

IBM Agile Lifecycle Manager
Version 2.1.0

Installation Guide and Configuration
30 March 2020



Note

Before using this information and the product it supports, read the information in [“Notices” on page 25.](#)

This edition applies to Version 2.1.0. of IBM® Agile Lifecycle Manager (product number 5737-E91) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2019, 2020.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface.....	V
New features overview.....	V
Chapter 1. Planning.....	1
Hardware requirements.....	1
Software prerequisites.....	1
Common Services deployment overview.....	2
Chapter 2. Installing and configuring.....	3
Before you install.....	3
Installing IBM Cloud Platform Common Services.....	3
Preparing to install services.....	3
Configuring authentication to Docker CLI.....	4
Installing services.....	4
Installing IBM Cloud Pak (cloudctl).....	6
Installing the Kubernetes CLI (kubectl).....	7
Installing the Helm CLI (helm).....	7
Uninstalling services.....	9
Installing IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform.....	9
Integrating IBM Agile Lifecycle Manager with an existing external directory service.....	12
Example 1: Configuring IBM Agile Lifecycle Manager to integrate with the external LDAP server using simple authentication.....	16
Example 2: Configuring IBM Agile Lifecycle Manager to integrate with the external LDAP server using ldapbind authentication.....	19
Example 3: Using a custom LDAP user to authenticate to IBM Agile Lifecycle Manager using a predefined group and role.....	20
Example 4: Creating a custom LDAP group and associating it to a predefined role.....	21
Example 5: Create a new IBM Agile Lifecycle Manager role and associate it to an OpenLDAP group.....	22
Displaying IBM Agile Lifecycle Manager logs in Kibana.....	23
Uninstalling IBM Agile Lifecycle Manager.....	23
Notices.....	25
Trademarks.....	26

Preface

This PDF document contains topics from the Knowledge Center in a printable format.

New features overview

IBM Agile Lifecycle Manager provides users with a toolkit to manage the lifecycle of both virtual and physical network services. This includes the design, test, deployment, monitoring and healing of services. IBM Agile Lifecycle Manager lets you design and integrate external resources into virtual production environments and then automate the management of end-to-end lifecycle processes.

A number of new features and enhancements have been introduced in IBM Agile Lifecycle Manager Version 2.1.0.

- **Carrier Grade Resource Manager**

A new Carrier Grade Resource Manager is provided with IBM Agile Lifecycle Manager. The new Resource Manager requires VIM drivers to integrate with virtual infrastructure and IBM Agile Lifecycle Manager Drivers to complete transitions and operations with different scripting mechanisms. This Resource Manager supports the following external drivers and functionality:

- **Openstack VIM Driver**

A resource manager driver to install and uninstall resource infrastructure modeled in HEAT.

- **Ansible Lifecycle Driver**

A resource manager driver to execute IBM Agile Lifecycle Manager operations using Ansible playbooks.

- **SOL003 Lifecycle Driver**

A resource manager driver to integrate directly with SOL003 compliant VNFMs.

- **SOL003 externalized grant logic**

Enhancements to allow SOL003 grant requests to be handled by external logic.

- **LMCTL enhancement for Resource Manager**

LMCTL tool enhanced to leverage IBM Agile Lifecycle Manager features.

- **Designer UI improvements**

Usability improvements to the IBM Agile Lifecycle Manager UI.

- **Volatile properties and reconfigure lifecycle**

Properties marked as volatile can be changed without causing a reinstall of assembly components during upgrade. A reconfigure transition is called to effect the change without leaving 'Active' state.

- **IBM Agile Lifecycle Manager microservice anti-affinity**

Clustered IBM Agile Lifecycle Manager microservices are deployed in different compute nodes where possible.

- **Authentication and authorization**

IBM Agile Lifecycle Manager can be configured to perform an LDAP bind using the authenticating user. Support for creating customized roles.

LDAP Directory Service groups can be associated with predefined roles.

- **Additional lifecycle state and transitions to split the existing install into infrastructure and application transitions (Create & Install)**

Chapter 1. Planning

This section helps you to plan your installation and use of IBM Agile Lifecycle Manager by listing the minimum software and hardware requirements for the Red Hat OpenShift Container Platform version of IBM Agile Lifecycle Manager.

Hardware requirements

This section lists the minimum hardware requirements for a deployment of IBM Agile Lifecycle Manager.

Your minimum hardware requirements are determined by the needs of the components of your specific solution. The requirements listed here focus on what you need to deploy each IBM Agile Lifecycle Manager instance.

On Red Hat OpenShift Container Platform, IBM Agile Lifecycle Manager requires a minimum of three worker nodes with a combined specification as shown in the following table.

<i>Table 1. IBM Agile Lifecycle Manager hardware requirements (spread across a minimum of three worker nodes)</i>	
Requirement	Setting
CPU	72 cores
Memory	96 GB of RAM
Disk space	1.5 TB

In addition to this, Common Services requires the following resources on Red Hat OpenShift Container Platform compute or worker nodes.

<i>Table 2. Common Services hardware requirements</i>	
Requirement	Setting
CPU	8 cores
Memory	32 GB of RAM
Disk space	100 GB

Software prerequisites

This section lists the software prerequisites for a deployment of IBM Agile Lifecycle Manager.

IBM Agile Lifecycle Manager has the following software prerequisites.

<i>Table 3. IBM Agile Lifecycle Manager software prerequisites</i>	
Prerequisites	
Red Hat OpenShift Container Platform 4.3	

<i>Table 3. IBM Agile Lifecycle Manager software prerequisites (continued)</i>
Prerequisites
Docker for Red Hat Enterprise Linux Version 1.12 or later (Required only on the boot node of Red Hat OpenShift Container Platform where the install is carried out from.)
openssl 1.1.1c or later

Note: The IBM Agile Lifecycle Manager User Interface is only supported on the Google Chrome browser.

Common Services deployment overview

Common Services requires the following resources on Red Hat OpenShift Container Platform compute or worker nodes: 8 Core, 32 GB RAM 100GB Storage.

Docker

Ensure Docker is installed by using the following commands:

```
yum install docker
systemctl start docker
```

Note: All commands should be performed as the root user.

Chapter 2. Installing and configuring

IBM Agile Lifecycle Manager is distributed as a self-contained package delivered as a Helm chart. To install IBM Agile Lifecycle Manager, you complete the required pre-installation tasks, then perform the installation.

Before you install

Before installing IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform, you must install IBM Common Services.

Note: You cannot upgrade an existing deployment of IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform. Instead, you must uninstall it as described in [“Uninstalling IBM Agile Lifecycle Manager”](#) on page 23, and then perform a fresh installation.

Installing IBM Cloud Platform Common Services

Follow these instructions to install Common Services 3.2.4 before you install IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform.

Preparing to install services

Before you install, review the following installation requirements.

Red Hat OpenShift Container Platform node sizing requirement

- Red Hat OpenShift Container Platform compute or worker nodes: 8 Core | 32 GB RAM

Note: The required resources are the allocable resources on the Red Hat OpenShift Container Platform nodes. If you want to install multiple Cloud Paks, you must add up all of the additional required resources.

- Storage requirements:
 - For offline installation, the Red Hat OpenShift Container Platform image registry requires at least 100 GB.
 - The management services MongoDB and logging each require 20 GB through the storage class.

Version of Red Hat OpenShift Container Platform

You must have a supported version of Red Hat OpenShift Container Platform (4.3), including the registry and storage services, which are installed and working in your cluster.

Healthy Red Hat OpenShift Container Platform status

To ensure that the Red Hat OpenShift Container Platform cluster is set up correctly, access the web console.

The web console URL can be found by running following command:

```
oc -n openshift-console get route console
```

Example output:

NAME	HOST/PORT	PATH	SERVICES	PORT
TERMINATION	WILDCARD			
console	console-openshift-console.apps.new-coral.purple-chesterfield.com			console
https	reencrypt/Redirect	None		

The console URL in this example is `https://console-openshift-console.apps.new-coral.purple-chesterfield.com`. Open the URL in your browser and check the result.

Available storage class

You must have a pre-configured storage class in the Red Hat OpenShift Container Platform.

The storage class can be found by running following command:

```
oc get storageclass
```

Exposed image registry (Offline only)

You must enable the Red Hat OpenShift Container Platform image registry route. You can use the commands in the following example to expose the docker-registry service:

```
oc patch configs.imageregistry.operator.openshift.io/cluster --patch '{"spec": {"defaultRoute": true}}' --type=merge
```

If you want to see details about the image-registry resource and its route, use the following commands:

```
oc get routes -n openshift-image-registry
```

For more information about exposing an Red Hat OpenShift Container Platform image registry route, see [Red Hat OpenShift Container Platform 4.3 route documentation](#).

Boot node

The boot node should be Red Hat Enterprise Linux (RHEL).

Configuring authentication to Docker CLI

Optional: If you want to access the private image registry from outside your cluster, set up authentication from your computer to the cluster.

Required user type or access level: Cluster administrator or team administrator

Before you begin

You must install Docker on your computer. For more information, see [Install Docker](#).

Steps

1. Configure docker to trust the image registry, run the following commands:

```
DOCKER_REGISTRY=$(oc get route -n openshift-image-registry default-route -o jsonpath='{.spec.hosts}')
mkdir -p /etc/docker/certs.d/$DOCKER_REGISTRY
openssl s_client -showcerts -servername $DOCKER_REGISTRY -connect $DOCKER_REGISTRY:443 2>/dev/null | openssl x509 -inform pem > /etc/docker/certs.d/$DOCKER_REGISTRY/ca.crt
```

2. Log in to Docker:

```
docker login -u kubeadmin -p $(oc whoami -t) $DOCKER_REGISTRY
```

Installing services

Prerequisites

Docker is installed and configured. For more information, see [Configuring authentication for the Docker CLI](#).

Review the information in [Preparing for installation](#).

About this task

Note: IBM Cloud Platform Common Services should not be installed on a master node.

Before you start installing services, you need to get the `icp-inception` image, the `config.yaml` file, and the `kubeconfig` file. Follow these steps to obtain the image and files and install the required services.


The `config.yaml` and `kubeconfig` file must be located under the same directory. For example, if the directory is called `cluster`, the directory structure is:

```
/path/to/cluster
|-- config.yaml
|-- kubeconfig
```

Get the icp-inception image

Ensure that you have the offline `icp-inception` image.

Offline package

1. Download the installation file or image for the type of nodes in your cluster from the [IBM Passport Advantage](#)®  website.
 - Download the `<CS-package-name.tar.gz>` file.
2. Extract the images and load them into Docker. Extracting the images can take a few minutes, during which time output is not displayed.
 - Run the following command:

```
tar xf <CS-package-name.tar.gz> -O | sudo docker load
```

Get the config.yaml file

1. Run the `mkdir cluster` command.
2. Extract the `config.yaml` file from the Common Services `icp-inception` image with the following commands:

Note: This is a single line command.

```
docker run --rm -v $(pwd)/cluster:/data -e LICENSE=accept ibmcom/icp-inception-amd64:3.2.4
cp cluster/config.yaml /data/config.yaml
```

3. Update the following parameters, in the extracted `config.yaml` file, to be similar to the offline file example:

- **master**
- **proxy**
- **management**
- **storage_class**
- **default_admin_user**
- **default_admin_password**
- **password_rules**

A full offline `config.yaml` example:

```
cluster_nodes:
  master:
    - worker1.ibm.com
  proxy:
    - worker1.ibm.com
  management:
    - worker1.ibm.com
```

```

storage_class: rook-ceph-cephfs-internal

roks_enabled: false
roks_url: <roks_url>
roks_user_prefix: "IAM#"

default_admin_password: admin
password_rules:
  - '(.*)'

## You can disable following services if they are not needed
management_services:
  # Common services
  iam-policy-controller: enabled
  metering: enabled
  licensing: disabled
  monitoring: enabled
  nginx-ingress: enabled
  common-web-ui: enabled
  catalog-ui: enabled
  mcm-kui: enabled
  logging: disabled
  audit-logging: disabled
  system-healthcheck-service: disabled
  multitenant-enforcement: disabled

```

Create the kubeconfig file

Run the following command to log in:

```

oc login -s $(grep api /root/wait_for_install_complete_output | cut -f10 -d' ' ) -u kubeadmin -p $(cat /root/auth/kubeadmin-password)

```

You can get the kubeconfig file from inside your Red Hat OpenShift Container Platform installation directory as follows:

```

cp /root/auth/kubeconfig cluster/kubeconfig

```

Run the common services installer

Complete the following step to install Common Services:

1. Start the installation.

- The config.yaml and kubeconfig files should be under the same directory, for example, cluster:

```

cd cluster

```

```

docker run -t --net=host -e LICENSE=accept -v $(pwd):/installer/cluster:z -v /var/run:/var/run:z -v /etc/docker:/etc/docker:z --security-opt label:disable ibmcom/icp-inception-amd64:3.2.4 addon

```

Installing IBM Cloud Pak (cloudctl)

You can use IBM Cloud Pak CLI (cloudctl) to view information about your cluster, manage your cluster, install Helm charts and workloads, and more.

After you install IBM Cloud Pak CLI (cloudctl), you can install another supported CLI on Windows, Linux, or macOS.

To download and install cloudctl, complete the following steps:

1. Get the service hostnames. Use the management-ingress service hostname in the command to download the installation file.

```

oc get route -n kube-system

```

Following is a sample output:

NAME	HOST/PORT	PATH	SERVICES	PORT
TERMINATION	WILDCARD			
icp-console	icp-console.apps.an.os.example.abc.com		management-ingress	<all>
passthrough/Redirect	None			
icp-proxy	icp-proxy.apps.an.os.example.abc.com		nginx-ingress	https
passthrough/Redirect	None			

2. Download the installation file.

```
curl -kLo cloudctl -X GET $(oc get routes -n kube-system icp-console -o jsonpath='{.spec.host}')
```

3. Move to required location and give required permissions.

```
chmod +x cloudctl
mv cloudctl /usr/local/bin
```

4. Confirm that cloudctl is installed:

```
cloudctl --help
```

The cloudctl usage is displayed.

5. Confirm login to your cluster using cloudctl:

```
cloudctl login -a $(oc get routes -n kube-system icp-console -o jsonpath='{.spec.host}') -u admin -p admin -n default
```

Installing the Kubernetes CLI (kubectl)

To access your cluster by using the command line interface (CLI), you must install and configure `kubectl`, the Kubernetes command line tool.

To download and install `kubectl`, complete the following steps:

1. Get the service hostnames. Use the `management-ingress` service hostname in the command to download the installation file.

```
oc get route icp-console -n kube-system
```

Following is a sample output:

NAME	HOST/PORT	PATH	SERVICES	PORT
TERMINATION	WILDCARD			
icp-console	icp-console.apps.an.os.example.abc.com		management-ingress	<all>
passthrough/Redirect	None			

2. Download the installation file.

```
curl -kLo kubectl-linux-amd64-v1.13.11 https://icp-console.apps.an.os.example.abc.com:443/api/cli/kubectl-linux-amd64
```

3. Move to required location and give it required permissions.

```
mv kubectl-linux-amd64-v1.13.11 /usr/local/bin/kubectl
chmod +x /usr/local/bin/kubectl
```

Installing the Helm CLI (helm)

You can use the Helm command line interface (CLI) to manage releases in your cluster.

For more information about Helm, see [Helm docs in GitHub](#).

When you use role-based access control, you must install a specific version of the Helm CLI client and provide certificates that contain the access token for a specific account.

Important: After you configure a connection, you must add the `--tls` option to Helm commands that access the server through Tiller.

Before you set up the Helm CLI, you must complete the following steps:

- Install the Kubernetes command line tool, `kubectl`, and configure access to your cluster.
- Install `cloudctl`, and log in to your cluster.
- Obtain access to the boot node and the cluster administrator account, or request that someone with that access level create your certificate. If you cannot access the cluster administrator account, you need an account that is assigned to the administrator role for a team and can access the `kube-system` namespace.

Downloading the installation file by using curl commands

You can complete the following steps to download the installation file:

1. Get the service hostnames. Use the `management-ingress` service hostname in the command to download the installation file.

```
oc get route -n kube-system
```

Following is a sample output:

NAME	HOST/PORT	PATH	SERVICES	PORT
TERMINATION	WILDCARD			
icp-console	icp-console.apps.an.os.example.abc.com		management-ingress	<all>
passthrough/Redirect	None			
icp-proxy	icp-proxy.apps.an.os.example.abc.com		nginx-ingress	https
passthrough/Redirect	None			

1. Download the installation file.

- For Linux x86_64, run the following command:

```
curl -kLo helm-linux-amd64-v2.12.3.tar.gz https://icp-console.apps.an.os.example.abc.com:443/api/cli/helm-linux-amd64.tar.gz
```

2. After you run the curl command, make a `helm-unpacked` directory and unpack the installation file into that directory with the following commands:

```
mkdir helm-unpacked
tar -xvzf ./<path_to_installer> -C helm-unpacked
```

3. Change the file to an executable, then move the file to your directory:

For Linux and macOS, run the following commands to change and move the file:

```
chmod 755 ./helm-unpacked/<unpacked_dir>/helm
sudo mv ./helm-unpacked/<unpacked_dir>/helm /usr/local/bin/helm
```

4. Delete the installer and extra unpacked archives:

```
rm -rf ./helm-unpacked ./<path_to_installer>
```

Verifying the installation

1. If you are using Helm 2.12.3, you must set `HELM_HOME`:

```
export HELM_HOME=~/.helm
```

2. Initialize your Helm CLI. **Important:** Do not use the `--upgrade` flag with the `helm init` command. Adding the `--upgrade` flag replaces the server version of Helm Tiller that is automatically installed.

- For environments with Internet access, run the following command:

```
helm init --client-only
```

- For air gap environments, run the following command:

```
helm init --client-only --skip-refresh
```

3. Verify that the Helm CLI is initialized. Run the following command:

```
helm version --tls
```

The output resembles the following content:

```
Client: &version.Version{SemVer:"v2.12.3",
GitCommit:"20adb27c7c5868466912eebdf6664e7390ebe710", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.12.3+icp",
GitCommit:"843201eceab24e7102ebb87cb00d82bc973d84a7", GitTreeState:"clean"}
```

Uninstalling services

You can uninstall services by removing all deployed Helm charts.

Offline uninstallation

1. Log in to the boot node as a user with root permissions.
2. Enter to the `cluster` directory within your installation directory:

```
cd /<installation_directory>/cluster
```

3. Uninstall by running the `uninstall-addon` command:

For Red Hat OpenShift Container Platform on Linux:

```
sudo docker run -t --net=host -e LICENSE=accept -v $(pwd):/installer/cluster:z -v /var/
run:/var/run:z -v /etc/docker:/etc/docker:z --security-opt label:disable ibmcom/icp-
inception-amd64:3.2.4 uninstall-addon
```

Installing IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform

This topic describes how to perform prerequisite tasks to prepare the installation environment, how to load the IBM Agile Lifecycle Manager archive into Red Hat OpenShift Container Platform, how to edit the installation configuration file, and then how to install IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform from the command line.

Before you begin

Before you perform this task make sure you have met the following prerequisites:

- You must have downloaded the eAssembly.
- Ensure you have completed all [prerequisite tasks](#).

Upgrade Note: You cannot upgrade an existing deployment of IBM Agile Lifecycle Manager on Red Hat OpenShift Container Platform. Instead, you must uninstall it as described in [“Uninstalling IBM Agile Lifecycle Manager”](#) on [page 23](#), and then perform a fresh installation.

Ensure that your Red Hat OpenShift Container Platform cluster has Internet access. You must enable IPsec in the cluster to ensure all data in motion is encrypted.

If you are installing into a **non-default namespace**, ensure that the user you are deploying with has adequate privileges to perform the installation.

About this task

You install IBM Agile Lifecycle Manager 2.1.0 as team administrator. You only deploy a single instance per namespace.

The IBM Agile Lifecycle Manager core and StatefulSet components are deployed during the installation process.

Security

Vault is deployed for secure storage of secrets and configuration, and some defaults are generated.

Optionally, **for testing purposes only**, you can install OpenLDAP to store user credentials and create your own secrets for certificates, credentials and LDAP configuration. To do so, you need the following utilities:

- openssl
- keytool

OpenLDAP limitation: OpenLDAP storage is not persistent, which means that any update or addition to existing roles and users are lost if OpenLDAP is restarted. Therefore **OpenLDAP should only be used for testing or proof-of-concept purposes**, and not in production environments.

You run the following three (included) prerequisite scripts before installation.

createSecurityClusterPrereqs.sh

Creates the PodSecurityPolicy and ClusterRole for all releases of the chart.

createSecurityNamespacePrereqs.sh

Creates the ClusterRoleBinding for the namespace.

createStorageVolumes.sh

Creates the required storage volumes for a single deployment of the chart.

Procedure

Prepare the IBM Agile Lifecycle Manager environment

Restriction: You **must** run the prerequisite scripts as cluster administrator.

1. **On each worker node**, set the Cassandra and Elasticsearch **vm.max_map_count** kernel parameter to a value of at least 262144.
 - a) Add the following line to the `/etc/sysctl.conf` configuration file:

```
vm.max_map_count=262144
```

- b) Reload the configuration.

```
sudo sysctl -p
```

2. Create a local folder on a boot node for the IBM Agile Lifecycle Manager images.

```
mkdir ./alm_images
cd ./alm_images
```

Prepare and load installation assets

3. Log in to the Red Hat OpenShift Container Platform cluster:

```
cloudctl login -a $(oc get routes -n kube-system icp-console -o jsonpath='{.spec.host}') -u
admin -p admin -n default
```

4. Instead of using an existing namespace, you create a new namespace for IBM Agile Lifecycle Manager:

```
oc create namespace <alm_namespace>
```

5. Trust the Common Service Red Hat OpenShift Container Platform host:


```
ICP_HOST=$(oc get routes -n kube-system | grep console | awk '{print $2}')
openssl s_client -showcerts -servername $ICP_HOST -connect ${ICP_HOST}:443 </dev/null >
ICP_ca_cert.pem
cp ICP_ca_cert.pem /etc/pki/ca-trust/source/anchors
update-ca-trust extract
```

6. Add the Red Hat OpenShift Container Platform internal Helm repository to the Helm CLI:

```
helm repo add local-charts https://<cluster_CA_domain>:<cluster_router_https_port>/helm-
repo/charts --ca-file ~/.helm/ca.pem
```

7. Trust the image registry cert and login to docker.

```
OCF_REGISTRY=$(oc get route -n openshift-image-registry default-route -o
jsonpath='{.spec.host}')
mkdir -p /etc/docker/certs.d/$OCF_REGISTRY
openssl s_client -showcerts -servername $OCF_REGISTRY -connect $OCF_REGISTRY:443 2>/dev/
null | openssl x509 -inform pem > /etc/docker/certs.d/$OCF_REGISTRY/ca.crt
```

8. Log in to the Red Hat OpenShift Container Platform docker registry:

```
docker login -u admin -p $(oc whoami -t) ${OCF_REGISTRY}
```

The default cluster domain name is `mycluster.icp`.

9. Load the IBM Agile Lifecycle Manager archive:

```
NAMESPACE=<alm_namespace>
```

```
cloudctl catalog load-archive --archive <ARCHIVE_NAME> --registry $OCF_REGISTRY/$NAMESPACE
--repo local-charts
```

The IBM Agile Lifecycle Manager Helm charts will be uploaded to the Helm Red Hat OpenShift Container Platform repository `local-charts`, and the Docker images to the Red Hat OpenShift Container Platform Docker registry. You can view the chart repositories on Red Hat OpenShift Container Platform:

```
cloudctl catalog repos
```

10. Update the local Helm repositories information:

```
helm repo update
```

Verify that the Helm charts are now visible from the Helm CLI:

```
helm search alm
```

Run the prerequisite scripts

11. Extract the prerequisite scripts from the `ibm_cloud_pak/pak_extensions/prereqs` directory. Use the following command:

```
tar -xvf <alm_archive_file> pak_extensions
```

12. Add all required configuration data, such as worker node URLs, disk locations and capacity, to the `storageConfig.env` file.

```
cd pak_extensions/prereqs/
vi storageConfig.env
```

For example:

```
WORKER1=worker1.an.os.example.abc.com
WORKER2=worker2.an.os.example.abc.com
WORKER3=worker3.an.os.example.abc.com
FS_ROOT=/opt/ibm/alm
CAPACITY_CASSANDRA=130
CAPACITY_KAFKA=295
```

```
CAPACITY_ELASTICSEARCH=50
CAPACITY_ZOOKEEPER=25
```

Note: For the WORKER1 to WORKER3 values, you must input the full worker name as found from the following command:

```
oc get nodes
```

13. Execute the prereqs scripts:

```
./createSecurityClusterPrereqs.sh
```

```
./createSecurityNamespacePrereqs.sh <alm_namespace>
```

```
./createStorageVolumes.sh <alm_namespace> <alm_release>
```

Where <alm_release> is the name you chose for the release.

Install IBM Agile Lifecycle Manager

You can install IBM Agile Lifecycle Manager either from the Helm CLI, or the Red Hat OpenShift Container Platform console.

14. To install IBM Agile Lifecycle Manager **from the Helm CLI**, use the following command:

```
NAMESPACE=<alm_namespace>
```

```
helm install --name <alm_helm-release> local-charts/ibm-alm-prod --namespace $NAMESPACE --
set license=accept,global.image.repository=image-registry.openshift-image-
registry.svc:5000/$NAMESPACE --tls
```

You can verify the deployment from using the `helm status alm --tls` command, where *alm* is the release name.

Healthcheck failure: A temporary Healthcheck failure may occur when Zookeeper generates new pod IP addresses, causing Vault to briefly lose connection. This issue resolves itself automatically, after which IBM Agile Lifecycle Manager resumes normal service.

What to do next

Validate that the UI is accessible by opening the URL obtained from this following command:

```
echo "https://$(oc get routes alm-nimrod -n <alm_namespace> -o jsonpath='{.spec.host}')
```

Then see [“Integrating IBM Agile Lifecycle Manager with an existing external directory service”](#) on page 12.

Integrating IBM Agile Lifecycle Manager with an existing external directory service

IBM Agile Lifecycle Manager requires an LDAP directory service to authenticate and authorize users. IBM Agile Lifecycle Manager can be configured to use groups and users from a directory service and associate those to roles and privileges.

IBM Agile Lifecycle Manager configuration for LDAP Directory Service

The configuration used by IBM Agile Lifecycle Manager to connect to an external Directory Service is specified in a generic Kubernetes secret in the form of literal key-value pairs.

Table 4. Supported configuration parameters

Parameter	Description	Required or Optional
LDAP_URL	Specifies the LDAP connection URL to the external Directory Service.	Required.
LDAP_BIND_DN	Specifies the user DN that is used to authenticate to the LDAP Directory Server. If not specified, an anonymous bind is attempted.	Optional.
LDAP_ADMIN_PASSWORD	Specifies the bind user's password. This is required if anonymous bind not available. No default values are set.	Optional.
LDAP_BASE_DN	Specifies the DN of the search base. The default is 'dc=lm,dc=com'.	Optional.
LDAP_USER_SEARCH_BASE	Specifies the search base for the user. If the parameter LDAP_BASE_DN is specified, the user search base must be relative to it. The default is 'ou=people'.	Optional.
LDAP_USER_SEARCH_FILTER	Specifies the filter used in the user search. The default is 'uid={0}'.	Optional.
LDAP_GROUP_SEARCH_BASE	Specifies the search base for groups of which the user is a member. If the parameter LDAP_BASE_DN is specified, the group search base must be relative to it. The default is 'ou=groups'.	Optional.
LDAP_GROUP_SEARCH_FILTER	Specifies the filter used in the group search The default is 'member={0}'.	Optional.
LDAP_PASSWORD_ATTR	Specifies the name of the attribute in the Directory Service that contain the user password.	Optional.

Table 4. Supported configuration parameters (continued)

Parameter	Description	Required or Optional
LDAP_PASSWORD_ENC	Specifies the encryption algorithm used to encrypt the user password. Possible values are BCrypt and PLAIN. If not specified, the default value BCrypt is used.	Optional.
AUTH_PROVIDER	Specifies the type of bind to be performed when the IBM Agile Lifecycle Manager client connect to the LDAP Directory Server. Possible values are ldapSimple and ldapBind. In the ldapSimple type, the bind user is used to perform LDAP activities. In the ldapBind type, the bind user is used to search the IBM Agile Lifecycle Manager user and to retrieve its password. A subsequent bind is performed using the user credentials. Any subsequent LDAP operation is carried out as the user. If not specified, the default value ldapSimple is used.	Optional.

Note: The properties LDAP_SEARCH_BASE and LDAP_SEARCH_FILTER available in previous versions of IBM Agile Lifecycle Manager have been renamed to LDAP_USER_SEARCH_BASE and LDAP_USER_SEARCH_FILTER. If you are using a secret definition from a previous version, make sure to update the name of these properties.

Configuring IBM Agile Lifecycle Manager to use an LDAP directory service

To configure IBM Agile Lifecycle Manager to use an LDAP directory service, you need to create a kubernetes secret containing the LDAP parameters.

The following code shows an example of a command to create an LDAP secret.

```
kubectl create secret generic alm-ldap-config-secret \
  --from-literal=LDAP_BASE_DN=<base-dn> \
  --from-literal=LDAP_BIND_DN=<bind-dn> \
  --from-literal=LDAP_ADMIN_PASSWORD=<bind-dn-password> \
  --from-literal=LDAP_PASSWORD_ATTR=<user-password-attribute> \
  --from-literal=LDAP_URL=ldap://<ldap-server-host>:<ldap-server-port> \
  --from-literal=LDAP_USER_SEARCH_BASE=<user-search-base> \
  --from-literal=LDAP_USER_SEARCH_FILTER=<user-search-filter>
```

Then you need to update IBM Agile Lifecycle Manager to use the LDAP configuration by running the following command:

```
helm upgrade alm --set global.security.ldapConfig.secretName=<alm-ldap-config-secret> --install \
  <path-to-the-alm-chart> --reuse-values --tls
```

The <path-to-the-alm-chart> must contain the path to the charts used to deploy.

IBM Agile Lifecycle Manager roles and privileges

Roles in IBM Agile Lifecycle Manager are defined by the privileges to which they are associated. IBM Agile Lifecycle Manager uses privileges to authorize users. IBM Agile Lifecycle Manager comes with a set of predefined roles, SLMAdmin, Portal, ReadOnly and RootSecAdmin. These predefined roles are associated by default to LDAP groups with the predefined group names: SLMAdmin, Portal, ReadOnly and RootSecAdmin. For a detailed description of the predefined roles and their associated privileges, see <http://servicelifecyclemanager.com/2.1.0/reference/security/default-security-users/>. To use predefined roles, you can create groups in the LDAP directory service named after the predefined roles and add members to them. Alternatively, you can associate custom LDAP directory service groups to predefined roles. For details see [Associate custom LDAP Directory Service groups to pre-defined roles](#).

Creating new roles in IBM Agile Lifecycle Manager

New roles can be defined in a kubernetes ConfigMap YAML file where the data section related to the alm-role is in JSON format.

The following code shows an example of a ConfigMap which defines a role called alm-operator:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: alm-operator
  namespace: alm-namespace
data:
  alm-operator.json: |-
    {
      "alm": {
        "roles": {
          "alm-operator": {
            "privileges": {
              "NsinstsMgt" : "read,execute",
              "nsDesMgt" : "read",
              "DeployLocMgt" : "read",
              "VnfInstsMgt" : "read",
              "BehvrScenDes" : "read",
              "VduInstsMgt" : "read",
              "IntentReqsMgt" : "read,execute",
              "VduGrpMgt" : "read",
              "BehvrScenExec" : "read,execute",
              "VduDesMgt" : "read",
              "RmDrv" : "read"
            },
            "ldapGroups": [
              "Operators"
            ]
          }
        }
      }
    }
```

The alm-operator role is associated to an LDAP group called Operators. Users who are members of the LDAP group Operators will inherit the privileges associated to the alm-operator role.

For details of the full list of privileges see <http://servicelifecyclemanager.com/2.1.0/reference/security/default-security-users/#available-privileges>.

Create the ConfigMap by running the following command:

```
kubectl create -f <ConfigMap-yaml-file>
```

After creating the ConfigMap, update IBM Agile Lifecycle Manager passing the ConfigMap in the global.security.almRoles.configMapName property to allow IBM Agile Lifecycle Manager to use the new role:

```
helm upgrade alm --set global.security.almRoles.configMapName=alm-operator --install
<path-to-the-alm-chart> --reuse-values --tls
```

The <path-to-the-alm-chart> must contain the path to the charts used to deploy.

Associate custom LDAP Directory Service groups to pre-defined roles

You may want to use existing LDAP directory service groups with predefined IBM Agile Lifecycle Manager roles. To do that, you need to create a ConfigMap that associates your groups to the predefined roles. For example, the following ConfigMap associates a group called `my_custom_group` to the predefined role `Portal`.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: alm-portal-group
  namespace: default
data:
  alm-portal-group.json: |-
    {
      "alm" : {
        "roles" : {
          "Portal" : {
            "ldapGroups" : [
              "my_custom_group"
            ]
          }
        }
      }
    }
```

After creating the ConfigMap, update IBM Agile Lifecycle Manager passing the ConfigMap in the `global.security.almRoles.configMapName`:

```
helm upgrade alm --set global.security.almRoles.configMapName=alm-portal-group --install
<path-to-the-alm-chart> --reuse-values --tls
```

The `<path-to-the-alm-chart>` must contain the path to the charts used to deploy.

Note: Only one ConfigMap can be passed to IBM Agile Lifecycle Manager, so all the configuration to create new roles and associate groups to roles have to be specified in one ConfigMap. It is not possible to modify predefined roles by adding or removing privileges.

Example 1: Configuring IBM Agile Lifecycle Manager to integrate with the external LDAP server using simple authentication

This example explains how to set up an external LDAP directory service with predefined IBM Agile Lifecycle Manager groups and users, and to authenticate them using simple authentication with one of those predefined users.

1. Setting up the LDAP directory service populated with predefined LDAP groups and users for simple authentication mechanism

Note: All commands (run as root or use `sudo`) in the following instructions are for OpenLDAP running on Ubuntu 18.04.

a. Installing OpenLDAP on Ubuntu 18.04.

1) Install OpenLDAP:

On the OpenLDAP server, run the following command:

```
sudo apt install slapd ldap-utils
```

The installation will request the creation of an administrator password for the LDAP directory. Make a note of the password that you chose. In this example "password" is used as the password of the LDAP administrator.

2) Reconfigure your existing OpenLDAP.

To reconfigure the suffix of your OpenLDAP server, run the following command:

```
sudo dpkg-reconfigure slapd
```

Use the following values when configuring the OpenLDAP Server:

```
Omit OpenLDAP Server Configuration: No
DNS Domain Name: example.org
Organization Name: dc=example,dc=org
Administrator Password: password
Confirm Password: password
Database backend to use: MDB
Do you want database to be removed when slapd is purged? : No
Move Old Database?: Yes
```

Note: All commands in the instructions below use the suffix `dc=example,dc=org`. For the purpose of the example we assume the user specifies `example.org` as the DNS domain and `dc=example,dc=org` as the organization during reconfiguration, and `password` as the password of the LDAP Administrator.

- b. Populating OpenLDAP with a predefined IBM Agile Lifecycle Manager LDAP configuration (consisting of roles, ldapgroups, and users).

Note: IBM Agile Lifecycle Manager mandates that all passwords are stored in `bcrypt` format. The `bcrypt` hashing mechanism that is used in IBM Agile Lifecycle Manager currently only supports hashed passwords with the `$2a` prefix. So ensure that you chose the `$2a` prefix for each LDAP user password, otherwise the user's login will fail. In the following `ldif` configuration file the `userPassword` value `$2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6` is "password" when encrypted using `bcrypt`.

- 1) Create a file named `predefined-user.ldif`.

On the OpenLDAP server, create a file called `predefined-user.ldif` with the following content, making sure to preserve the new lines between sections:

```
dn: ou=groups,dc=example,dc=org
objectClass: organizationalUnit
objectClass: top
ou: groups

dn: ou=people,dc=example,dc=org
changetype: add
objectClass: organizationalUnit
objectClass: top
ou: people

dn: uid=almuser,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: almuser
sn: almuser
uid: almuser
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: uid=almoperator,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: almoperator
sn: almoperator
uid: almoperator
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: uid=almoeditor,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: almoeditor
sn: almoeditor
uid: almoeditor
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: uid=almadmin,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: almadmin
```

```

sn: almadmin
uid: almadmin
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: uid=secadmin,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: secadmin
sn: secadmin
uid: secadmin
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: uid=readonly,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: readonly
sn: readonly
uid: readonly
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: cn=SLMAdmin,ou=groups,dc=example,dc=org
objectclass: groupOfNames
cn: SLMAdmin
member: uid=almadmin,ou=people,dc=example,dc=org

dn: cn=Portal,ou=groups,dc=example,dc=org
objectclass: groupOfNames
cn: Portal
member: uid=almuser,ou=people,dc=example,dc=org
member: uid=almoperator,ou=people,dc=example,dc=org
member: uid=almoeditor,ou=people,dc=example,dc=org

dn: cn=RootSecAdmin,ou=groups,dc=example,dc=org
objectclass: groupOfNames
cn: RootSecAdmin
member: uid=secadmin,ou=people,dc=example,dc=org

dn: cn=ReadOnly,ou=groups,dc=example,dc=org
objectclass: groupOfNames
cn: ReadOnly
member: uid=readonly,ou=people,dc=example,dc=org

```

2) Add the above data to the LDAP server.

On the OpenLDAP server, add the configuration created in the previous step.

```

ldapadd -H ldap://localhost:389 -D "cn=admin,dc=example,dc=org" -w password -f
predefined-user.ldif

```

2. Creating an LDAP secret

On the cluster in the IBM Agile Lifecycle Manager namespace, create a secret that will be used to connect to the LDAP server. The secret should contain the data that you provided while setting up the OpenLDAP (for example LDAP_URL as ldap://serverIP:port and LDAP_BASE_DN as dc=example,dc=org).

The following code shows the secret content for the Simple Authentication Mechanism:

```

kubectl create secret generic ldap-simple-auth \
--from-literal=LDAP_BASE_DN=dc=example,dc=org \
--from-literal=LDAP_BIND_DN=cn=admin,dc=example,dc=org \
--from-literal=LDAP_ADMIN_PASSWORD=password \
--from-literal=LDAP_PASSWORD_ATTR=userPassword \
--from-literal=LDAP_PASSWORD_ENC=BCRYPT \
--from-literal=LDAP_URL=ldap://<ldapHostMachineIp>:389 \
--from-literal=LDAP_USER_SEARCH_BASE=ou=people \
--from-literal=LDAP_USER_SEARCH_FILTER=uid={0} \
--from-literal=LDAP_GROUP_SEARCH_BASE=ou=groups \
--from-literal=LDAP_GROUP_SEARCH_FILTER=member={0}

```

When created, the following output message should appear: secret/ldap-simple-auth created

Note: If there was already a secret for LDAP, you must delete it and create a new secret with a different name. Otherwise, the upgrade will not take effect.

3. Installing or updating IBM Agile Lifecycle Manager using the newly created LDAP secret, and authenticating

- a. On the cluster, update IBM Agile Lifecycle Manager to pick up the secret.

```
helm upgrade alm --set global.security.ldapConfig.secretName=ldap-simple-auth --install  
<path-to-the-alm-chart> --reuse-values --tls
```

The parameter `<path-to-the-alm-chart>` must contain the path to the helm charts used to deploy.

- b. Wait for vault and ishtar pods to restart. You can use the following command to monitor them:

```
watch kubectl get pods -l 'app in \{(vault,ishtar)\}'
```

- c. When the pods have restarted, verify that you can log in to the IBM Agile Lifecycle Manager User Interface using the user `almadmin` and password `password`.

Example 2: Configuring IBM Agile Lifecycle Manager to integrate with the external LDAP server using `ldapbind` authentication

This example explains how to set up an external LDAP directory service with predefined IBM Agile Lifecycle Manager groups and users, and to authenticate using `ldapbind` authentication with one of the predefined users.

The following example assumes that an OpenLDAP server has been set up as described in [Example 1](#).

1. Modifying predefined IBM Agile Lifecycle Manager LDAP groups and users for the `ldapbind` mechanism

- a. On the OpenLDAP server, create a file called `predefined-bind-user.ldif` with the following content, making sure to preserve the new lines between sections:

```
dn: uid=almuser,ou=people,dc=example,dc=org  
changetype: modify  
add: objectClass  
objectClass: simpleSecurityObject  
-  
replace: userPassword  
userPassword: password  
  
dn: uid=almoperator,ou=people,dc=example,dc=org  
changetype: modify  
add: objectClass  
objectClass: simpleSecurityObject  
-  
replace: userPassword  
userPassword: password  
  
dn: uid=almoeditor,ou=people,dc=example,dc=org  
changetype: modify  
add: objectClass  
objectClass: simpleSecurityObject  
-  
replace: userPassword  
userPassword: password  
  
dn: uid=almadmin,ou=people,dc=example,dc=org  
changetype: modify  
add: objectClass  
objectClass: simpleSecurityObject  
-  
replace: userPassword  
userPassword: password  
  
dn: uid=secadmin,ou=people,dc=example,dc=org  
changetype: modify  
add: objectClass  
objectClass: simpleSecurityObject  
-  
replace: userPassword  
userPassword: password  
  
dn: uid=readonly,ou=people,dc=example,dc=org
```

```
changetype: modify
add: objectClass
objectClass: simpleSecurityObject
-
replace: userPassword
userPassword: password
```

- b. Apply the changes on the OpenLDAP server.

On the OpenLDAP server, run the following command:

```
ldapmodify -H ldap://localhost:389 -D "cn=admin,dc=example,dc=org" -w password -f
predefined-bind-user.ldif
```

2. Creating an LDAP secret

On the cluster in the IBM Agile Lifecycle Manager namespace, create a secret that will be used to connect to the LDAP server. The secret should contain the data that you provided while setting up the server (for example LDAP_URL as ldap://serverIP:port and LDAP_BASE_DN as dc=example,dc=org)

The following code shows the secret content for the Ldapbind Authentication Mechanism:

```
kubectl create secret generic ldapbind-secret \
--from-literal=LDAP_BASE_DN=dc=example,dc=org \
--from-literal=LDAP_BIND_DN=cn=admin,dc=example,dc=org \
--from-literal=LDAP_ADMIN_PASSWORD=password \
--from-literal=LDAP_PASSWORD_ATTR=userPassword \
--from-literal=LDAP_PASSWORD_ENC=PLAIN \
--from-literal=LDAP_URL=ldap://<ldapHostMachineIp>:389 \
--from-literal=LDAP_USER_SEARCH_BASE=ou=people \
--from-literal=LDAP_USER_SEARCH_FILTER=uid={0} \
--from-literal=LDAP_GROUP_SEARCH_BASE=ou=groups \
--from-literal=LDAP_GROUP_SEARCH_FILTER=member={0} \
--from-literal=AUTH_PROVIDER=ldapBind
```

When created, the following output message should appear: secret/ldapbind-secret created

Note: If there was already a secret for LDAP, you must delete it and create a new secret with a different name. Otherwise, the upgrade will not take effect.

3. Installing or updating IBM Agile Lifecycle Manager using the newly created LDAP secret

- a. On the cluster, update IBM Agile Lifecycle Manager to pick up the secret.

```
helm upgrade alm --set global.security.ldapConfig.secretName=ldapbind-secret --install
<path-to-the-alm-chart> --reuse-values --tls
```

The parameter <path-to-the-alm-chart> must contain the path to the helm charts used to deploy.

- b. Wait for vault and ishtar pods to restart. You can use the following command to monitor them:

```
watch kubectl get pods -l 'app in \{vault,ishtar\}'
```

- c. When the pods have restarted, verify that you can log in to the IBM Agile Lifecycle Manager User Interface using the user almadmin and password password.

Example 3: Using a custom LDAP user to authenticate to IBM Agile Lifecycle Manager using a predefined group and role

This example explains how to add a new user to one of the IBM Agile Lifecycle Manager predefined groups and to authenticate.

The following example assumes that an OpenLDAP server has been set up as described in [Example 1](#).

1. Create a new user (for example newuserpartofSLMADMIN and add it to the predefined group SLMAdmin).

On the OpenLDAP server, create a file newuserpartofSLMADMIN.ldif, making sure to preserve the new lines between sections:

```
dn: uid=newuserpartofSLMADMIN,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: newuserpartofSLMADMIN
sn: newuserpartofSLMADMIN
uid: newuserpartofSLMADMIN
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: cn=SLMAdmin,ou=groups,dc=example,dc=org
changetype: modify
add: member
member: uid=newuserpartofSLMADMIN,ou=people,dc=example,dc=org
```

Apply the changes on the OpenLDAP server:

```
ldapmodify -H ldap://localhost:389 -x -D "cn=admin,dc=example,dc=org" -w password -f
newuserpartofSLMADMIN.ldif
```

2. Check authentication.

On the OpenLDAP server, run the following command which should respond with a prompt to "Enter LDAP Password":

```
ldapsearch -x -H ldap://localhost:389 -b "dc=example,dc=org" -D
"uid=newuserpartofSLMADMIN,ou=people,dc=example,dc=org" -W
```

When prompted for the LDAP password, enter the userPassword value for user newuserpartofSLMADMIN as configured in the newuserpartofSLMADMIN.ldif file, namely \$2a\$10\$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

On the IBM Agile Lifecycle Manager UI, login with the username newuserpartofSLMADMIN and password password.

Example 4: Creating a custom LDAP group and associating it to a predefined role

This example explains how to create a new ldapgroup in Openldap and associate it to a pre-defined role in IBM Agile Lifecycle Manager.

The following example assumes that an OpenLDAP server has been set up as described in [Example 1](#).

1. Create a new group *cn=Designers,ou=groups,dc=example,dc=org* and add a member *uid=almdesigner,ou=people,dc=example,dc=org* to it.

On the OpenLDAP server, create a file called new-user-group.ldif with the following content, making sure to preserve the new lines between sections:

```
dn: uid=almdesigner,ou=people,dc=example,dc=org
changetype: add
objectClass: person
objectClass: uidObject
cn: almdesigner
sn: almdesigner
uid: almdesigner
userPassword: $2a$10$nyG5R781Vbowtj73XohFNOY6RHoMz6fInUMwtH00lkib.mh0yo4T6

dn: cn=Designers,ou=groups,dc=example,dc=org
objectClass: groupOfNames
cn: Designers
member: uid=almdesigner,ou=people,dc=example,dc=org
```

2. Add the group and user definition to OpenLDAP by running the following command:

```
ldapadd -H ldap://localhost:389 -D "cn=admin,dc=example,dc=org" -w password -f new-user-
group.ldif
```

3. Create a ConfigMap that associates the Designers group to the Portal role:

On the cluster, create a YAML file called alm-role-group.yaml:

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: alm-role-group
  namespace: default
data:
  alm-role-group.json: |-
    {
      "alm": {
        "roles": {
          "Portal": {
            "ldapGroups": [
              "Designers"
            ]
          }
        }
      }
    }

```

4. Create the ConfigMap by running the following command:

```
kubectl create -f alm-role-group.yaml
```

5. Update IBM Agile Lifecycle Manager to pick up the ConfigMap by running the following command:

```
helm upgrade alm --set global.security.almRoles.configMapName=alm-role-group --install <path-to-the-alm-chart> --reuse-values --tls
```

The parameter `<path-to-the-alm-chart>` must contain the path to the helm charts used to deploy.

6. Wait for vault and ishtar pods to restart. You can use the following command to monitor them:

```
watch kubectl get pods -l 'app in (vault,ishtar)\'
```

7. When the pods are restarted, verify that you can log in to the IBM Agile Lifecycle Manager User Interface using the user *almdesigner* and password *password*.

Example 5: Create a new IBM Agile Lifecycle Manager role and associate it to an OpenLDAP group

This example explains how to create a new IBM Agile Lifecycle Manager role and associate it to an OpenLDAP group.

This example assumes that an OpenLDAP server has been set up as described in [Example 1](#) and that a group *cn=Designers,ou=groups,dc=example,dc=org* and a member *uid=almdesigner,ou=people,dc=example,dc=org* have been defined in OpenLDAP. See [Example 4](#) for the steps on how to create the group *cn=Designers,ou=groups,dc=example,dc=org* and the member *uid=almdesigner,ou=people,dc=example,dc=org*.

1. Create a file called `alm-roles.yaml` containing a ConfigMap with the definition of the new role Designer, making sure to preserve the new lines between sections:

```

apiVersion: v1
kind: ConfigMap
metadata:
  name: alm-roles
  namespace: default
data:
  alm-roles.json: |-
    {
      "alm": {
        "roles": {
          "Designer": {
            "privileges": {
              "DeployLocMgt" : "read,write,execute",
              "VnfInstsMgt" : "read,write,execute",
              "VnfDesMgt" : "read,write,execute",
              "BehvrScenDes" : "read,write",
              "NsinstsMgt" : "read,write,execute",
              "IntentReqsOps" : "read",
              "nsDesMgt" : "read,write,execute",
            }
          }
        }
      }
    }

```

```

        "IntentReqsMgt" : "read",
        "VduMgt" : "read,write,execute",
        "BehvrScenExec" : "read,write,execute",
        "VduGrpMgt" : "read,write,execute",
        "VduDesMgt" : "read,write,execute",
        "MaintModeOride" : "read,execute",
        "VduInstsMgt" : "read,write,execute",
        "RmDrvr" : "read,write",
        "ResourcePkg" : "write"
    },
    "ldapGroups" : [
        "Designers"
    ]
}
}
}
}
}

```

2. Create the ConfigMap by running the following command:

```
kubectl create -f alm-roles.yaml
```

3. Update IBM Agile Lifecycle Manager to pick up the ConfigMap by running the following command:

```
helm upgrade alm --set global.security.almRoles.configMapName=alm-roles --install <path-to-the-alm-chart> --reuse-values --tls
```

The parameter <path-to-the-alm-chart> must contain the path to the helm charts used to deploy.

4. Wait for vault and ishtar pods to restart. You can use the following command to monitor them:

```
watch kubectl get pods -l 'app in (vault,ishtar)\'
```

5. When the pods are restarted, verify that you can log in to the IBM Agile Lifecycle Manager User Interface using the user `almadmin` and password `password`.

Displaying IBM Agile Lifecycle Manager logs in Kibana

You can configure the Red Hat OpenShift Container Platform cluster to collect IBM Agile Lifecycle Manager logs and to display them in Kibana.

To do this, you need to deploy and configure the Red Hat OpenShift Container Platform cluster logging feature within your cluster. For instructions, refer to the Red Hat OpenShift Container Platform documentation which is available on the following website:

<https://docs.openshift.com/>

Uninstalling IBM Agile Lifecycle Manager

Uninstall IBM Agile Lifecycle Manager from Red Hat OpenShift Container Platform by performing the following steps.

About this task

This procedure uninstalls the deployed version of IBM Agile Lifecycle Manager, but does not remove the load images or charts.

Note: When you uninstall a release, any unused docker images that were used as part of the installation remain on the master node and on a local repository on one or more worker nodes. From time to time, Kubernetes removes unused images as part of its garbage collection process. For more information, see [Configuring kubelet Garbage Collection](#).

Tip: If you have more than one instance of IBM Agile Lifecycle Manager, you can run the following Helm command to determine which releases are installed:

```
helm ls --tls
```

deleteSecurityClusterPrereqs.sh

Deletes the PodSecurityPolicy and ClusterRole for all releases of this chart

deleteSecurityNamespacePrereqs.sh

Deletes the RoleBinding for the namespace

deleteStorageVolumes.sh

Removes the persistent storage volumes and claims for a release, once you have uninstalled

Procedure

1. Delete the release to be removed by running the following Helm command:

```
helm delete --purge --tls release_name
```

Where *release_name* is the name the release to be uninstalled.

2. Run the (included) scripts.

```
./deleteStorageVolumes.sh <alm_release_name>
```

```
./deleteSecurityNamespacePrereqs.sh <alm_namespace>
```

```
./deleteSecurityClusterPrereqs.sh
```

After running the scripts, delete the folders described in the output of the deleteStorageVolumes script.

3. Clean up remaining cron job objects and their related pods.

After IBM Agile Lifecycle Manager has been uninstalled, orphaned job objects and related pods may remain on the system. Remove these as in the following example:

```
kubect1 delete job -l release=<alm_helm-release> --namespace  
<alm_namespace> --cascade
```

And execute the following command:

```
oc delete secret alm-vault-keys
```

Notices

This information applies to the PDF documentation set for IBM Agile Lifecycle Manager.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
958/NH04
IBM Centre, St Leonards

601 Pacific Hwy
St Leonards, NSW, 2069
Australia

IBM Corporation
896471/H128B
76 Upper Ground
London
SE1 9PZ
United Kingdom

IBM Corporation
JBF1/SOM1 294
Route 100
Somers, NY, 10589-0100
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of IBM or other companies.

