Oracle® Communications

Cloud Deployment Guide

Release 8.1

F82335-02

March 2024

Oracle Communications Cloud Deployment Guide, Release 8.1

F82335-02

Contents

Revision History

1 Cloud Deployment

2 Deploy with the Container Engine for Kubernetes

Revision History

Table 1 Revision History

Date	Description
Oct 2023	- Initial release
Mar 2024	- Updated download instructions and added link for occas81.zip.

1 Cloud Deployment

The cloud native Oracle Communications Converged Application Server is deployed on a Kubernetes cluster using container images from the Oracle Container Registry. Use these images to deploy the Converged Application Server Admin and Managed servers as Kubernetes Pods.

The high-level steps to deploy the Converged Application Server include:

1. Install the dependencies.
2. Install Podman.
3. Log in to the Oracle Container Registry.
4. Pull the Converged Application Server image from the Oracle Container Registry.
5. Install Kubernetes.
6. Install the WebLogic Kubernetes Operator.
7. Create a Persistent Volume
8. Deploy the domain.

Optional steps include:

- Bring up another managed server.
- Install the EFK stack (ElasticSearch, FluentD, Kibana).
- Install Prometheus and Grafana
- Apply a patch

Install Dependencies

The `java-devel` and `iproute-tc` packages are required dependencies for the Converged Application Server. The `git` package is required to clone the Kubernetes WebLogic Operator.

1. Install the required dependencies.

   ```
   sudo dnf install podman git java-devel iproute-tc
   ```

2. Set the JAVA_HOME variable.

   For example:

   ```
   export JAVA_HOME=/etc/alternatives/jre
   ```

   Append this line to your ~/.bashrc file.

Log in to Oracle Container Registry

Before deploying the Converged Application Server, you must log in to the Oracle Container Registry and accept the Java License Agreement for the JDK and the Middleware License Agreement for the Oracle Communications Converged Application Server.

Oracle provides access to the Oracle Container Registry for customers that have a Single Sign-On account at Oracle. The Oracle Container Registry contains Docker images for licensed commercial Oracle software products that you may use in your enterprise. Images may also be used for development and testing purposes. The license covers both production and non-production use. The Oracle Container Registry provides a web interface where customers are able to select Oracle Docker images and agree to terms of use before pulling the images using the standard Docker client software.

If necessary, create an account.

1. Navigate to the Oracle Container Registry.
2. In the top-right corner, click **Sign In**.
3. Enter your Oracle credentials and click **Submit**.
4. Click **Middleware** and then click **occas**.
5. Select your language and click **Continue**.
6. Scroll to the end of the License Agreement and click **Accept**.
7. From the server where you are installing the Converged Application Server, log in to the Oracle Container Registry.

   ```
   podman login container-registry.oracle.com
   ```

   > **Note:**
   >
   > If you are behind a proxy, follow these steps to set up the http_proxy and https_proxy variables for your docker client.

Download Domain Creation Files

In addition to downloading the `occas_generic.jar` file, you need to download the domain creation files.

1. Sign in to My Oracle Support.
2. Search for "Docker Script for OCCAS 8.1 Installation" (Doc ID 3007597.1).
3. Under the Details section, click the link to download occas81.zip.
4. Unzip the file.

   ```
   unzip occas81.zip
   ```

This creates the `occas81/` directory.

Choose the Image

The Oracle Communications Converged Application Server image is located on the Oracle Container Registry.

1. If using JDK 8, use the JDK 8 container.

   ```
   container-registry.oracle.com/middleware/occas:8.1.0.0.0-generic-8
   ```

2. If using JDK 11, use the JDK 11 container.

   ```
   container-registry.oracle.com/middleware/occas:8.1.0.0.0-generic-11
   ```

Install Kubernetes

Kubernetes orchestrates the Converged Application Server containers, deploying and scaling them as necessary. Because Kubernetes is complex with many customer-specific requirements, you should consider either using a hosted solution or reading the Kubernetes documentation in full to find the right solution for your needs. To use kubeadm, see Installing kubeadm for prerequisites, installation steps, and troubleshooting.

1. After Kubernetes is installed, run `kubeadm init` to start Kubernetes.
2. Follow the instructions returned by the kubeadm command.

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

3. Deploy a Pod network to the cluster.

   Converged Application Server supports Calico as the Pod network.

Install the Kubernetes Weblogic Operator

The WebLogic Kubernetes Operator supports running your WebLogic Server domains on Kubernetes, an industry standard, cloud neutral deployment platform. It lets you encapsulate your entire WebLogic Server installation and layered applications into a portable set of cloud neutral images and simple resource description files. You can run them on any on-premises or public cloud that supports Kubernetes where you've deployed the operator.

Kubernetes must be installed before following the procedure below. If you have not yet installed Kubernetes, see the Kubernetes documentation.

1. Install Helm, a package manager for Kubernetes.
   a. Download Helm.

      ```
      curl -O https://get.helm.sh/helm-v3.6.3-linux-amd64.tar.gz
      ```

      See the Releases page for other Linux architectures.

   b. Extract Helm.

      ```
      tar xf helm-v3.6.3-linux-amd64.tar.gz
      ```

   c. Move the Helm binary into your PATH.

      ```
      sudo install linux-amd64/helm /usr/local/bin/helm
      ```

2. Install the Oracle Weblogic Kubernetes Operator.

   The Oracle Weblogic Kubernetes Operator is used to configure the lifecycle of the Converged Application Server images. See the Oracle Weblogic Kubernetes Operator documentation.

   a. Clone the Weblogic Kubernetes Operator git repository.

      ```
      git clone --branch v4.0.5 https://github.com/oracle/weblogic-
      kubernetes-operator
      ```

   b. Navigate to the `weblogic-kubernetes-operator` directory.

      ```
      cd weblogic-kubernetes-operator
      ```

      > **Note:**
      >
      > Subsequent commands will assume your current directory is `weblogic-kubernetes-operator`.

3. Pull the Docker image for the WebLogic Kubernetes Operator and copy that image to all the nodes in your cluster.

   ```
   podman pull ghcr.io/oracle/weblogic-kubernetes-operator:4.0.5
   ```

4. Create the namespace for the Weblogic Kubernetes Operator.

   The syntax for the command is: `kubectl create namespace <unique name>`. For example:

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

5. Create the service account within that namespace for the WebLogic Kubernetes Operator.

   ```
   kubectl create serviceaccount -n sample-weblogic-operator-ns sample-
   weblogic-operator-sa
   ```

6. Update the following parameters in the file `weblogic-kubernetes-operator/kubernetes/charts/weblogic-operator/values.yaml`.

   If you plan on installing the EFK stack, set `elkIntegrationEnabled` to true; otherwise, leave `elkIntegrationEnabled` as false.

- `serviceAccount`—Set this to the previously created service account for the WebLogic Kubernetes Operator.

  ```
  serviceAccount: "sample-weblogic-operator-sa"
  ```

- `domainNamespaces`—Set this to the previously created namespace for the WebLogic Kubernetes Operator.

  ```
  domainNamespaces:
  - "sample-weblogic-operator-ns"
  ```

- `elkIntegrationEnabled`—Set this to true if you plan on installing the EFK stack; leave as false if you do not plan on installing the EFK stack.

  ```
  elkIntegrationEnabled: true
  ```

- `elasticSearchHost`—Change the word 'default' in 'elasticsearch.**default**.svc.cluster.local' to the namespace previously created for the WebLogic Kubernetes Operator.

  ```
  elasticSearchHost: "elasticsearch.sample-weblogic-operator-
  ns.svc.cluster.local"
  ```

7. Paste the following role-based access control (RBAC) definition into a YAML file called `rbac.yaml`.

   See Using RBAC Authorization.

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRoleBinding
   metadata:
     name: helm-user-cluster-admin-role
   roleRef:
     apiGroup: rbac.authorization.k8s.io
     kind: ClusterRole
     name: cluster-admin
   subjects:
   - kind: ServiceAccount
     name: default
     namespace: kube-system
   ```

8. Apply the RBAC file.

   ```
   kubectl apply -f rbac.yaml
   ```

9. From the `weblogic-kubernetes-operator` directory, start the operator.

   ```
   helm install sample-weblogic-operator kubernetes/charts/weblogic-operator
   \
     --namespace sample-weblogic-operator-ns \
     --set image=ghcr.io/oracle/weblogic-kubernetes-operator:4.0.5 \
     --set serviceAccount=sample-weblogic-operator-sa \
     --set "enableClusterRoleBinding=true"\
     --set "domainNamespaceSelectionStrategy=LabelSelector"\
   ```

```
    --set "domainNamespaceLabelSelector=weblogic-operator\=enabled"\
    --wait
```

You can verify the pod is running with the following command:

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

For example:

> **Note:**
>
> If you are installing the EFK stack and set `elkIntegrationEnabled` to true, then the READY column will display 2/2 when both containers are ready. If you are not installing the EFK stack and left `elkIntegrationEnabled` as false, then the READY column will display 1/1 when the container is ready.

```
[weblogic-kubernetes-operator]$ kubectl get pods -n sample-weblogic-
operator-ns
NAME                                 READY   STATUS    RESTARTS   AGE
weblogic-operator-7885684685-4jz5l   2/2     Running   0          2m32s
[weblogic-kubernetes-operator]$
```

Create the Persistent Volume

Converged Application Server uses a Persistent Volume (PV) to store logs that can be viewed with Kibana. The PV may be either a local path on the master node or a NFS location.

1. Create and label a namespace that can host one or more domains.

   ```
   kubectl create namespace sample-domain1-ns
   kubectl label ns sample-domain1-ns weblogic-operator=enabled
   ```

2. On the server where you'll store the log files, create the PV directory with open permissions.

   ```
   mkdir /u01/pv
   chmod 777 /u01/pv
   ```

   > **Note:**
   >
   > See Persistent Storage for details about permissions.

3. On the master node, navigate to the `occas81/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv` directory.

4. Open the `sample-domain1-weblogic-sample-pv.yaml` file.

5. Set the `storage` parameter to the number of gigabytes to allocate to the persistent volume.

   ```
   storage: 50Gi
   ```

6. Update the `hostPath` parameter.

   - If using a local PV, set the `path` parameter to the local path.

     ```
     hostPath:
       path: "/u01/pv"
     ```

   - If using a remote PV, uncomment the `nfs` parameter and set the `server` and `path` parameters to the NFS server where the PV is located.

     ```
     nfs:
       server: 10.0.0.1
       path: "/mnt/share/pv"
     ```

7. Use the `kubectl create` command to create the persistent volume and the persistent volume claim.

   ```
   kubectl create -f sample-domain1-weblogic-sample-pv.yaml
   ```

   ```
   kubectl create -f sample-domain1-weblogic-sample-pvc.yaml
   ```

8. Use the `kubectl get` command to verify the persistent volume and persistent volume claim was created.

   ```
   [doc@docker:occas-persistant-volume]$ kubectl get pv -n sample-domain1-ns
   NAME                                CAPACITY   ACCESS MODES   RECLAIM
   POLICY   STATUS   CLAIM
   STORAGECLASS                                   REASON   AGE
   sample-domain1-weblogic-sample-pv   50Gi       RWX
   Retain           Bound    sample-domain1-ns/sample-domain1-weblogic-
   sample-pvc   sample-domain1-weblogic-sample-storage-class            23m
   [doc@docker:occas-persistant-volume]$ kubectl get pvc -n sample-domain1-ns
   NAME                                 STATUS
   VOLUME                              CAPACITY   ACCESS MODES
   STORAGECLASS                                   AGE
   sample-domain1-weblogic-sample-pvc   Bound    sample-domain1-weblogic-
   sample-pv   50Gi       RWX            sample-domain1-weblogic-sample-
   storage-class   8m11s
   [doc@docker:occas-persistant-volume]$
   ```

Deploy the Replicated Domain

The Converged Application Server replicated domain is a logically related group of resources. Domains include the administration server and all managed servers. Converged Application Server domains extend Oracle WebLogic Server domains to support SIP and other telecommunication protocols.

See Understanding Oracle WebLogic Server Domains.

1. Navigate to the `occas8/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv` directory.
2. Run the `create-weblogic-credentials.sh` script to create the Secret for Weblogic.

   Kubernetes Secrets contain sensitive information like passwords. See Secrets.

   ```
   ./create-weblogic-credentials.sh -u weblogic -p <password> -n
   sample-domain1-ns -d sample-domain1 -s sample-domain1-weblogic-
   credentials
   ```

3. Navigate to the `domain-home-on-pv` folder.

   ```
   cd ~/occas8/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv/domain-home-
   on-pv
   ```

4. Open the `create-domain-inputs.yaml` file.
5. Set the following parameters.
   - domainHome—If you set the `path` parameter of the persistent volume to `/u01/pv`, then set the `domainHome` parameter to `/u01/pv/domains/sample-domain1`.
   - logHome—If you set the `path` parameter of the persistent volume to `/u01/pv`, then set the `logHome` parameter to `/u01/pv/logs/sample-domain1`.
   - domainPVMountPath—The `domainPVMountPath` parameter should match the `path` parameter of the persistent volume.
6. Use the `create-domain.sh` script to create the domain.

   The command requires two arguments: the input YAML file and the output directory.

   ```
   ./create-domain.sh \
       -i create-domain-inputs.yaml \
       -o ~/occas8/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv/
   domain-home-on-pv
   ```

7. Navigvate to the sample-domain1 folder.

   ```
   cd weblogic-domains/sample-domain1/
   ```

8. Deploy the WebLogic domain in the Kubernetes cluster.

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

   The domain will now be managed by the Weblogic Operator.
9. Open a browser and navigate to http://<IP address>:30701/console to visit the WebLogic Server's management interface.

For information on building a custom application router, see Configuring a Custom Application Router in the *Developer Guide*.

Modify the Domain File

For quick, on-demand changes to your domain, modify the domain.yaml file directly and use kubectl to apply your changes. By modifying the domain file you can scale up or down the number of managed servers, change JMV options, or update the mountPath for your PV.

Follow the steps below to manually add a second managed server.

1. Navigate to the sample-domain1 directory.

   ```
   cd ~/occas81/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv/domain-home-on-
   pv/weblogic-domains/sample-domain1
   ```

2. Open the file `domain.yaml`.
3. Set the replicatas attribute to the number of managed servers you want.

   For example:

   ```
   replicas: 2
   ```

4. Reapply the domain.yaml file.

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

Monitoring Tools

Converged Application Server can integrate with several third-party monitoring tools.

- The EFK stack—The EFK stack is the combination of Elasticsearch, FluentD, and Kibana working together to centralize log files for easy monitoring. FluentD gathers logs files from multiple nodes and sends them to Elasticsearch, which stores and indexes them, while Kibana provides the front-end web interface.
- Prometheus and Grafana—Converged Application Server can use the WebLogic Monitoring Exporter to export Prometheus-compatible metrics that are then exposed to Grafana, which visualizes the metrics over time to provide futher insights into the data.

Deploy the EFK Stack

Customers may deploy the EFK stack to increase visibility into the state of their Converged Application Server instances. Once the EFK stack is deployed, the Kibana web interface displays multiple Converged Application Server metrics.

> **Note:**
>
> Converged Application Server does not require the EFK stack and its installation is optional.

> **Note:**
>
> The FluentD pod runs in the sample-domain1-ns namespace and the ElasticSearch and Kibana pods run in the sample-weblogic-operator-ns namespace.

1. Download the Docker images for FluentD, busybox, ElasticSearch, and Kibana.

   ```
   podman pull fluent/fluentd-kubernetes-daemonset:v1.16-debian-
   elasticsearch8-1
   podman pull busybox
   podman pull logstash:8.7.1
   podman pull elasticsearch:8.7.1
   podman pull kibana:8.7.1
   ```

2. Navigate to the `occas81/dockerfiles/8.1.0.0.0/occas-efk-pv/FluentD/` directory.
3. Open the file `configmap.yml` and confirm your domain and namespace settings.
   - `weblogic.domainUID: sample-domain1`
   - `namespace: sample-domain1-ns`

   These two parameters identify the domain and namespace where the EFK stack will be deployed. They must match the `domainUID` and `namespace` parameters from the `create-domain-inputs.yaml` file that you used to create the domain.
4. Apply the FluentD ConfigMap.

   ```
   kubectl apply -f configmap.yml
   ```

5. Open the file FluentD domain file `fluentd.yml` and set the following values.
   - `weblogic.domainUID`—Confirm this is the same value in the ConfigMap.
   - `namespace`—Confirm this is the same value in the ConfigMap.
   - `LOG_PATH`—Set the beginning of the `value` parameter to your persistent volume. This value should match the `logHome` parameter in the create-domain-inputs.yaml file.

     ```
       - name:
     LOG_PATH


         value: /u01/pv/logs/sample-domain1
     ```

   - `ADMIN_LOG_PATH`—Set your admin log path.

     ```
       - name: ADMIN_LOG_PATH
         value: /u01/pv/logs/sample-domain1/servers/admin-server/logs
     ```

- `MS_LOG_PATH`—Set your managed server log path.

  ```
    - name: MS_LOG_PATH
      value: /u01/pv/logs/sample-domain1/servers/managed-server*/logs
  ```

- `SIP_MESSAGES_LOG_PATH`—Set the beginning of the `value` parameter to your persistent volume.

  ```
    - name: 
  SIP_MESSAGES_LOG_PATH


      value: /u01/pv/domains/sample-domain1/servers/managed-server*/
  logs/sip-messages*.log*
  ```

- `claimName`—Confirm this matches your persistent volume claim

  ```
  claimName: sample-domain1-weblogic-sample-pvc
  ```

- `mountPath`—Set the mount path for `occas-domain-storage-volume` to your persistent volume.

  > **Note:**
  >
  > There is more than one `mountPath` parameter. Make sure you edit the mount path for `occas-domain-storage-volume`.

  ```
   - mountPath: /u01/pv
     name: occas-domain-storage-volume
  ```

6. Base64 encode the values of elasticSearchHost and elasticSearchPort from the file `weblogic-kubernetes-operator/kubernetes/charts/weblogic-operator/values.yaml`.

   ```
   [k8smain:~]$ grep ^elasticSearch[HP] ~/weblogic-kubernetes-operator/
   kubernetes/charts/weblogic-operator/values.yaml
   elasticSearchHost: "elasticsearch.sample-weblogic-operator-
   ns.svc.cluster.local"
   elasticSearchPort: 9200
   [k8smain:~]$ echo -n elasticsearch.sample-weblogic-operator-
   ns.svc.cluster.local | base64
   ZWxhc3RpY3NlYXJjaC5zYW1wbGUtd2VibG9naWMtb3BlcmF0b3ItbnMuc3ZjLmNsdXN0ZXIubG
   9j
   YWw=
   [k8smain:~]$ echo -n 9200 | base64
   OTIwMA==
   [k8smain:~]$
   ```

7. Add those base64 encoded values to the `sample-domain1-weblogic-credentials` Kubernetes secret.

   ```
   kubectl edit secret sample-domain1-weblogic-credentials -n sample-
   domain1-ns
   ```

   The top of the resulting YAML:

   ```
   apiVersion: v1
   data:
     password: VGhpcyBpcyBub3QgcmVhbGx5IHRoZSBwYXNzd29yZC4K
     username: d2VibG9naWM=
     elasticsearchhost:
   ZWxhc3RpY3NlYXJjaC5zYW1wbGUtd2VibG9naWMtb3BlcmF0b3ItbnMuc3ZjLmNsdXN0
   ZXIubG9jYWw=
     elasticsearchport: OTIwMA==
   kind: Secret
   . . .
   ```

8. Navigate to the `weblogic-kubernetes-operator/kubernetes/samples/scripts/elasticsearch-and-kibana` directory.

   ```
   cd ~/weblogic-kubernetes-operator/kubernetes/samples/scripts/
   elasticsearch-and-kibana
   ```

9. Update the value of the namespace parameter in the file `elasticsearch_and_kibana.yaml`.

   Kibana and ElasticSearch must use the same namespace as the Kubernetes WebLogic Operator.

   ```
     namespace: "sample-weblogic-operator-ns"
     image: busybox
     image: "elasticsearch:8.7.1"
     image: "kibana:8.7.1"
   ```

10. Deploy ElasticSearch and Kibana in the Kubernetes WebLogic Operator namespace.

    ```
    kubectl apply -f elasticsearch_and_kibana.yaml
    ```

11. Verify the pods were created with the command `kubectl get pods -n sample-weblogic-operator-ns`.

    ```
    [k8smain]$ kubectl get pods -n sample-weblogic-operator-ns
    NAME                                 READY   STATUS
    RESTARTS      AGE
    elasticsearch-f7b7c4c4-zkhp8         1/1     Running   2 (52s
    ago)   2m26s
    kibana-57f6685789-25zsn              1/1     Running
    0             2m26s
    weblogic-operator-5789ddb9fc-jgdh6   2/2     Running
    0             15h
    ```

12. Enter the ElasticSearch pod and verify that ElasticSearch is running by making a curl request to "http://elasticsearch:9200/". Then exit the ElasticSearch pod.

```
[k8smain]$ kubectl exec -it elasticsearch-f7b7c4c4-wd94v -n sample-
weblogic-operator-ns -- /usr/bin/curl "http://elasticsearch:9200/"
Defaulted container "elasticsearch" out of: elasticsearch, set-vm-max-map-
count (init)
{
  "name" : "elasticsearchhost1",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "kkqeHA0LSUafAfHZ6MEktw",
  "version" : {
    "number" : "8.7.1",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "f229ed3f893a515d590d0f39b05f68913e2d9b53",
    "build_date" : "2023-04-27T04:33:42.127815583Z",
    "build_snapshot" : false,
    "lucene_version" : "9.5.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

13. Deploy FluentD.

```
kubectl apply -f ~/occas81/dockerfiles/8.1.0.0.0/occas-efk-pv/FluentD/
fluentd.yml
```

14. After a few minutes, verify the pods are running.

```
[k8smain:~]$ kubectl get pods -n sample-domain1-ns
NAME                                 READY   STATUS    RESTARTS
AGENAME                                                        READY
STATUS      RESTARTS   AGE
fluentd-container-d658498c7-6bd7q    1/1     Running   0          87m
sample-domain1-admin-server          1/1     Running   0          21h
sample-domain1-managed-server1       1/1     Running   0          21h
```

15. To view the Kibana interface, first use the `kubectl describe service` command to find the port number for Kibana. Then navigate to `http://<IP address>:<port number>/status` where <IP address> is the IP address of the master node.

```
[k8smain]$ kubectl describe service kibana -n sample-weblogic-operator-ns
Name:                     kibana
Namespace:                sample-weblogic-operator-ns
Labels:                   app=kibana
Annotations:              <none>
Selector:                 app=kibana
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.96.76.176
```

```
IPs:                      10.96.76.176
Port:                     <unset>  5601/TCP
TargetPort:               5601/TCP
NodePort:                 <unset>  32488/TCP
Endpoints:                172.16.145.212:5601
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

16. From the Kibana interface, click **Management**, and then **Index Management**.

    The following default patterns are available with Converged Application Server:

    Table 1-1 Index Patterns

Name	Description
occas-generic-index	Fluentd reads all log files under the log folder logHome and exports to this index.
engine-stdout-log	Fluentd reads Converged Application Server engine stdout logs and filters 'stdout'.
admin-server-critical-warnings	Fluentd reads Converged Application Server admin-server logs and filters critical/warning messages.
engine-all-log	Fluentd reads all engine logs and exports to this index.
occas-log	Fluentd reads engine logs and filters 'wlss' or 'msg' to get Converged Application Server specific logs.
occas-error-log	Fluentd reads engine logs and filters 'error' to get Converged Application Server error logs.
occas-warning-log	Fluentd reads engine logs and filters 'warning' to get Converged Application Server warning logs.
occas-sip-message-log	Fluentd reads engine/sip-message logs and filters 'CSeq' to get Converged Application Server SIP logs.

17. To create your own index patterns:

    a. From the Kibana interface, click **Management**, and then **Stack Management**, and then **Kibana**, and then **Data Views**, and then **Create Data View**.

    b. Enter a name.

    c. Enter an Index pattern.

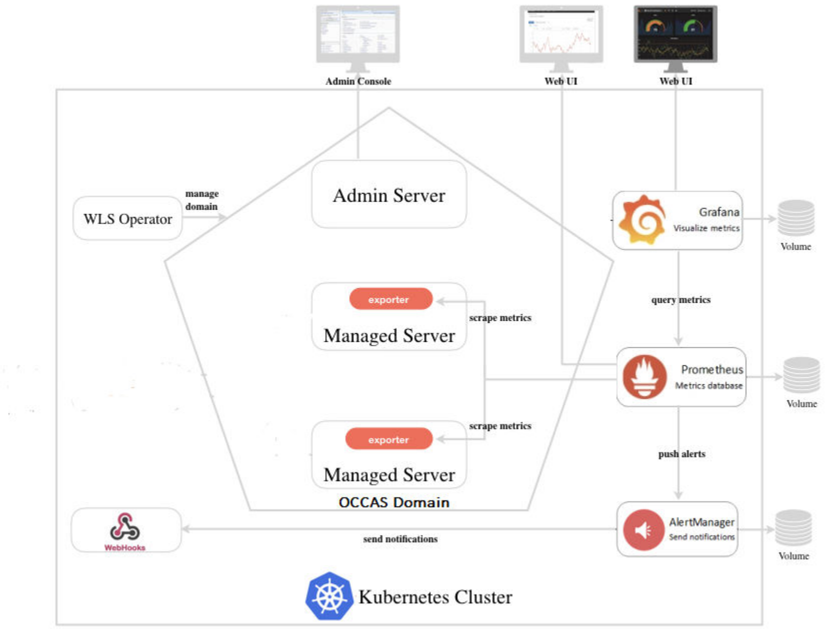
    d. Click **Save data view to Kibana**.

    After creating an index pattern, click **Discover** to view the log.

Deploy Prometheus and Grafana

Prometheus and Grafana are optional third-party monitoring tools that you can install to gather metrics on your Converged Application Server installation. When both are installed, Converged Application Server uses the WebLogic Monitoring Exporter to export Prometheus-compatible metrics that are then exposed to Grafana, which visualizes the metrics over time to provide futher insights into the data.

Figure 1-1 Cluster Architecture with Monitoring Tools



1. Pull the WebLogic Monitoring Exporter Docker image.

   ```
   podman pull ghcr.io/oracle/weblogic-monitoring-exporter:2.1.3
   ```

2. Navigate to the directory `occas81/dockerfiles/8.1.0.0.0/monitoring`.
3. Open the file `domain.yaml` and confirm the image, namespace,domainUID, and domainHome are correct.
   - `image: "oracle/occas:8.1.0.0.0-generic-8"`
   - `namespace: sample-domain1-ns`
   - `weblogic.domainUID: sample-domain1`
   - `domainHome: /u01/pv/domains/sample-domain1`
4. Depoy the WebLogic Monitoring Exporter .

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

5. Configure the Persistent Volume (PV) paths for Prometheus and Grafana.

   In this example, `/u01/pv` is the root PV directory.

   ```
   export PV_ROOT=/u01/pv
   sed -i 's@%PV_ROOT%@'"$PV_ROOT"'@' prometheus/persistence.yaml
   ```

```
sed -i 's@%PV_ROOT%@'"$PV_ROOT"'@' prometheus/alert-persistence.yaml
sed -i 's@%PV_ROOT%@'"$PV_ROOT"'@' grafana/persistence.yaml
```

6. Create a new namespace `monitoring`.

   ```
   kubectl create ns monitoring
   ```

7. Deploy the PV and PVC files for Prometheus and Grafana.

   ```
   kubectl apply -f prometheus/persistence.yaml
   kubectl apply -f prometheus/alert-persistence.yaml
   kubectl apply -f grafana/persistence.yaml
   ```

8. Add the Helm repositories for Prometheus and Grafana.

   ```
   helm repo add prometheus-community https://prometheus-
   community.github.io/helm-charts
   helm repo add grafana https://grafana.github.io/helm-charts
   ```

9. Update the `extraScrapeConfigs/basic_auth` values in the file `prometheus/values.yaml`.

10. Install the Prometheus chart.

    ```
    helm install --wait prometheus \
      --namespace monitoring \
      --values prometheus/values.yaml prometheus-community/prometheus \
      --version 17.0.0
    ```

11. Create the Grafana administrative credentials.

    ```
    kubectl --namespace monitoring create secret generic grafana-secret
    \
      --from-literal=username=admin --from-literal=password=<password>
    ```

12. Install the Grafana chart.

    ```
    helm install grafana \
      --namespace monitoring \
      --values grafana/values.yaml grafana/grafana \
      --version 6.52.9
    ```

13. (Optional) In versions 4.0.5 and higher, the Weblogic Kubernetes Operator sets the `enableRest` parameter to false. To enable it for autoscaling managed servers, run this command:

    ```
    helm upgrade sample-weblogic-operator weblogic-operator/weblogic-
    operator  \
      --version 4.0.5 \
      --namespace sample-weblogic-operator-ns \
      --set serviceAccount=sample-weblogic-operator-sa \
      --set "enableRest=true" \
      --wait
    ```

Refer to the Prometheus documentation and the Grafana documentation for how to configure and use those products.

Configure Grafana Data Sources

Once the WebLogic Monitoring Exporter starts exporting Converged Application Server metrics to Prometheus, configure Grafana to use Prometheus as its own source of data.

1. Navigate to the Grafana web interface.

   The URL is http://<IP address>:80/.

2. Click **Configuration**, and then **Data Source**, and then **Add Data Source** .
3. Select **Prometheus**.
4. Fill in the URL with the port number.

   The default URL for Prometheus is http://<IP address>:80/.

5. Click **Save & test**.
6. Click **Dashboards**, and then **Manage**, and then **Import** .
7. On the **Import** page, click **Upload .json File** and upload the file `oiccas81/dockerfiles/8.1.0.0.0/monitoring/grafana/OCCAS-DashBoard.json`.

Apply a Patch

The Dockerfile in the `occas81/dockerfiles/8.1.0.0.0/patch-opatch-update` directory extends the Converged Application Server image by updating OPatch and applying a patch. If an update is released for the Converged Application Server or one of its components (like WebLogic or Confluence), use OPatch to patch your system.

See Security Alerts to register for security notifications.

1. Log in to My Oracle Support.
2. Download the patches for the Converged Application Server or one of its components.
3. Identify the top-level folder name within the archive file.

   Syntax:

   ```
   unzip -l <zip file> | head
   ```

   For example:

   ```
   [user@host:~]$ unzip -l p33286174_141100_Generic.zip | head
   Archive:  p33286174_141100_Generic.zip
     Length      Date    Time    Name
   ---------  ---------- -----   ----
           0  09-22-2021 13:08   33286174/
           0  09-21-2021 19:11   33286174/etc/
           0  09-21-2021 19:11   33286174/etc/config/
           0  09-21-2021 19:11   33286174/files/
           0  09-21-2021 19:11   33286174/files/coherence/
           0  09-21-2021 19:11   33286174/files/coherence/bin/
           0  09-21-2021 19:11   33286174/files/coherence/lib/
   ```

In this example, the top-level folder is `33286174`.

4. Move the patch files into the `occas81/dockerfiles/8.1.0.0.0/patch-opatch-update` directory.

   For example:

   ```
   mv ~/Downloads/p28186730_139426_Generic.zip ~/occas81/dockerfiles/
   8.1.0.0.0/patch-opatch-update
   ```

5. Update `PATCH_PKG` variables in the Dockerfile to match the filename of patch files.

   ```
   ENV PATCH_PKG0="p28186730_139426_Generic.zip"
   ENV PATCH_PKG1="p33286174_141100_Generic.zip"
   ENV PATCH_PKG2="p33416881_141100_Generic.zip"
   ```

6. Update the `/u01/<folder>` directories within the Dockerfile with the top-level folder names of the patches you're installing.
7. Build the image.

   ```
   podman build --force-rm=true --no-cache=true -t oracle/occas:occas-
   generic-oct2021cpupatch-upgrade -f Dockerfile .
   ```

2 Deploy with the Container Engine for Kubernetes

The Converged Application Server can be deployed in OCI using Oracle's Container Engine for Kubernetes.

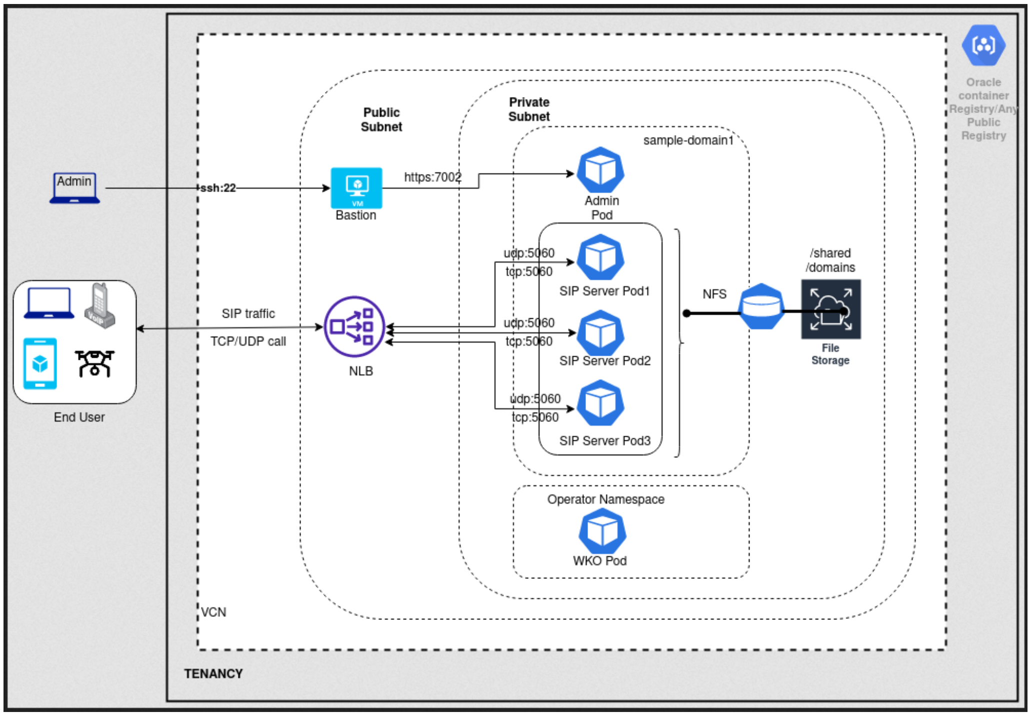
Oracle Cloud Infrastructure Container Engine for Kubernetes is a fully-managed, scalable, and highly available service that you can use to deploy your containerized applications to the cloud. Use Container Engine for Kubernetes (abbreviated OKE) when your development team wants to reliably build, deploy, and manage cloud-native applications. You specify whether to run applications on virtual nodes or managed nodes, and Container Engine for Kubernetes provisions them on Oracle Cloud Infrastructure in an existing OCI tenancy.

To learn more about OKE, view the OKE documentation.

Architecture Overview

The Converged Application Server deployed with OKE uses the following OCI services:

- **Container Engine for Kubernetes:** The Container Engine for the Kubernetes cluster has three worker nodes so that the OCCAS-Coherence clusters themselves have the highest availability.
- **File Storage:** The domain configuration files are stored on shared storage that is accessible from all WebLogic servers in the cluster, on File Storage. Because of this setup, you don't need to rebuild Docker images for changes in the domain configuration, backup is faster and centralized, and logs are stored by default on persistent storage.
- **Load Balancing:** By default, the WebLogic servers (admin or clustered managed servers) created by the operator (WKO) are not exposed outside the Container Engine for Kubernetes cluster. The Converged Application Server uses the OCI Network Load Balancer, which supports TCP and UDP, to expose an application to the outside world.
- **Registry:** A Converged Application Server image can be pulled from https://container-registry.oracle.com/. Optionally, the Docker images can be stored in a private Oracle Cloud Infrastructure Registry repository.



Create the Kubernetes Cluster

Before creating a Kubernetes cluster, follow the steps in Preparing for Container Engine for Kubernetes to set the correct Identity and Access Management (IAM) policy.

1. Log in to the Console for your OCI Tenancy.
2. From the OCI Menu, select **Developer Services** and then **Container Clusters (OKE)**.
3. If the compartment has not been set, select your compartment from the left-hand drop-down list.
4. Click **Create Cluster** and then **Quick Create** and then **Submit**.
5. Enter a name for the cluster.
6. Select the Kubernetes version.
7. Select Managed Node type.
8. Select the shape of each worker node in the cluster and the number of nodes in each subnet.
   To ensure the highest performance for those demanding workloads, Container Engine for Kubernetes allows you to select bare metal instances or even GPU instances on which to run your containers.
9. Click **Next** and then **Create cluster**.
10. After the cluster is created, you access the cluster in one of two ways:
    - Using kubectl in Cloud Shell
    - Using kubectl installed locally

See Accessing a Cluster Using Kubectl for more information.

11. Run `kubectl cluster-info` or `kubectl config view` to confirm the configuration.

After the cluster has been created:

1. Select **Menu**, and then **Networking**, and then**VNC**.
2. Note the subnets and their CIDR blocks.

Create and Configure the File Storage Service

When deployed with OKE, the Converged Application Server depends on the File Storage service.

1. From the OCI Menu, select **Storage** and then **Mount Targets** under the File Storage section.
2. Click **Create Mount Target**
3. Select the VCN of your cluster and the subnet of your worker nodes.
   If you require extra isolation, you may optionally setup additional VCN routing and security.
4. Click **Create**.
5. Select the created Mount Target.
6. Under the Exports section, click Create Export to create a file system for the Persistent Volume.
7. Accept the defaults and click **Create**.

Create a Virtual Network

1. From the Console, select **Networking** and then **Virtual Cloud Networks**.
2. Select the VCN assigned to your cluster and the subnet you previously selected when creating an export path.
3. Select the default Security List.
4. Create four ingress destination ports: two TCP ports (2048-2050 and 111) and two UDP ports (2048 and 111).

   The IP range must match the CIDR range of your VCN.
5. Create three egress source ports: two TCP ports (2048-2050 and 111) and one UDP port (111).

   The IP range must match the CIDR range of your VCN.

Create a Storage Class

1. Create a storage class file called ocistorageclass.yaml

   Use the OCID from the Mount Target as the value of the mntTargetId parameter.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: sample-domain1-weblogic-sample-storage-class
provisioner: oracle.com/sample-domain1-weblogic-sample-storage-class
parameters:
    mntTargetId: <OCID of the created mount target>
```

2. Apply the file to create the Storage Class.

   ```
   kubectl apply -f ocistorageclass.yaml
   ```

Install the Kubernetes Weblogic Operator with OKE

The WebLogic Kubernetes Operator supports running your WebLogic Server domains on Kubernetes. It lets you encapsulate your entire WebLogic Server installation and layered applications into a portable set of cloud neutral images and simple resource description files.

1. If Helm is not already installed, install Helm.

   ```
   curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/
   helm/main/scripts/get-helm-3
   chmod 700 get_helm.sh
   ./get_helm.sh
   ```

2. Clone the Oracle Weblogic Kubernetes Operator repository.

   The Oracle Weblogic Kubernetes Operator is used to configure the lifecycle of the Converged Application Server images. See the Oracle Weblogic Kubernetes Operator documentation.

   ```
   git clone --branch v4.0.5 https://github.com/oracle/weblogic-
   kubernetes-operator
   ```

3. Pull the Docker image for the WebLogic Kubernetes Operator.

   ```
   podman pull ghcr.io/oracle/weblogic-kubernetes-operator:4.0.5
   ```

   If your jumpbox or cluster cannot access ghcr.io, check the subnet and gateway settings of your network.

4. Create the namespace for the Weblogic Kubernetes Operator.

   The syntax for the command is: `kubectl create namespace <unique name>`. For example:

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

5. Create the service account within that namespace for the WebLogic Kubernetes Operator.

   ```
   kubectl create serviceaccount -n sample-weblogic-operator-ns sample-
   weblogic-operator-sa
   ```

6. Update the following parameters in the file `weblogic-kubernetes-operator/kubernetes/charts/weblogic-operator/values.yaml`.

If you plan on installing the EFK stack, set `elkIntegrationEnabled` to true; otherwise, leave `elkIntegrationEnabled` as false.

- `serviceAccount`—Set this to the previously created service account for the WebLogic Kubernetes Operator.

  ```
  serviceAccount: "sample-weblogic-operator-sa"
  ```

- `domainNamespaces`—Set this to the previously created namespace for the WebLogic Kubernetes Operator.

  ```
  domainNamespaces:
  - "sample-weblogic-operator-ns"
  ```

- `elkIntegrationEnabled`—Set this to true if you plan on installing the EFK stack; leave as false if you do not plan on installing the EFK stack.

  ```
  elkIntegrationEnabled: false
  ```

- `elasticSearchHost`—Change the word 'default' in 'elasticsearch.**default**.svc.cluster.local' to the namespace previously created for the WebLogic Kubernetes Operator.

  ```
  elasticSearchHost: "elasticsearch.sample-weblogic-operator-
  ns.svc.cluster.local"
  ```

7. From the `weblogic-kubernetes-operator` directory, start the operator.

   ```
   helm install sample-weblogic-operator kubernetes/charts/weblogic-operator
   \
     --namespace sample-weblogic-operator-ns \
     --set image=ghcr.io/oracle/weblogic-kubernetes-operator:4.0.5 \
     --set serviceAccount=sample-weblogic-operator-sa \
     --set "enableClusterRoleBinding=true"\
     --set "domainNamespaceSelectionStrategy=LabelSelector"\
     --set "domainNamespaceLabelSelector=weblogic-operator\=enabled"\
     --wait
   ```

Verify the pod is running with the following command:

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

For example:

```
[weblogic-kubernetes-operator]$ kubectl get pods -n sample-weblogic-operator-
ns
NAME                                 READY   STATUS    RESTARTS   AGE
weblogic-operator-7885684685-4jz5l   1/1     Running   0          2m32s
[weblogic-kubernetes-operator]$
```

> **Note:**
>
> If you are installing the EFK stack and set `elkIntegrationEnabled` to true, then the READY column will display 2/2 when both containers are ready. If you are not installing the EFK stack and left `elkIntegrationEnabled` as false, then the READY column will display 1/1 when the container is ready.

Download Domain Creation Files

In addition to downloading the `occas_generic.jar` file, you need to download the domain creation files.

1. Sign in to My Oracle Support.
2. Search for "Docker Script for OCCAS 8.1 Installation" (Doc ID 3007597.1).
3. Under the Details section, click the link to download occas81.zip.
4. Unzip the file.

   ```
   unzip occas81.zip
   ```

   This creates the `occas81/` directory.

Create a Persistent Volume

Create a persistent volume and a persistent volume claim to make the shared file system accessible from the pods.

1. Create and label a namespace that can host one or more domains.

   ```
   kubectl create namespace sample-domain1-ns
   kubectl label ns sample-domain1-ns weblogic-operator=enabled
   ```

2. Navigate to the `occas81/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv` directory.
3. Create Weblogic credentials.

   ```
   ./create-weblogic-credentials.sh -u <username> -p <PASSPHRASE> -n
   sample-domain1-ns -d sample-domain1 -s sample-domain1-weblogic-
   credentials
   ```

4. Open the `sample-domain1-weblogic-sample-pv_nfs.yaml` file.
   a. Set the `server` parameter to the IP address of the OFSS Export Path.
   b. Set the `path` parameter to the value of your previously configured Export Path.

   ```
   apiVersion: v1
   kind: PersistentVolume
   metadata:
     name: sample-domain1-weblogic-sample-pv
   ```

```
  labels:
    weblogic.domainUID: sample-domain1
spec:
  storageClassName: sample-domain1-weblogic-sample-storage-class
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteMany
  # Valid values are Retain, Delete or Recycle
  persistentVolumeReclaimPolicy: Retain
  nfs:
    server: 192.0.2.5
    path: "/FileSystem-20220604-1759-09"
```

5. Create the Persistent Volume.

```
kubectl create -f sample-domain1-weblogic-sample-pv_nfs.yaml
kubectl create -f sample-domain1-weblogic-sample-pvc_nfs.yaml
```

Choose the Image

The Oracle Communications Converged Application Server image is located on the Oracle Container Registry.

1. If using JDK 8, use the JDK 8 container.

```
container-registry.oracle.com/middleware/occas:8.1.0.0.0-generic-8
```

2. If using JDK 11, use the JDK 11 container.

```
container-registry.oracle.com/middleware/occas:8.1.0.0.0-generic-11
```

Create Kubernetes Secret

Create a Kubernetes secret in order to pull from a private or public registry.

1. Use kubectl to create the secret.

   Use your own email address in the docker-username and docker-email parameters.

```
kubectl create secret docker-registry occas-image-sec \
  --docker-server=container-registry.oracle.com \
  --docker-username=first.last@oracle.com \
  --docker-password=<token> \
  --docker-email=first.last@oracle.com \
  -n sample-domain1-ns
```

2. Log in to the Oracle Container Registry.

   Use the same username and password as above.

```
podman login container-registry.oracle.com
```

Create Domain

The Converged Application Server domain is a logically related group of resources. Domains include the administration server and all managed servers. Converged Application Server domains extend Oracle WebLogic Server domains to support SIP and other telecommunication protocols.

See Understanding Oracle WebLogic Server Domains.

1. Navigate to the `occas81/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv/domain-home-on-pv` directory.
2. Open the `create-domain-inputs.yaml` file.
3. Set the following parameters.

Table 2-1 Parameters

Parameter	Definition	Default
configuredManagedServerCount	Name of the WebLogic cluster instance to generate for the domain.	5
createDomainFilesDir	Directory on the host machine to locate all the files to create a WebLogic domain, including the script that is specified in the `createDomainScriptName` property. By default, this directory is set to the relative path `wlst`, and the create script will use the built-in WLST offline scripts in the `wlst` directory to create the WebLogic domain. It can also be set to the relative path `wdt`, and then the built-in WDT scripts will be used instead. An absolute path is also supported to point to an arbitrary directory in the file system. The built-in scripts can be replaced by the user-provided scripts or model files as long as those files are in the specified directory. Files in this directory are put into a Kubernetes ConfigMap, which in turn is mounted to the `createDomainScriptsMountPath`, so that the Kubernetes Pod can use the scripts and supporting files to create a domain home.	wlst

Table 2-1 (Cont.) Parameters

Parameter	Definition	Default
domainHome	Home directory of the WebLogic domain. If not specified, the value is derived from the domainUID as /shared/domains/ <domainUID>.	/ FileSystem-20220305-1124-54/domains/sample-domain1
domainPVMountPath	Mount path of the domain persistent volume.	/ FileSystem-20220305-1124-54
domainUID	Unique ID that will be used to identify this particular domain. Used as the name of the generated WebLogic domain as well as the name of the Domain. This ID must be unique across all domains in a Kubernetes cluster. This ID cannot contain any character that is not valid in a Kubernetes Service name.	sample-domain1
exposeAdminNodePort	Boolean indicating if the Administration Server is exposed outside of the Kubernetes cluster.	true
exposeAdminT3Channel	Boolean indicating if the T3 administrative channel is exposed outside the Kubernetes cluster.	true
httpAccessLogInLogHome	Boolean indicating if server HTTP access log files should be written to the same directory as `logHome`. Otherwise, server HTTP access log files will be written to the directory specified in the WebLogic domain home configuration.	true
image	OCCAS generic image	container-registry.oracle.com/ middleware/occas:8.1.0.0.0-generic-11
imagePullSecretName	Name of the Kubernetes Secret to access the container registry to pull the WebLogic Server image. The presence of the secret will be validated when this parameter is specified	occas-image-sec
initialManagedServerReplicas	Number of Managed Servers to start initially for the domain.	2

Table 2-1 (Cont.) Parameters

Parameter	Definition	Default
javaOptions	Java options for starting the Administration Server and Managed Servers. A Java option can have references to one or more of the following pre-defined variables to obtain WebLogic domain information: $(DOMAIN_NAME), $(DOMAIN_HOME), $(ADMIN_NAME), $(ADMIN_PORT), and $(SERVER_NAME).	-Dweblogic.StdoutDebugEnabled=false
logHome	The in-pod location for domain log, server logs, server out, introspector out, Node Manager log, and server HTTP access log files. If not specified, the value is derived from the domainUID as /shared/logs/<domainUID>.	/FileSystem-20220305-1124-54/domain-home-in-pv/logs
managedServerPort	Port number for each Managed Server.	8001
namespace	Kubernetes Namespace in which to create the domain.	sample-domain1-ns
persistentVolumeClaimName	Name of the persistent volume claim. If not specified, the value is derived from the domainUID as <domainUID>-weblogic-sample-pvc	sample-domain1-weblogic-sample-pvc
productionModeEnabled	Boolean indicates if production mode is enabled for the domain.	true
serverStartPolicy	Determines which OCCAS will be started. Legal values are `NEVER`, `IF_NEEDED`, `ADMIN_ONLY`.	IF_NEEDED
t3ChannelPort	Port for the T3 channel of the network access point.	30012

Table 2-1 (Cont.) Parameters

Parameter	Definition	Default
t3PublicAddress	Public address for the T3 channel. This should be set to the public address of the Kubernetes cluster. This would typically be a load balancer address. For development environments only, in a single server (all-in-one) Kubernetes Deployment, this may be set to the address of the master, or at the very least, it must be set to the address of one of the worker nodes.	If not provided, the script will attempt to set it to the IP address of the Kubernetes cluster.
sipPublicAddress	The public listen to address to embed in SIP headers when the channel is used for an outbound connection. This is typically the IP address presented by the IP sprayer or external load balancer as the virtual IP (VIP) for the telecommunication servicesif sipPublicAddress is blank, Operator will assign kubernetes internal address same as ListenAddress Like sample-domain1-managed-server1	Load Balancer private address
weblogicCredentialsSecretName	Name of the Kubernetes Secret for the Administration Server user name and password. If not specified, the value is derived from the domainUID as <domainUID>-weblogic-credentials.	domain1-weblogic-credentials
serverPodCpuRequest serverPodMemoryRequest serverPodCpuCLimit serverPodMemoryLimit	The maximum amount of compute resources allowed, and minimum amount of compute resources required, for each server pod. Please refer to the Kubernetes documentation on "Managing Compute Resources for Containers" for details.	Resource requests and resource limits are not specified.

In the generated YAML files, the names of the Kubernetes resources are formed from values specified in the `create-inputs.yaml` file. Invalid characters like uppercase letters or underscores are converted to lowercase letters and hyphens.

4. Update the `fix-pvc-owner` item of `create-domain-job-template.yaml` file.

   ```
   ...
   initContainers:
     - name: fix-pvc-owner
       image: %WEBLOGIC_IMAGE%
       command: ["sh", "-c", "chown 1000:0 %DOMAIN_ROOT_DIR%/. && find
   %DOMAIN_ROOT_DIR%/. -maxdepth 1 ! -name '.snapshot' ! -name '.' -
   print0 | xargs -r -0 chown -R 1000:0"]
       volumeMounts:
   ...
   ```

5. Use the `create-domain.sh` script to create the domain.

   The command requires two arguments: the input YAML file and the output directory.

   ```
   ./create-domain.sh \
       -i create-domain-inputs.yaml \
       -o ~/occas81/dockerfiles/8.1.0.0.0/occas-domain-home-on-pv/
   domain-home-on-pv
   ```

6. Navigvate to the sample-domain1 folder.

   ```
   cd weblogic-domains/sample-domain1/
   ```

7. Deploy the WebLogic domain in the Kubernetes cluster.

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

   The domain will now be managed by the Weblogic Operator.

8. Create an SSH tunnel through your bastion host to access the Administrator Console. Open a browser and navigate to http://<IP address>:30701/console to visit the WebLogic Server's management interface.

Create Load Balancers

1. Create a nlb.yaml file that defines the network load balancer.

   ```
   apiVersion: v1
   kind: Service
   metadata:
     name: occas-nlb
     namespace: sample-domain1-ns
     annotations:
       oci.oraclecloud.com/load-balancer-type: "nlb"
       oci-network-load-balancer.oraclecloud.com/security-list-
   management-mode: "All"
   spec:
     type: LoadBalancer
     ports:
     - port: 5060
       targetPort: 5060
   ```

```
    protocol: TCP
  selector:
    weblogic.clusterName: bea-engine-tier-clust
    weblogic.createdByOperator: "true"
    weblogic.domainUID: sample-domain1
```

2. Apply the file.

   ```
   kubectl apply -f nlb.yaml
   ```

3. From the OCI Console, select **Networking** and then **Load Balancer** and confirm the load balancer was successfully created.