DevOps Quick Start Guide

/ ForgeRock Identity Platform 6.5

Latest update: 6.5.2

Gina Cariaga David Goldsmith Shankar Raman

ForgeRock AS.
201 Mission St., Suite 2900
San Francisco, CA 94105, USA
+1 415-599-1100 (US)
www.forgerock.com

Abstract

Quick introduction to ForgeRock Identity Platform™ deployment using DevOps techniques for new users and readers evaluating the software.

Table of Contents

Preface

The *DevOps Quick Start Guide* provides instructions for quickly installing the ForgeRock DevOps Examples.

This guide is written for anyone who wants to evaluate the ForgeRock DevOps Examples. This guide covers the tasks you need to quickly get the DevOps Examples running on your system.

Before You Begin

Before deploying the ForgeRock Identity Platform in a DevOps environment, read the important information in Start Here.

About ForgeRock Identity Platform Software

ForgeRock Identity Platform™ serves as the basis for our simple and comprehensive Identity and Access Management solution. We help our customers deepen their relationships with their customers, and improve the productivity and connectivity of their employees and partners. For more information about ForgeRock and about the platform, see https://www.forgerock.com.

The platform includes the following components:

- ForgeRock® Access Management (AM)
- ForgeRock® Identity Management (IDM)
- ForgeRock® Directory Services (DS)
- ForgeRock® Identity Gateway (IG)

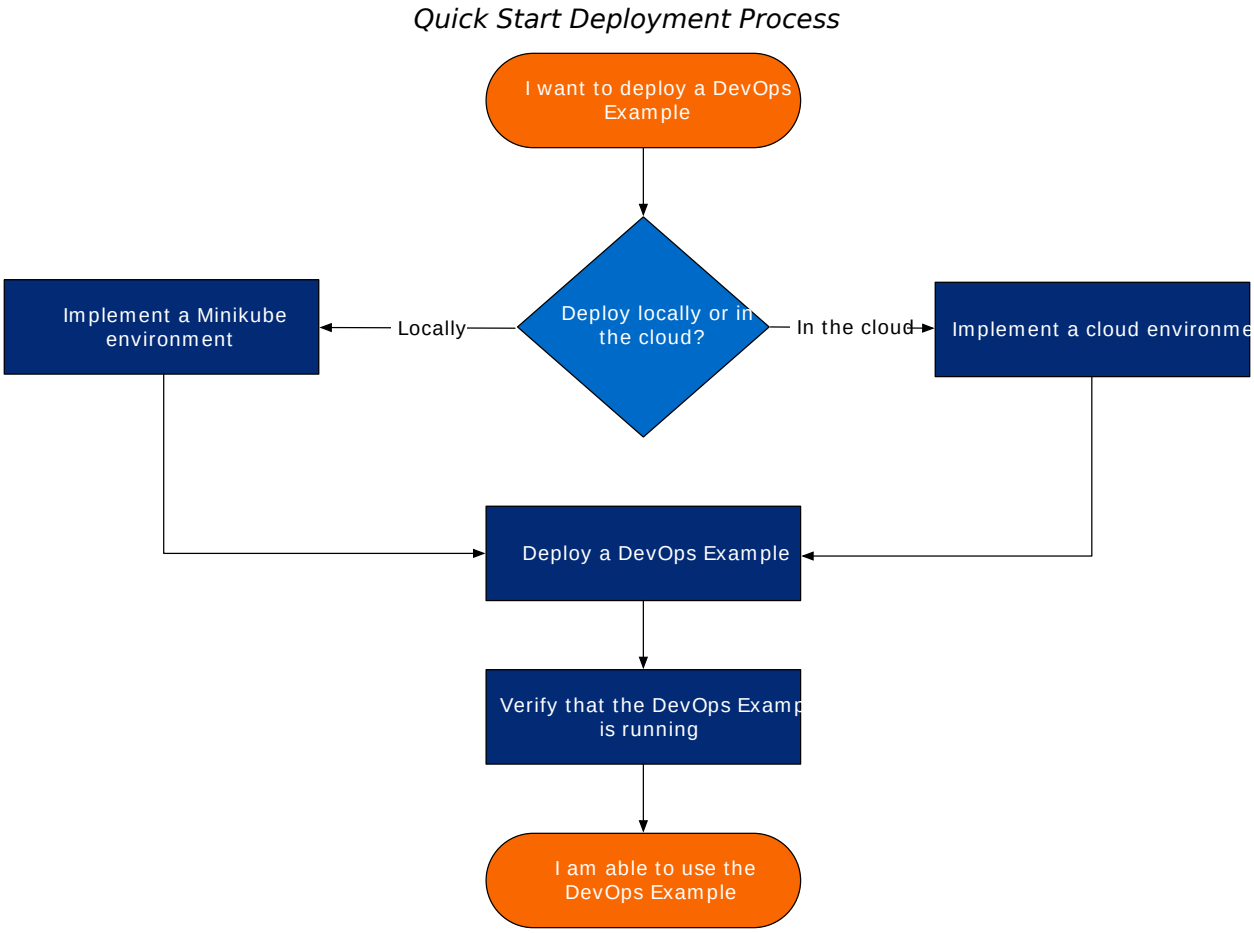
Chapter 1

Introducing DevOps for the ForgeRock Identity Platform

This *Quick Start Guide* provides instructions for quickly deploying and running the ForgeRock Identity Platform in a DevOps environment. The guide is designed to demonstrate how easy it can be to deploy the ForgeRock Identity Platform using DevOps techniques.

See "About the Example Deployment" for information about this example deployment and for guidance on performing more complex ForgeRock Identity Platform DevOps deployments.

The following diagram illustrates a high-level workflow you'll use to set up a DevOps environment and deploy ForgeRock Identity Platform:



Finer-grained workflows in "*Implementing a DevOps Environment*" and "*Deploying the ForgeRock Identity Platform*" provide more detailed task breakouts.

1.1. About the Example Deployment

The example deployment presented in this guide lets you get a simple ForgeRock Identity Platform deployment up and running in a DevOps environment as quickly as possible. The deployment uses the most minimal configuration possible for AM, IDM, and IG. This minimalist configuration is suitable for evaluation and demonstration purposes only.

This section describes several characteristics of the example deployment, and provides resources you can use for more complex deployments.

1.1.1. ForgeRock Identity Platform Configuration

The example deployment configures ForgeRock Identity Platform components as simply as possible:

- AM's configuration is empty: no realms, service configurations, or policies are configured in addition to the default configuration.
- IDM's configuration implements bidirectional data synchronization between IDM and LDAP described in Synchronizing Data Between LDAP and IDM in the *Samples Guide*.
- IG's configuration is as simple as possible.

This simple configuration, available from ForgeRock's public Git repository, is suitable for demonstration purposes only. A more robust ForgeRock Identity Platform deployment requires more complex configuration. For example, you would probably want to store your configuration in a private Git repository.

See the following links for information about using a more complex ForgeRock Identity Platform configuration:

- "Configuration as an Artifact" in the *DevOps Developer's Guide*
- "Creating Your Configuration Repository" in the *DevOps Developer's Guide*
- "About AM Configuration" in the *DevOps Developer's Guide*
- "Deploying the Example" in the *DevOps Developer's Guide*
- "Modifying the AM Configuration" in the *DevOps Developer's Guide*
- "Deploying the Example" in the *DevOps Developer's Guide*
- "Modifying the IDM Configuration" in the *DevOps Developer's Guide*
- "Deploying the Example" in the *DevOps Developer's Guide*
- "Modifying the IG Configuration" in the *DevOps Developer's Guide*

1.1.2. Docker Images

The example deployment uses evaluation-only Docker images for the ForgeRock Identity Platform from ForgeRock's public Docker registry at `gcr.io/forgerock-io`. ForgeRock does not support using these Docker images in production deployments.

See the following links for information about building production-ready Docker images and storing them in your own Docker registry:

- "To Create a Kubernetes Secret for Accessing a Private Docker Registry" in the *DevOps Developer's Guide*
- "*Building and Pushing Docker Images*" in the *DevOps Developer's Guide*

1.1.3. Secure Communication With ForgeRock Identity Platform Services

The example deployment provides secure access over HTTPS to ForgeRock Identity Platform server web UIs and REST APIs.

See "Configuring and Installing the frconfig Helm Chart" in the *DevOps Developer's Guide* for more information about securing communication to ForgeRock Identity Platform servers.

1.1.4. Runtime Changes to the AM Web Application

The example deployment installs the default AM `.war` file. You can customize this `.war` file to provide enhancements such as custom authentication modules, cross-origin resource sharing (CORS) support, or a custom look and feel for web UIs.

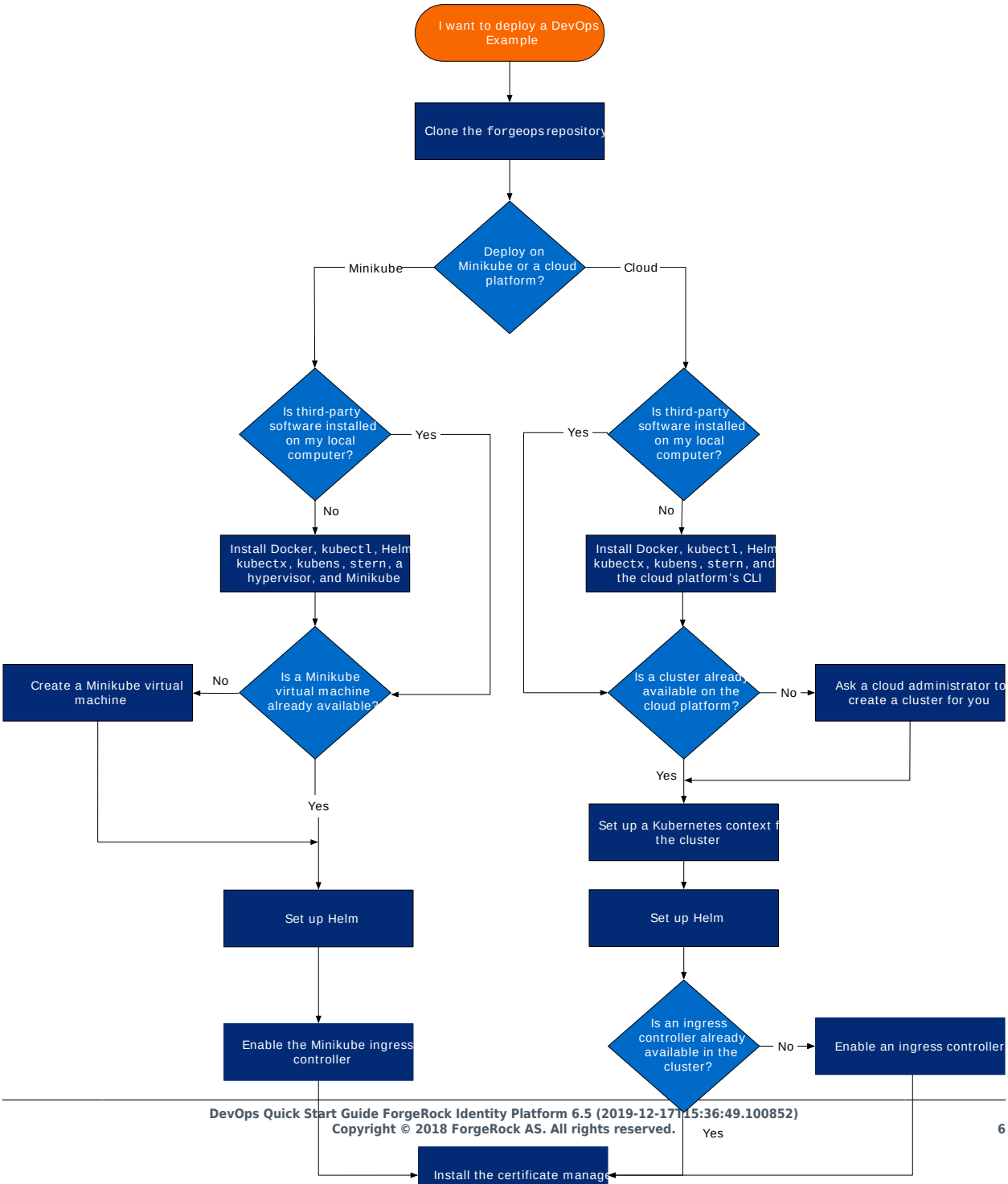
See "Customizing the AM Web Application" in the *DevOps Developer's Guide* for details about customizing the AM `.war` file when running in a DevOps environment.

Chapter 2

Implementing a DevOps Environment

The following diagram illustrates a high-level workflow you'll use to implement a DevOps environment:

Quick Start Environment Implementation



To implement a DevOps environment, perform the tasks listed in one of the following tables:

- Minikube: "Setting up a Minikube Environment"
- Cloud: "Setting up a Cloud Environment"

Setting up a Minikube Environment

Task	Steps
Clone the `forgeops` repository.	Follow the instructions in "forgeops Repository" in the *DevOps Release Notes*.
Install third-party software.	Follow the instructions in "Installing Required Third-Party Software" in the *DevOps Release Notes*.
Create a Minikube virtual machine (if necessary).	Follow the instructions in "Configuring Your Kubernetes Cluster" in the *DevOps Developer's Guide*.
Set up Helm.	Follow the instructions in "Setting up Helm" in the *DevOps Developer's Guide*.
Enable the Minikube ingress controller.	Perform "Deploying an Ingress Controller" in the *DevOps Developer's Guide*.
Install the certificate manager.	Perform "To Install the Certificate Manager" in the *DevOps Developer's Guide*.
Create a new Kubernetes namespace.	Perform "To Create a Namespace to Run the DevOps Examples" in the *DevOps Developer's Guide*. Do not use an existing namespace. If you do not start with an empty namespace, the steps in this guide might not work.

Setting up a Cloud Environment

Task	Steps
Clone the `forgeops` repository.	Follow the instructions in "forgeops Repository" in the *DevOps Release Notes*.
Install third-party software.	Follow the instructions in "Installing Required Third-Party Software" in the *DevOps Release Notes*.
Ask an administrator to create a cluster for you.	Refer to your cloud provider's documentation for more information.
Set up a Kubernetes context for your cluster.	Follow the instructions in "Setting up a Kubernetes Context" in the *DevOps Developer's Guide*.
Set up Helm.	Follow the instructions in "Setting up Helm" in the *DevOps Developer's Guide*.
Enable an ingress controller.	Perform the relevant procedure in "Deploying an Ingress Controller" in the *DevOps Developer's Guide*.

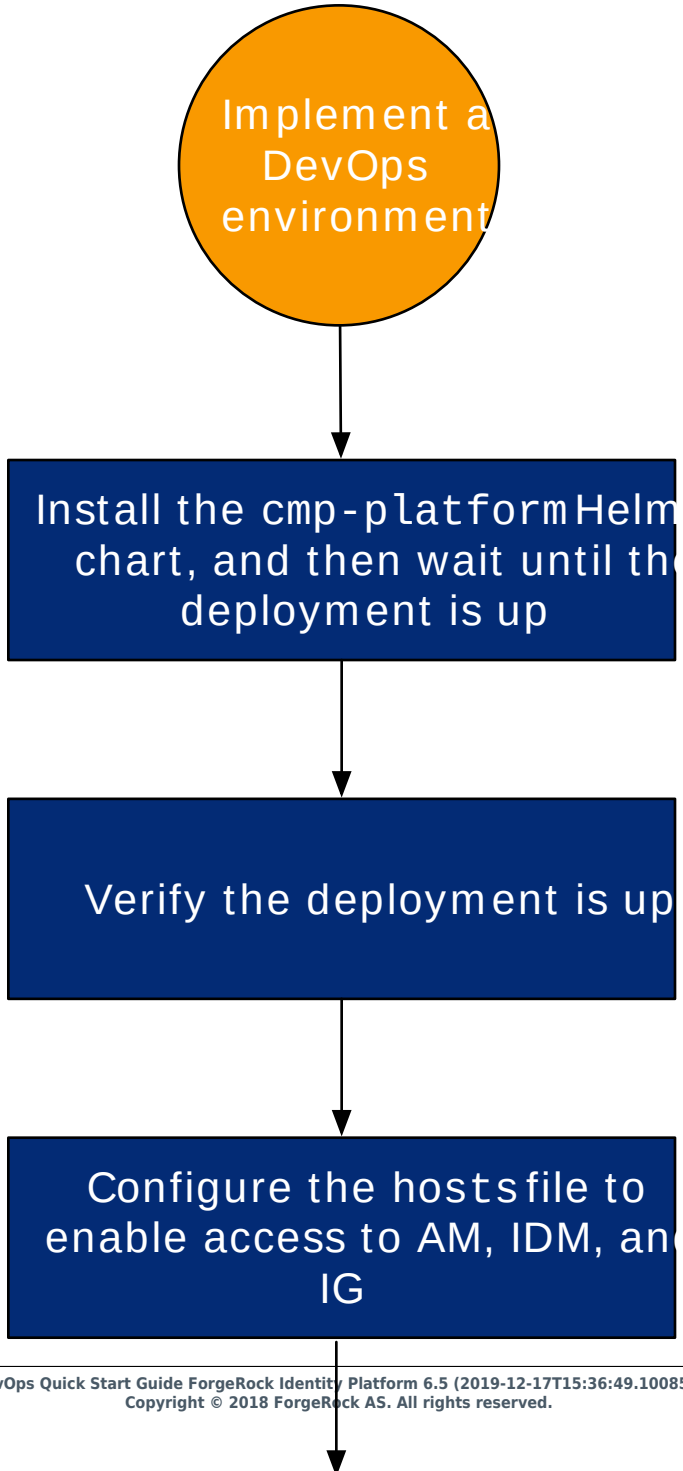
Task	Steps
Install the certificate manager.	Perform "To Install the Certificate Manager" in the *DevOps Developer's Guide*.
Create a new Kubernetes namespace.	Perform "To Create a Namespace to Run the DevOps Examples" in the *DevOps Developer's Guide*. Do not use an existing namespace. If you do not start with an empty namespace, the steps in this guide might not work.

Chapter 3

Deploying the ForgeRock Identity Platform

The following diagram illustrates the workflow you'll use to deploy ForgeRock Identity Platform:

Quick Start Deployment



To deploy ForgeRock Identity Platform, perform the following tasks:

Tasks	Steps
Install the `cmp-platform` Helm chart, and then wait until the deployment is up.	Perform "To Install the Helm Chart for the ForgeRock Identity Platform".
Verify the deployment is up.	Perform "To Determine Whether ForgeRock Identity Platform Components Are Up and Running".
Configure the `hosts` file.	Perform "To Configure the Hosts File".
Access the AM console, the IDM admin UI, and IG.	Perform "To Access ForgeRock Identity Platform Web User Interfaces".

To Install the Helm Chart for the ForgeRock Identity Platform

After you complete the following steps, AM, IDM, IG, and DS are automatically installed, minimally-configured, and started for you:

1. Make sure you've checked out the release/6.5.2 branch of the `forgeops` repository.
2. Change to the Helm charts directory in your `forgeops` repository clone:

   ```
   $ cd /path/to/forgeops/helm
   ```

3. Update the `cmp-platform` Helm chart's dependencies:

   ```
   $ helm dependency update cmp-platform
   Hang tight while we grab the latest from your chart repositories..
   .
   ...Unable to get an update from the "local" chart repository (http://127.0.0.1:8879/charts):
    Get http://127.0.0.1:8879/charts/index.yaml: dial tcp 127.0.0.1:8879: connect: connection
    refused
   ...Successfully got an update from the "nfs-provisioner" chart
    repository
   ...Successfully got an update from the "forgerock" chart
    repository
   ...Successfully got an update from the "stable" chart repository
   Update Complete. *Happy Helming!*
   Saving 8 charts
   Deleting outdated charts
   ```

4. Install the `cmp-platform` Helm chart:

   ```
   $ helm install cmp-platform
   ```

   Helm installs the `cmp-platform` Helm chart from the `forgeops` repository using default configuration values. The `cmp-platform` Helm chart installs and starts AM, IDM, IG, and DS.

   Output similar to the following appears in the terminal window:

   ```
   NAME:   bald-owl
   LAST DEPLOYED: Wed Jul 31 15:11:10 2019
   NAMESPACE: my-namespace
   STATUS: DEPLOYED
   ```

```
RESOURCES:
==> v1/Secret
NAME                   AGE
amster-secrets         1s
configstore            1s
certmanager-ca-secret  1s
frconfig               1s
openam-secrets         1s
openam-pstore-secrets  1s
openidm-secrets-env    1s
openidm-secrets        1s
openig-secrets-env     1s
postgres-openidm       1s
userstore              1s

==> v1/ConfigMap
amster-config           1s
amster-bald-owl         1s
configstore             1s
frconfig                1s
am-configmap            1s
boot-json               1s
idm-boot-properties     1s
bald-owl-openidm        1s
idm-logging-properties  1s
openig-bald-owl         1s
openidm-sql             1s
userstore               1s

==> v1/PersistentVolumeClaim
postgres-openidm  1s

==> v1beta1/Deployment
amster            1s
bald-owl-openam   1s
bald-owl-openig   1s
postgres-openidm  1s

==> v1beta1/Ingress
openam   1s
openidm  1s
openig   1s

==> v1alpha1/Certificate
wildcard.my-namespace.example.com  1s

==> v1/Pod(related)

NAME                               READY  STATUS    RESTARTS  AGE
amster-544794d567-kbhdt            0/2    Init:0/1  0         1s
bald-owl-openam-57cdb9c748-4rv95   0/1    Pending   0         1s
bald-owl-openig-6649668879-rwqwr   0/1    Pending   0         1s
postgres-openidm-79496f548b-f2gz9  0/1    Init:0/1  0         1s
configstore-0                      0/1    Pending   0         1s
bald-owl-openidm-0                 0/2    Pending   0         1s
userstore-0                        0/1    Pending   0         0s

==> v1/Service
```

```
NAME         AGE
configstore  1s
openam       1s
openidm      1s
openig       1s
postgresql   1s
userstore    1s

==> v1beta1/StatefulSet
configstore       1s
bald-owl-openidm  1s
userstore         1s

==> v1alpha1/Issuer
ca-issuer  1s


NOTES:

ForgeRock Platform

If you are on minikube, get your ip address using `minikube ip`

In your /etc/hosts file you will have an entry like:

192.168.100.1 login.my-namespace openidm.my-namespace openig.my-namespace


Get the pod status using:

kubectl get po

Get the ingress status using:

kubectl get ing



When the pods are ready, you can open up the consoles:

http://login.my-namespace.example.com/
http://openidm.my-namespace.example.com/admin
http://openig.my-namespace.example.com/
```

To Determine Whether ForgeRock Identity Platform Components Are Up and Running

1. Query the status of pods that comprise the deployment until all pods are ready:

    a. Run the **kubectl get pods** command:

```
$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
amster-544794d567-kbhdt             2/2     Running   0          1m
bald-owl-openam-57cdb9c748-4rv95    1/1     Running   0          1m
bald-owl-openidm-0                  2/2     Running   0          1m
bald-owl-openig-6649668879-rwqwr    1/1     Running   0          1m
configstore-0                       1/1     Running   0          1m
postgres-openidm-79496f548b-f2gz9   1/1     Running   0          1m
userstore-0                         1/1     Running   0          1m
```

b. Review the output. Deployment is complete when:

- All pods are completely ready. For example, a pod with the value `1/1` in the `READY` column of the output is completely ready, while a pod with the value `0/1` is not completely ready.
- All pods have attained `Running` status.

c. If necessary, repeat the **kubectl get pods** command until all the pods are ready.

2. Review the Amster pod's log to determine whether AM deployment completed successfully.

   Use the **kubectl logs amster-xxxxxxxxxx-yyyyy -c amster -f** command to stream the Amster pod's log to standard output.

   The following output appears as the deployment clones the Git repository containing the initial AM configuration, then waits for the AM server and DS instances to become available:

```
+ trap exit_script SIGINT SIGTERM SIGUSR1 EXIT
+ case $1 in
+ ./amster-install.sh
Waiting for AM server at http://openam:80//config/options.htm
Got Response code 000
response code 000. Will continue to wait
Got Response code 000
response code 000. Will continue to
 wait
. . .
```

   When Amster start to configure AM, the following output appears:

```
Got Response code 200
AM web app is up and ready to be configured
About to begin configuration
Extracting amster version
Amster version is:  6.5.2
Executing Amster to configure AM
Executing Amster script /opt/amster/scripts/00_install.amster
Jul 31, 2019 11:12:43 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Amster OpenAM Shell (6.5.2 build 314d553429, JVM: 1.8.0_151)
Type ':help' or ':h' for help
.
--------------------------------------------------------------------------------
am> :load /opt/amster/scripts/00_install.amster
07/31/2019 11:12:46:960 PM GMT: Checking license acceptance...
07/31/2019 11:12:46:966 PM GMT: License terms accepted.
07/31/2019 11:12:46:975 PM GMT: Checking configuration directory /home/forgerock/openam.
07/31/2019 11:12:46:976 PM GMT: ...Success.
07/31/2019 11:12:46:983 PM GMT: Tag swapping schema files.
07/31/2019 11:12:47:029 PM GMT: ...Success.
07/31/2019 11:12:47:029 PM GMT: Loading Schema odsee_config_schema.ldif
07/31/2019 11:12:47:123 PM GMT: ...Success
.
. . .
```

The following output indicates that deployment is complete:

```
07/31/2019 11:13:04:675 PM GMT: Installing new plugins...
07/31/2019 11:13:09:672 PM GMT: Plugin installation complete.
07/31/2019 11:13:12:162 PM GMT: Setting up monitoring authentication file.
Configuration complete!
Executing Amster script /opt/amster/scripts/01_import.amster
Amster OpenAM Shell (6.5.2 build 314d553429, JVM: 1.8.0_151)
Type ':help' or ':h' for help
.
--------------------------------------------------------------------------------
am> :load /opt/amster/scripts/01_import.amster
Importing directory /git/config/6.5/default/am/empty-import
Import completed successfully
Configuration script finished
+ pause
+ echo 'Args are 0 '
+ echo 'Container will now pause. You can use kubectl exec to run export.sh'
+ true
+ wait
+ sleep 1000000
```

To Configure the Hosts File

After you have installed the Helm chart for the example, configure the `/etc/hosts` file on your local computer so that you can access ForgeRock Identity Platform web UIs:

1. Get the ingress controller's IP address:

   - For Minikube, run the **minikube ip** command.

- For cloud environments, ask your cluster administrator which IP address to use to access your cluster's ingress controller.

2. To enable cluster access through the ingress controller, add an entry in the `/etc/hosts` file. For example:

```
192.168.99.100 login.my-namespace.example.com openidm.my-namespace.example.com
               openig.my-namespace.example.com
```

The entire entry should go on a single line in the `/etc/hosts` file with no line breaks.

In this example, `login.my-namespace.example.com`, `openidm.my-namespace.example.com`, and `openig.my-namespace.example.com` are the hostnames you use to access ForgeRock Identity Platform components, and `192.168.99.100` is the ingress controller's IP address.

To Access ForgeRock Identity Platform Web User Interfaces

1. If necessary, start a web browser.
2. To start the AM console:
   a. By default, the `amster` Helm chart dynamically generates a random password for the `amadmin` user. It stores the password in the `amster-config` configmap.

      To obtain the password, look for the `adminPwd` value in the `amster-config` configmap:

      ```
      $ kubectl get configmaps amster-config -o yaml | grep adminPwd
      --adminPwd "74xW6IShJF" \
      ```

   b. (Optional) Delete the `amster-config` configmap so that the `amadmin` password is not publicly available.

      ```
      $ kubectl delete configmaps amster-config
      ```

   c. Navigate to the AM deployment URL, https://login.*my-namespace*.example.com/XUI/?service=adminconsoleservice.

      The Kubernetes ingress controller handles the request and routes you to a running AM instance.

   d. Log in to the AM console as the `amadmin` user with password `password`.
3. To start the IDM Admin UI:
   a. Navigate to the IDM Admin UI's deployment URL, https://openidm.*my-namespace*.example.com/admin.

      The Kubernetes ingress controller handles the request and routes you to a running IDM instance.

b. Log in to the IDM Admin UI as the `openidm-admin` user with password `openidm-admin`.

4. To access IG, navigate to https://openig.*my-namespace*.example.com.

   The Kubernetes ingress controller handles the request, routing it to IG.

   You should see a message similar to the following:

   ```
   hello and Welcome to OpenIG. Your path is /. OpenIG is using the default handler for this route.
   ```

Appendix A. Getting Support

This appendix contains information about support options for the ForgeRock DevOps Examples and the ForgeRock Identity Platform.

A.1. ForgeRock DevOps Support

ForgeRock has developed artifacts in the `forgeops` and `forgeops-init` Git repositories for the purpose of deploying the ForgeRock Identity Platform in the cloud. The companion ForgeRock DevOps documentation provides examples, including the ForgeRock Cloud Deployment Model (CDM), to help you get started.

These artifacts and documentation are provided on an "as is" basis. ForgeRock does not guarantee the individual success developers may have in implementing the code on their development platforms or in production configurations.

A.1.1. Commercial Support

ForgeRock provides commercial support for the following DevOps resources:

- Dockerfiles and Helm charts in the `forgeops` Git repository
- ForgeRock DevOps guides.

ForgeRock provides commercial support for the ForgeRock Identity Platform. For supported components, containers, and Java versions, see the following:

- *ForgeRock Access Management Release Notes*
- *ForgeRock Identity Management Release Notes*

- *ForgeRock Directory Services Release Notes*
- *ForgeRock Identity Message Broker Release Notes*
- *ForgeRock Identity Gateway Release Notes*

A.1.2. Support Limitations

ForgeRock provides no commercial support for the following:

- Artifacts other than Dockerfiles or Helm charts in the `forgeops` and `forgeops-init` repositories. Examples include scripts, example configurations, and so forth.
- Non-ForgeRock infrastructure. Examples include Docker, Kubernetes, Google Cloud Platform, Amazon Web Services, and so forth.
- Non-ForgeRock software. Examples include Java, Apache Tomcat, NGINX, Apache HTTP Server, and so forth.
- Production deployments that use the DevOps evaluation-only Docker images. When deploying the ForgeRock Identity Platform using Docker images, you must build and use your own images for production deployments. For information about how to build Docker images for the ForgeRock Identity Platform, see "*Building and Pushing Docker Images*" in the *DevOps Developer's Guide*.

A.1.3. Third-Party Kubernetes Services

ForgeRock supports deployments on Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (Amazon EKS), Microsoft Azure Kubernetes Service (AKS), and Red Hat OpenShift.

Red Hat OpenShift is a tested and supported platform using Kubernetes for deployment. ForgeRock uses OpenShift tools such as Minishift, as well as other representative environments such as Amazon AWS for the testing. We do not test using bare metal due to the many customer permutations of deployment and configuration that may exist, and therefore cannot guarantee that we have tested in the same way a customer chooses to deploy. We will make commercially reasonable efforts to provide first-line support for any reported issue. In the case we are unable to reproduce a reported issue internally, we will request the customer engage OpenShift support to collaborate on problem identification and remediation. Customers deploying on OpenShift are expected to have a support contract in place with IBM/Red Hat that ensures support resources can be engaged if this situation may occur.

A.2. Accessing Documentation Online

ForgeRock publishes comprehensive documentation online:

- The ForgeRock Knowledge Base offers a large and increasing number of up-to-date, practical articles that help you deploy and manage ForgeRock software.

While many articles are visible to community members, ForgeRock customers have access to much more, including advanced information for customers using ForgeRock software in a mission-critical capacity.

- ForgeRock product documentation, such as this document, aims to be technically accurate and complete with respect to the software documented. It is visible to everyone and covers all product features and examples of how to use them.

A.3. How to Report Problems or Provide Feedback

If you are a named customer Support Contact, contact ForgeRock using the Customer Support Portal to request information or report a problem with Dockerfiles or Helm charts in the DevOps Examples or the CDM.

If you have questions regarding the DevOps Examples or the CDM that are not answered in the documentation, file an issue at https://github.com/ForgeRock/forgeops/issues.

When requesting help with a problem, include the following information:

- Description of the problem, including when the problem occurs and its impact on your operation.
- Steps to reproduce the problem.

  If the problem occurs on a Kubernetes system other than Minikube, GKE, EKS, OpenShift, or AKS, we might ask you to reproduce the problem on one of those.
- HTML output from the **debug-logs.sh** script. For more information, see "Running the debug-logs.sh Script" in the *DevOps Developer's Guide*.
- Description of the environment, including the following information:
  - Environment type: Minikube, GKE, EKS, AKS, or OpenShift.
  - Software versions of supporting components:
    - Oracle VirtualBox (Minikube environments only).
    - Docker client (all environments).
    - Minikube (all environments).
    - **kubectl** command (all environments).
    - Kubernetes Helm (all environments).
    - Google Cloud SDK (GKE environments only).
    - Amazon AWS Command Line Interface (EKS environments only).

  - Azure Command Line Interface (AKS environments only).
- `forgeops` repository branch.
- Any patches or other software that might be affecting the problem.

A.4. Getting Support and Contacting ForgeRock

ForgeRock provides support services, professional services, training through ForgeRock University, and partner services to assist you in setting up and maintaining your deployments. For a general overview of these services, see https://www.forgerock.com.

ForgeRock has staff members around the globe who support our international customers and partners. For details on ForgeRock's support offering, including support plans and service level agreements (SLAs), visit https://www.forgerock.com/support.

Glossary

affinity (AM)	AM affinity based load balancing ensures that the CTS token creation load is spread over multiple server instances (the token origin servers). Once a CTS token is created and assigned to a session, all subsequent token operations are sent to the same token origin server from any AM node. This ensures that the load of CTS token management is spread across directory servers. Source: *Best practices for using Core Token Service (CTS) Affinity based load balancing in AM*
Amazon EKS	Amazon Elastic Container Service for Kubernetes (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on Amazon Web Services without needing to set up or maintain your own Kubernetes control plane. Source: *What is Amazon EKS* in the Amazon EKS documentation.
ARN (AWS)	An Amazon Resource Name (ARN) uniquely identifies an Amazon Web Service (AWS) resource. AWS requires an ARN when you need to specify a resource unambiguously across all of AWS, such as in IAM policies and API calls. Source: *Amazon Resource Names (ARNs) and AWS Service Namespaces* in the AWS documentation.
AWS IAM Authenticator for Kubernetes	The AWS IAM Authenticator for Kubernetes is an authentication tool that enables you to use *Amazon Web Services (AWS)* credentials for authenticating to a Kubernetes cluster. Source: *AWS IAM Authenticator for Kubernetes* `README` file on `GitHub`.

cloud-controller-manager	The `cloud-controller-manager` daemon runs controllers that interact with the underlying cloud providers. `cloud-controller-manager` is an alpha feature introduced in Kubernetes release 1.6. The `cloud-controller-manager` daemon runs cloud-provider-specific controller loops only. Source: *cloud-controller-manager* section in the Kubernetes Concepts documentation.
Cloud Developer's Kit (CDK)	The developer artifacts in the `forgeops` Git repository, together with the ForgeRock Identity Platform documentation form the Cloud Developer's Kit (CDK). Use the CDK to stand up the platform in your developer environment.
Cloud Deployment Model (CDM)	The Cloud Deployment Model (CDM) is a common use ForgeRock Identity Platform architecture, designed to be easy to deploy and easy to replicate. The ForgeRock Cloud Deployment Team has developed Helm charts, Docker images, and other artifacts expressly to build the CDM.
CloudFormation (AWS)	CloudFormation is a service that helps you model and set up your Amazon Web Services (AWS) resources. You create a template that describes all the AWS resources that you want. AWS CloudFormation takes care of provisioning and configuring those resources for you. Source: *What is AWS CloudFormation?* in the AWS documentation.
CloudFormation template (AWS)	An AWS CloudFormation template describes the resources that you want to provision in your AWS stack. AWS CloudFormation templates are text files formatted in JSON or YAML. Source: *Working with AWS CloudFormation Templates* in the AWS documentation.
cluster	A container cluster is the foundation of Kubernetes Engine. A cluster consists of at least one cluster master and multiple worker machines called nodes. The Kubernetes objects that represent your containerized applications all run on top of a cluster. Source: *Container Cluster Architecture* in the Kubernetes Concepts documentation.
cluster master	A cluster master schedules, runs, scales and upgrades the workloads on all nodes of the cluster. The cluster master also manages network and storage resources for workloads. Source: *Container Cluster Architecture* in the Kubernetes Concepts docuementation.
ConfigMap	A configuration map, called `ConfigMap` in Kubernetes manifests, binds the configuration files, command-line arguments, environment

	variables, port numbers, and other configuration artifacts to the assigned containers and system components at runtime. The configuration maps are useful for storing and sharing non-sensitive, unencrypted configuration information. Source: *ConfigMap* in the Kubernetes Cocenpts documentation.
container	A container is an allocation of resources such as CPU, network I/O, bandwidth, block I/O, and memory that can be "contained" together and made available to specific processes without interference from the rest of the system. Source *Container Cluster Architecture* in the Google Cloud Platform documentation
DaemonSet	A set of daemons, called `DaemonSet` in Kubernetes manifests, manages a group of replicated pods. Usually, the daemon set follows an one-pod-per-node model. As you add nodes to a node pool, the daemon set automatically distributes the pod workload to the new nodes as needed. Source *DaemonSet* in the Google Cloud Platform documentation.
Deployment	A Kubernetes deployment represents a set of multiple, identical pods. A Kubernetes deployment runs multiple replicas of your application and automatically replaces any instances that fail or become unresponsive. Source: *Deployment* in the Google Cloud Platform documentation.
deployment controller	A deployment controller provides declarative updates for pods and replica sets. You describe a desired state in a deployment object, and the deployment controller changes the actual state to the desired state at a controlled rate. You can define deployments to create new replica sets, or to remove existing deployments and adopt all their resources with new deployments. Source: *Deployments* in the Google Cloud Platform documentation.
Docker Cloud	Docker Cloud provides a hosted registry service with build and testing facilities for Dockerized application images; tools to help you set up and manage host infrastructure; and application lifecycle features to automate deploying (and redeploying) services created from images. Source: About Docker Cloud in the Docker Cloud documentation.
Docker container	A Docker container is a runtime instance of a Docker image. A Docker container is isolated from other containers and its host machine. You can control how isolated your container's network,

	storage, or other underlying subsystems are from other containers or from the host machine. Source: Containers section in the Docker architecture documentation.
Docker daemon	The Docker daemon (`dockerd`) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A Docker daemon can also communicate with other Docker daemons to manage Docker services. Source: *Docker daemon* section in the Docker Overview documentation.
Docker Engine	The Docker Engine is a client-server application with these components: • A server, which is a type of long-running program called a daemon process (the `dockerd` command) • A REST API, which specifies interfaces that programs can use to talk to the daemon and tell it what to do • A command-line interface (CLI) client (the `docker` command) Source: Docker Engine section in the Docker Overview documentation.
Dockerfile	A Dockerfile is a text file that contains the instructions for building a Docker image. Docker uses the Dockerfile to automate the process of building a Docker image. Source: *Dockerfile* section in the Docker Overview documentation.
Docker Hub	Docker Hub provides a place for you and your team to build and ship Docker images. You can create public repositories that can be accessed by any other Docker Hub user, or you can create private repositories you can control access to. An image is an application you would like to run. A container is a running instance of an image. Source: *Overview of Docker Hub* section in the Docker Overview documentation.
Docker image	A Docker image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.

	A Docker image includes the application code, a runtime engine, libraries, environment variables, and configuration files that are required to run the application. An image is an application you would like to run. A container is a running instance of an image. Source: *Docker objects* section in the Docker Overview documentation. Hello Whales: Images vs. Containers in Dockers.
Docker namespace	Docker namespaces provide a layer of isolation. When you run a container, Docker creates a set of namespaces for that container. Each aspect of a container runs in a separate namespace and its access is limited to that namespace. The `PID` namespace is the mechanism for remapping process IDs inside the container. Other namespaces such as net, mnt, ipc, and uts provide the isolated environments we know as containers. The user namespace is the mechanism for remapping user IDs inside a container. Source: *Namespaces* section in the Docker Overview documentation.
Docker registry	A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can also run your own private registry. Source: *Docker registries* section in the Docker Overview documentation.
Docker repository	A Docker repository is a public, certified repository from vendors and contributors to Docker. It contains Docker images that you can use as the foundation to build your applications and services. Source: *Repositories on Docker Hub* section in the Docker Overview documentation.
Docker service	In a distributed application, different pieces of the application are called “services.” Docker services are really just “containers in production.” A Docker service runs only one image, but it codifies the way that image runs including which ports to use, the number replicas the container should run, and so on. By default, the services are load-balanced across all worker nodes. Source: *About services* in the Docker Get Started documentation.
dynamic volume provisioning	The process of creating storage volumes on demand is called dynamic volume provisioning. Dynamic volume provisioning allows storage

	volumes to be created on-demand. It automatically provisions storage when it is requested by users. Source: *Dynamic Volume Provisioning* in the Kubernetes Concepts documentation.
egress	An egress controls access to destinations outside the network from within a Kubernetes network. For an external destination to be accessed from a Kubernetes environment, the destination should be listed as an allowed destination in the whitelist configuration. Source: *Network Policies* in the Kubernetes Concepts documentation.
firewall rule	A firewall rule lets you allow or deny traffic to and from your virtual machine instances based on a configuration you specify. Each Kubernetes network has a set of firewall rules controlling access to and from instances in its subnets. Each firewall rule is defined to apply to either incoming glossary-ingress(ingress) or outgoing (egress) traffic, not both. Source: *Firewall Rules Overview* in the Google Cloud Platform documentation.
garbage collection	Garbage collection is the process of deleting unused objects. Kubelets perform garbage collection for containers every minute and garbage collection for images every five minutes. You can adjust the high and low threshold flags and garbage collection policy to tune image garbage collection. Source: *Garbage Collection* in the Kubernetes Concepts documentation.
Google Kubernetes Engine (GKE)	The Google Kubernetes Engine (GKE) is an environment for deploying, managing, and scaling your containerized applications using Google infrastructure. The GKE environment consists of multiple machine instances grouped together to form a container cluster. Source: *Kubernetes Engine Overview* in the Google Cloud Platform documentation.
ingress	An ingress is a collection of rules that allow inbound connections to reach the cluster services. Source: *Ingress* in the Kubernetes Concepts documentation.
instance group	An instance group is a collection of instances of virtual machines. The instance groups enable you to easily monitor and control the group of virtual machines together.

	Source: *Instance Groups* in the Google Cloud Platform documentation.
instance template	An instance template is a global API resource that you can use to create VM instances and managed instance groups. Instance templates define the machine type, image, zone, labels, and other instance properties. They are very helpful in replicating the environments. Source: *Instance Templates* in the Google Cloud Platform documentation.
kubectl	The **kubectl** command-line tool supports several different ways to create and manage Kubernetes objects. Source: *Kubernetes Object Management* in the Kubernetes Concepts documentation.
kube-controller-manager	The Kubernetes controller manager is a process that embeds core controllers that are shipped with Kubernetes. Logically each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process. Source: *kube-controller-manager* in the Kubernetes Reference documentation.
kubelet	A kubelet is an agent that runs on each node in the cluster. It ensures that containers are running in a pod. Source: *kubelets* in the Kubernetes Concepts documentation.
kube-scheduler	The `kube-scheduler` component is on the master node and watches for newly created pods that do not have a node assigned to them, and selects a node for them to run on. Source: *Kubernetes components* in the Kubernetes Concepts documentation.
Kubernetes	Kubernetes is an open source platform designed to automate deploying, scaling, and operating application containers. Source: *Kubernetes Concepts*
Kubernetes DNS	A Kubernetes DNS pod is a pod used by the kubelets and the individual containers to resolve DNS names in the cluster. Source: *DNS for services and pods* in the Kubernetes Concepts documentation.

Kubernetes namespace	A Kubernetes namespace is a virtual cluster that provides a way to divide cluster resources between multiple users. Kubernetes starts with three initial namespaces: • `default`: The default namespace for user created objects which don't have a namespace • `kube-system`: The namespace for objects created by the Kubernetes system • `kube-public`: The automatically created namespace that is readable by all users Kubernetes supports multiple virtual clusters backed by the same physical cluster. Source: *Namespaces* in the Kubernetes Concepts documentation.
Let's Encrypt	Let's Encrypt is a free, automated, and open certificate authority. Source: Let's Encrypt web site.
network policy	A Kubernetes network policy specifies how groups of pods are allowed to communicate with each other and with other network endpoints. Source: *Network policies* in the Kubernetes Concepts documentation.
node (Kubernetes)	A Kubernetes node is a virtual or physical machine in the cluster. Each node is managed by the master components and includes the services needed to run the pods. Source: *Nodes* in the Kubernetes Concepts documentation.
node controller (Kubernetes)	A Kubernetes node controller is a Kubernetes master component that manages various aspects of the nodes such as: lifecycle operations on the nodes, operational status of the nodes, and maintaining an internal list of nodes. Source: *Node Controller* in the Kubernetes Concepts documentation.
persistent volume	A persistent volume (PV) is a piece of storage in the cluster that has been provisioned by an administrator. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins that have a lifecycle independent of any individual pod that uses the PV. Source: *Persistent Volumes* in the Kubernetes Concepts documentation.
persistent volume claim	A persistent volume claim (PVC) is a request for storage by a user. A PVC specifies size, and access modes such as:

	• Mounted once for read and write access • Mounted many times for read-only access Source: *Persistent Volumes* in the Kubernetes Concepts documentation.
pod anti-affinity (Kubernetes)	Kubernetes pod anti-affinity allows you to constrain which nodes can run your pod, based on labels on the **pods** that are already running on the node rather than based on labels on nodes. Pod anti-affinity enables you to control the spread of workload across nodes and also isolate failures to nodes. Source: *Inter-pod affinity and anti-affinity*
pod (Kubernetes)	A Kubernetes pod is the smallest, most basic deployable object in Kubernetes. A pod represents a single instance of a running process in a cluster. Containers within a pod share an IP address and port space. Source: *Understanding Pods* in the Kubernetes Concepts documentation.
replication controller	A replication controller ensures that a specified number of Kubernetes pod replicas are running at any one time. The `replication controller` ensures that a pod or a homogeneous set of pods is always up and available. Source: *ReplicationController* in the Kubernetes Concepts documentation.
secret (Kubernetes)	A Kubernetes secret is a secure object that stores sensitive data, such as passwords, OAuth 2.0 tokens, and SSH keys in your clusters. Source *Secrets* in the Kubernetes Concepts documentation.
security group (AWS)	A security group acts as a virtual firewall that controls the traffic for one or more compute instances. Source: *Amazon EC2 Security Groups* in the AWS documentation.
service (Kubernetes)	A Kubernetes service is an abstraction which defines a logical set of pods and a policy by which to access them. This is sometimes called a microservice. Source: *Services* in the Kubernetes Concepts documentation.
shard	Sharding is a way of partitioning directory data so that the load can be shared by multiple directory servers. Each data partition, also

	known as a *shard*, exposes the same set of naming contexts, but only a subset of the data. For example, a distribution might have two shards. The first shard contains all users whose name begins with A-M, and the second contains all users whose name begins with N-Z. Both have the same naming context. Source: *Class Partition* in the *OpenDJ Javadoc*.
stack (AWS)	A stack is a collection of AWS resources that you can manage as a single unit. You can create, update, or delete a collection of resources by using stacks. All the resources in a stack are defined by the template. Source: *Working with Stacks* in the AWS documentation.
stack set (AWS)	A stack set is a container for stacks. You can provision stacks across AWS accounts and regions by using a single AWS template. All the resources included in each stack of a stack set are defined by the same template. Source: *StackSets Concepts* in the AWS documentation.
volume (Kubernetes)	A Kubernetes volume is a storage volume that has the same lifetime as the pod that encloses it. Consequently, a volume outlives any containers that run within the pod, and data is preserved across container restarts. When a pod ceases to exist, the Kubernetes volume also ceases to exist. Source: *Volumes* in the Kubernetes Concepts documentation.
VPC (AWS)	A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. Source: *What Is Amazon VPC?* in the AWS documentation.
worker node (AWS)	An Amazon Elastic Container Service for Kubernetes (Amazon EKS) worker node is a standard compute instance provisioned in Amazon EKS. Source: *Worker Nodes* in the AWS documentation.
workload (Kubernetes)	A Kubernetes workload is the collection of applications and batch jobs packaged into a container. Before you deploy a workload on a cluster, you must first package the workload into a container. Source: *Understanding Pods* in the Kubernetes Concepts documentation.