# Oracle WebLogic Server on Oracle Private Cloud Appliance and Kubernetes

Deploying Oracle WebLogic Server applications on Oracle Private Cloud Appliance and Kubernetes, including migration from Oracle Exalogic Elastic Cloud systems

WHITE PAPER / AUGUST 28, 2019

ORACLE

## PURPOSE STATEMENT

This document describes how to deploy Oracle WebLogic Server applications on Kubernetes on Oracle Private Cloud Appliance or Oracle Private Cloud at Customer, enabling you to run these applications in cloud native infrastructure that is fully supported by Oracle, and that is portable across cloud environments. The document highlights how applications deployed on Oracle Exalogic Elastic Cloud systems can be migrated to this infrastructure without application changes, enabling you to preserve your application investment as you adopt modern cloud native infrastructure.

## DISCLAIMER

TABLE OF CONTENTS

## INTRODUCTION

Oracle WebLogic Server has been certified to run on Docker Containers and Kubernetes. Oracle has developed open source tools to make it easy to migrate Oracle WebLogic Server applications from existing physical or virtual systems to run in Kubernetes. The tools allow you to introspect existing Oracle WebLogic Server domain configurations and applications, create Docker images from them, deploy applications to Kubernetes, monitor them, persist logs, manage application lifecycle, and use automated CI/CD and DevOps processes for ongoing maintenance and application update processes.

Oracle Private Cloud Appliance and Oracle Private Cloud at Customer fully support Oracle Linux Cloud Native Environment, including Oracle Container Runtime for Docker and Oracle Container Services for Use with Kubernetes. They provide an ideal runtime for Oracle WebLogic Server applications to run in Docker and Kubernetes with full, integrated system support from Oracle. We recommend that customers who are running existing Oracle WebLogic Server applications on Oracle Exalogic Elastic Cloud systems, and that wish to adopt cloud native infrastructure and DevOps practices, consider migrating to Oracle Private Cloud Appliance and Oracle Private Cloud at Customer.

This white paper describes how to perform such a migration, including an overview of supporting tooling and recommended practices. The discussion and example refers specifically to a scenario where Oracle Private Cloud Appliance is the target system, but the tools and practices are also applicable to Oracle Private Cloud at Customer.

## MARKET AND TECHNOLOGY TRENDS TOWARDS DOCKER AND KUBERNETES

Industry momentum towards adoption of private and public cloud technology continues to grow. Enterprise customers building cloud applications are leveraging microservices and container-based architectures for application development and deployment, and these architectures have driven adoption of Docker containers and Kubernetes orchestration technology as the de facto standard for cloud native infrastructure. Vendors and open source projects are responding with delivery of software tools and capabilities that support deployment of enterprise applications on Kubernetes.

These trends affect Oracle WebLogic Server customers, who may be running tens, or hundreds, or thousands of applications on more traditional physical or virtualized systems. Many Oracle WebLogic Server customers are running, or are planning to run, their applications in the same cloud native infrastructure that is being built for microservices application architectures. They want to use and leverage the same tools and capabilities across their applications for deploying, monitoring, logging, tracing, and load balancing – technologies such as Helm, Prometheus, Grafana, Elastic Stack, and Ingress. And they want to integrate their Oracle WebLogic Server applications with new microservices.

Oracle has responded to these industry trends by developing tools that make it easy for customers to migrate their Oracle WebLogic Server applications from traditional physical and virtual systems to Kubernetes. Migrated applications can run flexibly in private clouds or in public clouds because Oracle WebLogic Server and the supporting tools are portable.

## DEPLOYING AND MANAGING ORACLE WEBLOGIC SERVER APPLICATIONS ON KUBERNETES

Oracle WebLogic Server 12.2.1.3 has been certified to run in Docker and Kubernetes, and future versions will be certified as well. In the GitHub project for Oracle WebLogic Server Docker containers there are Dockerfiles and scripts that users can use to customize and build Oracle

WebLogic Server Docker images. There are numerous samples that show users how to create images, deploy applications, configure data sources and JMS servers etc. Oracle WebLogic Server Docker images are published in public repositories both in the Oracle Container Registry as well as in the Docker Store. The Oracle WebLogic Server images follow Docker best practices of layering, persisting state outside of the image, and running a single server in the container. The images are supported in both development and production use cases - refer to My Oracle Support Document 2017945.1.

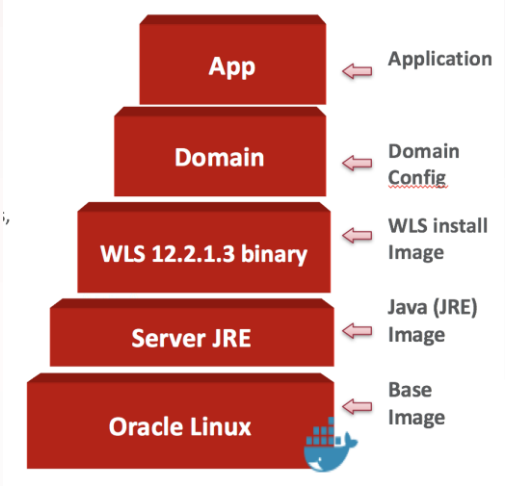


Figure 1. Oracle WebLogic Server Docker image.

To see the certified and supported versions of Docker and Kubernetes please refer to Supported Virtualization page.

To run Oracle WebLogic Server domains in Kubernetes, the open source tools described below are available to manage, migrate, monitor, persist logs, and maintain Oracle WebLogic Server Docker images and applications.

**WebLogic Kubernetes Operator**

A Kubernetes Operator is an application specific controller that extends the Kubernetes API to create, configure, and manage instances of specific application types.

We have adopted the Operator pattern to provide the WebLogic Kubernetes Operator and simplify use of Kubernetes as a container infrastructure for Oracle WebLogic Server instances. The WebLogic Kubernetes Operator extends Kubernetes to create, configure, and manage an Oracle WebLogic Server domain. Find the GitHub project for the WebLogic Kubernetes Operator and documentation that includes a Quick Start guide and samples.

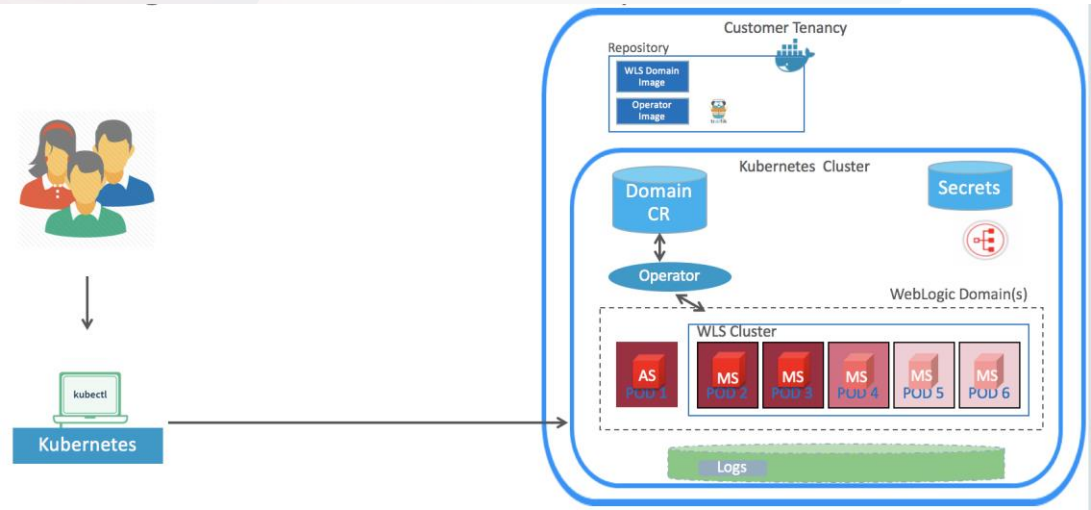


Figure 2. Oracle WebLogic Server on Kubernetes managed by the WebLogic Kubernetes Operator

The WebLogic Kubernetes Operator incorporates knowledge of how to perform lifecycle operations on an Oracle WebLogic Server domain. For example, it knows how to perform graceful shutdowns of WebLogic Server as appropriate when performing operations such as a rolling restart of a domain, in order to deliver service levels expected of Oracle WebLogic Server applications.

Some of the operations performed by the Operator are:

- Provisioning an Oracle WebLogic Server domain.
- Life cycle management
- Updating Docker images
- Scaling and shrinking the Oracle WebLogic Server cluster
- Defining Role-Based Access Control (RBAC) roles
- Creation of Kubernetes services for communication between Kubernetes pods, and for load balancing requests across clustered managed servers.

**WebLogic Deploy Tooling**

WebLogic Deploy Tooling (WDT) makes the automation of Oracle WebLogic Server domain provisioning and application deployment easy. Instead of writing WebLogic Scripting Tool (WLST) scripts that need to be maintained, WDT creates a declarative, metadata model that describes the domain, applications, and the resources used by the applications. This metadata model makes it easy to provision, deploy, and perform domain lifecycle operations in a repeatable fashion, which makes it perfect for Continuous Delivery of applications. The GitHub project for WebLogic Deploy Tooling provides documentation and samples. The blog Make WebLogic Domain Provisioning and Deployment Easy! walks you through a complete sample.

WDT enables you to introspect a domain configuration and application binaries and create Docker images that either contain the domain in the Docker image or reference the domain configuration on a persistent volume. WDT supports introspecting 10.3.6, 12.1.3, 12.2.1.3 domains, and migrating and recreating these domains in Oracle WebLogic Server 12.2.1.3.

When introspecting domains using the WDT Discover Domain Tool, two files are created:

1- A yaml model of the domain configuration.
2- An archive containing the application binaries.

Once the yaml model has been created you can customize and validate your configuration to meet Kubernetes requirements. Invoking the WDT Create Domain Tool takes the domain yaml model and archive and creates a domain home with the application deployed. You can find a sample of creating a Docker image with the domain home and application deployed to it with WDT at the WDT GitHub project.

**WebLogic Image Tool**

The WebLogic Image Tool is an open source tool that allows you to automate building, patching, and updating your Oracle WebLogic Server Docker images, including your own customized images. This tool can be scripted and used in CI/CD processes. Find the WebLogic Image Tool GitHub project at https://github.com/oracle/weblogic-image-tool.

There are four major use cases for this tool:

1. Create a customized Oracle WebLogic Server Docker image where the user can choose:
    a. The OS base image (e.g. Oracle Linux 7.5).
    b. The version of Java (e.g. 8u202).
    c. The version of Oracle WebLogic Server or Oracle Fusion Middleware Infrastructure (FMW Infrastructure) installer (e.g. 12.2.1.3).
    d. A specific Patch Set Update (PSU).
    e. One or more interim or “one-off” patches.
2. Patch a base install image of WebLogic.
3. Patch and build a domain image of Oracle WebLogic Server or FMW Infrastructure using a WebLogic Deploy Tool model.
4. Patch and update a domain configuration, or deploy a new application to an already existing image.

Using the WebLogic Image Tool, you can incorporate the use cases above into an automated process for patching and updating all of your Oracle WebLogic Server applications running in Docker and Kubernetes. Oracle recommends that you apply quarterly Patch Set Updates (PSU) to your Oracle WebLogic Server binaries, latest Java update, and latest OS on an ongoing basis for security reasons. Use of the Image Tool will simplify this update process.

The WebLogic Image Tool leverages an important new capability built into My Oracle Support (MOS) that provides a REST API for specifying and downloading patches. The Image Tool automatically downloads all the one-off patches and PSUs you specify from MOS using this REST API. The tool checks for patch conflicts invoking MOS APIs. You must provide the MOS credentials with the necessary support entitlements, and manually download the Oracle WebLogic Server and Java installers before invoking the tool. Patches and installers are cached to prevent having to download them multiple times.

The WebLogic Image Tool follows Docker best practices to automatically build an image with the recommended image layering, and to perform cleanup to minimize image size. The tool ensures that the image remains patchable and only uses standard and publicly documented Oracle tools and APIs.

A YouTube video demonstrates using the WebLogic Image Tool to create a customized Oracle WebLogic Server install image. To learn more about how to use the Image Tool to automate CI/CD processes when deploying Oracle WebLogic Server domains in Kubernetes, please refer

to our documentation and a demonstration YouTube video using Jenkins.

### WebLogic Monitoring Exporter

The WebLogic Monitoring Exporter exposes Oracle WebLogic Server metrics that can be read and collected by monitoring tools such as Prometheus, and displayed in Grafana dashboards. The WebLogic Monitoring Exporter tool is available in open source here.

Oracle WebLogic Server generates a rich set of metrics and runtime state information that provides detailed performance and diagnostic data about the servers, clusters, applications, and other resources that are running in an Oracle WebLogic Server domain. The WebLogic Monitoring Exporter enables administrators to easily monitor this data using Prometheus and Grafana, tools that are commonly used for monitoring Kubernetes environments.

Figure 3. Oracle WebLogic Server on Kubernetes Grafana Dashboards

With Oracle WebLogic Server runtime metrics being exported to Prometheus, you can auto-scale Oracle WebLogic Server clusters by setting rules in Prometheus based on the metrics. When the rule is met, Prometheus calls onto the WebLogic Kubernetes Operator to scale the Oracle WebLogic Server cluster.

For more information on the design and implementation of the WebLogic Monitoring Exporter, see Exporting Metrics from WebLogic Server. For more information on Kubernetes monitoring using Prometheus and Grafana see Using Prometheus and Grafana to Monitor WebLogic Server on Kubernetes. Try the end-to-end sample that shows you how to monitor your Oracle WebLogic Server domain running in Kubernetes with Prometheus and out of the box Grafana dashboards. The sample also shows you how to set alerting rules in Prometheus. When alert conditions are met, the Prometheus server fires alerts that can send notifications to various receivers such as email, Slack, and webhooks.

### WebLogic Logging Exporter

The WebLogic Logging Exporter exports Oracle WebLogic Server logs directly from the server instances to Elastic Stack. Logs can be analyzed and then displayed in Kibana dashboards. You can find the logging exporter code in its GitHub project along with samples.

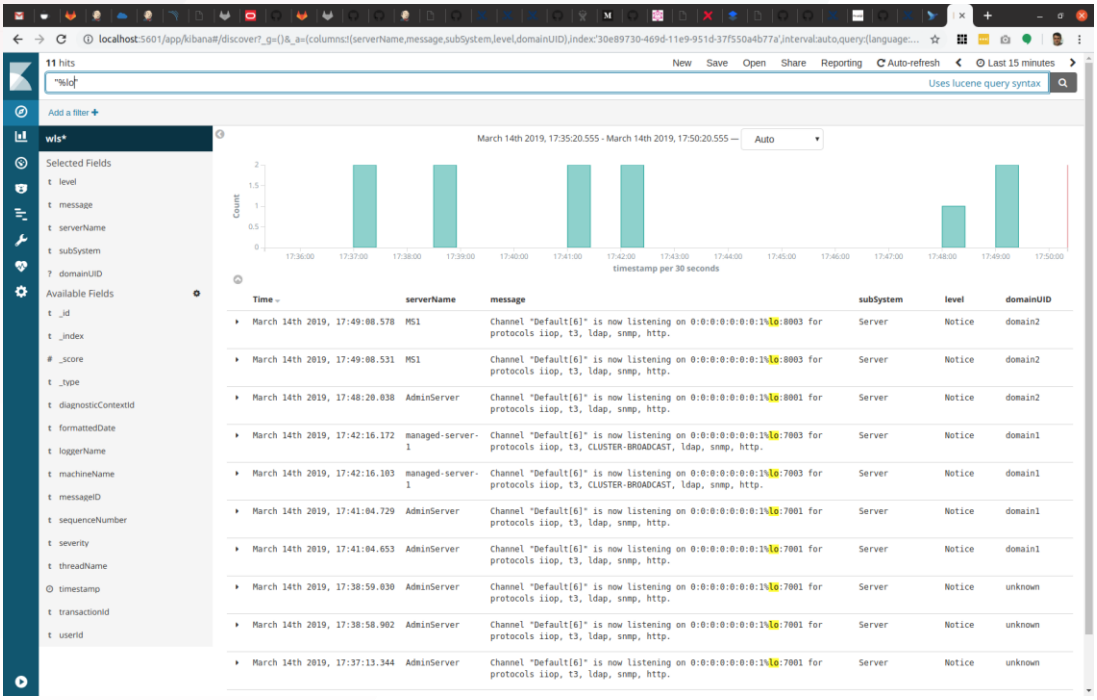


Figure 4. Oracle WebLogic Server logs in Kibana

Taken together, the tools described above simplify the process of deploying Oracle WebLogic Server applications running in Kubernetes clusters. They can enable you to leverage the new capabilities of modern cloud native infrastructure to manage and evolve your applications, and to integrate them with other applications running in the same environment.

## MIGRATING ORACLE WEBLOGIC SERVER DEPLOYMENTS ON ORACLE EXALOGIC ELASTIC CLOUD TO ORACLE PRIVATE CLOUD APPLIANCE

Oracle Private Cloud Appliance is an Oracle Engineered System designed for rapid deployment of private clouds. Compute resources, network hardware, storage providers, operating systems and applications are engineered to work together, and are managed and operated as a single unit. Oracle Private Cloud Appliance is similar to Oracle Exalogic Elastic Cloud systems in this respect.

In addition to the above, Oracle Private Cloud Appliance supports running Kubernetes in an environment fully supported by Oracle. Oracle Linux Cloud Native Environment can be used to create Kubernetes clusters on Oracle Private Cloud Appliance. The white paper Deploy Application Containers on Oracle Private Cloud Appliance/Private Cloud at Customer describes this process. Oracle Private Cloud Appliance running Kubernetes is an attractive migration target for Oracle Elastic Cloud customers running Oracle WebLogic Server workloads who wish to migrate to a container-based Kubernetes platform.

In general applications can be migrated without changes. Customers should consider the following general comparisons between the environments when planning such migrations. More detail is provided in the following sections of this document:

- Oracle WebLogic Server 10.3.6 and 12.1.3 versions are nearing end of life. New features such as updated REST support, JSON processing, auto-scaling and REST management in Oracle WebLogic Server 12.2.1.X enable better integration with cloud systems. Consequently Kubernetes support has focused on 12.2.1.X (and later version) support. Customers using prior versions should migrate to 12.2.1.3 or later as part of the migration process. WDT will generally support migration of such applications and domain configurations "as is", unless the applications are using deprecated APIs. See our Upgrade Documentation for more detail on upgrading Oracle WebLogic Server versions.
- WDT and the WebLogic Image Tool will support the creation of the required Docker images.
- The migration will change the underlying compute infrastructure used by applications. Managed servers in the migrated environment will run in Kubernetes pods, nodes and clusters as defined by the WebLogic Kubernetes Operator Custom Resource.
- Oracle WebLogic Server management tools such as the console and WLST are supported in the migrated environment. However users should consider how native Kubernetes tools might be used for managing their Oracle WebLogic Server applications going forward.
- Although Oracle Traffic Director is supported for migrations to Oracle Public Cloud Appliance, native Kubernetes load balancers such as Traefik and Voyager are more appropriate for Kubernetes, and are recommended as replacements for Oracle Traffic Director.
- Access to external systems via HTTP and T3 protocol is supported, including access to databases, including Oracle RAC clusters running in Oracle Exadata systems. SDP protocols are not supported on Oracle Private Cloud Appliance, so any existing usage of SDP within domains running on Oracle Exalogic Cloud systems must be removed.
- Oracle WebLogic Server, Oracle Coherence, and Oracle Application Development Framework are currently supported for use in Kubernetes with the WebLogic Kubernetes tools. Support for additional Oracle Fusion Middleware products is planned.
- See the licensing documentation for migration of Oracle Exalogic Elastic Cloud Software licenses.

The following sections of this paper describe in detail how to use the WebLogic Kubernetes tools to migrate Oracle WebLogic Server applications running on Oracle Exalogic Elastic Cloud systems to Kubernetes clusters running on Oracle Private Cloud Appliance.

### Prepare the Environment

In this section we will describe how to prepare your environment to deploy your Oracle WebLogic Server applications to the Kubernetes cluster you have created. It is assumed you have already created the Kubernetes cluster on Oracle Private Cloud Appliance per the Oracle white paper referenced above.

The first step is for you to access to the Kubernetes cluster. This is done through *kubectl* commands. *Kubectl* is a command line interface for running commands against Kubernetes clusters. You need to create a *kubeconfig* file so that you can perform *kubectl* commands from your local box. The white paper Deploy application containers on Oracle Private Cloud Appliance/Private Cloud at Customer describes how to create a `kubeconfig` file in the section "Setting up Kubernetes Master Node":

```
$ mkdir -p $HOME/.kubesudo
$ cp -i /etc/kubernetes/admin.conf $HOME/.kube/configsudo
$ chown $(id -u):$(id -g) $HOME/.kube/config
```

To run *kubectl* commands in your local box run:

```
$ export KUBECONFIG=$HOME/.kube/config
```

## INSTALL HELM AND TILLER

Helm is an application package manager running atop Kubernetes. It allows describing the application structure through convenient helm-charts and managing it with simple commands.

The WebLogic Kubernetes Operator project has created a Helm chart to install the Operator in a Kubernetes cluster. For Helm installation and usage information, see Install Helm and Tiller.

The Load Balancer, Prometheus, Grafana, and the Elastic Stack are all installed in the Kubernetes cluster using Helm charts.

## DOWNLOAD FILES FROM THE WEBLOGIC KUBERNETES OPERATOR PROJECT

You need to clone the Operator repository to your local machine so that you have access to the various sample files mentioned throughout this paper. First create a directory for running your commands that contains the necessary files to run Oracle WebLogic Server on Oracle Private Cloud Appliance.

```
$ mkdir –p ~/WLS_K8S_PCA
$ cd ~/WLS_K8S_PCA
$ git clone https://github.com/oracle/weblogic-kubernetes-operator
```

## BUILD THE ORACLE WEBLOGIC SERVER DOCKER IMAGE

The WebLogic Kubernetes Operator supports two Oracle WebLogic Server Docker image models - one where the domain home is stored on a Persistent Volume (PV/PVC), and one where the domain home is stored within the Docker image. In this paper we will deploy an Oracle WebLogic Server 12.2.1.3 domain within the Docker image. The image will be layered, with the Oracle Linux Slim base image, the JDK, the Oracle WebLogic Server binaries, necessary patches applied to run with the Operator, the domain home, and the application. We will use the integration between the WebLogic Image Tool and the WebLogic Deploy Tooling (WDT) to build the image. This integration makes it very simple to create the Oracle WebLogic Server image in one command line.

The first step is to introspect the domain configuration in Oracle Exalogic Elastic Cloud using WDT. The WDT Discover Domain Tool provides a bootstrapping mechanism to create a model and archive file by inspecting an existing domain and gathering configuration and binaries from it. Note that the model file produced by the tool is not directly usable by the WDT Create Domain Tool or the Deploy Applications Tool because the WDT Discover Domain Tool does not discover the passwords from the existing domain. Instead, it puts a `--FIX ME--` placeholder for passwords it finds. Domain users are also not discoverable so the tool does not create the `AdminUserName` and `AdminPassword` fields in the `domainInfo` section, which are needed by the Create Domain Tool. The tool provides a starting point from which the user can edit the generated model and archive file to suit their needs for running one of the other tools.

To run the WDT Discover Domain Tool, provide the domain location on Oracle Exalogic Elastic Cloud and the name of the archive file. For example:

```
$ weblogic-deploy\bin\discoverDomain.cmd -oracle_home c:\wls12213 -
domain_home domains\DemoDomain -archive_file archive.zip -model_file
simple-topology.yaml
```

The `archive.zip` contains the applications you wish to deploy to the domain. If you were to

unzip the `archive.zip` generated by WDT you would find the applications, under a specific directory *`wlsdeploy`*.

For example, the `archive.zip` directory structure for deploying an application called `testwebapp.war` is:

```
Archive:  archive.zip
   creating: META-INF/
  inflating: META-INF/MANIFEST.MF
   creating: wlsdeploy/
   creating: wlsdeploy/applications/
  inflating: wlsdeploy/applications/testwebapp.war

The domain yaml model generated by WDT introspection that describes the
application deployment testwebapp.war looks like:

appDeployments:
    Application:
        'test-webapp':
            SourcePath:
'wlsdeploy/applications/testwebapp.war'
            Target: '@@PROP:CLUSTER_NAME@@'
            ModuleType: war
            StagingMode: nostage
            PlanStagingMode: nostage
```

The value of `@@PROP:CLUSTER_NAME@@` is contained in the `domain.properties` file and stands for the Oracle WebLogic Server cluster where the exporter and the application will be deployed to.

In this paper we will use the *`create`* command of the Image Tool to build the Oracle WebLogic Server image. In preparation to create the image you must follow the following steps:

1- Create directory where you will unzip the Image Tool

```
$ mkdir –p  ~/ImageTool
```

```
$ cd ~/ImageTool
```

2- You need to have your credentials for My Oracle Support so that the Image Tool can have access to the patches.
3- From releases https://github.com/oracle/weblogic-image-tool/releases download the `imagetool-1.1.1.zip`
4- Unzip `imagetool-1.1.1.zip`
5- `cd imagetool-1.1.0/bin`
6- `source setup.sh`
7- You need to set 3 environment variables:

```
export WLSIMG_CACHEDIR="/path/to/cachedit"
export WLSIMG_BLDDIR="/path/to/dir"
export MYPWD="My Oracle Support password"
```

The Oracle WebLogic Server and JDK installers that you downloaded will be copied to the `$WLSIMG_CACHEDIR`. In the build directory the Image Tool puts all the files it uses to build the image.
The Image Tool will automatically download the patches using your My Oracle Support credentials. You can store the password in an environment variable or file, in this particular case it is saved to an environment variable called `MYPWD`.

8- Download JDK and Oracle WebLogic Server installers

9- Copy the Oracle WebLogic Sever and JDK installer to `$WLSIMG_CACHEDIR`.
10- After copying the JDK to the cache directory, invoke `imagetool cache`.

```
$ imagetool cache addInstaller \
--type jdk --version 8u221 –path \  $WLSIMG_CACHEDIR/jdk-8u221-
linux-x64.tar.gz
```

11- After copying the Oracle WebLogic Server installer to the cache directory you need to add it to the cache, invoke `imagetool cache`.

```
$ imagetool cache addInstaller --type wls \
--version 12.2.1.3.0 –path \
$WLSIMG_CACHEDIR/fmw_12.2.1.3.0_wls_quick_Disk1_1of1.zip
```

12- In order to deploy an Oracle WebLogic Server domain to Kubernetes managed by the WebLogic Kubernetes Operator, Oracle WebLogic Server 12.2.1.3 requires 2 patches: 29135930 and 27117282. The Image Tool *`create`* command takes the following options: the version of the JDK, the version of WebLogic, the patches to be applied, the MOS credentials, the WDT domain yaml model, the properties passed into the model, and the archive containing the applications. For an explanation of the options to use with the Image Tool command run `imagetool create –h`.

*Note: For this example, the domain home needs to include the sample domain named* `sample-domain1`*, e.g:* `/u01/oracle/user_projects/domains/sample-domain`.

```
$ imagetool create --tag domain-home-in-image:12.2.1.3 \
--version 12.2.1.3.0 --user / <MOS user>   \
--patches=29135930_12.2.1.3.0,27117282_12.2.1.3.0 \
-- passwordEnv MYPWD --jdkVersion=8u221   \
--wdtArchive=<Directory>/archive.zip  \
--wdtModel=<Directory>/simple-topology.yaml \
--wdtDomainHome=/u01/oracle/user_projects/domains/sample-domain1
\
--wdtVariables=<Directory>/domain.properties \
--wdtVersion=1.1.1
```

## PUSH IMAGES TO WORKER NODES IN ORACLE PRIVATE CLOUD APPLIANCE

After the image has been created by the Image Tool we need to make the image available to each node of the Kubernetes cluster.  The Kubernetes cluster in Oracle Private Cloud Appliance has 2 worker nodes, these can be obtained by running “kubectl get nodes” command.

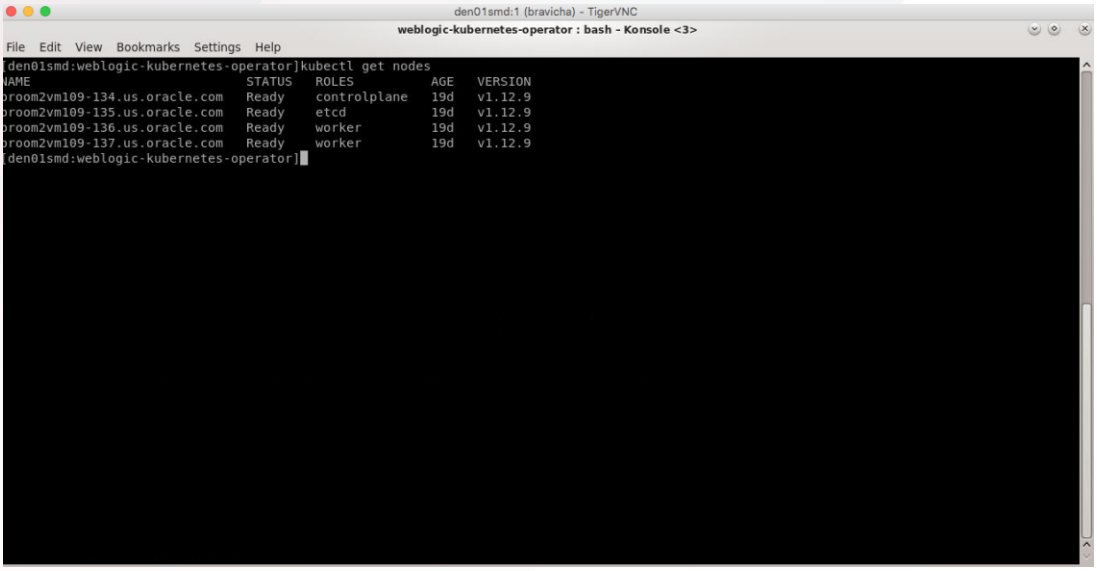

Figure 5. Kubernetes Nodes in the Oracle Private Cloud Appliance machine, 2 worker nodes

Save the image from your local box to a tar file:

```
$ docker save domain-home-in-image:12.2.1.3 -o domain-home-in-image.tar
```

Copy the tar'd image to each VM on Oracle Private Cloud Appliance:

```
$ scp -i ~/.ssh.id_rsa domain-home-in-image.tar root@broom2vm109-
137.us.oracle.com:/tmp/domain-home-in-image.tar

$ scp -i ~/.ssh.id_rsa domain-home-in-image.tar root@broom2vm109-
136.us.oracle.com:/tmp/domain-home-in-image.tar
```

Then ssh into each of the worker nodes and load the image to each node:

```
$ docker load -i /tmp/domain-home-in-image.tar
```

### Deploy Operator

The WebLogic Kubernetes Operator project provides a Helm chart to deploy the Operator. In the section "Prepare your Environment" you cloned the WebLogic Kubernetes Operator GitHub project to your local box. You will need to go to the directory where the project was cloned.

```
$ cd ~/WLS_K8S_PCA/weblogic-kubernetes-operator
```

Create the namespace where the Operator will run:

```
$ kubectl create namespace sample-weblogic-operator-ns
```

Run Helm to install the Operator in its namespace:

```
$ helm install kubernetes/charts/weblogic-operator \
  --name sample-weblogic-operator \
  --namespace sample-weblogic-operator-ns \
  --set image=oracle/weblogic-kubernetes-operator:2.2.1\
  --set serviceAccount=sample-weblogic-operator-sa \
```

```
  --set "domainNamespaces={}" \
  --wait
```

Verify that the Operator is running:

```
$ kubectl get pods -n sample-weblogic-operator-ns
```

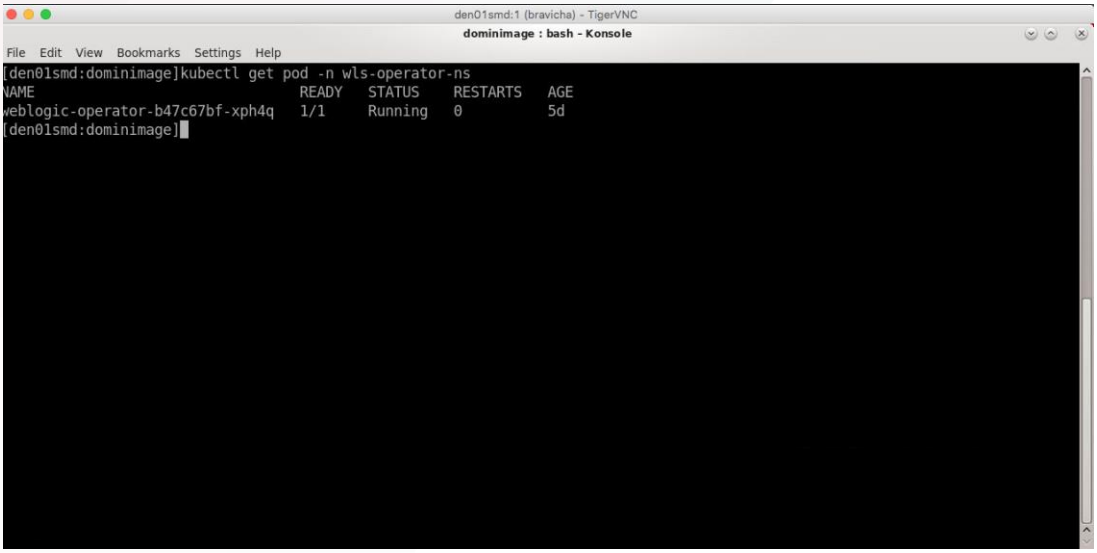


Figure 6. WebLogic Kubernetes Operator running in the Kubernetes cluster.

### Deploy Load Balancer

Load balancers in Kubernetes support Ingress. Ingress is a Kubernetes resource that encapsulates a collection of rules and configuration for routing external HTTP(S) traffic to internal services. We have certified Traefik, Voyager, and Apache with Oracle WebLogic Server on Kubernetes and provide samples that show how to install these load balancers with Helm charts. When running on Kubernetes, Oracle WebLogic Server also supports other load balancers that support Ingress and are supported by default in these Kubernetes clusters.

In Kubernetes use of Oracle Traffic Director (OTD) or Oracle HTTP Server (OHS) load balancers are not required for adaptive load balancing across clusters. The WebLogic Kubernetes Operator creates a cluster service with the end-points of the managed servers running in the Oracle WebLogic Server cluster. When the Oracle WebLogic Server cluster scales and the application is ready to receive traffic, the Operator adjusts the cluster service and adds the new managed server end-point. Ingress starts routing traffic to the newly added managed server. This kind of traffic routing through Kubernetes services is supported only with load balancers, like Traefik and Voyager, which support Ingress.

In this example we will deploy Traefik to the Kubernetes cluster running in Oracle Private Cloud Appliance.

We need to create the namespace of Traefik:

```
$ kubectl create namespace traefik
```

Install Traefik:

```
$ helm install stable/traefik \
  --name traefik-operator \
  --namespace traefik \
```

```
  --values kubernetes/samples/charts/traefik/values.yaml  \
  --set "kubernetes.namespaces={traefik}" \
  --wait
```

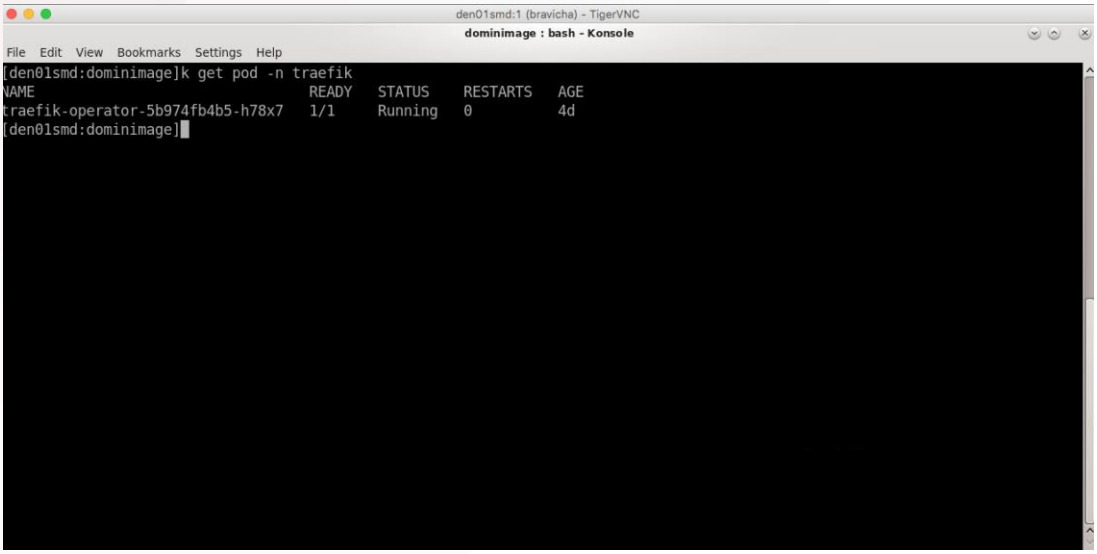


Figure 7. Traefik pod deployed in Kubernetes cluster

**Deploy the Domain**

To deploy the domain we will create a Kubernetes Custom Resource (CR) to represent the domain object in Kubernetes, and to extend the Kubernetes APIs to orchestrate the Oracle WebLogic Server Domain. The domain CR is defined and created by applying the `domain.yaml` file. The WebLogic Kubernetes Operator monitors the domain CR and matches the running state to the definition in the domain CR. For example, if you define in the domain CR that 2 managed servers will run in the cluster, then the Operator will start two pods with two managed servers running in the Oracle WebLogic Server cluster.

Create the namespace where the domain will run:

```
$ kubectl create namespace sample-domain1-ns
```

Configure the operator to manage the domain in the domain's namespace:

```
$ helm upgrade \
  --reuse-values \
  --set "domainNamespaces={sample-domain1-ns}" \
  --wait \
  sample-weblogic-operator \
  kubernetes/charts/weblogic-operator
```

Configure Traefik to manage Ingress created in this namespace:

```
$ helm upgrade \
  --reuse-values \
  --set "kubernetes.namespaces={traefik,sample-domain1-ns}" \
  --wait \
  traefik-operator \
  stable/traefik
```

The admin server credentials are stored in a Kubernetes secret to keep the credentials secure. Create the Kubernetes secret:

```
$ kubernetes/samples/scripts/create-weblogic-domain-credentials/create-
weblogic-credentials.sh \
  -u <username> -p <password> -n sample-domain1-ns -d sample-domain1
```

We will need to modify the `domain.yaml` with correct information for the deployment - for example the image name, namespace of the domain, domain UID, etc. before creating the domain Custom Resource.

Make a copy of the `domain.yaml` to your local directory:

```
$ cp ./kubernetes/samples/scripts/create-weblogic-domain/manually-
create-domain/domain.yaml .
$ vi domain.yaml
```

Replace the following information in the `domain.yaml`

```
kind: Domain
metadata:
  # Update this with the `domainUID` of your domain:
  name: sample-domain1
  # Update this with the namespace your domain will run in:
  namespace: sample-domain1-ns
  labels:
    weblogic.resourceVersion: domain-v2
    # Update this with the `domainUID` of your domain:
    weblogic.domainUID: sample-domain1

spec:
  # This parameter provides the location of the Oracle WebLogic Server
Domain Home (from the container's point of view).
  # Note that this might be in the image itself or in a mounted volume
or network storage.
  domainHome: /u01/oracle/user_projects/domains/sample-domain1

  # If the domain home is inside the Docker image, set this to `true`,
otherwise set `false`:
  domainHomeInImage: true

# Update this with the name of the Docker image that will be used to
run your domain:
  image: "domain-home-in-image:12.2.1.3"

# Identify which Secret contains the Oracle WebLogic Server admin
credentials (note that there is an example of
# how to create that Secret at the end of this file)
webLogicCredentialsSecret:
  # Update this with the name of the secret containing your
  name: sample-domain1-weblogic-credentials

adminServer:
    # serverStartState legal values are "RUNNING" or "ADMIN"
    # "RUNNING" means the listed server will be started up to "RUNNING"
mode
    # "ADMIN" means the listed server will be start up to "ADMIN" mode
```

```
    serverStartState: "RUNNING"
    adminService:
      channels:
       # Update this to set the NodePort to use for the Admin Server's
default channel (where the
       # admin console will be available):
       - channelName: default
         nodePort: 30701
```

In order to access the Administration console, you need to expose a node port for the Admin server. The Operator will create a `NodePort` service with the port defined in the CR. This value can be changed to a port in the range of 30000-32767.

Create the domain CR:

```
$ kubectl apply -f domain.yaml
```

Verify that the domain CR was created:

```
$ kubectl describe domain -n sample-domain1-ns
```

As soon as the Operator sees the domain CR created it will standup the domain according to definitions in the `domain.yaml`. Check that the domain has started

```
$ kubectl get pods -n sample-domain1-ns -o wide
```

You should see an admin server and two managed servers running:

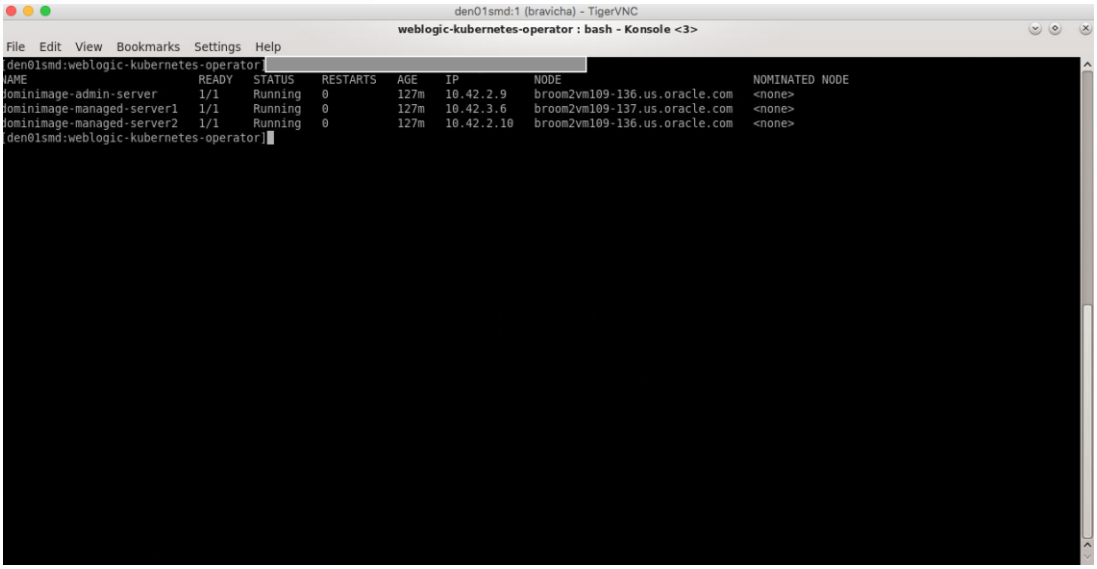

Figure 8. Oracle WebLogic Server domain running on Kubernetes managed by Operator

Create an Ingress for the domain, in the domain namespace, by using the sample Helm chart:

```
$ helm install kubernetes/samples/charts/ingress-per-domain \
  --name sample-domain1-ingress \
  --namespace sample-domain1-ns \
  --set wlsDomain.domainUID=sample-domain1 \
  --set traefik.hostname=sample-domain1.org
```

To invoke the Administration console, go to your browser, use the host and `NodePort` for the Admin Server you defined in the CR, and enter the URL:

```
http:// ${myhost}:<Node Port>/console
```

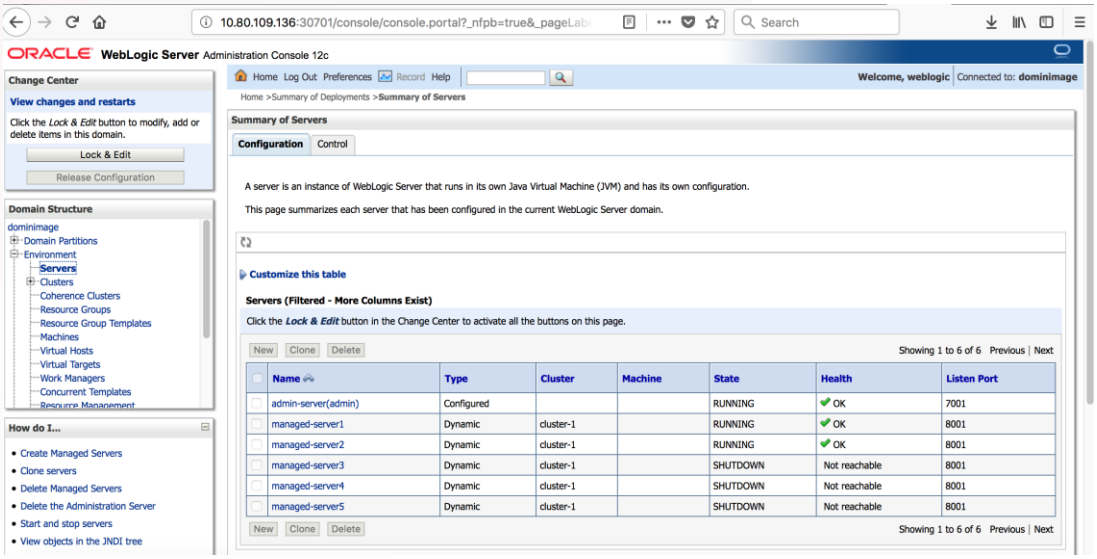

Figure 9. Oracle WebLogic Server Administration Console, server page

## SCALE THE ORACLE WEBLOGIC SERVER CLUSTER

There are three ways you can scale/shrink the Oracle WebLogic Server cluster:

1- Edit the `domain.yaml` and change the number of replicas, for example

```
clusters:
clusterName: cluster-1
serverStartState: "RUNNING"
replicas: 3
```

and then run:

```
$ kubectl apply –f domain.yaml
```

As soon as the Operator sees the change in the CR it will start a new pod with a Managed server running.

2- You can also set rules in WebLogic Diagnostic Framework (WLDF). When the rules are met the Operator is asked to perform a scaling action. The Operator will start a new pod with a Managed server running in it. Please refer to the blog Automatic Scaling of WebLogic Clusters in Kubernetes.

3- The Monitoring Exporter enables exporting Oracle WebLogic Server metrics to Prometheus. In Prometheus you can set rules and when the rules are met, Prometheus calls on the Operator to scale the Oracle WebLogic Server cluster.

## LIFECYCLE MANAGEMENT/OPERATIONS

The WebLogic Kubernetes Operator uses two probes: a Liveness probe and a Readiness probe. The Liveness probe lets the WebLogic Kubernetes Operator know when the Oracle WebLogic Server instance running in the pod is unhealthy. The Operator will restart the pod making sure servers are highly available. The Readiness probe tells the Operator that the application is not only up but ready to receive traffic. When the Readiness probe indicates that the application is "ready" the Operator modifies the Kubernetes cluster service with the end-point of the Managed server and Ingress starts routing traffic to the ready application.

The WebLogic Kubernetes Operator also supports user initiated lifecycle operations like start/stop/restart a server/cluster/domain and initiating a rolling restart of the domain. Please refer to the documentation about lifecycle operations.

These lifecycle operations are performed through the CR. For example, to ask the Operator to shut down the domain edit the `domain.yaml` and set the `serverStartPolicy` to `NEVER`, and run `kubectl –f apply domain.yaml`. When the Operator sees the `serverStartPolicy` change in the CR it will shutdown the domain.

For a rolling restart, setting the `restartVersion` to some `string version`, will make the Operator initiate a rolling restart of the domain. The `string version` helps the Operator know what servers in the domain have already been restarted and which servers still need to be restarted.

```
# serverStartPolicy legal values are "NEVER", "IF_NEEDED", or
"ADMIN_ONLY"
  # This determines which Oracle WebLogic Servers the Operator will
start up when it discovers this Domain
  # - "NEVER" will not start any server in the domain
  # - "ADMIN_ONLY" will start up only the administration server (no
managed servers will be started)
  # - "IF_NEEDED" will start all non-clustered servers, including the
administration server and clustered servers up to the replica count
  serverStartPolicy: "NEVER"
  restartVersion: "RollDomain1"
```

### Monitoring the Oracle WebLogic Server Domain in Kubernetes

Oracle WebLogic Server domains can be monitored in Kubernetes by using the WebLogic Monitoring Exporter to export all metrics in a format that can be read by Prometheus and displayed in Grafana dashboards. We have an end-to-end sample that shows how to deploy Prometheus and Grafana to display the domain metrics.

The WebLogic Monitoring Exporter is a web application that is deployed to the servers of the Oracle WebLogic Server cluster. Download the exporter wls-exporter.war from the GitHub project releases of the Monitoring Exporter. For more details on how to set up and run Prometheus and Grafana with the WebLogic Monitoring Exporter, see the blog, Using Prometheus and Grafana to Monitor WebLogic Server on Kubernetes.

You need to deploy the `wls-exporter` web application to the domain created in the image `domain-home-in-image:12.2.1.3`. We will use the Image Tool and integration with WDT to deploy the exporter.

Create an archive that contains the exporter and create a model yaml with the application deployment configuration. The `wls-exporter.war` needs to be under the archive directory

format:

```
wlsdeploy/applications
```

For example, the archive.zip directory structure for deploying wls-exporter.war is:

```
Archive:  archive.zip
   creating: META-INF/
  inflating: META-INF/MANIFEST.MF
   creating: wlsdeploy/
   creating: wlsdeploy/applications/
  inflating: wlsdeploy/applications/wls-exporter.war
```

Create a metadata model that describes the exporter to be deployed. For this example, the metadata model file is called `simple-topology.yaml` that specifies the `wls-exporter.war`:

```
appDeployments:
    Application:
        'wls-exporter':
            SourcePath: 'wlsdeploy/applications/wls-exporter.war'
            Target: '@@PROP:CLUSTER_NAME@@'
            ModuleType: war
            StagingMode: nostage
            PlanStagingMode: nostage
```

The value of `@@PROP:CLUSTER_NAME@@` is contained in the `domain.properties` file and references the Oracle WebLogic Server cluster where the exporter and the application will be deployed.

The Image Tool update command takes the following options: the name of the image to be created, what image will be updated, the WDT domain yaml model created above, the properties passed into the model, the archive containing the applications, and the WDT operation of update. For an explanation of the options to use with the `imagetool` command run:

```
imagetool update –h.
```

*Note: In this example, the domain home must be:*
`/u01/oracle/user_projects/domains/sample-domain1`

```
$ imagetool update --tag domain-home-in-image-monitor:12.2.1.3 \
--fromImage=domain-home-in-image:12.2.1.3 \
--wdtArchive=<Directory>/archive.zip \
--wdtModel=<Directory>/simple-topology.yaml \
--wdtDomainHome=/u01/oracle/user_projects/domains/sample-domain1 \
--wdtVariables=<Directory>/domain.properties \
--wdtOperation=DEPLOY --wdtVersion=1.1.1
```

We need to push the updated image to each node of the Kubernetes cluster in Oracle Private Cloud Appliance.

Save the image from your local box to a tar file:

```
$ docker save domain-home-in-image-monitor:12.2.1.3 -o domain-home-in-
image-monitor.tar
```

Copy the tar'd image to each VM on Oracle Private Cloud Appliance:

```
$ scp -i ~/.ssh.id_rsa domain-home-in-image-
monitor.tar root@broom2vm109-137.us.oracle.com:/tmp/domain-home-in-
image.tar

$ scp -i ~/.ssh.id_rsa domain-home-in-image-
monitor.tar root@broom2vm109-136.us.oracle.com:/tmp/domain-home-in-
image.tar
```

Then ssh into each of the worker nodes and load the image to each node:

```
$ docker load -i /tmp/domain-home-in-image-monitor.tar
```

Edit the `domain.yaml` to add the new updated image:

```
# Update this with the name of the Docker image that will be used to
run your domain:
  image: "domain-home-in-image-monitor:12.2.1.3"
```

Apply the `domain.yaml` to update the CR:

```
$ kubectl apply -f domain.yaml
```

When the Operator sees the change in the CR with the new updated image name it will roll the Oracle WebLogic Server domain starting new pods based on the new image. Now that the exporter has been deployed the next step is to deploy Prometheus and Grafana.

## DEPLOY PROMETHEUS

Create a namespace called “monitoring” where both Prometheus and Grafana pods and services will run.

```
$ kubectl create namespace monitoring
```

Prometheus is installed using a Helm chart, follow the instructions to deploy Prometheus in the sample Setting up Prometheus.

Updating Prometheus configuration to scrape new Oracle WebLogic Server metrics can be done dynamically. When installing with the Prometheus Helm chart, there are two containers running in the Prometheus server pod: one is for the Prometheus server, the other is to detect changes in the ConfigMap. To change, apply the latest configuration to the ConfigMap of the Prometheus server. The sample describes how to update Prometheus to scrape new metrics.

## DEPLOY GRAFANA

Grafana is installed using a Helm chart. Follow the instructions to deploy Grafana in the sample Setting up Grafana.

In the out of the box Grafana dashboards you can drill down to monitoring information about your Data Sources. You can track information such current capacity, active number of connections, total connections, and see them in graphs.

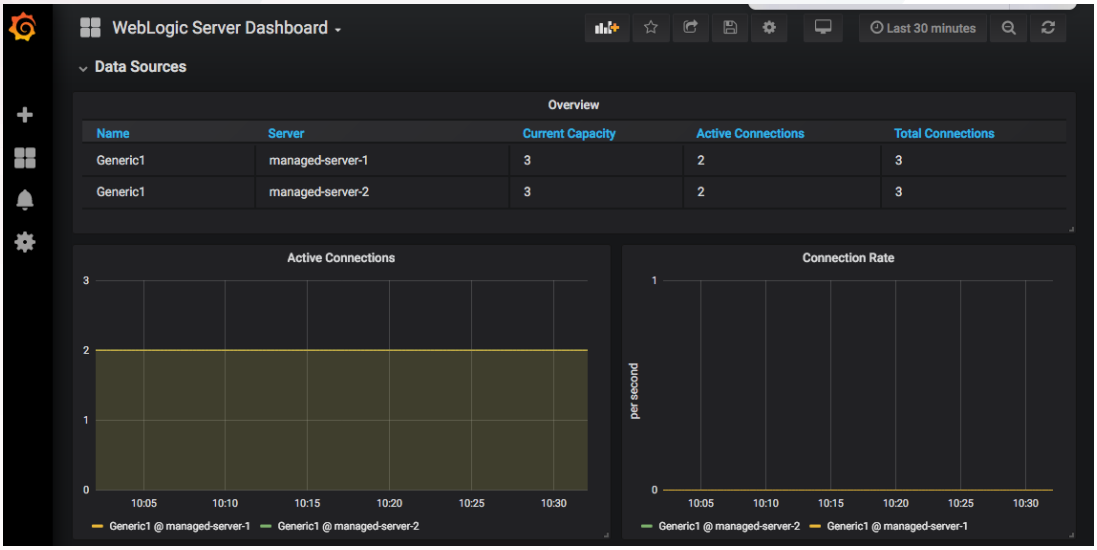

Figure 10. Grafana dashboards for Oracle WebLogic Server data sources

## PROMETHEUS ALERT MANAGER

Prometheus can be configured to set alerting rules. When alert conditions are met, the Prometheus server fires alerts that can send notifications to various receivers like email, Slack, webhook. It can also notify the WebLogic Kubernetes Operator to scale the Oracle WebLogic Server cluster. A sample demonstrates how to deploy and configure the alert manager in Prometheus and Setting up a Webhook or Firing Alerts.

### Oracle WebLogic Server Kubernetes Logging

Oracle WebLogic Server offers different options for persisting server logs. The server logs can be persisted to a Persistent Volume on Kubernetes, they can be sent to the Elastic Stack using Logstash or FluentD, or the WebLogic Logging Exporter can send the server logs directly to the Elastic Stack.

The core of Elastic Stack is three open source products: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.

The WebLogic Logging Exporter adds a log event handler to Oracle WebLogic Server, such that Oracle WebLogic Server logs can be integrated into Elastic Stack in Kubernetes directly, by using the Elasticsearch REST API. The WebLogic Logging Exporter has its own GitHub Project that includes step-by-step instructions on how to install the WebLogic Logging exporter.

### Summary of Migration Outcomes

After completion of the above migration process, you will have laid the foundation for running Oracle WebLogic Server applications on Kubernetes on Oracle Private Cloud Appliance, including:

- Monitoring applications in production
- Lifecycle management of applications

- Scaling configurations as required to meet workload demands
- Analyzing production issues
- Broader adoption of CI/CD and DevOps for application development and management
- Enabling ongoing updates to production applications
- Enabling ongoing patching of production systems
- Migration and deployment of additional applications to the environment
- Migration and deployment of Oracle Coherence and Oracle Application Development Framework applications
- Development and deployment of new applications
- Future deployment of Oracle Fusion Middleware product applications

You can continue to leverage your existing application investment with the new capabilities provided in Kubernetes, all fully supported by Oracle.

In addition, you will have migrated to a cloud native infrastructure that has emerged as a standard platform for cloud deployments, giving you greater application portability and integration across clouds, including public clouds such as Oracle Cloud Infrastructure. You will be positioned for adoption of new microservices technologies for application development. For example, you may consider usage of Helidon, an open source project sponsored and supported by Oracle, that provides a set of Java libraries for microservices, for extension of existing applications, or development of net new applications, with support for integration with Oracle WebLogic Server and Coherence. Oracle is also working on tooling that will support integrated management of hybrid applications across these and additional technologies, and that will also support management across hybrid private/public cloud deployments. You will have many options for evolving your application portfolio to meet your business needs.

## CONCLUSION

Oracle provides Oracle Private Cloud Appliance or Oracle Private Cloud at Customer for your private cloud deployments, and provides tools for creating and managing Kubernetes clusters on these systems. Oracle also provides tools that will enable you to migrate existing Oracle WebLogic Server applications, including applications on Exalogic Elastic Cloud systems, to these private cloud systems on open cloud native infrastructure that is supported by Oracle. By doing so, you can preserve your investment in Oracle WebLogic Server applications, leverage new Kubernetes and technologies to enhance these applications, and position your organization for evolving applications as business requirements dictate.

ORACLE CORPORATION

**Worldwide Headquarters**
500 Oracle Parkway, Redwood Shores, CA 94065 USA

**Worldwide Inquiries**
TELE + 1.650.506.7000 + 1.800.ORACLE1
FAX + 1.650.506.7200
oracle.com

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com. Outside North America, find your local office at oracle.com/contact.

blogs.oracle.com/oracle facebook.com/oracle twitter.com/oracle

Integrated Cloud Applications & Platform Services



White Paper Oracle WebLogic Server on Oracle Private Cloud Appliance **and Kubernetes**
August 2019
Author: Monica Ricelli
Contributing Authors: [OPTIONAL]

Oracle is committed to developing practices and products that help protect the environment

ORACLE