Oracle® Fusion Middleware

Running Oracle WebLogic Server on Docker

12*c* (12.2.1.1.0)

E70166-01

June 2016

ORACLE

Oracle Fusion Middleware Running Oracle WebLogic Server on Docker, 12c (12.2.1.1.0)

E70166-01

Primary Author: Jeffrey Schieli

Contributing Authors: Monica Riccelli

Contributors: Bruno Borges

Contents

Preface

This section describes the intended audience, how to use this guide, related documents, and provides information about documentation accessibility.

Audience

This document is intended for application developers who want to quickly create lightweight clustered and non-clustered WebLogic Server domain configurations on Docker, a Linux-based container technology, in order to run a single host OS or on VMs, for either development or production environments.

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.

Related Documents

For more information, see the following documents in the Oracle WebLogic Server documentation set:

- *Understanding Domain Configuration for Oracle WebLogic Server*
- *Creating WebLogic Domains Using the Configuration Wizard*
- *Administering Server Startup and Shutdown for Oracle WebLogic Server*
- *Administering Node Manager for Oracle WebLogic Server*
- *Understanding the WebLogic Scripting Tool*
- *Developing Applications for Oracle WebLogic Server*
- *Deploying Applications to Oracle WebLogic Server*

Conventions

The following text conventions are used in this document:

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

1

Getting Started

This section discusses how Oracle WebLogic Server 12.2.1 can be configured to run inside a Docker container. Docker is a Linux-based container technology that enables you to quickly create lightweight clustered and non-clustered WebLogic Server domain configurations on a single host OS or virtual machines, for either development or production environments.

Topics include:

1.1 About Docker

Docker is a platform that enables users to build, package, ship and run distributed applications. Docker users package up their applications, and any dependent libraries or files, into a Docker image.

Docker images are portable artifacts that can be distributed across Linux environments. Images that have been distributed can be used to instantiate containers where applications can run in isolation from other applications running in other containers on the same host operating system.

1.2 About WebLogic Server Images on Docker

Oracle, as part of the certification, has released Dockerfiles and supporting scripts on GitHub that are used to build images for WebLogic Server. The posted files are examples to help you get started. The WebLogic Server images are built as an extension of the Oracle Linux image 7.0, with JDK 7 or 8, and the Oracle WebLogic Server 12c (12.2.1) installations. For more information, see About the Dockerfiles and Scripts on GitHub.

To support building your Docker images, Oracle has certified using WebLogic Server 12.2.1 with various combinations of JDK versions, Oracle Linux and Red Hat Linux OS versions, Kernel versions, and Docker versions. For detailed certification information about supported WebLogic Server Docker images, see `http://www.oracle.com/technetwork/middleware/ias/oracleas-supported-virtualization-089265.html`.

1.2.1 Custom WebLogic Server Docker Images

You can also create your own WebLogic Server Docker images. To facilitate this process, Oracle has posted Dockerfiles and scripts on GitHub as examples that can help you to get started.

These are the prerequisites to build custom WebLogic Server Docker images:

- Supported Oracle Linux or Red Hat Linux base image
- Dockerfiles and scripts from GitHub
- Oracle WebLogic Server 12*c* (12.2.1) generic installer or Developer installer
- Corresponding supported JDK

1.3 About the Dockerfiles and Scripts on GitHub

To facilitate the building and running of WebLogic Server Docker images, Oracle has posted Dockerfiles and supporting scripts on GitHub. Download the entire directory structure to build your Oracle WebLogic Server images and start your containers. To access these files, go to `https://github.com/oracle/docker/tree/master/OracleWebLogic/`. Refer to the `/wokshops` directory for guided labs that provide step-by-step insructions.

The Dockerfiles and scripts enable you to extend your image and create clustered and non-clustered Oracle WebLogic Server domain configurations, including both development and production, running on multiple host operating system or on VMs. Please note that the Dockerfiles and scripts on Github are only intended to be samples for you to write your own Dockerfiles and build your WebLogic Server images.

Each server running in the resulting domain configurations runs in its Docker container and can communicate as required with other servers. Other configurations and approaches are possible, as described in Building WebLogic Server Images on Docker.

1.3.1 What Are the Dockerfiles on GitHub?

There are two types of Dockerfiles available on GitHub for WebLogic Server 12*c* (12.2.1), located under the `/OracleWebLogic/dockerfiles/12.2.1` subdirectory:

- `Dockerfile.developer` – builds a WebLogic Server "developer" install image.
- `Dockerfile.generic` – builds a WebLogic Server "generic" domain image.

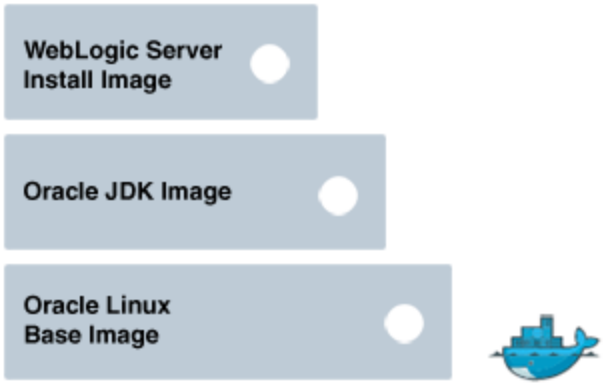
WebLogic Server Install Image Dockerfile

The Dockerfile to create an WebLogic Server install image performs the following functions:

- Extends the Oracle JDK image
- Installs WLS using the (WLS generic/Developer installers) in silent mode

Figure 1-1 illustrates the WebLogic Server Docker image.

Figure 1-1 Oracle WebLogic Server Docker Image



After creating your WebLogic Server install images, you can extend them to have a base WebLogic Server domain configured.

WebLogic Server Domain Image Dockerfile

The Dockerfile to create an WebLogic Server domain image performs the following functions:

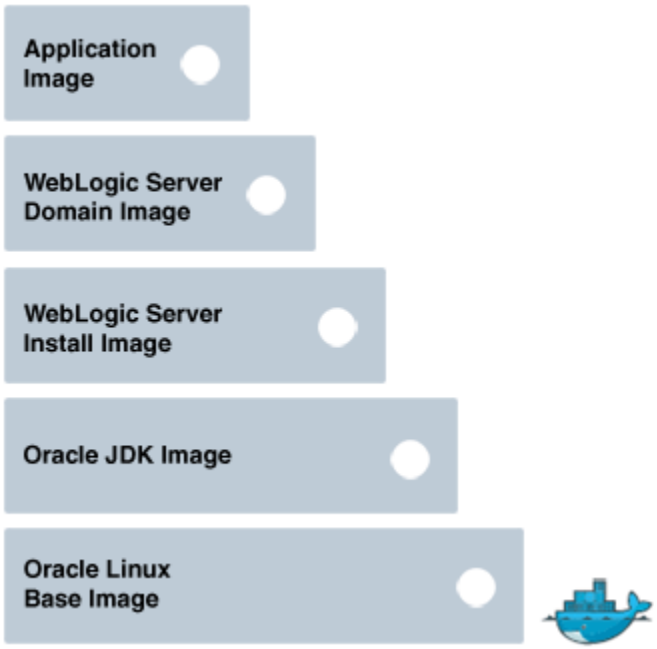
- Extends the WebLogic Server install image
- Configures a WebLogic Server domain by calling WLST scripts. The domain has one Admin server, a JMS server, a Data Source, and enables JAX-RS 2.0

There are additional Dockerfiles to create an Application image. The Dockerfile performs the following functions:

- Extends the domain image
- Deploys an application to an Oracle WebLogic Server domain

Figure 1-2 illustrates a WebLogic Server domain and application image.

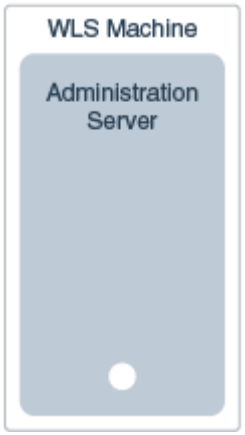
Figure 1-2 WebLogic Server Domain and Application Image



You can create two types of containers using the Oracle WebLogic Server domain image:

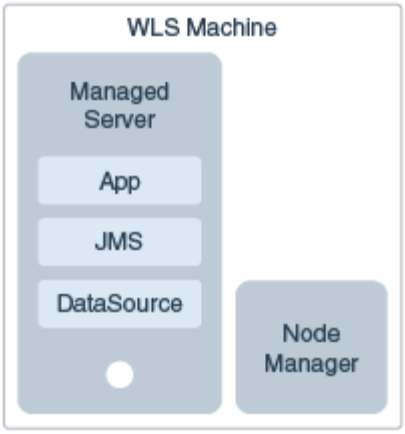
- Administration server container with a single WebLogic Server Administration server, as shown in Figure 1-3.

  Figure 1-3 Administration Server Container

- Managed server container with a node manager, which adds itself as a machine to the Administration Server and a Managed Server, as shown in Figure 1-4.

Figure 1-4 Managed Server with Node Manager Container



1.3.2 What Are the Scripts on GitHub?

The supported scripts aid in the creation of a WebLogic Server 12*c* (12.2.1) Docker image and serve as examples to extend the image with the configuration of a WebLogic Server domain. The scripts are located under the subdirectories `/OracleWebLogic/dockerfiles`, `/OracleWebLogic/samples`, and `/OracleWebLogic/samples/1221-domain/container-scripts`.

The scripts in Table 1-1 help in the creation of a WebLogic Server install image and the starting of WebLogic servers inside of a Docker container.

Table 1-1 Supported WebLogic Server Scripts for Docker on GitHub

Script	What it does
`buildDockerImage.sh`	Builds the image using the WebLogic Server installation Dockerfile instructions.
`add-machine.py`	WLST scripts to create a machine using the Managed Server container name.
`add-server.py`	WLST scripts to create a Managed Server.
`create-wls-domain.py`	WLST script configures a base domain with one Administration Server, JMS server, JSP, and data source.
`createMachine.sh`	Starts a Node Manager in the container and calls `addMachine.sh` to start Node Manager and add the Node Manager machine.
`createServer.sh`	Starts a Node Manager in the container and calls `add-server.py` to configure a Managed Server in the machine create by `add-machine.py`.
`rm_containers.sh`	Removes all running containers.
`clean-up-docker.sh`	Removes all ghost containers and all ghost images.

1.4 Clustering WebLogic Server on Docker Containers

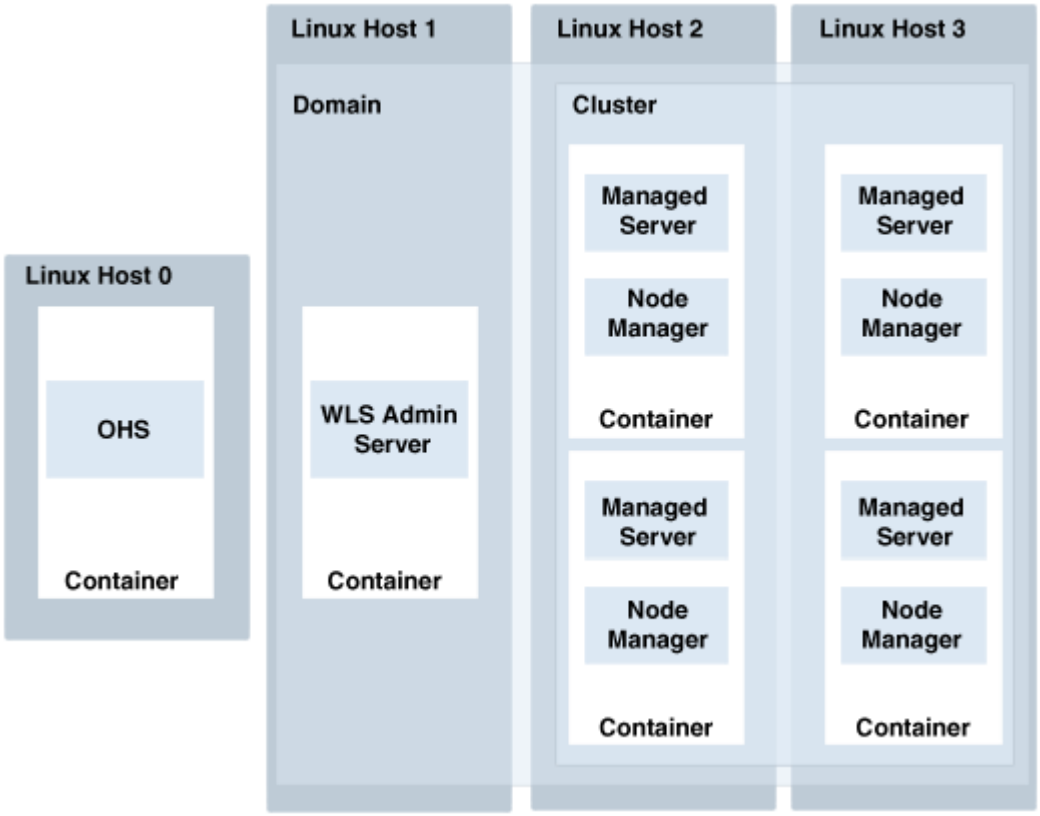
WebLogic Server uses a *machine* concept, which is an operational system with an agent– the *Node Manager*. This machine resource allows an Administration Server to create and assign Managed Servers to a domain and/or cluster, expand a domain and/or cluster, and to deploy applications and resources to the Managed Servers.

By using machines in containers, you can configure a Dynamic Cluster and easily scale up your cluster by starting new Managed Server containers. Using the WebLogic Server Scripting Tool (WLST), your cluster can quickly be scaled in and out. For more information about the Node Manager, see *Administering Node Manager for Oracle WebLogic Server*, and for more information about using WLST, refer to *Understanding the WebLogic Scripting Tool*.

The Docker containers enable you to create clustered and non-clustered WebLogic Server domain configurations. Each server in the domain runs in its own Docker container and is capable of communicating as required with other servers.

Figure 1-5 illustrates clustering WebLogic Server on Docker Containers.

Figure 1-5 Clustering Oracle WebLogic Server on Docker Containers



The advantages of this topology are:

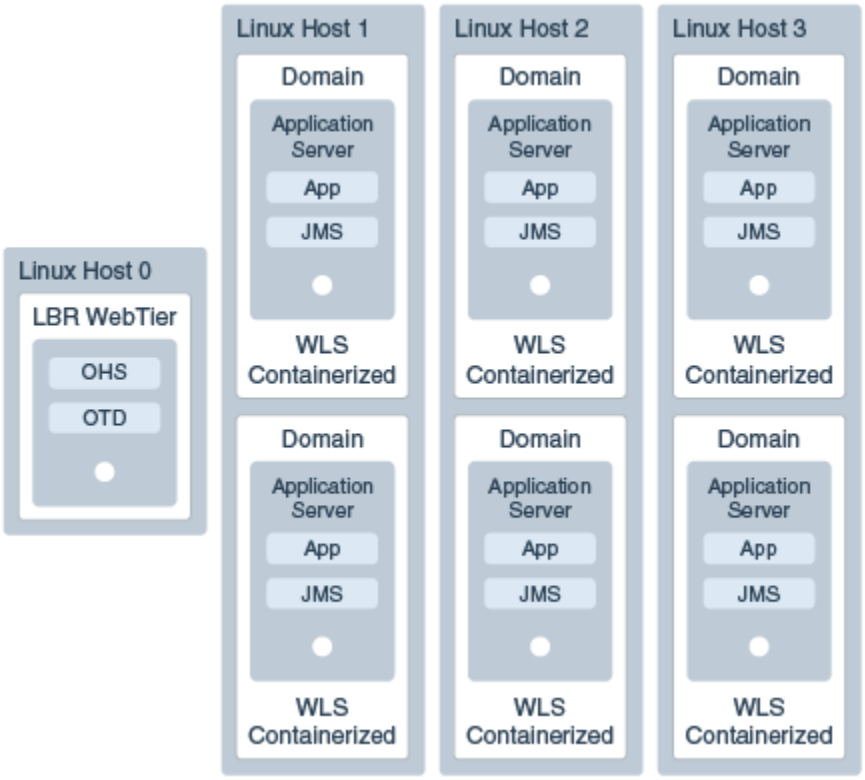
- Good for traditional-like deployments
- Easy to deploy containers from Oracle WebLogic Server domain images
- Easy to scale a cluster up or down

- Great for developers looking for a lightweight, repeatable, and shareable WebLogic environment
- Good for continuous deployments
- Easy to mange because there is no need to install or configure anything on a host except for Docker binaries

1.4.1 Non-Clustered WebLogic Server Domain in a Docker Container

A recommended topology that is in line with the "Docker way" for configuring containerized applications and services, consists of a container designed to run only an WebLogic Administration Server that contains all resources, shared libraries, and deployments. The Docker image includes all predefined domain resources, applications, and shared libraries deployed up-front and no Managed Servers or clusters are configured as shown in Figure 1-6.

Figure 1-6 Containerized WebLogic Server Applications on a Multiple Hosts



The advantages of this topology are:

- Uses the Docker "recommended" way for configuring containerized applications and services.
- Containers are easily repeatable.
- Each container is an instance of the same WebLogic domain.

2

Building WebLogic Server Images on Docker

This section discusses how to use the Dockerfiles and supporting scripts that Oracle has made available on GitHub to build your own WebLogic Server 12.2.1 Docker images.

Topics include:

2.1 Building WebLogic Server Images on Docker

To build the Oracle WebLogic install image, you must first have the Oracle Linux and Oracle JDK images already running. Pull the Oracle Linux 6 or 7 image from Docker Hub. To create the Oracle JDK image, download the Dockerfile from GitHub/Oracle JDK; this Docker file extends the Oracle Linux image and installs the JDK. Download the JDK into the `/OracleJDK/java-x` directory.

```
$ sudo docker build -t oracle/jdk:8
```

Before you begin, choose the installation type you want to use, either the Generic or Developer installer, as described in About WebLogic Server Images on Docker. Then follow these steps:

1. Download the required WebLogic Server installer in the `dockerfiles/12.2.1` directory.
2. Change to the `/dockerfiles` directory and run the `buildDockerImage.sh` script as root:

   ```
   $ sudo sh buildDockerImage.sh -h
   ```

   Usage: `buildDockerImage.sh [-d|-g] [-v] 12.2.1`

   where `d:` creates image based on the developer distribution.

where `g:` creates image based on the generic distribution.

where `v:` specifies the WebLogic version.

Note:

The resulting image will *not* have a domain preconfigured. Oracle provides a separate Dockerfile and supporting scripts to extend the WebLogic Server install image and create a WebLogic Server domain image.

2.1.1 Samples for WebLogic Server Domain Creation

To give you an idea on how to create a domain from a custom Dockerfile to extend the WebLogic Server install image, Oracle provides some samples for WebLogic Server 12*c* (12.2.1) for both the Developer and Generic distributions. The samples are in the `samples/1221-domain` directory.

2.1.2 Sample Domain for WebLogic Server 12*c* (12.2.1)

This Dockerfile will create an image by extending `oracle/weblogic:12.2.1-dev` (from the Developer distribution). It will configure a `base_domain` with the following settings:

- Domain Name: `base_domain`
- Admin Port:`8001`
- Administrator Username: `weblogic`
- Administrator Password: `welcome1`
- Oracle Linux Username: `oracle`
- Oracle Linux Password: `welcome1`
- Cluster Name: `DockerCluster`
- Managed Server Port: `7001`
- NodeManager Port : `5556`
- JVM Memory Settings: `-Xms236m –Xmx512m –XXLMacPermSize=2048m`

2.1.3 Write Your Own Oracle WebLogic Server Domain with WLST

The best way to create your own domain, or to extend domains, is by using WLST. The WLST script used to create domains in the Dockerfile container is `create-wls-domain.py`. This script by default adds JMS resources and a few other settings.

You may want to tune this script with your own setup to create data sources and connection pools, security realms, deploy artifacts, and so on. You can also extend images and override the existing domain, or create a new one with WLST.

For more information about using WLST, refer to *Understanding the WebLogic Scripting Tool*.

2.1.4 Building a Sample Docker Image of a WebLogic Server Domain

To build a sample of a WebLogic Server image with a domain configured, follow these steps:

1. Make sure you have `oracle/weblogic:12.2.1-dev` image built as described in Building WebLogic Server Images on Docker. If not, change to the `/dockerfiles` directory and run the `buildDockerImage.sh` script as root:

   ```
   $ sudo sh buildDockerImage.sh -v 12.2.1 [-d|-g]
   ```

2. Change to the `/samples/1221-domain` directory and run the `docker build` command:

   ```
   $ sudo docker build -t samplewls:12.2.1 –build-arg ADMIN_PASSWORD=welcome1
   ```

3. Verify that you have the image in place by running the `docker images` command:

   ```
   $ sudo docker images
   ```

2.2 Running an Administration Server Container

When you use the WebLogic Server domain image to start your container, an Administration Server starts running in the container by default. The default Administration Server name is `AdminServer`; the default port configuration is `8001`; and the default Administration Server container name is `wlsadmin`. When running more than one domain in the same single host, you must change the Administration Server name, port number, and container name.

To start the Administration Server:

1. Execute the `docker run` command:

   ```
   $ sudo docker run -d --name=wlsadmin samplewls:12.2.1
   ```

   where `samplewls:12.2.1` is the WebLogic Server domain image tag. The samples Dockerfiles define `startWebLogic.sh` as the default CMD (command).

2. To obtain the IP address of the Administration Server Container run the `docker inspect` command:

   ```
   $ sudo docker inspect --format '{{ .NetworkSettings.IPAddress }}' wlsadmin
   ```

3. Open the Administration Server's web-based console at `http:// xxx.xx.x.xx:8001/console`.

Note:

If you have multiple WebLogic Server domains running on the same host (such as multiple Administration Servers), change the `-name` parameter (name of the Administration Server container) and the `-p` parameter (port of the Administration Server).

2.3 Running a Managed Server Container

Managed Server containers have a Node Manager and a Managed Server running in it. These Managed Server containers communicate to an Administration Server container by linking (`-link` command) using the Administration Server container name. The Administration Server container name defaults to `wlsadmin`.

When there are more than one domain running on the same host, then the Administration Server container name needs to be unique; you need to change the Administration Server container name by using the `-name` parameter and matching the name given in the `-link` command of each Managed Server container.

There are three different ways to start a Managed Server container:

- Start Node Manager (manually):

  ```
  $ sudo docker run -d -link wlsadmin:wlsadmin <image-name> startNodeManager.sh
  ```

- Start Node Manager and create a Machine automatically:

  ```
  $ sudo docker run -d -link wlsadmin:wlsadmin <image-name>  createMachine.sh
  ```

- Start Node Manager, create a Machine, and create a Managed Server automatically:

  ```
  $ sudo docker run -d -link wlsadmin:wlsadmin <image-name>  createServer.sh
  ```

2.3.1 Sample Managed Server Command and Parameters

A sample `docker run` command for a Managed Server container and a listing of available parameters is as follows:

```
$ sudo docker run -d -link wlsadmin:wlsadmin \
     -p <NM Port>:5556 -p <MS Port>:<MS Port> \
     -name=<Container name> \
     -e MS_HOST=<Host address where Managed Server container runs> \
     -e MS_PORT=<Managed Server port> \
     -e NM_HOST=<Host address where Managed Server container runs> \
     -e NM_PORT=<Node Manager Port (should match the port in the -p)> \
     <image name> \
     <createMachine.sh, startNodeManager.sh, createServer.sh>
```

2.3.2 Script Variables

The supported scripts have a list of variables that must be properly configured, as defined in Table 2-1.

Table 2-1 Script Variables and Definitions

Variable	Definition
`ADMIN_USERNAME`	Username of the AdminServer `weblogic` user. Default: `weblogic`
`ADMIN_PASSWORD`	Password of `ADMIN_USERNAME`. Defaults to value passed during Dockerfile build. (`welcome1` in the samples)

Table 2-1 (Cont.) Script Variables and Definitions

Variable	Definition
ADMIN_URL	t3 URL of the AdminServer. Default: t3://wlsadmin:8001
CONTAINER_NAME	Name of the Machine to be created. Default: node manager_ + hash of the container
NM_HOST	IP address where Node Manager can be reached. Default: IP address of the container
NM_PORT	Port of Node Manager. Default: 5556
MS_HOST	IP address where Managed Server can be reached. Default: IP address of the container
MS_PORT	Port of Managed Server. Default: 7001

2.3.3 Examples of Using the Script Variables

If you want to run a "Single-Host" configuration on a remote server, you must expose ports and addresses of the Admin server, Managed Servers and Node Manager as shown in the following examples:

```
$ sudo docker run -d -link wlsadmin:wlsadmin -p 5556:5556 -name="wlsnm0" -e
NM_HOST="xx.xxx.xx.xxx" -e NM_PORT="5556" samplewls:12.2.1 createMachine.sh

$ sudo docker run -d -link wlsadmin:wlsadmin   -p 7003:7003
-e MS_HOST=xx.xxx.xx.xxx -e MS_PORT=7003 samplewls:12.2.1 createServer.sh

$ sudo docker run -d -link wlsadmin:wlsadmin   -p 7002:7002
-e MS_HOST= xx.xxx.xx.xxx  -e MS_PORT=7002 samplewls:12.2.1 createServer.sh
```

Note:

You must assign a new, unique listen port when you create an additional Managed Server container on a host OS where a Managed Server container is already running. This prevents multiple Managed Servers running on the same host OS from listening on the same listen port.

If you used the `createServer.sh` command:

1. Access the Administration Server console at `http://admin-container-ip:8001 /console`.
2. In the **Domain Structure** tree, expand **Environment**.
3. Select **Machines** to open the **Summary of Machines** page and verify that you have a Machine registered.

4. Click your registered Machine in the table, and then use the **Node Manager** and **Servers** tabs to verify that your Node Manager and Managed Server are also configured.
5. Open the **Servers** page, select the **Control** tab, and start your server.

2.4 Communicating With Servers on a Remote Host

Another possible topology is to run a single Admin Server container communicating with Oracle WebLogic Server running on a remote host. Use `-add-host` so that the container assumes the IP address of the host where it is running instead of the local container IP address:

```
$ sudo docker run -d -p 8001:8001 -net=host -add-host=hostname: <host ip address
where container is running> -name wlsadmin samplewls:12.2.1
```

For this topology to work, the following configurations are necessary:

- The listen address of the AdminServer in the Docker container has to be configured.
- The listen address of the AdminServer in the remote host has to be configured.
- The client must use the hosts IP addresses to get the initial context for JNDI lookup.

2.5 Creating a WebLogic MedRec Sample Domain

The Supplemental Quick Installer is a lightweight installer that contains all the necessary artifacts to develop and test applications on Oracle WebLogic Server 12.2.1. You can extend the WebLogic developer install image (`oracle/weblogic:12.2.1-dev`) to create a WebLogic Server domain image with the MedRec application deployed. The Supplemental Quick Installer is located at `/samples/1221-medrec`.

To create the MedRec sample domain:

1. Make sure you have the oracle/weblogic:12.2.1-dev image built. If not go into dockerfiles and call:

   ```
   $ sudo sh buildDockerImage.sh -v 12.2.1 –d
   ```

2. Changed directories to `/samples/1221-domain` and run the following command:

   ```
   $ sudo docker build -t samplewls:12.2.1
   ```

3. Build the medrec image:

   ```
   $ sudo docker build -t 1221-medrec
   ```

4. Run a container from this new sample domain image:

   ```
   $ sudo docker run –d –p 7011:7011 1221-medrec
   ```

5. Access the AdminServer Console at `http://localhost:7011/medrec`.

2.6 Running an Oracle WebLogic Server Domain in a Multi Host Environment

Docker 1.9 introduced the ability to network containers together running on multi-host operating systems or virtual machines. Oracle WebLogic Server 12c domains are certified on servers running in Docker containers that were distributed in different physical hosts or virtual machines. In this Docker environment, the WebLogic servers running in the cluster have all the high availability properties of a WebLogic Server cluster (for example Session Replication and Singleton Service Migration).

Docker has developed tools that make it significantly easier to create such multi-host environments and network them together:

- **Docker Machine**: Docker Machine is a tool that lets you install Docker Engine on virtual hosts and manage the hosts with docker-machine commands.
- **Docker Swarm**: Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Because Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.
- **Docker Overlay Network**: Dockers Overlay network driver supports multi-host networking natively out-of-the-box while still providing better container isolation.
- **Docker Compose**: Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application services. Then, using a single command, you create and start all the services from your configuration.
- **Docker Registry**: The Registry is a stateless, highly scalable server side application that stores and lets you distribute Docker images.
- **Consul**: Consul makes it simple for services to register themselves and to discover other services via a DNS or HTTP interface.

Note:

To run Docker 1.10 or higher, you need UEK4

2.6.1 Building Application Images

In a WebLogic Server domain running in a Docker environment, you deploy applications by extending the WebLogic Server Domain image to create an Application image. Samples are provided on GitHub at `/samples/1221-appdeploy`. The WLST script used to deploy the sample application and create the `1221-appdeploy` image is `/samples/1221-appdeploy/container-scripts/app-deploy.py`. This script by default deploys the sample application to all servers in the domain. You can deploy your own applications by modifying the WLST scripts or create a new one with WLST.

To build an application image:

```
$ cd ~/docker-images/OracleWebLogic/samples/1221-appdeploy
$ sudo docker build -t 1221-appdeploy
```

2.6.2 Apache Plugin Web Tier Images

The Apache Plugin provides the ability to load balance traffic to WebLogic Managed servers in a WebLogic cluster. Each Managed server is running in its own Docker container and the Apache Plugin Web tier is also running inside of its own Docker container. In a multi-host environment, the traffic can be routed to any Managed Server container running in the Docker Swarm cluster and networked together by the Docker Overlay Network.

A sample of how to create the Apache Plugin Web tier images from a custom Dockerfile is provided at `/samples/1221-webtier-apache`. The best way to create your own image is to edit the Dockerfile and the `weblogic.conf` file to fit your environment.

The Dockerfile extends the **httpd:2.4** image and installs the Apache Plugin. To build the WebLogic Web tier image:

```
$ cd ~/docker-images/OracleWebLogic/samples/1221-webtier-apache
$ sudo docker build -t webtier
```

2.6.3 Building WebLogic Distributed Domains and Clusters

The following images are required to create Oracle WebLogic distributed domains and clusters across multiple hosts or virtual machines:

- Oracle Linux
- Oracle JDK Image
- WebLogic Install Image
- WebLogic Domain Image
- WebLogic Application Image
- Web tier Image

To distributed WebLogic domains and clusters you must install the Docker Engine in every host where you plan on running containers. Run the images and start the containers from those images. The only requirement for these containers to network together is the Docker Overlay network. The overlay network requires a valid key-value store service. Currently, Docker supports Consul, Etcd, and ZooKeeper (Distributed store), Refer to Docker networking for the steps to enable it.

Alternatively, you can use the Docker tools previously described to create WebLogic distributed domains and clusters. The Docker Machine starts a Virtual Box with the Docker Engine running inside. You can have as many Docker Machines in your environment as you want. To start a machine:

```
$ docker-machine create
```

Every Docker Machine participates in a Docker Swarm cluster and are networked using the Docker Overlay network. The Docker Registry allows you to push images into the registry and then run containers from these images from outside the VM using scripts. Every virtual machine that is part of the Docker Swarm is networked together with the Docker Overlay network. Every container running in the VM can communicate with any other container running in a different VM in the Docker Swarm. This allows you to run the WebLogic servers in many different VMs and

distribute the WebLogic Server domain or cluster across several VMs. To look at the Docker networks, run the following commands:

```
$ sudo docker network ls
$ sudo docker network inspect <overlay network name>
```

Scripts to create a WebLogic Server Domain in a multi-host environment using the tools previously described are available at `/samples/1221-multihost`. The `bootstrap.sh` script starts two Docker Machines: the `weblogic-orchestrator` and the `weblogic-master`. The `weblogic-orchestrator` has the Docker Registry that registers the images that are required to run the containers and has Consul to help start services. The `weblogic-master` has the Docker Swarm, the Overlay Network, and the WebLogic Admin Server container running in the VM. After starting the two Docker machines, the `appdeploy` image is pushed into the registry running in the `weblogic-orchestrator` machine. Lastly, the bootstrap script calls the post-bootstrap script; this script runs an Admin server Docker Container in the `weblogic-master` machine from the `app-deploy` image that has been pushed to the registry.

To start a new Docker machine where the Managed Server runs, call `/samples/1221-multihost/create-machine.sh`. The new Docker Machine is part of the Docker Swarm and the Managed Server containers running in this VM can network using the Overlay network with other containers in the Swarm. After running the `create-machine.sh` script, you can see the Docker Machines running in your environment by invoking the following command:

```
$ sudo docker-machine ls
```

To start Managed Server containers from the `app-deploy` image, call `/samples/1221-multihost/create-container.sh`. If you want the Managed server container to be started in a particular Docker machine, provide the machine name as a parameter (`./create-container.sh <machine name>`); otherwise, the container can start in any of the Docker Machines in the Swarm.

To load balance requests to the Managed Servers in the Oracle WebLogic Cluster, create a Docker container running the Apache Plugin Web tier. This container runs in one of the Docker Machines in the Docker Swarm and is networked to all other containers in the Swarm. The `/samples/1221-multihost/start-webtier.sh` script discovers all Managed Servers running in the Swarm; creates the string `WEBLOGIC_CLUSTER`; pushes the webtier image to the registry; and starts a webtier container on the `weblogic-master` machine. The `WEBLOGIC_CLUSTER` string is set as an environment parameter when starting the Apache Web tier container and is set in the `weblogic.conf` file. Also, note that the Apache Web tier container is bound to port 80 of the `weblogic-master` machine.

Call the sample application using the Apache Web tier. In your browser, use the IP address of the weblogic-master machine and port 80:

```
http://xxx.xxx.xxx:80/sample
```

To obtain the IP address of the `weblogic-master` machine, run the command:

```
$ sudo docker-machine ls
```

You can repeat the steps above to create additional Docker Machines and Managed Servers running in the machines.

2.7 Additional Considerations When Running WebLogic Server Images in Docker Containers

This section addresses some additional considerations when running WebLogic Server images in Docker containers.

2.7.1 How the File System Is Managed in Containers

WebLogic Server configuration files, server logs, file stores, and so on, are all kept in the container file system. When a Docker container is destroyed you will lose your entire file system. There are two alternatives that you can use to avoid losing your file system, described below and illustrated in Figure 2-1:

- Maintain a "data-only" container to store your domain file system. (Container1 and DataCont-1 in the figure.)
- Use the host file system to store the container's local file system (Container2 in the figure.)

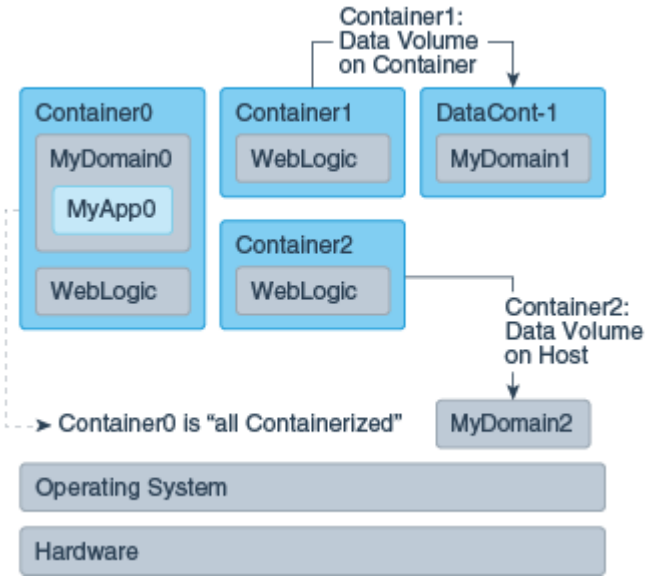
Note:

Container0 in the figure represents a single WebLogic Server that is stateless. Its only function is to deploy applications and resources and does not need to maintain its file system. If it is destroyed, you can just start a new one from the image.

To minimize the dependency on the file system, Oracle recommends:

- Keeping your stores, such as TLog and JMS stores, in the database.
- If you use XA Transactions, use XATransactions without TLog write, because this minimizes the writing to the TLog. See "XA Transactions without Transaction TLog Write" in *Developing JTA Applications for Oracle WebLogic Server*.

Figure 2-1 Managing the File System with Docker Containers



2.7.2 Patching and Upgrading WebLogic Server Images

To patch or upgrade your WebLogic Server images created with the WebLogic Server generic installation image, follow these steps:

1. Upgrade or patch the image by extending the WebLogic Server install Docker image.
2. Use the Docker `cp` (copy) command to copy your domain directory to a destination directory on either the host or a "data-only" container.
3. Remove the container.
4. Run the new container from the extended image (with upgrade/patch).
5. Use the Docker `cp` command to copy your domain directory back to the upgraded container.

2.7.3 Security Concerns Regarding Docker and Linux Containers

The following security concerns have been raised regarding Docker and Linux containers:

- One area of concern is whether it is possible to isolate code running in separate containers from each other. There are no known issues impacting the ability to run WebLogic Server in such an environment at this time.
- Another security concern is the source of the Docker images. You should only obtain Docker images from trusted sources and you need to be aware of the frequency of updates and the nature of the controls on Docker Hub.
- You should stay current with Docker and Linux technology and remain aware of security issues that are raised in each.
- Docker containers default network mode of "Bridge Networking" does not support multicast. Docker containers "Host Networking" supports multicast, but provides less isolation since it uses the host networking stack. Oracle recommends the use of unicast as the WebLogic Server clustering protocol when running in Docker containers.

3

Frequently Asked Questions for Running WebLogic Server Images on Docker

This section provides answers to frequently asked questions about running WebLogic Server 12.2.1 images in Docker containers.

3.1 Is Oracle Weblogic Server 12.2.1 certified on Oracle Linux 6.6/7 and Red Hat Linux 7 Docker images?

Yes WebLogic 12.2.1 is supported and certified on Oracle Linux 6.6/7 and Red Hat Linux 7. For detailed certification information about supported WebLogic Server Docker images, see `http://www.oracle.com/technetwork/middleware/ias/oracleas-supported-virtualization-089265.html`.

3.2 Can I create my own WebLogic Server Docker Images?

Yes, you can use the Dockerfiles and scripts posted on GitHub as examples. For more information, see "Building WebLogic Server Images on Docker".

3.3 Does Oracle post WebLogic Server Docker images on the Docker Hub?

No, but Oracle has posted some Dockerfiles and supporting scripts on GitHub as samples to create WebLogic Server Docker images. For more information, see "About the Dockerfiles and Scripts on GitHub".

3.4 Where is the domain file system stored for WebLogic Server images in a Docker environment?

Containers are *instances* of images. Each gets its own file system on a copy-on-write approach. Docker uses Union Filesystems. Data of a container can be persisted in three ways:

- Container Union Filesystem (default) — Copy-on-write
- Data Volume — Folder mapped to a folder on the host
- Data Volume Container — Folder mapped to a folder on another container

3.5 Is Weblogic Server on Docker supported on multiple hosts?

Yes. Docker containers can be networked together on multi-host operating systems or virtual machines. Each server in the domain and cluster runs in its own Docker container and is capable of communicating as required with other servers.

For more information, see "Clustering WebLogic Server on Docker Containers" and "Running an Oracle WebLogic Server Domain in a Multi Host Environment".

3.6 What is the recommended procedure for patching WebLogic Server on Docker containers?

For information about patching and/or upgrading WebLogic Server on Docker, see "Patching and Upgrading WebLogic Server Images".

3.7 Is there an orchestration layer to support Weblogic Server in a Docker container?

There is not a supported orchestration layer at this time.

3.8 Does Oracle support third-party software running on Docker with WebLogic Server?

Oracle only certifies WebLogic Server 12.2.1 on Docker. See `http://www.oracle.com/technetwork/middleware/ias/oracleas-supported-virtualization-089265.html`.