles must be provided in the access tokens for the user. The roles must appear in JSON Web token (JWT) under the Token Claim Name **groups**. A user must be assigned with the appropriate roles. In addition to the roles assigned to a user, the user token must also include the appropriate scope.

The user interface access is controlled using Role Based Access Control (RBAC) and implemented using OAUTH Roles from the OIDC provider. Users are assigned appropriate roles based on their needs. All the different types of roles are also independent of each other. A user could have access to Initiative entity in Solution Designer, but may not have access to Initiative application to interact with the entity.

Landing Page Roles

On the landing page, you will see only those applications that you have access to, based on the roles assigned to them.

Table 2-2 Landing Page Roles

Role	Description
Service Specialist	The Service Specialist has access to the following applications: • Administration • Domains • Infrastructure Specifications • Product Specifications • PSR models • Resource Specifications • Service Specifications
Service Catalog Admin	The Service Catalog Admin can work with all the functionality available in the Solution Designer application. The Service Catalog Admin has access to the following applications: • Administration • Domains • Infrastructure Specifications • Product Specifications • PSR models • Resource Specifications • Service Specifications • Initiatives • Workspaces

Initiative Lifecycle Roles

Initiatives lifecycle management is controlled using the following roles. The users can be assigned various roles depending on the business requirements.

Table 2-3 Initiative Lifecycle Roles

Role	Description
Initiative Approve Functional Testing	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Approve Rollout	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Complete Definition	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Complete Testing	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Content Provider	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Discard	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Launch	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Reopen	A state transition for an initiative to progress an initiative through its life cycle.
Initiative Start Acceptance Testing	A state transition for an initiative to progress an initiative through its life cycle.

Table 2-3 (Cont.) Initiative Lifecycle Roles

Role	Description
Initiative Complete Advanced Configuration	A state transition for an initiative to progress an initiative through its life cycle. The user assigned to this role can perform simulated publish and also can complete the Advanced Configuration state.

> **Note:**
>
> You must have the **Service Catalog Admin** role to be able to use the Initiative Lifecycle roles. If you have access to the lifecycle roles does not imply that you have access to Initiatives application.

Domain Filter Roles

Domain filter roles can be assigned to the user based on the domains that you create in Solution Designer. After you create a service or technical domain in Solution Designer, you must assign appropriate domain filter roles to the users. You need these roles to access all the entities in Solution Designer User Interface that are associated with the specified domain. To define the domain filter roles, you must prefix a role with **sDOMAIN:** for service domains and **tDOMAIN:** for technology domains. These domain roles can be defined as:

```
sDOMAIN:Service_Domain_Id
tDOMAIN:Technology_Domain_Id
DOMAIN:all
```

> **Note:**
>
> You must use the ID of the domain.

Table 2-4 Domain Filter Roles

Role	Description
sDOMAIN	This is a service domain role. This role gives access to all the entities such as PSR models, Service Specifications and Resource Specifications that the specified service domain is associated with. For example, `sDOMAIN:BroadbandInternet` gives access to all the entities associated with the `BroadbandInternet` domain.
tDOMAIN	This is a technical domain role. This role gives access to all the entities such as PSR models, Service Specifications, and Resource Specifications that the specified service domain is associated with. For example, `tDOMAIN:Wireless5G` gives access to all the entities associated with the `Wireless5G` technology domain.

Table 2-4 (Cont.) Domain Filter Roles

Role	Description
DOMAIN:all	This role acting as a superuser gives access to all the entities in all the domains.

> **Note:**
>
> Users must have at least one of the domain filter roles assigned to them to model a service using entities such as PSR models, Service Specifications, and Resource Specifications in Solution Designer.

Domain Management Role

Domain management role provides access to the **Domains** application in Solution Designer. When you have the domain administrator role, you can create, update, and delete service and technology domains in the **Domains** application.

Table 2-5 Domain Management Role

Role	Description
DOMAIN:admin	This is a domain administrator role. This user can manage service and technology domains in Solution Designer. They can create, update, and delete domains.

About Relying Party

Relying party is an application that relies upon the user's credentials such as user ID and password to grant access to an application. Relying Party outsources its user authentication function to an identity provider. The Solution Designer CNTK provides a sample Apache Relying Party.

The Relying Party must handle the following authorization flows:

Table 2-6 Relying Party authorization flows and its URL prefixes

Authorization Flows	Description	URL prefixes
User Interface flow	Accessing the Solution Designer application using a web browser.	Protect by OpenID Connect • /scd/ • /apps/
API flow	Accessing the TMF REST APIs used for integration.	Protect by OAuth2 • /scd/tmf-api/

About Object Store

For use with Solution Designer cloud native, an Object Store S3 provider must be designated. Examples of S3 Object Store providers are Minio, Oracle OCI Object Store and so on. Only one S3 provider is supported. Setup a bucket to store build dependencies for the UIM participant. Create and record an access key and secret.

Given the dynamic nature of the Solution Designer cloud native instance, you must have a set of operational mechanisms to:

- Monitor disk usage and warn when the usage crosses a threshold.
- Clean out the artifacts that are no longer needed.

Management of Secrets

Solution Designer cloud native leverages Kubernetes Secrets to store sensitive information securely. This sensitive information is, at a minimum, the database credentials. Additional credentials may be stored to authenticate with the external authenticator system. Your cartridges may need to communicate with other systems, such as Oracle Communications Unified Inventory Management (UIM). The credentials for such systems too are managed as Kubernetes Secrets.

These secrets need to be secured over their lifecycle by the Kubernetes cluster administration.

Solution Designer cloud native scripts assume that a set of prerequisite secrets exist when they are invoked. As such, creation of the secrets is a prerequisite step in the pipeline. Solution Designer cloud native toolkit provides a script to create some of the common secrets it needs, but this script is interactive and therefore not suitable for Continuous Delivery (CD) automation pipelines. The script serves to provide a basic mechanism to add secrets and illustrates the names and structure of the secrets that Solution Designer cloud native requires. You can create the secrets manually by using the script for each instance.

Using Kubernetes Monitoring Toolchain

A multi-node Kubernetes cluster with multiple users requires a capable set of tools to monitor and manage the cluster. There are tools that provide data, rich visualizations and other capabilities such as alerts. Solution Designer cloud native does not require any particular system to be used, but recommends using such a monitoring, visualization and alerting capability.

For Solution Designer, the key aspects of monitoring are:

- Worker capacity in CPU and memory. Monitoring the free capacity leads to predictable Solution Designer instance creation and scale-up.
- Worker node disk pressure
- Worker node network pressure
- Health of the core Kubernetes services

The namespaces and pods that Solution Designer cloud native uses provide a cross instance view of Solution Designer cloud native.

Planning Your Container Engine for Kubernetes (OKE) Cloud Environment

This section provides information about planning your cloud environment if you want to use Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE) for Solution Designer cloud native. Some of the components, services, and capabilities that are required and recommended for a cloud native environment are applicable to the Oracle OKE cloud environment as well.

- **Kubernetes** and **Container Images**: You can choose from the version options available in OKE as long as the selected version conforms to the range described in the section about planning cloud native environment.
- **Container Image Management**: Solution Designer cloud native recommends using Oracle Cloud Infrastructure Registry with OKE. Any other repository that you use must be able to serve images to the OKE environment in a quick and reliable manner.
- **Oracle Multitenant Database**: It is strongly recommended to run Oracle DB outside of OKE, but within the same Oracle Cloud Infrastructure tenancy and the region as an Oracle DB service (BareMetal or VM). The database version should be 19c.
- **Helm** : Install Helm on a Linux host that has access to kubectl of the cluster.
- **Persistent Volumes**: Use NFS-based persistence. Solution Designer cloud native recommends the use of Oracle Cloud Infrastructure File Storage service in the OKE context.
- **Authentication** and **Secrets Management**: These aspects are common with the cloud native environment. Choose your mechanisms to deliver these capabilities and implement them in your OKE instance.
- **Monitoring Toolchains**: While the Oracle Cloud Infrastructure Console provides a view of the resources in the OKE cluster, it also enables you to use the Kubernetes Dashboard. Any additional monitoring capability must be built up.
- **CI** and **CD Pipelines**: The considerations and actions described for CI and CD pipelines in the cloud native environment apply to the OKE environment as well.

Compute Disk Space Requirements

Given the size of the Solution Designer cloud native container images, the size of the Solution Designer containers, and the volume of the logs generated, it is recommended that the Kubernetes worker nodes have at least 40 GB of free space for the container storage and the image storage.

Work with your Cloud Infrastructure administrator to ensure worker nodes have enough disk space. Common options are to use Compute shapes with larger boot volumes or to mount an Oracle Cloud Infrastructure Block Volume to container storage.

Connectivity Requirements

Solution Designer cloud native assumes the connectivity between the OKE cluster and the Oracle CDBs is a LAN-equivalent in reliability, performance and throughput. This can be achieved by creating the Oracle CDBs within the same tenancy as the OKE cluster, and in the same Oracle Cloud Infrastructure region.

Using Load Balancer as a Service (LBaaS)

For load balancing, you have the option of using the services available in OKE. The infrastructure for OKE is provided by Oracle's IaaS offering, Oracle Cloud Infrastructure. In OKE, the master node IP address is not exposed to the tenants. The IP addresses of the worker nodes are also not guaranteed to be static. This makes DNS mapping difficult to achieve. Additionally, it is also required to balance the load between the worker nodes. In order to fulfill these requirements, you can use Load Balancer as a Service (LBaaS) of Oracle Cloud Infrastructure.

The relying party must be configured to use a service load balancer.

For additional details, see the following:

- "Creating Load Balancers to Distribute Traffic Between Cluster Nodes" in Oracle Cloud Infrastructure documentation.
- "Load Balancer Annotations" in Oracle GitHub documentation.

About Using Oracle Cloud Infrastructure Domain Name System (DNS) Zones

While a custom DNS service can provide the addressing needs of Solution Designer cloud native even when Solution Designer is running in OKE, you can evaluate the option of Oracle Cloud Infrastructure Domain Name System (DNS) zones capability. Configuration of DNS zones (and integration with on-premise DNS systems) is not within the scope of Solution Designer cloud native.

Using Persistent Volumes and File Storage Service (FSS)

In the OKE cluster, Solution Designer cloud native can leverage the high performance, high capacity, high reliability File Storage Service (FSS) as the backing for the persistent volumes of Solution Designer cloud native. It is recommended to have at least 25 GB of free space. There are two flavors of FSS usage in this context:

- Allocating FSS by setting up NFS mount target
- Native FSS

To use FSS through an NFS mount target, see instructions for allocating FSS and setting up a Mount Target in "Creating File Systems" in the Oracle Cloud Infrastructure documentation. Note down the Mount Target IP address and the storage path and use these in the Solution Designer cloud native instance specification as the NFS host and path. This approach is simple to set up and leverages the NFS storage provisioner that is typically available in all Kubernetes installations. However, the data flows through the mount target, which models an NFS server.

FSS can also be used natively, without requiring the NFS protocol. This can be achieved by leveraging the FSS storage provisioner supplied by OKE. A broad outline of how to do this is available in the blog post "Using File Storage Service with Container Engine for Kubernetes" on the Oracle Cloud Infrastructure blog.

Leveraging Oracle Cloud Infrastructure Services

For your OKE environment, you can leverage existing services and capabilities that are available with Oracle Cloud Infrastructure. The following table lists the Oracle Cloud Infrastructure services that you can leverage for your OKE cloud environment.

Table 2-7 Oracle Cloud Infrastructure Services for OKE Cloud Environment

Type of Service	Service	Indicates Mandatory / Recommended / Optional
Developer Service	Container Clusters	Mandatory
Developer Service	Registry	Recommended
Core Infrastructure	Compute Instances	Mandatory
Core Infrastructure	File Storage	Recommended
Core Infrastructure	Block Volumes	Optional

Table 2-7 (Cont.) Oracle Cloud Infrastructure Services for OKE Cloud Environment

Type of Service	Service	Indicates Mandatory / Recommended / Optional
Core Infrastructure	Networking	Mandatory
Core Infrastructure	Load Balancers	Recommended
Core Infrastructure	DNS Zones	Optional
Database	BareMetal and VM	Recommended

3

Creating Solution Designer Cloud Native Images

Solution Designer cloud native requires container images to be made available to create and manage its cloud native instances. This chapter describes how to create those Solution Designer cloud native images.

Solution Designer cloud native requires two types of container images. The Solution Designer DB Installer images are used to manage Solution Designer schema. The other images are the Solution Designer image itself. These images are the basis for all of the long running pods and all the services that comprise Solution Designer cloud native instance. Each image is built on top of a Linux base image and adds Java and Solution Designer product components on top.

Solution Designer Cloud native images are created using the Solution Designer cloud native builder toolkit and a dependency manifest file. The Solution Designer cloud native Image Builder is intended to be run as part of a Continuous Integration process that generates images. It needs to run on Linux and have access to the local container image builder. The versions of these are as per the Service Catalog and Design statement of certification in the Service Catalog and Design documentation. The dependency manifest is a file that describes all the versions required to build out the image.

See the following topics for further details:

- Downloading the Solution Designer Cloud Native Image Builder
- Prerequisites for Creating Solution Designer Images
- Configuring the Solution Designer Cloud Native Images
- Creating Solution Designer Cloud Native Images

Downloading the Solution Designer Cloud Native Image Builder

You download the Solution Designer image builder (**ocscd-image-builder.tgz**) and the base service model (**ServiceModelBase.zip**) from the Oracle software delivery website: http://edelivery.oracle.com.

To build the Solution Designer cloud native container image for each microservice, the **ocscd-image-builder.tgz** file is required.

The Solution Designer cloud native image builder is bundled with the following components:

- A manifest file named **ocscd_cn_ci_manifest.yaml**.
- Solution Designer cloud native builder kit.
- Staging directory structure that contains the corresponding files for creating images for each microservice.

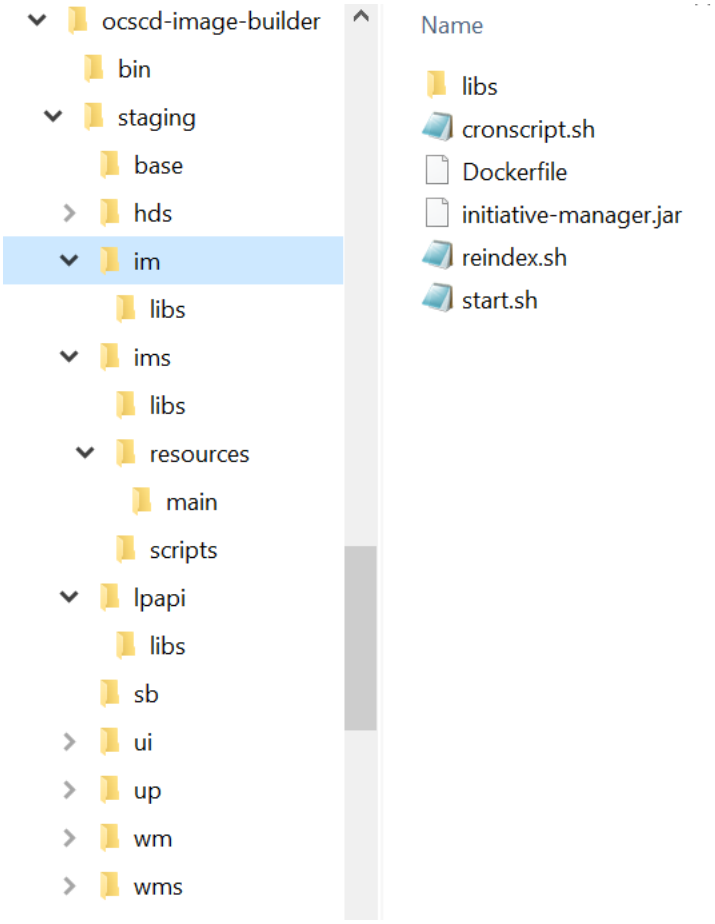
  Table 3-1 lists the microservices in Solution Designer.

Table 3-1 Microservices and its description

Microservices name	Description
im	Microservice for the initiative manager.
ims	Microservice for the initiative manager DB schema installer.
wm	Microservice for the workspace manager.
wms	Microservice for the workspace manager DB schema installer.
lpapi	Microservice for the landing page api.
lpapis	Microservice for the landing page api DB schema installer.
ui	Microservice for the solution designer UI.
up	Microservice for the UIM Participant.
hds	Microservice for the headless Design Studio.

The staging directory structure contains the files for the microservices. Each directory structure contains a Dockerfile and the contents of the container image. Figure 3-1 illustrates the staging directory structure.

Figure 3-1 Staging Directory Structure



Prerequisites for Creating Solution Designer Images

You must download the prerequisite software from Oracle software delivery website `http://edelivery.oracle.com` You must add these contents to the Staging directory. Use `ocscd_cn_ci_manifest.yaml` for the complete list of prerequisite software.

The prerequisites for building Solution Designer cloud native images are:

- Podman on the build machine.
- Java 21

In addition to the prerequisites for building Solution Designer images, the following are the prerequisites for building the headless Design Studio microservice:

- Eclipse
- Studio plugin
- UIM SDK
- UniOps Reference Solution
- Java 8
- JDeveloper Suite jars
- Log4j

See *Service Catalog and Design Compatibility Matrix* for details about the required and supported versions of these prerequisite software.

Configuring the Solution Designer Cloud Native Images

The image builder uses a manifest file. The manifest file describes the input as a list of the files that must be downloaded and its location. It is consumed by the image build process. The default configuration in the latest manifest file provides all the necessary components and required patches for creating the Solution Designer cloud native images easily.

You can also modify the manifest file to extend it to meet your requirements. This enables you to:

- Specify any Linux image as the base, as long as its binary is compatible with Oracle Linux.
- Upgrade the Oracle Enterprise Linux version to a newer version to uptake a quarterly Critical Patch Updates.
- Upgrade the JDK version to a newer JDK version to uptake a quarterly Critical Patch Updates.
- Change the Solution Designer artifacts to newer artifacts to uptake a new Service Catalog and Design patch.

The breakdown of each section in the dependency manifest file is as follows:

```
ocscdBuilder:
    schemaVersion: cn-manifest/v1
    date: 2022-11-30

ocscdBaseImage:
    name: ocscd-base
```

```
    tag: latest
    # Specify the details of the Linux base image for OCSCD.
    # Refer to the OCSCD documentation for certification statement on
supported
    # types and versions.  This information is coded into the OCSCD image as a
    # LABEL, for tracking purposes.
    linux:
        vendor:Oracle
        version: 8.7
        image: container-registry.oracle.com/os/oraclelinux:8-slim

    jdk:
        vendor: Oracle
        version: jdk-21.0.4
        path: java/jdk-21.0.4_linux-x64_bin.tar.gz

studioBaseImage:
    name: ocscd-studio-base
    tag: latest
    jdk:
        vendor: Oracle
        version: jdk-8u341
        path: java/jdk-8u341-linux-x64.tar.gz
    studioPlugin:
        version: 8.2.0
        path: studio/DesignStudio-8.2.0.0.0-b16827283.zip
    eclipse:
        version: 4.33
        path: studio/eclipse-jee-2024-09-R-linux-gtk-x86_64.tar.gz
    uimSdk:
        version: 7.6.0
        path: uim/UIM_SDK.zip
    snoConfig:
        version: 7.4.3.1
        path: sno/UniOps-RI_configuration-7.4.3.1.0.B363.zip
    jdev:
        version: 12.2.1.4.0
        path1: fmw/jdev_suite_122140.jar
        path2: fmw/jdev_suite_1221402.jar
    log4j:
        version: 2.23.1
        path1: log4j/log4j-api-2.23.1.jar
        path2: log4j/log4j-core-2.23.1.jar

imImage:
    name: ocscd/initiative-manager
    tag: latest

imsImage:
    name: ocscd/initiative-manager-semele
    tag: latest

wmImage:
    name: ocscd/workspace-manager
    tag: latest
```

```
wmsImage:
    name: ocscd/workspace-manager-semele
    tag: latest

lpapiImage:
    name: ocscd/landing-page-api
    tag: latest

lpapisImage:
    name: ocscd/landing-page-api-semele
    tag: latest

uiImage:
    name: ocscd/unifieddesigner-ui
    tag: latest

upImage:
    name: ocscd/uim-participant
    tag: latest

studioImage:
    name: ocscd/headless-design-studio
    tag: latest
```

> **Note:**
>
> The `schemaVersion` and `date` parameters are maintained by Oracle. Do not modify these parameters.
>
> Version numbers provided here are only examples. The manifest file used specifies the actual versions currently recommended.

- **Solution Designer Cloud Native Infrastructure Image**

  While not required by Solution Designer cloud native to create or manage Solution Designer instances, this infrastructure image is a necessary building block of the final Solution Designer container image.

  ```
  linux:
      vendor:Oracle
      version: 8.7
      image: container-registry.oracle.com/os/oraclelinux:8-slim
  ```

  The `linux` parameter specifies the base Linux image to be used as the base container image. The version is the two-digit version from **/etc/redhat-release**.

  The vendor and version details are specified and used for:

  – Validation when an image is built.

– Querying at run-time. To troubleshoot issues, Oracle support requires you to provide these details in the manifest file used to build the image.

```
jdk:
    vendor: Oracle
    version: jdk-21.0.4
    path: java/jdk-21.0.4_linux-x64_bin.tar.gz
```

The `jdk` parameter specifies the JDK vendor, version, and the staging path. In the given staging path, the staging directory must contain a directory named java that contains the JDK installer. The JDK installer must be downloaded and placed in the location `staging/java/jdk-11.0.17.0.2_linux-x64_bin.tar.gz`

- **Service Catalog and Design - Design Studio Image**

```
studioBaseImage:
    name: ocscd-studio-base
    tag: latest
    jdk:
        vendor: Oracle
        version: jdk-8u341
        path: java/jdk-8u341-linux-x64.tar.gz
    studioPlugin:
        version: 8.2.0
        path: studio/DesignStudio-8.2.0.0.0-b16827283.zip
    eclipse:
        version: 4.33
        path: studio/eclipse-jee-2024-09-R-linux-gtk-x86_64.tar.gz
    uimSdk:
        version: 7.6.0
        path: uim/UIM_SDK.zip
    snoConfig:
        version: 7.4.3.1
        path: sno/UniOps-RI_configuration-7.4.3.1.0.B363.zip
    jdev:
        version: 12.2.1.4.0
        path1: fmw/jdev_suite_122140.jar
        path2: fmw/jdev_suite_1221402.jar
    log4j:
        version: 2.23.1
        path1: log4j/log4j-api-2.23.1.jar
        path2: log4j/log4j-core-2.23.1.jar
```

The `studioBaseImage` contains the various parameters that are required for Design Studio. This is required for designing complex design policies and also for deploying the cartridges in the run-time application. All the parameters mentioned in the `studioBaseImage` must be downloaded and placed in the appropriate directories as mentioned in the manifest file. You must download reference implementation cartridges `UniOps-RI_configuration-7.4.3.1.0.B363.zip` from the URL:

https://www.oracle.com/downloads/applications/communications/rsdod-downloads.html#

Place `UniOps-RI_configuration-7.4.3.1.0.B363.zip` in a directory, and enter the path in `snoConfig`.

- **Solution Designer Cloud Native Image**

```
imImage:
    name: ocscd/initiative-manager
    tag: latest
imsImage:
    name: ocscd/initiative-manager-semele
    tag: latest
wmImage:
    name: ocscd/workspace-manager
    tag: latest
wmsImage:
    name: ocscd/workspace-manager-semele
    tag: latest
lpapiImage:
    name: ocscd/landing-page-api
    tag: latest
uiImage:
    name: ocscd/unifieddesigner-ui
    tag: latest
upImage:
    name: ocscd/uim-participant
    tag: latest
studioImage:
    name: ocscd/headless-design-studio
    tag: latest
```

These parameters specify the microservices of Solution Designer that can be installed. The `name` specifies the name of the image that is created and the `tag` specifies the version of the microservice. The *`ocscd`* is the default directory in which the image is created. You can also specify the container repository instead of the default directory.

For container images, you can use a standard tag or specify the sha256 digest depending on your business needs. Using a sha256 digest is more secure and mitigates the tampering of your images from your container repository. The following is an example using a standard tag.

```
wmImage:
    name: ocscd/workspace-manager
    tag: latest
```

The following is an example using sha256 digest. You must have **@sha256** at the end of the name line.

```
wmImage:
    name: ocscd/workspace-manager@sha256
    tag: a6b7be1808b8443dde696c5f108be1cb6e7641d6b281ef7598df012c1d6871f8
```

Creating Solution Designer Cloud Native Images

To create the Solution Designer image, the image builder does the following:

- Starts with a base-level operating system image (for example, **oraclelinux:8-slim**).

You can specify any Linux image as the base, as long as its binary is compatible with Oracle Linux and conforms to the compatibility matrix. See *Service Catalog and Design Compatibility Matrix* for details about the supported software.

The following packages must be installed onto the given base image, or be already present:

- gzip
- tar
- unzip

Creating the Solution Designer Images

To create the Solution Designer images:

1. Create the workspace directory:

   ```
   mkdir workspace
   ```

2. Obtain and untar the Solution Designer image builder file: **ocscd-image-builder.tgz** to the workspace directory:

   ```
   tar -xf ./ocscd-image-builder.tgz --directory workspace
   ```

3. Download the prerequisite software listed in Prerequisites for Creating Solution Designer Images to the **workspace/ocscd-image-builder/staging/** directory. For example, the following command downloads JDK to the **workspace/ocscd-image-builder/staging/java** directory.

   ```
   cp jdk-8u251-linux-x64.tar.gz ./workspace/ocscd-image-builder/staging/java/
   jdk-8u251-linux-x64.tar.gz
   ```

4. Run **build-images.sh** and pass the manifest file, staging path, and the type of image to be created.

   ```
   export DMANIFEST=$(pwd)/ocscd-image-builder/bin/ocscd_cn_ci_manifest.yaml
   export STAGING=$(pwd)/ocscd-image-builder/staging
   ```

   Run one of the following commands:

   - To create all the images, use "-c all" as shown:

     ```
     $(pwd)/ocscd-image-builder/bin/build-images.sh -f $DMANIFEST -
     s $STAGING -c all
     ```

   - To create individual images, use "-c *image name*" as shown:

     ```
     $(pwd)/ocscd-image-builder/bin/build-images.sh -f $DMANIFEST -
     s $STAGING -c im,wm
     ```

   - To create Solution Designer image using proxy or firewall:

     ```
     $(pwd)/ocscd-image-builder/bin/build-images.sh -f $DMANIFEST -
     s $STAGING -c all -p http://my-corporate-proxy
     ```

These steps can be included into your CI pipeline as long as the required components are already downloaded to the staging area.

Post-build Image Management

The Solution Designer cloud native image builder creates images with names and tags based on the settings in the manifest file. By default, this results in the following images:

- ocscd/headless-design-studio:latest
- ocscd-studio-base:latest
- ocscd/uim-participant:latest
- ocscd/unifieddesigner-ui:latest
- ocscd/workspace-manager-semele:latest
- ocscd/workspace-manager:latest
- ocscd/initiative-manager-semele:latest
- ocscd/initiative-manager:latest
- ocscd/landing-page-api-semele:latest
- ocscd/landing-page-api:latest

Once images are built in a CI pipeline, the pipeline uniquely tags the images and pushes them to an internal container repository. An uptake process can then be triggered for the new images:

- Sanity Test
- Development Test (for explicit retesting of scenarios that triggered the rebuild, if any)
- System Test
- Integration Test
- Pre-Production Test
- Production

4

Creating Solution Designer Cloud Native Instances

This chapter describes how to create a Solution Designer cloud native instance in your cloud environment using the operational scripts and the base configuration provided in the Solution Designer cloud native toolkit.

This procedure is intended to validate that you are able to create a Solution Designer instance in your environment. Before you create an instance, you must download the Solution Designer cloud native tar file and extract the **ocscd-cntk.tgz** file.

Installing the Solution Designer Artifacts and the Toolkit

Build container image for the Solution Designer application using the Solution Designer cloud native Image Builder.

You must create a container image private repository for this image, ensuring that all nodes in the cluster have access to the repository. See "About Container Image Management" for more details.

Download the Solution Designer cloud native toolkit archive and do the following:

- **On Oracle Linux**: Where Kubernetes is hosted on Oracle Linux, download and extract the tar archive to each host that has connectivity to the Kubernetes cluster.
- **On OKE**: For an environment where Kubernetes is running in OKE, extract the contents of the tar archive on each OKE client host. The OKE client host is the bastion host/s that is set up to communicate with the OKE cluster.

Set the variable for the installation directory by running the following command, where *ocscd_cntk_path* is the installation directory of the Solution Designer cloud native toolkit:

```
export OCSCD_CNTK=ocscd_cntk_path
```

Creating the Namespace

After you set the environment variables, you create the namespace that Solution Designer is installed. The namespace requires a namespace administrator service account so Solution Designer services and manages the secrets and configuration maps.

To create namespace, run the following:

```
kubectl create namespace ocscd
```

To create the namespace administrator. You can modify the administrator service account by modifying the details in **secrets-admin.yaml** file.

```
kubectl -n ocscd apply -f $OCSCD_CNTK/samples/rbac/secrets-admin.yaml
```

Installing Kafka

Solution Designer requires asynchronous messaging to interact with Solution Designer services. Kafka is used for asynchronous messaging. You can configure a native Kafka broker.

Setup Strimzi Kafka

You can use a native Kafka broker for asynchronous messaging. Strimzi operator provides a way to run an Apache Kafka cluster on Kubernetes in various deployment configurations.

The CNTK provides a sample Strimzi Kafka which is located at $OCSCD_CNTK/**samples/strimzi-kafka**. Follow the instructions in the README file located at $OCSCD_CNTK/**samples/strimzi-kafka/README.md** to install sample Strimzi Kafka.

> **Note:**
>
> This sample does not use the PVC backend storage and must be considered for production use.

Setup OCI Streams as Kafka

You can use OCI Streams as Kafka for asynchronous messaging. When setting up OCI Streams, you must setup the following items when creating a stream pool:

1. Create a private endpoint.
2. Under advanced options, select auto create topics = true

After you create the Stream Pool, go to **Stream Pool Detail** and copy the data in **Kafka Connection Settings** page.

The user account that has created the Stream Pool is the owner of the Pool and you must get an OCI Access Token for that account to use the Stream Pool.

You can use the copied information to update the Kafka details by running the **manage-instance-credentials.sh** script.

```
$OCSCD_CNTK/scripts/manage-instance-credentials.sh -i scd update kafka
Kafka End-point (hostname:port): host_name:port
Kafka security protocol (PLAINTEXT,SASL_PLAINTEXT, SASL_SSL, SSL): Security
Protocol
SASL mechanism: SASL_Mechanism
SASL connection string: Copy_SASL_Connection_string;
```

Where

- *host_name*:*port* is the host name and port address of the Kafka that you copied from the **Bootstrap Servers** information in the **Kafka Connection Settings** page.
- *Security Protocol* is the security protocol that you copied in the **Kafka Connection Settings** page.
- SASL_Mechanism is the security mechanism that you copied in the **Kafka Connection Settings** page.

- `Copy_SASL_Connection_string` is the SASL Connection strings that you copied in the **Kafka Connection Settings** page. For example:

```
org.apache.kafka.common.security.plain.PlainLoginModule required
username="tenancy/user/poolId" password="AUTH_TOKEN";
```

> **Note:**
>
> Replace AUTH_TOKEN with your access token.

Installing UIM Participant PVC

The UIM Participant requires storage space as scratch space for compiling the UIM artifacts and storing the artifacts for download. A basic template example is in **$OCSCD_CNTK/samples/pvc**.

Perform the following steps:

1. Create a PVC that references read/write storage which could be block storage or other technology like NFS, Oracle OCI FSS, or the nfs-subdir-external-provisioner. Record the name of the PVC you created.
   For NFS, this can be a static PV (Persistent Volume) as provided in the sample README or a dynamic provisioner like Kubernetes nfs-subdir-external-provisioner.
2. Update the instance specification file to reference the same PVC name that you created:

```
uim-participant:
    storageVolume:
        pvc: pvc_name
```

Installing OpenID Connect (OIDC) Relying Party

A Relying Party, is an application or website that outsources its user authentication function to an IDP. There are several commercial and non-commercial offerings. The CNTK provides a sample Apache Rely Party which is located at $OCSCD_CNTK/**samples/apache-rely-party**. Follow the instructions in the README file located at $OCSCD_CNTK/**samples/apache-rely-party/README.md** to install sample Apache Rely Party.

Installing OpenSearch

Solution Designer uses OpenSearch which is a distributed search and analytics engine. OpenSearch is an elastic data store that is used to search the entities used in the Solution Designer application.

The CNTK provides a sample OpenSearch which is located at $OCSCD_CNTK/**samples/opensearch**. Follow the instructions in the README file located at $OCSCD_CNTK/**samples/opensearch/README.md** to install sample OpenSearch.

> **Note:**
>
> This sample does not use the PVC backend storage and must be considered for production use. The samples include an example using Oracle OCI **oci-bv** storage.

Creating a Solution Designer Cloud Native Instance

This section describes how to create a Solution Designer cloud native instance.

Setting Environment Variables

Solution Designer cloud native relies on access to certain environment variables to run seamlessly. Ensure that the path to your private specification repository variable is set in your environment:

To set the environment variables:

1. Create a directory that serves as your specification repository, by running the following command, where *spec_repo_path* is the path to your private specification repository:

   ```
   $ export SPEC_PATH=spec_repo_path
   ```

Creating Secrets

You must store sensitive data and credential information in the form of Kubernetes Secrets that the scripts and Helm charts in the toolkit consume. Managing secrets is out of the scope of the toolkit and must be implemented while adhering to your organization's corporate policies.

As a prerequisite to using the toolkit for installing the Solution Designer database or creating an instance, you must create secrets for access to the following:

- Solution Designer database
- Kafka
- Authentication
- Object Store

The toolkit provides sample scripts for this purpose. However, they are not pipeline-friendly. The scripts should be used for creating an instance manually and quickly, but not for any automated process for creating instances. The scripts also illustrate both the naming of the secret and the layout of the data within the secret that Solution Designer cloud native requires. You must create secrets prior to running the install-db.sh or create-instance.sh scripts.

The following is an example of running the script to create the secrets that you need:

```
$OCSCD_CNTK/scripts/manage-instance-credentials.sh -i ocscd create db,db-
schema,kafka,ocscd-oidc,s3
```

Where

- `db` specifies the database connectivity details and the administrator credentials for connecting to the Solution Designer database. This is consumed by the Solution Designer

database installer and Oracle Communications Service Catalog and Design- Solution Designer.

- `db-schema` specifies the schema credentials that the Solution Designer services use to persist data.
- `kafka` specifies the connectivity details and the credentials for connecting to Kafka provider.
- `ocscd-oidc` specifies OpenID Connect (OIDC) connectivity and credentials for connecting to the OIDC provider for Service Catalog and Design - Solution Designer.
- `s3` specifies the connectivity details and the credentials for connecting to an S3 Object Store provider.

Assembling the Specifications

To assemble the specifications:

Copy the instance specification to your $SPEC_PATH and rename:

```
cp $OCSCD_CNTK/samples/instance.yaml $SPEC_PATH/ocscd.yaml
```

> **Note:**
>
> The file name must match the instance name or the namespace name.

You edit this file as per the instructions described in the sections that follow.

Configuring the Instance Specification

Edit the instance specification (`ocscd.yaml`) to set the container image location and tag in each microservice location. The following is an instance of a few entries:

```
initiative-manager: {}
#  env:
#    elasticsearch:
#      host: opensearch-cluster-master
#    externalProductCatalog: false
#  replicaCount: 1
#
#  image:
#    repository: ocscd/initiative-manager
#    tag: latest
#    pullSecrets: []
#  db:
#    type: "STANDARD" # Acceptable values are STANDARD and ADB
workspace-manager: {}
#  replicaCount: 1
#  image:
#    repository: ocscd/workspace-manager
#    tag: latest
#    pullSecrets: []
#  db:
```

```
#    type: "STANDARD" # Acceptable values are STANDARD and ADB
landing-page-api: {}
#  replicaCount: 1
#  image:
#    repository: ocscd/landing-page-api
#    tag: latest
#    pullSecrets: []
#  db:
#    type: "STANDARD" # Acceptable values are STANDARD and ADB
```

> **Note:**
>
> YAML formatting is case-sensitive. While the next step uses vi editor for editing, if you are not familiar with editing YAML files, use a YAML editor to ensure that you do not make any syntax errors while editing. Follow the indentation guidelines for YAML, as incorrect spacing can lead to errors.

When the `externalProductCatalog` is set to `true`, you can integrate Solution Designer with External Product Catalog using TMF Open API TMF620 (Product Catalog Management). The default value is `false` and if set to `false`, you manage the product catalog in Solution Designer.

If your environment requires a password to download the container images from your repository, create a Kubernetes secret with the image pull credentials. See the Kubernetes documentation for details. Reference the secret name in the instance specification.

```
# uncomment and set if required
#  image:
#    pullSecrets: []
#      - name: my-regcred
```

By default, Solution Designer is configured to use a PDB for its database.

```
#  db:
#   type: "STANDARD" # Acceptable values are STANDARD and ADB
```

If you choose a tablespace name that is different from the default name **USERS**, then you must enter the new name in the instance specification.

```
#  db:
    #  type: "STANDARD"
    #  defaultTablespace: "USERS"
    #  tempTablespace: "TEMP"
```

Installing the Schemas

This procedure configures an empty PDB or ADB. Depending on the database strategy for your team, you may have already performed this procedure as described in "Provisioning Oracle Multitenant Container Database (CDB)". Before continuing, ensure that the PDB or

ADB being used for creating the Solution Designer instance includes the schema installation. If the database already has the schema installed, skip this procedure.

After the PDB or ADB is created, it is configured with the Solution Designer schema.

Run the following to install the Solution Designer schema.

```
$OCSCD_CNTK/scripts/install-db.sh -i ocscd -s $SPEC_PATH -c 4
```

Create a Solution Designer Instance

This procedure describes how to create a Solution Designer instance in your environment using the scripts that are provided with the toolkit.

To create a Solution Designer instance:

1. Run the following command:

   ```
   $OCSCD_CNTK/scripts/create-instance.sh -i ocscd -s $SPEC_PATH
   ```

   The **create-instance.sh** script creates and deploys the microservices that are specified in the instance specification file. If the scripts fails, see "Troubleshooting Issues with the Scripts", before you make additional attempts.

2. If you query the status of the pods, the **READY** state of the microservices may display **0/1** for several minutes while the Solution Designer application is starting.
   When the **READY** state shows **1/1**, your Solution Designer instance is up and running. You can then validate the instance by running the Oracle Communications Service Catalog and Design - Solution Designer application and creating a sample PSR model.

Validating the Solution Designer Instance

After creating an instance, you can validate it by checking the domain configuration and the client UIs.

Run the following command to display the deployment details of the Solution Designer instance that you have created:

```
kubectl get deployment -n ocscd
```

The command displays the domain configuration information.

To verify the client UIs, log into the Service Catalog and Design - Solution Designer Web client user interface with the service catalog administrator login credentials created in IAM using the URL:

```
http://hostname:port/apps/scd/
```

Where

- *hostname* is the Solution Designer host name.
- *port* is the port number where Solution Designer is installed.

> **Note:**
>
> After a Solution Designer instance is created, it may take a few minutes for the user interface to become active.

Scaling the Solution Designer Application Cluster

You can create multiple instances of the microservices components of Solution Designer by setting the replicaCount accordingly in the **MANIFEST.yaml** file while creating the Solution Designer instance and during the upgrade. By default, the replicaCount is set to 1. You cannot modify the replicaCount for **UIM-Participant** component which is set to 1, by default.

To scale the Solution Designer instances, edit the **MANIFEST.YAML** file.

```
initiative-manager:
        replicaCount: 1

workspace-manager:
        replicaCount: 1

landing-page-api:
        replicaCount: 1
```

The following guidelines are for the high availability of the third party deployments such as OpenSearch, Kafka. These guidelines can be altered depending on the desired resiliency.

Guidelines for third party deployments:

- OpenSearch can have minimum 3 replicas and have backend PVC storage.
- Kafka can have minimum 3 replicas and backend PVC storage.
- For database, you can create snapshots, backups, or standby databases for high availability.
- For OIDC Relying Party, a single replica is sufficient. If the Relying Party implementation supports replicas, you can add more replicas based on your business needs and the technology chosen.

Deleting and Recreating Your Solution Designer Instance

Deleting Your Solution Designer Instance

To delete your Solution Designer instance, run the following command:

```
$OCSCD_CNTK/scripts/delete-instance.sh -i ocscd
```

Recreating Your Solution Designer Instance

When you delete a Solution Designer instance, the database state for that instance still remains unaffected. You can re-create the Solution Designer instance by using the same container image.

To re-create an instance, run the following command:

```
$OCSCD_CNTK/scripts/create-instance.sh -i ocscd -s $SPEC_PATH
```

Cleaning Up the Environment

To clean up the environment:

1. Delete the instance:

   ```
   $OCSCD_CNTK/scripts/delete-instance.sh -i ocscd
   ```

2. Delete OpenSearch, Kafka, Rely Party, PVC/PV
3. Delete the namespace, which in turn deletes the Kubernetes namespace and the secrets:

   ```
   kubectl delete namespace ocscd
   ```

4. Drop the PDB or ADB.

Troubleshooting Issues with the Scripts

This section provides information about troubleshooting some issues that you may come across when running the scripts.

Timeout Issue

While running **install-db.sh** script, the install fails with a timeout.

```
fatal: [local]: FAILED! => {"changed": true, "duration": 124, "method":
"create", "msg": "\"Job\" \"pre-deploy-initiative-manager-semele\": Timed out
waiting on resource"
```

To resolve this issue, observe the Kubernetes Pod logs for the Kubernetes Job **pre-deploy-initiative-manager-semele** mentioned in the error message to determine the database error. Mostly, it is related to the connectivity from the Pod to the database or incorrect credentials. Check the connectivity from the pod to the database and provide correct credentials.

Recreating an Instance

If you face issues when creating an instance, do not try to re-run the **create-instance.sh** script as this will fail. Instead, perform the cleanup activities and then run the following command:

```
$OCSCD_CNTK/scripts/create-instance.sh -i ocscd -s $SPEC_PATH
```

Cleanup Failed Instance

When a create-instance script fails, you must clean up the instance before making another attempt at instance creation.

To clean up the failed instance, delete the instance:

```
$OCSCD_CNTK/scripts/delete-instance.sh -i ocscd
```

Pre-deploy Initiative Manager Failure

The **pre-deploy-initiative-manager-semele** fails with the following log:

```
com.oracle.communications.semele.cli.SemeleCli getExitCode
SEVERE: Execution failed:
com.oracle.communications.semele.DbManagerException: java.sql.SQLException:
ORA-28009: connection as SYS should be as SYSDBA or SYSOPER
```

You must check if the database admin user has the required permissions.

When running **manage-instance-credentials** to give the database details, add 'as sysdba' to the database admin username.

Example:

```
OCSCD DB Admin(sys) Username: sys as sysdba
```

Post-Installation Failure

You may receive the following error in the post-installation process:

```
Error: failed post-install: job failed: BackoffLimitExceeded
```

When you receive the errors, check the microservice in the Kubernetes namespace.

To see what pod has failed, run `kubectl get pods`.

Run `kubectl describe` *`PodName`* on the pod.

Run `kubectl logs` *`PodName`* on the pod.

Post-Installation Workspace Manager Failure

The **post-install-workspace-manager** may fail with the error,

```
Participants could not be configured. HTTP return code 401.
```

You can use one of the following two methods to fix the error:

- Remove any self-signed certificates as self-signed certificates are not supported in Solution Designer.
- If there is an issue with the OIDC parameters or the OIDC provider, connect to the **workspace-manager-pod** and find out what error occurs.

  Run the following command to connect to the **workspace-manager-pod**:

  ```
  kubectl -n namespace exec -ti workspace-manager-pod -- bash
  ```

After you connect to the **workspace-manager-pod**, run the following script to find out what error has occurred:

```
real_scope=${LCM_OPERATION_SCOPE}
base64_data=$(echo -n "$OIDC_CLIENT_ID:$OIDC_CLIENT_SECRET" | base64 -w 0)
header1="Authorization: Basic "$base64_data
header2="Content-Type: application/x-www-form-urlencoded;charset=UTF-8"
data1="grant_type=client_credentials"
curl -s -H "$header1" -H "$header2" --request POST "$OIDC_TOKEN_ENDPOINT" -
d "$data1" -d "scope=$real_scope"
```

Creating New Connections

After you create the Solution Designer instance, in the Solution Designer Web client user interface, create a new connection in the Workspace application to publish the initiatives to the UIM Participant.

See "Creating New Connections" in *Solution Designer User's Guide* on how to create new connections.

Next Steps

The Solution Designer instance is ready for use. To understand details on infrastructure setup and structuring of the instances for your organization, proceed to Planning Infrastructure.

5

Planning Infrastructure

This chapter provides details about setting up infrastructure and structuring Solution Designer instances for your organization.

See the following topics:

- Sizing Considerations
- Managing Configuration as Code
- Setting Up Automation
- Securing Operations in Kubernetes Cluster

Sizing Considerations

This section describes the hardware sizing guidelines and memory sizing guidelines for Solution Designer on Oracle Linux. It also includes general sizing guidelines applicable to any solution type. These guidelines are intended to assist in estimating the total Solution Designer system requirements. These guidelines do not contain express or implied warranties of any kind. This is a guideline for cluster sizing and altering as needed based on the values used in the **instance.yaml** file.

Table 5-1 provides recommended system hardware requirements for Solution Designer installed on an Oracle Linux platform.

Table 5-1 Hardware Requirements for Linux

Software Components	Recommended Size Linux (2 threads per core)
Solution Designer	4 vCPUs
Peristent Volumes	25 GB
Disk space	40 GB
Total Memory	11 GB approximately

The default memory limits defined for the deployment in the Kubernetes cluster:

Table 5-2 Memory Sizing

Component	Memory Limit
initiative-manager	4Gi
workspace-manager	1Gi
landing-page-api	1Gi
unifieddesigner-ui	1Gi
ocscd-router	512Mi
uim-participant	1Gi + 2Gi

The default settings that get used for **instance.yaml** are as follows:

```
initiative-manager:
    resources:
        limits:
            memory: 4Gi
workspace-manager:
    resources:
        limits:
            memory: 1Gi
landing-page-api:
    resources:
        limits:
            memory: 1Gi
unifieddesigner-ui:
    resources:
        limits:
            memory: 1Gi
ocscd-router:
    resources:
        limits:
            memory: 1Gi
uim-participant:
    resources:
        limits:
            memory: 1Gi
    builder:
        resources:
            limits:
                memory: 2Gi
```

> **Note:**
>
> As the usage of OpenSearch and Kafka for Solution Designer is very minimal, the memory size allocation can be minimal.

Managing Configuration as Code

Managing Configuration as Code involves the following tasks:

- Creating Source Control Repository
- Setting the Repository Path During Instance Creation

Creating Source Control Repository

Managing Configuration as Code (CAC) is a central tenet of using Solution Designer cloud native. You must create a source control repository to store all configuration that is necessary to re-create a Solution Designer instance (or database) if it is lost. This does not include the toolkit scripts.

Setting the Repository Path During Instance Creation

To offer flexibility in how the repository directory structure develops, the **create-instance** script takes as input, the path to the specification files.

The **-s specPath** parameter is mandatory in **create-instance.sh** and can point to several directories at once (directories are separated by a colon).

Setting Up Automation

This section describes the complete sequence of activities for setting up a Solution Designer environment with the aim of grouping repeatable steps into high-level categories. You should start to plan the steps that you can automate to some degree.

> **Note:**
>
> These steps exclude any one-time setup activities. For details on one-time setup activities, see the tasks you must do before creating an instance in "Creating a Solution Designer Cloud Native Instance".

The toolkit provides sample scripts for this purpose. However, they are not pipeline-friendly. The scripts should be used for creating an instance manually and quickly, but not for any automated process for creating instances. The scripts also illustrate both the naming of the secret and the layout of the data within the secret that Solution Designer cloud native requires. You must create secrets prior to running the install-db.sh or create-instance.sh scripts.

Configuring Code for Creating a Solution Designer Instance

To configure code for creating an instance, you assemble the instance configuration. While some of these activities could be automated, much of the work is manual in nature.

1. Assemble the configuration.
   To assemble the configuration for an instance:

   To assemble the configuration for an instance, copy **$OCSCD_CNTK/samples/instance.yaml** to your file repository and rename to align with your instance naming decisions made earlier (for example, *instance*.yaml).
2. Create the required secrets for Solution Designer DB access, OIDC, and so on.

```
$SCD_CNTK/scripts/manage-instance-credentials.sh -i ocscd create
db,kafka,ocscd-oidc,s3
```

> **Note:**
>
> Passwords and other secret input must adhere to the rules specified of the corresponding component.

After the configuration and the input are available, the remaining activities are focused on Continuous Delivery, which can be automated.

1. Create one Solution Designer PDB or ADB per instance.

2. Create the instance.

```
$OCSCD_CNTK/scripts/install-db.sh -i ocscd -s $SPEC_PATH -c 4
$OCSCD_CNTK/scripts/create-instance.sh -i ocscd -s $SPEC_PATH
```

Deleting an Instance

This section describes the sequence of activities for deleting and cleaning up various aspects of the Solution Designer environment.

To delete the application instance:

1. Run the following command:

```
$OCSCD_CNTK/scripts/delete-instance.sh -i ocscd
```

To clean up the database, drop PDB or ADB.

To clean up the configuration as code:

1. Delete the Solution Designer instance and the database instance specification files.
2. Delete the secrets:

```
$OCSCD_CNTK/scripts/manage-instance-credentials.sh -i ocscd delete
db,kafka,ocscd-oidc,s3
```

Securing Operations in Kubernetes Cluster

This section describes how to secure the operations of Solution Designer cloud native users in a Kubernetes cluster. A well organized deployment of Solution Designer cloud native ensures that individual users have specific privileges that are limited to the requirements for their approved actions. The Kubernetes objects concerned are service accounts, roles, and container registry secrets.

Service Accounts

The Solution Designer application requires a Kubernetes service account to manage Secrets. Solution Designer uses Kubernetes Secrets to store connection credentials to other systems. Solution Designer requires a namespace service account in the namespace. This namespace service account need not be a cluster role.

```
kubectl -n ocscd apply -f $OCSCD_CNTK/samples/rbac/secrets-admin.yaml
```

Roles

Assign the appropriate role or cluster role for the service accounts based on:

- The deployment choice for OpenSearch and Kafka.
- Whether the deployment uses a Kubernetes operator.

The Service Catalog Administrator's RBAC can be much more limited. For a start, it would be limited to only that instance's namespace. Further, it would be limited to the set of actions and objects that the instance-related scripts manipulate when run by the Administrator. This set of

actions and objects is documented in the Service Catalog and Design toolkit sample located in the **samples/rbac** directory.

There are many ways to structure permissions within a Kubernetes cluster. There are clustering applications and platforms that add their own management and control of these permissions. Given this, the Service Catalog and Design toolkit provides a set of RBAC files as a sample.

These samples are in **samples/rbac** directory within the toolkit. The key files are **scd-admin-role.yaml** and **scd-admin-rolebinding.yaml**. These files govern the basic RBAC for a Service Catalog Administrator.

Container Registry Secrets

You must protect the container registries using authentication. Before deploying Solution Designer, create a kubernetes.io/dockerconfigjson secret with your credentials. After you create the secret, reference that secret name in the instance specification file. The following is an example for the initiative manager.

```
initiative-manager:
    image:
        pullSecrets:
            - name: my-regcred
```

Managing Logs

Solution Designer uses the Kubernetes architecture for collecting logs. Kubernetes captures logs from each container in a running Pod. Solution Designer logs the data to the stdout(standard output) stream from the container it is running in. A log collection infrastructure can be used to collect the logs from the Kubernetes Cluster. You can use any log collection tool such as OpenSearch, fluentd and so on. After the logs are collected, you can use any dashboard to view the logs.

For OpenSearch, use the following container images:

- opensearchproject/opensearch
- opensearchproject/opensearch-dashboards

For using Fluentd to collect logs and push to OpenSearch, use the following container image:

- fluent/fluentd-kubernetes-daemonset:v1-debian-opensearch

An optional example pattern for the logs using Fluentd is shown for the body of the log message:

```
/^(?<time_stamp>\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2}.\d{3})\s+\[(?
<thread>[^\]]*)\]\s(?<severity>\w*)\s+(?<class>[^\s]*)\s-\s(?
<messagebody>.*)$/m
```

The CNTK provides a sample implementation for Fluentd which is located at **$OCSCD_CNTK/logging/fluentd**.

6
Upgrading the Solution Designer Environment

This chapter describes the tasks you perform in order to apply a change or upgrade a component in your Solution Designer cloud native environment.

Before you upgrade your cloud native environment, you must compare the sample instance specification of the new toolkit with the sample from the current one and migrate your customizations to the new specification.

Solution Designer Upgrade Procedures

The Solution Designer cloud native owned upgrade procedures are:

1. Specification File Update
2. Database Schema upgrade
3. Solution Designer application upgrade

Change or upgrade procedures that are dictated by Solution Designer are applied using the scripts and the configuration provided in the toolkit.

Specification File Update

You can update the instance specification file for the changes in database schema and the Solution Designer application. Examples include updating the Solution Designer DB Installer image, changing an existing value, changing the Solution Designer images, or supplying something new such as a secret.

Edit the instance specification to set the container image location and tag in each microservice location. See "Configuring the Instance Specification" for more information on configuring the instance specification file.

Database Schema Upgrade Procedure

Changes impacting the database Schema can be found in any of the instance specification file.

To perform a database schema upgrade:

1. Scale down initiative-manager, workspace-manager, and landing-page-api.

   ```
   $OCSCD_CNTK/scripts/scale-down.sh -i ocscd -s $SPEC_PATH -m im,wm,lpapi
   ```

2. Run the following command to install schema in database:

   ```
   $OCSCD_CNTK/scripts/install-db.sh -i ocscd -s $SPEC_PATH -c 4
   ```

Solution Designer Application Upgrade

Changes impacting the Oracle Communications Service Catalog and Design - Solution Designer application can be found in any of the instance specification files.

Run the following command to upgrade the Solution Designer instance to push out the changes you just made to the running instance:

```
$OCSCD_CNTK/scripts/update-instance.sh -i ocscd -s $SPEC_PATH
```

Upgrades to Infrastructure

From the point of view of Solution Designer instances, upgrades to the cloud infrastructure is rolling upgrades.

> **Note:**
>
> All infrastructure upgrades must continue to meet the supported types and versions listed in the Service Catalog and Design documentation's certification statement.

Rolling upgrades are where, with proper high-availability planning (like anti-affinity rules), the instance as a whole remains available as parts of it undergo temporary outages. Examples of this are Kubernetes worker node OS upgrades, Kubernetes version upgrades.

Kubernetes Infrastructure Upgrades

Follow standard Kubernetes practices to upgrade these components. The impact at any point should be limited to one node - master (Kubernetes and OS) or worker (Kubernetes, and OS). If a worker node is going to be upgraded, drain and cordon the node first. This will result in all pods moving away to other worker nodes. This is assuming your cluster has the capacity for this - you may have to temporarily add a worker node or two. For Solution Designer instances, any pods on the cordoned worker will suffer an outage until they come up on other workers. As each worker undergoes this process in turn, pods continue to terminate and start up elsewhere, but as long as the instance has pods in both affected and unaffected nodes, it will continue to process orders.

Miscellaneous Upgrade Procedures

This section describes miscellaneous upgrade scenarios.

Network File System (NFS)

If an instance is created successfully, but a change to the NFS configuration is required, then the change cannot be made to a running Solution Designer instance. In this case, the procedure is as follows:

1. Stop the Solution Designer instance or specifically the uim-participant.
2. Update the `nfs` details in the instance specification.
3. Start the instance.

7

Maintaining Solution Designer Environment

This chapter describes the tasks you perform to maintain the Solution Designer environment.

Running Operational Procedures

This section describes the tasks you perform for a planned upgrade to the Solution Designer cloud native environment. You must consider if the change in the environment fundamentally affects the processing to the extent that Solution Designer should not run when the upgrade is applied or Solution Designer can run during the upgrade but must be restarted to properly process the change.

The operational procedures are performed using the Solution Designer cloud native specification files and scripts.

The operational procedures you perform for upgrading your cloud environment are:

- Scaling down the instance
- Scaling up the instance
- Restarting the instance

Rolling Restart

Occasionally, you may need to restart microservices in a rolling fashion, one at a time. This does not result in downtime, but only reduced capacity for a limited period. A rolling restart can be triggered by invoking the **restart-instance.sh** script. This script can restart the whole instance in a rolling fashion, or only the admin server or all the managed servers in a rolling fashion. Some operations may automatically trigger rolling restart. These include changes such as image updates, parameter changes, and so on pushed via the **upgrade-instance.sh** script.

Scaling Down the instance

The scaling down is bringing down to 0 microservices. This does not include any of the third party services running in the cluster like Opensearch, Kafka, and Relying party.

Scale down using the following command:

```
$OCSCD_CNTK/scripts/scale-down.sh -m full
```

Scaling Up the instance

Scaling up is up to the initial replica count. A generalized scaling can change the number of replicas up to a value between 0 and 1.

To scale up the instance, run the update instance script:

```
$OCSCD_CNTK/scripts/update-instance.sh -i ocscd -s $SPEC_PATH
```

Restarting the Instance

The Solution Designer cloud native toolkit provides a script (**restart-instance.sh**) that you can use to perform different flavors of restarts on a running instance of Solution Designer cloud native.

Following is the usage of the **restart-instance.sh** script

```
restart-instance.sh parameters
      -i instanceName : mandatory
      -s specPath : mandatory; locations of specification files
      -r restartType : mandatory; what kind of restart is requested
    # restartType can take the following values:
      * full        - Restart the whole instance (rolling restart)
      * lpapi       - Restart all the Landing Page API Servers (rolling
restart)
      * im          - Restart all the Initiative Manager Servers (rolling
restart)
      * wm          - Restart all the Workspace Manager Servers (rolling
restart)
      * ocscdui     - Restart all the OCSCD UI Servers (rolling restart)
      * up          - Restart all the UIM Participant Servers (rolling
restart)
      * router      - Restart all the ocscd-router Servers (rolling restart)

    # or just -h for help
```

For example, to restart a complete instance, run the following command:

```
$OCSCD_CNTK/scripts/restart-instance.sh -i ocscd -s $SPEC_PATH -r full
```

Rotating Secrets

Rotating secrets and passwords is an important part of securing your instance. To perform credential update, run the following:

1. Update the password or secret in the system that is being updated For example, OIDC secret in your IDP.
2. Run the following command to update the secret into SCD.

   ```
   $OCSCD_CNTK/scripts/manage-instance-credentials.sh -i ocscd update
   <secrets>
   ```

   Choose one or more of the following as needed:

   - **ocscd-oidc**: OIDC client ID and secret
   - **s3**: access key and secret
   - **db-schema**: Database schema passwords
   - **db**: Administrator user password
3. Restart Solution Designer for the microservices to uptake the new secrets. See "Restarting the Instance" on how to restart the instances.