Java Platform, Standard Edition

Containerized Advanced Management Console Deployment Guide

Release 3
F51657-02
January 2022

ORACLE

Java Platform, Standard Edition Containerized Advanced Management Console Deployment Guide, Release 3

F51657-02

Contents

Preface

The Containerized Advanced Management Console (AMC) deployment guide provides instructions on how to deploy Containerized AMC to Kubernetes environment. This guide also details the administration and maintenance aspects of Containerized AMC.

Audience

This document is intended for system administrators who manage the Java desktop environment in their enterprise. To deploy the Containerized AMC, you need to be familiar with Containers, Kubernetes, and Helm.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

- Advanced Management Console User Guide
- Oracle WebLogic Server Kubernetes Operator

Conventions

The following text conventions are used in this document:

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

Abbreviations

The following table lists the commonly used abbreviations in this guide.

Abbreviations	Full Form
AMC	Advanced Management Console
DBaaS	Database as a Service
OCR	Oracle Container Registry
WKO	Oracle WebLogic Kubernetes Operator
WLS	WebLogic Server
WIT	WebLogic Image Tool
WDT	WebLogic Deploy Tooling

1

About Containerized Advanced Management Console

Advanced Management Console (AMC) is a feature available as part of Java SE Subscription. Oracle provides an easy-to-deploy containerized version of AMC.

Unlike a non-containerized AMC, you don't need to install and maintain the WebLogic Server (WLS) separately. It uses WebLogic Kubernetes Operator (WKO) for deploying or managing AMC. This solution can be easily deployed in cloud or non-cloud environments and leverages scaling functionality of the operator within a Kubernetes framework. This will address the complexity involved in installing and maintaining the non-containerized AMC without the intervention of IT or any specialized technical skills.

The Containerized AMC provides the following capabilities:

- Rely only on well-known and easily available resources to deploy and manage containerized AMC in Kubernetes environment, either on cloud or on premise.
- Reduces installation, configuration, and deployment time.
- Ability to containerize existing compatible versions of the AMC application with underlying WLS component.
- Supports all features provided by non-containerized AMC installation.
- Performance on par with a non-containerized version.
- Easily scale containerized AMC deployments using Kubernetes.
- Ease of migration from non-containerized AMC to containerized AMC without requiring changes or redeploying already-deployed agents; as long as the same address is used for the WLS server or WLS load balancer.
- Supports other external database solutions such as Database as a Service (DBaaS).
- Reduces AMC upgrading overhead.

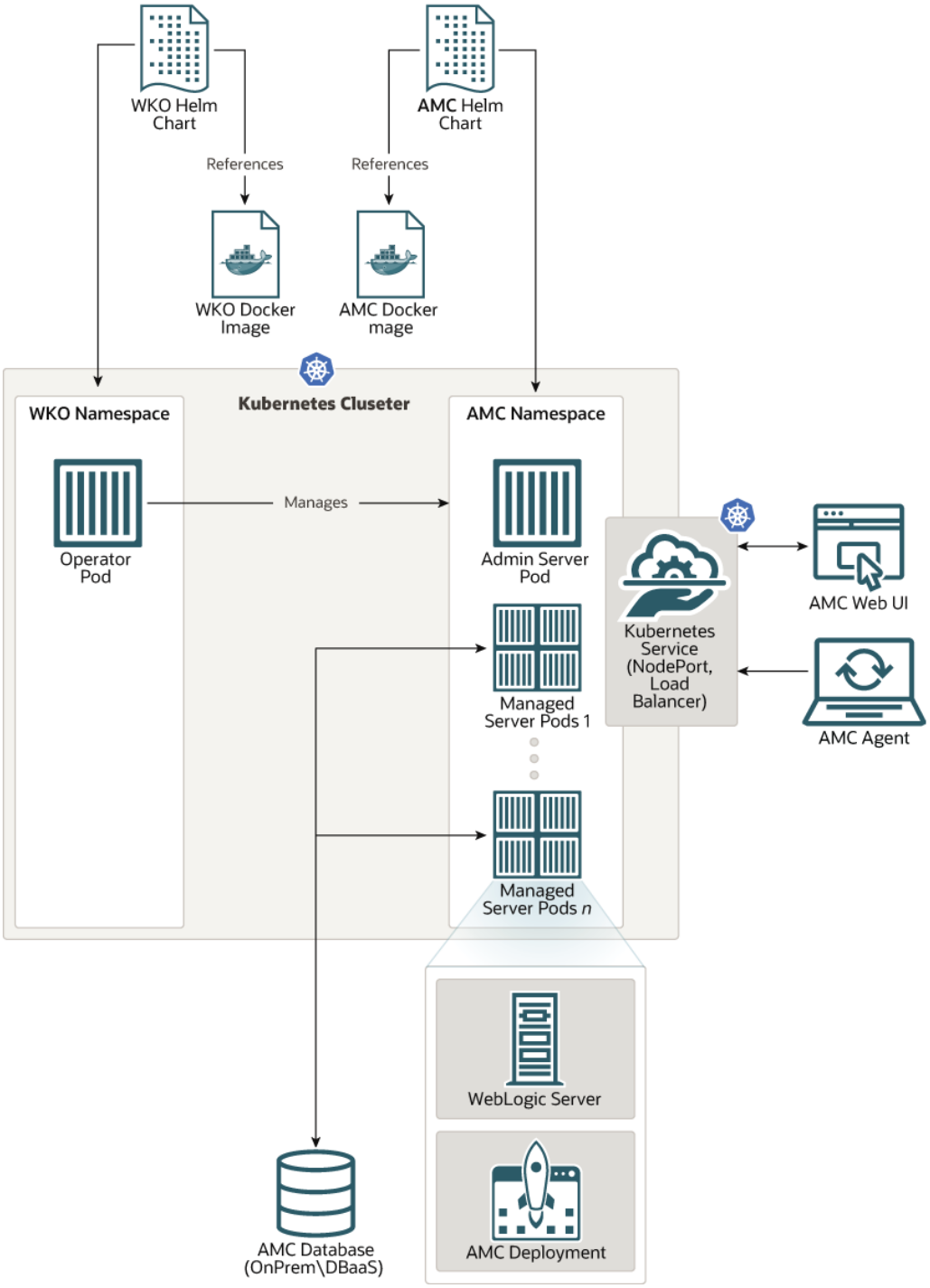
Overview

The Containerized AMC solution leverages the Oracle WebLogic Kubernetes Operator (WKO) Model in the image pattern for domain creation and management.

See Kubernetes Model in Image.

Architecture Diagram:

Figure 1-1 Containerized AMC Deployment Architecture



As part of this Containerized AMC solution, a custom AMC container image has to be created comprising of an AMC application archive, WebLogic server, and base WDT model that defines the domain home configuration. The Domain resource and other Kubernetes manifests for deploying AMC are packaged and distributed as a Helm chart. This Helm chart points to the container image that is created. The Helm chart values are configurable to support custom deployment settings.

The database installation and configuration for AMC is not included in the containerized AMC. The externally configured database is referenced through data source settings supplied as a part of the chart configuration.

The WKO instance handles the AMC domain creation and management. The installation of WKO is a prerequisite for Containerized AMC and has to be done through the WKO Helm chart, which is hosted on GitHub. After the successful installation of the AMC Helm chart, the admin server and managed server pods hosting the AMC deployment will be up and ready to use.

The Kubernetes service endpoints are exposed either as NodePort or LoadBalancer based on the chart configuration. The AMC Web UI and AMC Agent communicate with the server through these service endpoints.

Advantages

Containerized AMC provides ease of deployment and various advantages as compared to non-containerized AMC.

Some of the advantages of Containerized AMC are:

- It takes approximately 30 minutes to deploy containerized AMC for the first time as opposed to the multi-hour effort to install non-containerized AMC.
- WLS is bundled with Containerized AMC; therefore, you don’t have to install it separately.
- Maintaining Containerized AMC is relatively easy as compared to applying patches manually and regularly upgrading to current versions.
- Scaling flexibility in Containerized AMC removes the complexity of introducing additional components, such as load balancers in non- containerized AMC, which complicates the installation, configuration, and maintenance of WLS and database.
- You can update the images used by the solution with up-to-date versions of WLS and other required components available from Oracle, and other widely available sources, for example, container hub for load balancers.
- Easy to upgrade AMC and WLS versions.
- Leverage WKO features for domain administration and scaling.

2
Setup the Environment

Before you start the deployment, install the prerequisite software and setup the environment.

You need to build the AMC container image using AMC ear file. See Create the AMC Container Image.

Prerequisites

The following are the prerequisites to deploy the Containerized AMC:

- Familiarity with Containers, Kubernetes, and Helm
- Ability to setup the Kubernetes cluster
- Availability of Oracle DB or MySQL database
- Access to Oracle Container Registry (OCR)
- Java SE Subscription license to download AMC ear from Oracle Technology Resources (OTN) or My Oracle Support (MOS)
- Privilege to perform common administrative tasks on Database

System Requirements

System requirements to deploy WebLogic Kubernetes Operator (WKO) and Containerized AMC.

- Prerequisites for WKO release: See WKO Prerequisites
- Setup either of the following database, which is ready to accept external connections from AMC application:
  - Oracle DB: 19c, 12c, or 11g
  - MySQL: 8, 5.7, or 5.6
- WebLogic Cluster, as a publicly accessible Kubernetes Service, must be accessible from all agent machines

> **Note:**
>
> Change in the host name provided at the time of AMC initialization might stop AMC Agent to communicate with the server and block agent bundle downloads.

- WKO deployment in Kubernetes Cluster
- Kubernetes worker nodes must have internet access as AMC connects to external network to fetch Java release information
- All agents must be able to access the server

Environment Setup

You need to setup the environment for seamless deployment of Containerized AMC.

Ensure you have the following environment setup:

- Setup the Kubernetes environment. See:
  - Oracle WebLogic Server Kubernetes Operator User Guide
  - Oracle Linux Cloud Native Environment
  - Oracle Container Engine for Kubernetes
- WKO is a prerequisite to manage the AMC domain resource. You can either make use of an already running WKO deployment in a cluster, or setup a new WKO and configure it for AMC deployment.
- Setup the database, either on-premise or DBaaS. Oracle database or MySQL is supported.
- Install and configure the database. See Database Installation and Configuration for Advanced Management Console. Also, see the Configurable Parameters, Table 3-1 to know more about the configuration parameters that are required to setup the database for Containerized AMC.
- Ensure that Oracle or MySQL database is ready to accept the connection.

Create the AMC Container Image

You can use WebLogic Image Tooling (WIT) to create the AMC Container Image.

This topic provides a high level procedure to create the AMC container image. For detailed instructions, see Image Creation Guide.

> **Note:**
>
> - Ensure that the operating system user has adequate permissions to perform install and deploy commands. Use `sudo` or `sudo -E` commands as required.
> - The folder names mentioned in this topic are for illustrative purposes. You can name the folders as per your requirement.

1. Create a working folder `model-images`.

   Example Command:

   ```
   $ mkdir model-images
   $ cd model-images/
   ```

2. Download latest version of WIT and place it in `model-images` folder. Extract the zip into this folder.

Example Command:

```
$ wget https://github.com/oracle/weblogic-image-tool/releases/download/release-1.9.5/imagetool.zip
$ unzip imagetool.zip
```

3. Download the latest version of WebLogic Deploy Tooling (WDT) and place it in the `model-images` folder.

   Example Command:

   ```
   $ wget https://github.com/oracle/weblogic-deploy-tooling/releases/download/release-1.9.6/weblogic-deploy.zip
   ```

4. Download the WebLogic container image from Oracle Container Registry. Goto https://container-registry.oracle.com/ then **Middleware** and then to **weblogic** repository to download the image. You need to sign in and accept the license proceeding. This is a one time requirement.

   Use the following command to download the WebLogic container image:

   ```
   docker login container-registry.oracle.com
   docker pull container-registry.oracle.com/middleware/weblogic:12.2.1.4
   ```

   Example Command:

   ```
   $ docker login container-registry.oracle.com
   Username: <abc>
   Password: <123>
   $ docker pull container-registry.oracle.com/middleware/weblogic:12.2.1.4
   ```

5. Create the `archive-AMC` inside the `model-images` folder to save the AMC application bundle. The folder structure within `archive-AMC` must be `wlsdeploy/applications`. WDT archives have a well-defined directory structure, which always has `wlsdeploy` as the top directory. Download the latest AMC EAR (JavaAMC-2_20.ear) from AMC download page. Save the file in the `wlsdeploy/applications` folder.

   Example Command:

   ```
   $ mkdir archive-AMC
   $ cd archive-AMC
   $ mkdir -p wlsdeploy/applications
   $ cp JavaAMC-2_20.ear wlsdeploy/applications/
   $ zip -r archive.zip wlsdeploy
   ```

6. Export JAVA_HOME:

   ```
   export JAVA_HOME=</jdk/home/>
   ```

   Example Command:

   ```
   $ export JAVA_HOME=/usr/java/jdk1.8.0_271-amd64
   ```

7. Cache the WDT for successful image creation using the following command from the `model-images` directory:

```
$ cd model-images
./imagetool/bin/imagetool.sh cache addInstaller \
--type wdt \
--version latest \
--path ./weblogic-deploy.zip
```

8. Stage the model files. Copy the following code snippet into a `model.amc.yaml` file and save it in the `model-images` folder.

   Sample `model.amc.yaml` file:

```
# Copyright (c) 2020, Oracle Corporation and/or its affiliates.
# Licensed under the Universal Permissive License v 1.0 as shown at
https://oss.oracle.com/licenses/upl.

domainInfo:
    AdminUserName: '@@SECRET:__weblogic-credentials__:username@@'
    AdminPassword: '@@SECRET:__weblogic-credentials__:password@@'
    ServerStartMode: 'prod'

appDeployments:
    Application:
        JavaAMC:
            SourcePath: 'wlsdeploy/applications/JavaAMC-2_20.ear'
            ModuleType: ear
            Target: '@@ENV:AMC-CLUSTER-NAME@@'
    Library:
        'jax-rs#2.0@2.22.4.0':
            SourcePath: '@@WL_HOME@@/common/deployable-libraries/
jax-rs-2.0.war'
            ModuleType: war
            Target: '@@ENV:AMC-CLUSTER-NAME@@'
```

9. Build the image using WIT:

```
./imagetool/bin/imagetool.sh update \
  --tag amc-<version>:latest \
  --chown oracle:root \
  --fromImage container-registry.oracle.com/middleware/
weblogic:12.2.1.4 \
  --wdtModel      ./model.amc.yaml \
  --wdtArchive    ./archive-AMC/archive.zip \
  --wdtModelOnly \
  --wdtDomainType WLS
```

   The `fromImage` is the WebLogic container image that is referenced from Oracle Container Registry.

Example Script:

```
#!/bin/bash
wls_image="container-registry.oracle.com/middleware/weblogic:12.2.1.4"

./imagetool/bin/imagetool.sh update \
  --tag amc2u20:latest \
  --chown oracle:root \
  --fromImage ${wls_image} \
  --wdtModel      ./model.amc.yaml \
  --wdtArchive    ./archive-AMC/archive.zip \
  --wdtModelOnly \
  --wdtDomainType WLS
```

The image is successfully built and following message appears:

```
[INFO ] Build successful. Build time=36s. Image tag=amc2u20:latest
```

Also, run the container (docker) image commands to verify if the image is created successfully.

Distribute the Image to all Kubernetes Nodes

The AMC container image is built on one machine. During deployment, the image must be available on all Kubernetes nodes.

To distribute the image to all Kubernetes nodes, `push` the image to a private container registry and `pull` the image from this registry during deployment. If the registry is not available, then you can distribute the image using the Save and Load method.

Follow these steps to distribute the image to all cluster nodes using the Save and Load method:

1. From the machine where the image is built, save the image as a `tar` file using the command:

   ```
   docker save <amc2u20:latest> -o <amc2u20>.tar
   ```

   > **Note:**
   >
   > Ensure that you use the same <repo/image:tag> (for example, *<amc2u20:latest>*) while configuring the parameters in the deployment configuration file. This configuration file is referred to as `custom-values.yaml` file in this deployment guide.

2. Copy the `tar` file to all the node machines using the command:

   ```
   scp <amc2u20>.tar <user>@<machine>:/home/<user>/
   ```

If you are using `ssh` key-based authentication, use the command:

```
scp -i private_key <amc2u20>.tar <user>@<machine>:/home/<user>/
```

3. On the Kubernetes node machines, load the image using the following command:

```
cat <amc2u20>.tar | docker load
```

3

Deploy Containerized Advanced Management Console on Kubernetes Environment

Ensure that all prerequisites are met before you start the deployment.

See Setup the Environment for details.

Create a working directory on your Kubernetes controller machine (on the machine where helm package and kubectl of your cluster is configured). Place all the required artifacts such as helm package, `custom-values.yaml` file, `jks` files, and so on in this working directory. You can perform helm install, update, and uninstall, and kubectl create and delete from this working directory. See Helm Commands for the complete list of CLI commands for helm.

Follow these steps to deploy Containerized AMC on Kubernetes environment:

1. Configure the Database
2. Deploy WebLogic Kubernetes Operator
3. Create Kubernetes Secrets and ConfigMap
4. Configure AMC Helm Chart
5. Deploy Containerized AMC

Configure the Database

Ensure that you have installed and configured MySQL or Oracle database; either on-premise or on DBaaS.

See Database Installation and Configuration for Advanced Management Console to setup and configure the database.
The DB service locator, name, and user credentials must be provided for WebLogic data source setup through AMC helm chart.

Deploy WebLogic Kubernetes Operator

Follow these steps to deploy WKO:

1. Add the WKO chart repository to the local cache by using the following command:

   ```
   helm repo add weblogic-operator https://oracle.github.io/weblogic-kubernetes-operator/charts
   ```

2. Create namespaces for WKO deployment and AMC deployment. A WebLogic domain will be deployed in this namespace and it will be managed by WKO. Use the following commands to create the namespace:

   ```
   kubectl create namespace <operator-namespace>
   kubectl create namespace <amc-namespace>
   ```

Example Command:

```
$ kubectl create namespace wko-ns
$ kubectl create namespace amc-ns
```

3. Install the WKO in the Operator's namespace *<operator-namespace>* by using the following command:

```
helm install <release-name> weblogic-operator/weblogic-operator
\
--set "domainNamespaces={<amc-namespace>}" \
-n <operator-namespace>
```

Example Command:

```
$ helm install wko weblogic-operator/weblogic-operator \
--set "domainNamespaces={amc-ns}" \
-n wko-ns
```

You can verify the status of your installation using the following commands:

- Verify if the operator's pod is running. Note that it might take a while for the operator's pod to be up and running.

```
 kubectl get pods -n <operator-namespace> --watch
```

Example Command:

```
$ kubectl get pods -n wko-ns  --watch
```

Expected Output:

```
NAME                                READY   STATUS    RESTARTS
AGE
weblogic-operator-75c85f5649-l6cbx   1/1     Running   0
107s
```

- View the operator's pod log to verify if the operator is up and running:

```
 kubectl logs -n <operator-namespace> -c weblogic-operator
deployments/weblogic-operator
```

Example Command:

```
$ kubectl logs -n wko-ns -c weblogic-operator deployments/weblogic-
operator
```

Create Kubernetes Secrets and ConfigMap

A Secret is an object that contains sensitive data, such as passwords, tokens, or keys. A ConfigMap is an object that stores non-confidential data in key-value pairs.

Create Kubernetes Secrets

You can store and manage confidential information, such as passwords, authentication tokens, and ssh keys in Kubernetes Secret. Storing confidential information in a Secret is safer and more flexible than using them directly in a container image.

You can create Secrets to store user credentials, database connection information, LDAP server credentials, mail server configuration, and so on. Each Secret must have a unique name, which is updated in the `custom-values.yaml` file. The Secrets are referenced through this file during deployment.

Create Secrets as shown in the following example. The credentials and namespace depicted in the examples are sample values. These values might change based on your requirement.

```
# Create Secret for WLS Admin Credentials
$ kubectl create secret generic amc-wls-credentials \
--from-literal=username=weblogic \
--from-literal=password='Welcome@123' \
-n amc-ns

##Create Secret for Database Connection
$ kubectl create secret generic amc-ds-credentials \
--from-literal=username=amc2 \
--from-literal=password='amc2' \
-n amc-ns

##Create Secret for mail session in Weblogic
$ kubectl create secret generic amc-mail-credentials \
--from-literal=username=first.last@example.com \
--from-literal=password='mailserverpassword' \
-n amc-ns

##Create Secret for connecting to LDAP server, LDAP Server integration from
WLS
$ kubectl create secret generic amc-ldap-credentials \
--from-literal=password='AMC2Admin$' \
-n amc-ns
```

The Secret names are updated in the `custom-values.yaml` file. See the sample custom-values.yaml file.

Create a ConfigMap

ConfigMaps are useful for storing and sharing non-sensitive, unencrypted configuration information. It binds configuration files, command-line arguments, environment variables, port numbers, and other configuration artifacts.

You can use a ConfigMap to override any environment-specific configuration.

The WLS for Containerized AMC is configured with default Demo Identity and Demo Trust keystore certificates. During deployment, if you want to overwrite the default configuration, use Custom Identity and Java Standard Trust method. For detailed steps, see Configure Keystores. However, you must have a valid Identity keystore to overwrite the default configuration.

Create a ConfigMap to add a keystore file as shown in the following example. Change the values based on your requirement.

```
# For Reusing existing JKS or Overriding default JKS Certificate
$ kubectl create configmap amc-keystore --from-file=<path>/
amckeystore.jks -n amc-ns
```

Provide the details of the ConfigMap and the file name in `custom-values.yaml` file. You can also create a Secret for the passwords associated with keystore and alias, and update the secret name in `custom-values.yaml` file.

Example Command:

```
$ kubectl create secret generic amc-jks-credentials \
--from-literal=password='keystore_password' \
-n amc-ns
```

Configure AMC Helm Chart

Ensure to download the AMC Helm Chart from GitHub. The Kubernetes manifest resources including the domain resource file as required by the WKO is packaged as Helm chart.

The Kubernetes manifest resources including the domain resource file as required by the WKO is packaged as a Helm chart. Ensure that you download the AMC Helm Chart from GitHub and then install it from the local path. The configuration parameters that are required to deploy the Containerized AMC are part of the default `values.yaml` included in the AMC Helm Chart.

Configurable Parameters

Default values are configured in the AMC Helm Chart.

You can download the Helm Chart from GitHub.

The table provides the default values that are configured in the AMC Helm Chart.

Table 3-1 AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
`namespace`		Kubernetes namespace that the AMC domain should be a part of.	amc	Mandatory	
`domain`		Domain name for the WebLogic Kubernetes Operator to manage.	`amc-domain`	Optional	
`image`	`repo`	Repository hosting the AMC container image.	None	Mandatory	
	`tag`	Tag for the AMC container image.	None	Mandatory	
	`pullPolicy`	Policy for image pull.	`IfNotPresent`	Optional	Always, IfNotPresent,Never
	`pullSecrets`	For image pull authentication with container registry hosting the AMC container image.	None	Optional	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
`weblogicCredentials`	`secretsName`	Name of the Kubernetes Secret containing the credentials for WebLogic server.	`<domain-name>-weblogic-credentials`	Depends on the option selected	Either of these scenarios: (a): User creates a Kubernetes secret containing `username` and `password` as keys in the same namespace beforehand. The name of the secret is passed against `secretsName` field (b): User provides plain text input against `username` and `password` fields. The Kubernetes secret is auto generated by Helm chart. Name of the secret can be optionally passed by the user. If not, the default value would be of the form `<domain-name>-weblogic-credentials`
	`username`	Username to be used for WebLogic admin server.	None	Depends on the option selected	
	`password`	Password to be used for WebLogic admin server.	None	Depends on the option selected	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
`weblogicSSLCertificateOverride`	`enabled`	Enable to override the default SSL certificate used by WebLogic server.	`false`	Optional	
	`configmapName`	The ConfigMap should be created from the `jks` file in the AMC namespace beforehand. The name should be updated against this field.		Mandatory, if enabled	
	`keystore: filename`	The name of the `jks` file used to create the configuration map.		Mandatory, if enabled	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`keystore:secretsName`	Name of the Kubernetes Secret containing the keystore password.	`<domain-name>-weblogic-keystore-credentials`	Depends on the option selected	Either of these scenarios: (a): User creates a Kubernetes secret containing `password` as key in the same namespace beforehand. The name of the secret is passed against `secretsName` field. (b): User provides plain text input against `password` field. The Kubernetes secret is auto generated by Helm chart. Name of the secret can be optionally passed by the user. If not, the default value would be of the form `<domain-name>-weblogic-keystore-credentials`.
	`keystore:password`	Plain text password for keystore.		Depends on the option selected	
	`keystorealias:alias`	Name of the alias in the `jks`.		Mandatory, if enabled	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`keystorealias:secretsName`	Name of the Kubernetes Secret containing the keystore alias password.	`<domain-name>-weblogic-keystorealias-credentials`	Depends on the option selected	Either of these Scenarios: (a): User creates a Kubernetes secret containing `password` as key in the same namespace beforehand. The name of the secret is passed against `secretsName` field. (b): User provides plain text input against `password` field. The Kubernetes secret is auto generated by Helm chart. Name of the secret can be optionally passed by the user. If not, the default value would be of the form `<domain-name>-weblogic-keystorealias-credentials`.
	`keystorealias:password`		Depends on the option selected		
`replicaCount`		Number of managed server pods to spin up. **Note:**Automatic scaling of managed servers is not yet supported.	2	Mandatory	2 to 10

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
restartVersion		An increment in this value forces domain restart and introspect on upgrade of deployment.	1	Optional	
extraEnv	JAVA_OPTIONS	Java Options(for memory, proxy, and so on) to be applied to WebLogic containers.	-Xmx2g-Xms512m -Dweblogic.StdoutDebugEnabled=false	Optional	
	USER_MEM_ARGS	User memory arguments to be supplied to WebLogic containers.	-XX:+UseContainerSupport	Optional	
resources	limits	Set the Kubernetes resource limit for CPU or Memory on the WebLogic containers.	3Gi for admin server memory and 5Gi for managed server memory	Optional	
	requests	Set the Kubernetes resource request for CPU or Memory on the WebLogic containers.	1Gi for both admin and managed server memory	Optional	
isVersionUpgrade		Set to true for AMC version upgrades.	false	Optional	
database	type	Type of database.	mysql	Mandatory	mysql, oracle
	name	Name of the database.	amc2	Mandatory	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`credentials.secretsName`	Name of Kubernetes Secret containing the credentials for database connection.	`<domain-name>-datasource-credentials`	Depends on the option selected	Either of these scenarios: (a): User creates a Kubernetes secret containing `username` and `password` as keys in the same namespace beforehand. The name of the secret is passed against `secretsName` field. (b): User provides plain text input against `username` and `password` fields. The Kubernetes secret is auto generated by Helm chart. Name of the secret can be optionally passed by the user. If not, the default value would be of the form `<domain-name>-datasource-credentials`.
	`credentials.username`	Username for the database connection.	None	Depends on the option selected	
	`credentials.password`	Password for the database connection.	None	Depends on the option selected	
	`host`	Hostname where the Database instance is running.	None	Mandatory	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`port`	Port number of the database instance is running on.	3306	Mandatory	
	`isOracle11`	To be enabled if using Oracle 11g as Database for AMC.	`false`	Optional	
	`use_cj_driver_mysql8`	Enables the usage of `com.mysql.cj.jdbc.Driver`.	`false`	Optional	To be enabled only if using MySQL8 and WLS 12.2.1.4
`nodePort`	`enabled`	Set to `true` to enable NodePort service.	`false`	Optional	
	`adminserver`	Port number exposing WebLogic admin server interface for access outside cluster using Kubernetes NodePort service.	Random port assignment	Optional	
	`managedserver`	Port number exposing AMC interface for access outside cluster using Kubernetes NodePort service.	Random port assignment	Optional	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
`loadBalancer`	`enabled`	Set to `true` to enable LoadBalancer service. **Note:**Once the LoadBalancer service is enabled for the Containerized AMC, it is persisted even on disabling the option. This is done on purpose to retain the assigned external IP address. Ensure that this service is not deleted inadvertently, else this will lead to AMC services being inaccessible. **Note:**If only LoadBalancer service is enabled in the `custom-values.yaml` file, the WebLogic admin server console will not be accessible. If there is a need to access the Weblogic admin server console, enable the NodePort service as well.	`false`	Optional	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`port`	Port number exposing AMC interface for access outside cluster using Kubernetes LoadBalancer service.	8002	Mandatory, if enabled	
	`annotations`	Annotations for the LoadBalancer service.	None	Optional	
`mailServer`	`enabled`	Set to `true` to enable Mail Server configuration.	`false`	Optional	
	`properties`	Settings for the mail server.	None	Mandatory, if enabled	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`credentials.secretsName`	Name of Kubernetes Secret containing the credentials for mailserver connection.	`<domain-name>-mailserver-credentials`	Depends on the option selected	Either of these scenarios: (a): User creates a Kubernetes secret containing `username` and `password` as keys in the same namespace beforehand. The name of the secret is passed against `secretsName` field. (b): User provides plain text input against `username` and `password` fields. The Kubernetes secret is auto generated by Helm chart. Name of the secret can be optionally passed by the user. If not, the default value would be of the form `<domain-name>-mailserver-credentials`.
	`credentials.username`	Username for the Mail Server connection.	None	Depends on the option selected	
	`credentials.password`	Password for the Mail Server connection.	None	Depends on the option selected	
`ldap`	`enabled`	Set to `true` to enable LDAP Server configuration.	`false`	Optional	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`credentials.secretsName`	Name of Kubernetes Secret containing the credential for LDAP connection.	`<domain-name>-ldap-credentials`	Depends on the option selected	Either of these scenarios: (a): User creates a Kubernetes secret containing `password` as key in the same namespace beforehand. The name of the secret is passed against `secretsName` field. (b): User provides plain text input against `password` field. The Kubernetes secret is auto generated by Helm chart. Name of the secret can be optionally passed by the user. If not, the default value would be of the form `<domain-name>-ldap-credentials`.
	`credentials.password`	Password for the LDAP Server connection.	None	Depends on the option selected	
	`host`	Host name of LDAP server.	None	Mandatory, if enabled	
	`port`	Port number for LDAP service.	389	Optional	
	`enableSSL`	Set to `true` to use SSL for LDAP.	`false`	Optional	
	`principal`	LDAP principal	None	Mandatory, if enabled	

Table 3-1 (Cont.) AMC Helm Chart Configurable Values

Configuration Keys	Sub-Keys (if any)	Description	Default (if any)	Mandatory or Optional	Accepted Range of Values
	`userbaseDN`	Base domain name (DN) for user.	None	Mandatory, if enabled	
	`groupbaseDN`	Base domain name (DN) for group.	None	Mandatory, if enabled	

Deploy Containerized Advanced Management Console

Ensure WKO is deployed successfully before proceeding with the AMC deployment.

Follow these steps to deploy the Containerized AMC:

1. Create the AMC container image using WIT and distribute the container image to all Kubernetes cluster nodes. See Create the AMC Container Image.
2. Clone the AMC repository from GitHub:

   ```
   git clone https://github.com/oracle/helm-charts.git
   ```

3. Create the AMC helm chart package:

   ```
   helm package helm-charts/java-amc/
   ```

   The helm package command will create the `java-amc-1.0.0.tgz` file.

4. Configure the deployment parameters. See Configurable Parameters. You can override the configurable parameters based on your requirement and save as `custom-values.yaml` file in the working directory.

   Find below the sample `custom-values.yaml` file for creating an AMC Domain in a Weblogic dynamic cluster with two managed servers. In this example, the AMC domain is deployed into a Kubernetes namespace `amc-ns`, on an OKE Kubernetes cluster in OCI. The WebLogic domain is integrated with an Oracle DBaas 19c on OCI, Mail Server, LDAP server, and uses an external Identity JKS file. Both the NodePort and LoadBalancer (OCI LoadBalancer) are enabled for the deployment, that is the WebLogic Admin Server can be accessed as a NodePort Service and the AMC WebUI can be accessed either through an OCI LoadBalancer or a NodePort Service.

   > **Note:**
   >
   > Ensure that the parameters in the `custom-values.yaml` file is indented properly to avoid execution error.

   ```
   namespace: amc-ns
   domain: amc-domain
   ```

```
replicaCount: 2
isVersionUpgrade: false
extraEnv:
  adminServer:
    JAVA_OPTIONS: "-Xmx2g -Xms512m"
    USER_MEM_ARGS: ""
  managedServer:
    JAVA_OPTIONS: "-Xmx4g -Xms1g -
Dhttps.proxyHost=proxy.example.com -Dhttps.proxyPort=80 -
Dhttp.proxyHost=proxy.example.com -Dhttp.proxyPort=80"
    USER_MEM_ARGS: ""
resources:
  adminServer:
    limits:
      memory: "2Gi"
    requests:
      memory: "1Gi"
  managedServer:
    limits:
      memory: "5Gi"
    requests:
      memory: "1Gi"

image:
  repo: amc2u20
  tag: latest
  pullPolicy: IfNotPresent


weblogicCredentials:
  secretsName: amc-wls-credentials

weblogicSSLCertificateOverride:
  enabled: true
  configmapName: amc-keystore
  keystore:
    filename: amckeystore.jks
    secretsName: amc-jks-credentials
  keystorealias:
    alias: amctest
    secretsName: amc-jks-credentials

database:
  type: oracle
  name: amc2
  credentials:
    secretsName: amc-ds-credentials
  host: dbaas.s4.example.com
  port: 1521

nodePort:
  enabled: true
  adminserver: 30720
  managedserver: 30721
```

```
loadBalancer:
  enabled: true
  port: 6503
  annotations:
    service.beta.kubernetes.io/oci-load-balancer-internal: "true"
    service.beta.kubernetes.io/oci-load-balancer-subnet1:
"ocid1.subnet.oc1.<country.region.xyz>"

mailServer:
  enabled: true
  properties:
    mail.smtp.host: mail.example.com
    mail.transport.protocol: smtp
    mail.smtp.auth: true
    mail.smtp.ssl.enable: true
    mail.debug: true
  credentials:
    secretsName: amc-mail-credentials

ldap:
  enabled: true
  credentials:
    secretsName: amc-ldap-credentials
  host: ldap.example.com
  port: 389
  enableSSL: false
  principal: cn=amc,dc=example,dc=com
  userbaseDN: ou=users,ou=amc,dc=example,dc=com
  groupbaseDN: ou=groups,ou=amc,dc=example,dc=com
```

5. Install the Containerized AMC container image that you have created, using the following commands:

   - To validate the configured parameters through a dry run:

     ```
     helm install <release-name> <path to amc helm package> \
      --values custom-values.yaml  -n <amc-namespace> --dry-run
     ```

     Example Command:

     ```
     $ helm install amc  ./java-amc-1.0.0.tgz \
     --values custom-values.yaml -n amc-ns  --dry-run
     ```

     > **Note:**
     >
     > Helm is not allowed connections to the Kubernetes API server during a dry run and thus cannot determine the presence of secrets. Ignore this warning **One or more secrets specified are not present in this namespace**, if any, during dry run.

- Install the chart, thus deploying the Containerized AMC:

```
helm install <release-name> <path to amc helm package> \
--values custom-values.yaml -n <amc-namespace>
```

Example Command:

```
$ helm install amc  ./java-amc-1.0.0.tgz \
--values custom-values.yaml -n amc-ns
```

The installation might take a while to complete. You can track the status of the installation using the following commands:

```
kubectl get events -n <amc-namespace> --watch
kubectl get all -n <amc-namespace> -o wide
```

Example Commands:

```
$ kubectl get events -n amc-ns --watch
$ kubectl get all -n amc-ns -o wide
```

You should see the details about Pods and Services, and the server status as `Running` after sometime.

Access the Containerized AMC

You can access Containerized AMC using the NodePort service or the LoadBalancer service or both, depending on the configuration in the `custom-values.yaml`.

After you deploy the AMC on Kubernetes environment and verify the status of deployment using the command `kubectl get all -n amc-ns`, the complete node and server information is displayed. The following is an example output:

Figure 3-1 Server Details

```
NAME                                        READY   STATUS    RESTARTS   AGE
pod/amc-domain-admin-server                 1/1     Running   1          6h37m
pod/amc-domain-managed-server1              1/1     Running   0          6h36m
pod/amc-domain-managed-server2              1/1     Running   0          6h36m
NAME                                            TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)              AGE
service/amc-domain-admin-server                 ClusterIP      None           <none>          7777/TCP             6h37m
service/amc-domain-admin-server-external        NodePort       192.0.2.155    <none>          7777:30720/TCP       6h37m
service/amc-domain-cluster-amc-cluster          ClusterIP      192.0.2.204    <none>          6504/TCP,7002/TCP    6h36m
service/amc-domain-lb-service                   LoadBalancer   192.0.2.214    203.0.113.10    6503:31587/TCP       6h39m
service/amc-domain-managed-server-external      NodePort       192.0.2.254    <none>          6502:30721/TCP       6h39m
service/amc-domain-managed-server1              ClusterIP      None           <none>          6501/TCP,7002/TCP    6h36m
service/amc-domain-managed-server2              ClusterIP      None           <none>          6500/TCP,7002/TCP    6h36m
```

The details of the servers and nodes, including internal and external ports are displayed.

Access Containerized AMC using LoadBalancer Service

In the LoadBalancer service type, note down the external IP and port number. You can use the values to create the URL to access Containerized AMC. Considering the example, the URL to access the Containerized AMC will be `https://203.0.113.10:6503/amcwebui/login.html`.

Access Containerized AMC using NodePort Service

In the NodePort service type, you can create URLs of the Admin Server and AMC Server using any of the Kubernetes node IP addresses/host name and the external port number mapped against respective NodePort services.

- Admin Server: `http://<kubernetes-node>:30720/console/login/LoginForm.jsp`
- AMC Server: `https://<kubernetes-node>:30721/amcwebui/login.html`

Run the following command to retrieve the Kubernetes node IP addresses:

```
kubectl get nodes -o wide
```

You can login using the credentials provided in the `custom-values.yaml` file.

> **Note:**
>
> - It is recommended to access the Containerized AMC using LoadBalancer service rather than NodePort service.
> - If only LoadBalancer service is enabled in the `custom-values.yaml` file, the WebLogic admin server console will not be accessible. If there is a need to access the admin server console, enable the NodePort service as well.

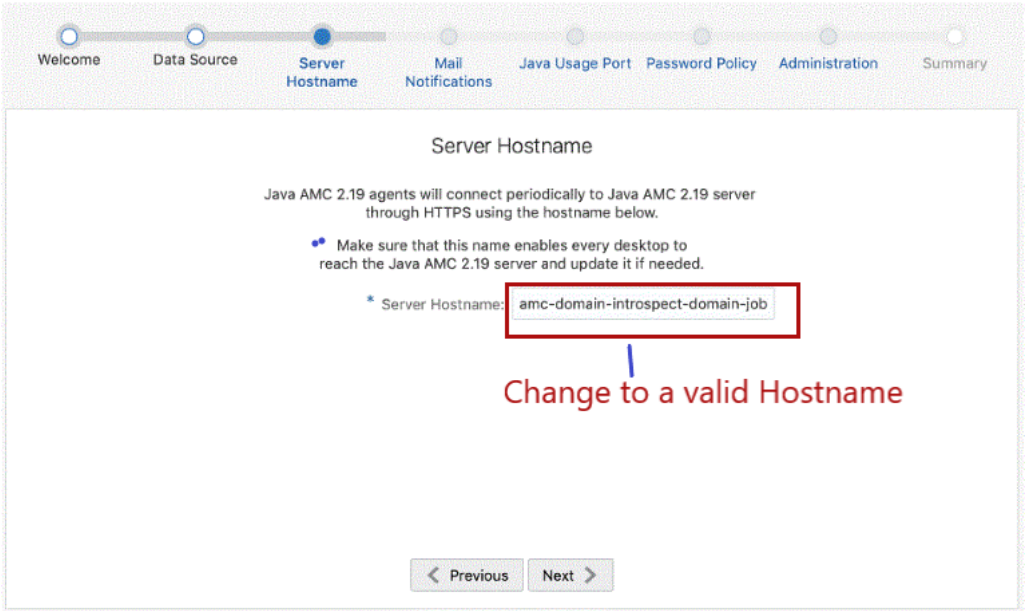
Initialize the Containerized AMC

During initialization, the host name (or host IP) has to be provided. This host name (or host IP) is to be used by all agents to communicate with the server.

By default, the AMC initialization wizard fetches the server host name from the CN (common name) of SSL certificate. However, the CN name might not be the valid server host name for Containerized AMC. Hence, during initialization, provide a valid server host name to access Containerized AMC. Also, ensure that you don't have to change the host name (or host IP) once you specify.

For example, as shown in the following screen shot, the host name is fetched from SSL certificate by default. This is not a valid host name. You need to provide a valid host name or IP for the Containerized AMC to communicate with the server.

Figure 3-2 Initializing Containerized AMC



4

Manage the AMC Deployment

The upgrade and maintenance of WKO and AMC can be achieved seamlessly using the Containerized AMC.

You can also migrate non-containerized AMC to Containerized AMC with minimal down time and continuity of services.

Migrate Non-containerized AMC to Containerized AMC

Migration of non-containerized AMC to Containerized AMC involves reusing of existing on-premise AMC database connection with Containerized AMC. In other words, the database connecting to on-premise (non-containerized) AMC is disconnected and connected to Containerized AMC. However, the WebLogic domain configuration of on-premise AMC is not migrated to Containerized AMC.

While migrating from on-premise AMC to Containerized AMC, you can either migrate to same version of AMC or a higher version. However, you cannot migrate to a lower version of AMC.

> **Note:**
>
> When upgrading the Advanced Management Console to version 2.17 or later, and if the **Agent Auto Update** option is enabled, you must manually replace the signing certificate in the agent machine. For detailed steps, see Automatic Update of Advanced Management Console Agent.

When migrating from on-premise AMC to Containerized AMC, you can plan to either:

- Reuse On-premise AMC Server Address: The AMC server address is the host name and port number you set during AMC initialization. This address is used by all agents connected to the on-premise AMC. You can retain the same AMC server address for the migrated Containerized AMC. However, to reuse the AMC server address, the server address has to be front ended by an external load balancer.

  OR

- Change AMC Server Address: Plan the migration by changing the AMC server address.

Reuse On-premise AMC Server Address

Before you begin the migration, ensure that the on-premise AMC is front ended through an external load balancer application.

1. Stop or un-deploy the running AMC application in your on-premise WebLogic cluster or server. You can also stop the WebLogic server or cluster if its used only for AMC deployment.
2. Deploy Containerized AMC using the database connection properties that were used by the on-premise AMC.

3. Identify the service end points (that is, server host name or IP address, and port number) of your Containerized AMC deployment. See Access the Containerized AMC.
4. Replace the external load balancer backend with Containerized AMC service end points and restart the load balancer application.
5. If you want to upgrade to higher version of AMC, see Upgrade Containerized AMC deployment.

Containerized AMC

```
------------------------------------------------
 HA Proxy front ending on-premise AMC
------------------------------------------------
frontend AMCServer_001
bind *:8189 ssl crt /etc/haproxy/cert/server.pem
reqadd X-Forwarded-Proto:\ https
default_backend AMCServer_001_Back

backend AMCServer_001_Back
cookie JSESSIONID prefix nocache
server AMCServer_001_MS1 <onprem-managed-server1:port> cookie amc-
cluster  ssl verify none
server AMCServer_001_MS2 <onprem-managed-server2:port> cookie amc-
cluster  ssl verify none

------------------------------------------------------
 HA Proxy front ending Containerized AMC
------------------------------------------------------

frontend AMCServer_001
bind *:8189 ssl crt /etc/haproxy/cert/server.pem
reqadd X-Forwarded-Proto:\ https
default_backend AMCServer_001_Back

backend AMCServer_001_Back
cookie JSESSIONID prefix nocache
server AMCServer_001_MS1 <cont-amc-server-host:port> cookie amc-
cluster  ssl verify none
```

Change AMC Server Address

Migrate on-premise AMC to Containerized AMC by providing different AMC server address (host name or IP address, and port number).

1. Stop or un-deploy the running AMC application in your on-premise WebLogic cluster or server. You can also stop the WebLogic server or cluster if its used only for AMC deployment.
2. Deploy Containerized AMC using the database connection properties that were used by on-premise AMC.
3. Identify the service end points (that is, server host name or IP, and port number) of your Containerized AMC deployment.

4. Connect to the AMC database from any database client application and update the AMC configuration table using the following SQL statements and commit the changes:

```
update config set configvalue = '<cont-amc-service-host>' where configkey
= 'hostname';
update config set configvalue = '<cont-amc-service-port>' where configkey
= 'amc_port';
```

Example

```
update amc2.config set configvalue = '203.0.113.10' where configkey =
'hostname';
update amc2.config set configvalue = '6503' where configkey = 'amc_port'
```

5. On every agent machine, modify the properties `server.name` and `server.port` in `AMCServer.properties` file present in the `conf` directory of `AMC_Agent` (`%AMC_DIR%\conf\AMCUser.properties`). To ensure continuity of managing agents through the new deployment, update the properties for all agents.

```
server.name: <cont-amc-service-host>
server.port: <cont-amc-service-port>
server.protocol: https
agentId: 3
authCookieName: amc_auth
authCookieValue: c78b2652-1d86-41b6-a673-c4b309d9ec06
```

6. Restart all agents after this change:
   - Windows: Restart the AMC system service through the task manager
   - macOS and Linux: Execute - `sudo bash ${AMC_DIR}/bin/AMCAgent.sh -restart`

Upgrade the Deployments

Run the `upgrade` command to fetch the latest version of WKO, AMC, and Helm chart values.

Upgrade WKO Deployment

Use the following command to upgrade WKO deployment:

```
helm upgrade <release-name> weblogic-operator/weblogic-operator \
--set "javaLoggingLevel=FINE" -n <operator-namespace> \
--reuse-values
```

Example Command:

```
$ helm upgrade wko weblogic-operator/weblogic-operator \
--set "javaLoggingLevel=FINE" -n wko-ns \
--reuse-values
```

For detailed steps to upgrade WKO deployment, see Upgrade the operator.

Upgrade Containerized AMC deployment

You can upgrade the Containerized AMC in any of these scenarios:

- A newer version of AMC container image is available
- If you need to increase or decrease the number of pods running (Scale-in or Scale-out)
- Changes in WebLogic domain configuration
- Switch to a different database, mail server, or LDAP server

To update the configuration values, edit the `custom-values.yaml` file.

Run the following command to upgrade the Containerized AMC deployment:

```
helm upgrade <release_name> <path to amc helm package> --values custom-
values.yaml --wait \
-n <amc-namespace> --reuse-values
```

Where:

- `custom-values.yaml` are the custom values that overwrite the standard values in the local or remote repository.
- `release_name` is the name of the Containerized AMC deployment.
- `path to amc helm package` is the local helm package containing the standard AMC values.
- `amc-namespace` is the AMC version that is being upgraded.
- `wait` pauses the output until all pods and services are in the ready state.
- `reuse-values` uses the last release values and merge overrides, if any, while upgrading.

Example Command:

```
$ helm upgrade amc ./java-amc-1.0.0.tgz --values custom-values.yaml --
wait \
   -n amc-ns   --reuse-values
```

Uninstall Deployments

Uninstall will remove all resources that you have created during deployment. Ensure to uninstall the Containerized AMC before you uninstall WKO.

Use the following commands in the specified order to uninstall AMC and WKO deployments:

Uninstall Containerized AMC Deployment

Delete the ConfigMap:

```
kubectl delete configmap <configmap-name> -n <amc-namespace>
```

Delete the secrets created during AMC deployment:

```
kubectl delete secret <secret-name> -n <amc-namespace>
```

Remove AMC namespace:

```
kubectl delete namespace <amc-namespace>
```

Remove AMC deployment:

```
helm uninstall <release_name> -n <amc-namespace>
```

Uninstall WKO Deployment

Delete the Custom Resource Definition object created by the operator:

```
kubectl delete customresourcedefinition domains.weblogic.oracle -n <operator-
namespace>
```

Remove the operator's namespace:

```
kubectl delete namespace <operator-namespace>
```

Remove the operator:

```
helm unistall <release-name> -n <operator-namespace>
```

Here is an example bash script that you can use to uninstall AMC and WKO deployments completely.

```
#!/bin/bash

kubectl delete configmap amc-keystore -n amc-ns
kubectl delete secret  amc-wls-credentials -n amc-ns
kubectl delete secret  amc-ds-credentials -n amc-ns
kubectl delete secret  amc-mail-credentials -n amc-ns
kubectl delete secret  amc-ldap-credentials -n amc-ns
kubectl delete secret  amc-jks-credentials -n amc-ns

helm uninstall amc -n amc-ns
helm uninstall wko -n wko-ns

kubectl delete customresourcedefinition domains.weblogic.oracle -n wko-ns


kubectl delete namespace amc-ns
kubectl delete namespace wko-ns
```

5

Administer and Maintain the Containerized AMC

Follow the best practices mentioned in this topic to efficiently maintain the Containerized AMC.

Security Guidelines

Here are some security recommendations for your Containerized AMC deployments.

- **Secrets**: When providing passwords to Containerized AMC deployment, it is recommended to use Kubernetes Secrets instead of plain text.
- **UpToDate images**: When a new version of AMC or WebLogic container image is available, download the latest version of `Java_AMC_<version>.ear` file and WebLogic container image, and create a new container image. Applying the latest Critical Patch Update (CPU) or Security Alert is required, older versions are not updated with the latest security patches. Upgrade the currently running Containerized AMC with the newly built image.
- **Firewall**: Follow the industry-standard practices, such as configuring firewall rules to restrict the network traffic and prevent accidental or malicious threats.

Containerized AMC Maintenance

The WLS and AMC updates are aligned with the Java CPU release cycles. Applying the latest CPU or Security Alert is required, older versions are not updated with the latest security patches. Ensure that the container images are up to date with these versions. When you update the container images, ensure that you upgrade your Containerized AMC deployment.

Database Administration

Any change in the database configuration will require a domain restart. If there are corresponding changes to the database section of `values.yaml`, an upgrade of your Containerized AMC will internally trigger a domain restart.

6
Troubleshooting

Some known issues and ways to troubleshoot that you need to be aware of.

Error Messages

- WKO error message:

```
unable to build kubernetes objects from release manifest: unable to
recognize "": no matches for  kind "Domain" in
      version "weblogic.oracle/v8"
```

  If you encounter this error, check if the WKO is running and AMC namespace is correctly assigned to it.

- Dry run error message:

```
 execution error at(amc/templates/validation.yaml:19:49): One or more
secrets specified are not present in this
      namespace.
```

  Ignore this warning if you are executing the dry run.

- Error due to formatting issues in the deployment configuration file:

```
Error: execution error at (java-amc/templates/validation.yaml:5:51):
Docker image repository or tag not
      configured
```

  If you encounter this error, check if the `custom-values.yaml` is properly indented or formatted.

Known Issues

Here are some known issues:

- While initializing the Containerized AMC, the default common name (CN) is populated from the SSL certificate. Therefore, you have to use a valid host name while initializing. See Initialize the Containerized AMC.
- If only the LoadBalancer service is enabled in the `custom-values.yaml` file, the WebLogic admin server console will not be accessible. If there is a need to access the admin server console, enable the NodePort service as well.
- If the server hostname provided during initialization becomes inaccessible later, then AMC Agents stop communicating with the server . Also the AMC agent bundle download will fail.

Best Practices

Here are some tips or best practices that you can use to avoid communication (or access) issues:

- If the Kubernetes is setup in a private network and it needs to access a public network, for example to pull the WKO container image, then configure the correct proxy in the container engine.
- If there are network accessibility issues while accessing a server or a database, ensure that the firewall settings in the Kubernetes cluster is configured correctly.
- The Advanced Management Console requires access to Java Security Baselines and Java releases servers. Internet access is also required to introspect JNLP files if they're outside of the corporate network. Ensure that the required proxy is configured for the managed server pods in the `custom-values.yaml` file.
- Once the LoadBalancer service is enabled for the Containerized AMC, it is persisted even on disabling the option. This is done on purpose to retain the assigned external IP address. Ensure that this service is not deleted inadvertently, else this will lead to AMC services being inaccessible.
- The operations such as container image building, distribution, deployment, upgrade, and so on are performed by the operating system user having adequate permission. If required, prefix the commands with `sudo` or `sudo -E` as applicable in your environment.
- Create a working directory for storing all the artifacts such as helm package, `custom-values.yaml`, and JKS files. Run the commands such as helm install or update, kubectl create or delete from this work directory.
- Before you start Containerized AMC deployment, you can script the following operations as per you requirement and run the commands from a bash terminal:
  - Namespace creation
  - WKO installation
  - AMC secretes creation
  - ConfigMap creation
  - Helm install dry run command

  This will ensure that all prerequisites are met before proceeding with the deployment.
- The default pod resource requests and limits as configured in the `values.yaml` should suffice for normal functioning. However, if the pods get terminated silently without any apparent reasons, recheck this configuration.