NetIQ® Identity Manager

Using Identity Manager Containers in Kubernetes

July 2020

Legal Notice

For information about NetIQ legal notices, disclaimers, warranties, export and other use restrictions, U.S. Government restricted rights, patent policy, and FIPS compliance, see https://www.netiq.com/company/legal/.

Copyright (C) 2020 NetIQ Corporation. All rights reserved.

Contents

About this Book and the Library **5**
About NetIQ Corporation **7**

Part I Overview and Planning **9**

1 Planning Your Container Deployment in Kubernetes **11**

System Requirements . . . 11
Obtaining the Docker Images . . . 11
Loading the Docker Images . . . 11

2 Preparing for Deployment **13**

Prerequisites and Considerations . . . 13
Managing Container Volume Data . . . 14
Generating YAML Files . . . 14

3 Deploying the Identity Manager Containers in Kubernetes **17**

Deploying Identity Manager Engine Container . . . 17
Deploying Remote Loader Container . . . 18
Deploying Fanout Agent Container . . . 18
Generating Certificates With Identity Vault Certificate Authority . . . 18
Generating Certificates for OSP . . . 19
Generating Certificates for Identity Applications . . . 20
Generating Certificates for Identity Reporting . . . 21
Deploying OSP Container . . . 22
Deploying Identity Applications Container . . . 23
Deploying Form Renderer Container . . . 24
Deploying ActiveMQ Container . . . 24
Deploying Identity Reporting Container . . . 25

4 Completing the Deployment **27**

Configuring an NGINX Proxy Service to Expose HTTPS Outside the Cluster . . . 27

About this Book and the Library

This guide provides instructions for deploying the NetIQ Identity Manager (Identity Manager) containers.

Intended Audience

This book is intended for identity architects and identity administrators responsible for installing Identity Manager using containers.

Other Information in the Library

For more information about the library for Identity Manager, see the Identity Manager documentation website.

About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

Our Viewpoint

Adapting to change and managing complexity and risk are nothing new

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

Enabling critical business services, better and faster

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

Our Philosophy

Selling intelligent solutions, not just software

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate—day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

Driving your success is our passion

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with—for a change. Ultimately, when you succeed, we all succeed.

Our Solutions

- Identity & Access Governance
- Access Management
- Security Management
- Systems & Application Management
- Workload Management
- Service Management

Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

Worldwide:	www.netiq.com/about_netiq/officelocations.asp
United States and Canada:	1-888-323-6768
Email:	info@netiq.com
Website:	www.netiq.com

Contacting Technical Support

For specific product issues, contact our Technical Support team.

Worldwide:	www.netiq.com/support/contactinfo.asp
North and South America:	1-713-418-5555
Europe, Middle East, and Africa:	+353 (0) 91-782 677
Email:	support@netiq.com
Website:	www.netiq.com/support

Contacting Documentation Support

Our goal is to provide documentation that meets your needs. The documentation for this product is available on the NetIQ website in HTML and PDF formats on a page that does not require you to log in. If you have suggestions for documentation improvements, click **comment on this topic** at the bottom of any page in the HTML version of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

Contacting the Online User Community

NetIQ Communities, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, NetIQ Communities helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit https://www.netiq.com/communities/.

Overview and Planning

Kubernetes is a container-orchestration system for automating application deployment and management. Kubernetes works with a range of container tools like Docker. For more information, see Kubernetes documentation.

The following Identity Manager components can be deployed in Kubernetes:

- Identity Manager Engine
- Remote Loader
- One SSO Provider (OSP)
- Fanout Agent
- ActiveMQ
- Identity Applications
- Form Renderer
- Identity Reporting

1 Planning Your Container Deployment in Kubernetes

The following sections describe the high-level planning required for a container-based deployment in Kubernetes environment:

- "System Requirements" on page 11
- "Obtaining the Docker Images" on page 11
- "Loading the Docker Images" on page 11

System Requirements

You must ensure that the following requirements are met for deploying the containers:

Software	Certified Versions
Kubernetes	1.18.5 or later

Obtaining the Docker Images

Perform the following steps to on all the Kubernetes nodes:

1. Download the `Identity_Manager_4.8.1_Containers.tar.gz` from the download page.
2. Run the following command to extract the `.tar` file:

   ```
   tar -zxvf Identity_Manager_4.8.1_Containers.tar.gz
   ```

Loading the Docker Images

Perform the following steps on all the Kubernetes nodes:

1. From the location where you have extracted the `Identity_Manager_4.8.1_Containers.tar.gz file`, navigate to the `Identity_Manager_4.8.1_Containers` directory.
2. Load the images by running the following commands:

   ```
   docker load --input IDM_481_identityengine.tar.gz
   docker load --input IDM_481_remoteloader.tar.gz
   docker load --input IDM_481_fanoutagent.tar.gz
   docker load --input IDM_481_osp.tar.gz
   docker load --input IDM_481_identityutils.tar.gz
   docker load --input IDM_481_identityapplication.tar.gz
   docker load --input IDM_481_formrenderer.tar.gz
   ```

```
docker load --input IDM_481_activemq.tar.gz
docker load --input IDM_481_identityreporting.tar.gz
```

2 Preparing for Deployment

The Identity Manager containers deployment process requires pre-installation, installation, and post-installation work. The examples used in this section provides information on deploying containers in an Advanced Edition. However, the containers can also be deployed in Standard edition.

Prerequisites and Considerations

The sample deployment described in this guide is for on-premises deployment of containers in Kubernetes. Ensure that the following considerations are met before deploying containers in Kubernetes.

- Create a cluster setup and identify the Master and Worker nodes.
- Set up a Calico network for communication between pods across Kubernetes nodes. For information about setting up a Calico network, see Calico networking for Kubernetes.
- DNS resolution happens between all the pods in your cluster.
- To ensure that the communication between pods happen through Fully Qualified Domain Name (FQDN), ensure that you update the FQDN entries of all the master and worker nodes on all the nodes where pods will be deployed.
- Based on the Identity Manager components you want to deploy in your environment, ensure that the appropriate container resources like memory, CPU, and so on, are available in all the nodes in your environment.

  For more information about assigning the memory resource, see Kubernetes documentation.

  For more information about assigning the CPU resource, see Kubernetes documentation.
- Ensure that the appropriate volumes are available in each of the nodes.
- Persistent Volume Claim is used for requesting storage space, through the YAML files, for each pods deployed on your nodes.
- Ensure that certificates are available before deploying the pods. For information about generating the certificates, see “Generating Certificates With Identity Vault Certificate Authority” on page 18.
- (Conditional) If you want to establish a communication between an on-premise environment and Kubernetes cluster present in a different environment, set up a Virtual Private Networking (VPN) between Kubernetes cluster and on-premise for secure communication.
- Ensure NGINX service is set up to expose HTTPS outside the Kubernetes cluster environment. For information on setting up the NGINX service in a Kubernetes environment, see Kubernetes documentation.
- Some containers are dependent on others. The following table provides details on those containers that are dependent on other containers.

Table 2-1 Dependent Containers

Container	Dependent containers
OSP	◆ Identity Engine ◆ iManager **NOTE:** The iManager container can be deployed as a standalone container using Docker. For more information, see Deploying iManager Container in the *NetIQ Identity Manager: Deploying Identity Manager 4.8.1 Containers*.
Identity Applications	◆ OSP ◆ Databases for Identity Applications
Form Renderer	Identity Applications
Identity Reporting	◆ Identity Applications ◆ Databases for Identity Reporting

Managing Container Volume Data

Kubernetes supports several mechanisms for data storage and persistence. One such mechanism of persisting container data is by using shared volumes.

The examples used in this guide assumes that you create and use shared volumes. For example, create a shared volume called `/data` on all the Kubernetes nodes.

```
mkdir /data
```

However, you can use other volumes that Docker supports. For more information, see Docker documentation.

NOTE: Ensure that you have read-write permissions for the shared volumes.

Generating YAML Files

In Kubernetes cluster, Identity Manager containers can be deployed in the form of pods using YAML file. YAML is a human-readable language in a text-based format that is mainly used for configuration.

Perform the following steps to create the YAML file:

1. Download the `Identity_Manager_4.8.1_Containers.tar.gz` from the download page.
2. Run the following command to extract the `.tar` file:

   ```
   tar –zxvf Identity_Manager_4.8.1_Containers.tar.gz
   ```
3. Navigate to the `Identity_Manager_4.8.1_Containers` directory.
4. Run the following command to load the image:

   ```
   docker load --input IDM_481_idm_conf_generator.tar.gz
   ```
5. Deploy the container using the following sample command:

```
docker run –it --name=idm_conf_generator -v /data:/config
idm_conf_generator:idm-4.8.1
```

6 Specify the values for the configuration parameters:

NOTE: Ensure that the YAML file is available in the volumized location, for example, `/data`. The `silent.properties` and `secret.properties` files are also created with the YAML file.

Table 2-2 *Silent Installation*

Parameter	Description
Generate inputs for Kubernetes Orchestration	*Applies only if you have selected y in the* **Configure the Silent properties for Docker Containers** *option.* Specifies whether you want to generate the YAML file for Kubernetes.
Directory name with absolute path for creating kube yaml file	*Applies only if you have selected* **y** *in the* **Configure the Silent properties for Docker Containers** *option.* Specifies the path for creating the YAML file for Kubernetes. **NOTE:** It is recommended that you provide different paths for the Identity Applications and Identity Reporting YAML files.
Kubernetes volume mount path	*Applies only if you have selected y in the* **Configure the Silent properties for Docker Containers** *option.* Specifies the path for the Kubernetes volume.
Identity Manager Engine hostname for Kubernetes deployment	*Applies only if you have selected* **y** *in the* **Configure the Silent properties for Docker Containers** *option.* Specifies the hostname of the Identity Manager Engine.

For information on the configuration parameters, see the Understanding the Configuration Parameters in the *NetIQ Identity Manager Setup Guide for Linux*.

NOTE: If you select Y for the Do you want to use the SSPR Docker container prompt, then you must deploy SSPR using the standalone Docker deployment procedure. For more information, see Deploying SSPR Container Deploying SSPR Container in the *NetIQ Identity Manager: Deploying Identity Manager 4.8.1 Containers*.

7 Review the generated YAML files before deploying the pods.

3 Deploying the Identity Manager Containers in Kubernetes

The Identity Manager containers deployment process requires pre-installation, installation, and post-installation work. The examples used in this section provides information on deploying containers in an Advanced Edition. However, the containers can also be deployed in Standard edition.

Some containers are dependent on others. For more information on containers that are dependent on other containers, see Table 2-1 on page 14.

Deploying Identity Manager Engine Container

1. Log in to the Kubernetes master node.
2. Copy the `secret.properties` into the shared volume path, for example, `/data`.

   NOTE: Identity Manager Engine uses host-network mode. Provide the Kubernetes worker node name for deployment of Identity Manager Engine.

3. Run the following command to deploy the container:

   ```
   kubectl create -f <location of the YAML file>
   ```

   For example:

   ```
   kubectl create -f /data/Identity_Manager_Engine.yaml
   ```

4. To verify whether the pod was successfully deployed, check the log file by running the following command:

   ```
   kubectl exec <pod name> -- /usr/bin/tail -f /var/opt/netiq/idm/log/
   idmconfigure.log
   ```

   For example:

   ```
   kubectl exec identity-engine -- /usr/bin/tail -f /var/opt/netiq/idm/log/
   idmconfigure.log
   ```

5. (Conditional) Run the following command to verify the status of the deployed pod:

   ```
   kubectl get pods -o=wide
   ```

6. To check the details of the deployed pod, run the following command:

   ```
   kubectl describe pod <pod name>
   ```

   For example,

   ```
   kubectl describe pod identity-engine
   ```

7. To log in to the pod, run the following command:

   ```
   kubectl exec -it <pod name> -- su nds
   ```

   For example,

   ```
   kubectl exec -it identity-engine -- su nds
   ```

Deploying Remote Loader Container

1. Log in to the Kubernetes master node.
2. Run the following command to deploy the container:

   ```
   kubectl create -f <location of the YAML file>
   ```

   For example:

   ```
   kubectl create -f /data/RemoteLoader.yaml
   ```

3. (Conditional) Run the following command to verify the status of the deployed pod:

   ```
   kubectl get pods -o=wide
   ```

4. To log in to the pod, run the following command:

   ```
   kubectl exec -it <pod name> -- /bin/bash
   ```

   For example,

   ```
   kubectl exec -it remoteloader -- /bin/bash
   ```

5. Configure Remote Loader. For more information, see Configuring the Remote Loader and Drivers in the *NetIQ Identity Manager Driver Administration Guide*.

Deploying Fanout Agent Container

1. Log in to the Kubernetes master node.
2. Run the following command to deploy the container:

   ```
   kubectl create -f <location of the YAML file>
   ```

   For example:

   ```
   kubectl create -f /data/FanOutAgent.yaml
   ```

3. (Conditional) Run the following command to verify the status of the deployed pod:

   ```
   kubectl get pods -o=wide
   ```

4. To log in to the pod, run the following command:

   ```
   kubectl exec -it <pod name> -- /bin/bash
   ```

   For example,

   ```
   kubectl exec -it fanoutagent -- /bin/bash
   ```

5. Configure the Fanout Agent. For more information, see Configuring the Fanout Agent in the *NetIQ Identity Manager Driver for JDBC Fanout Implementation Guide*.

Generating Certificates With Identity Vault Certificate Authority

(Conditional) This section applies only if you are using Identity Vault as the Certificate Authority.

The following components require you to generate certificates before they are deployed. Before you generate the certificates for the following components, ensure that you deploy the Identity Manager Engine and iManager containers.

NOTE: The iManager container can be deployed as a standalone container using Docker. For more information, see Deploying iManager Container in the *NetIQ Identity Manager: Deploying Identity Manager 4.8.1 Containers*.

- OSP
- Identity Applications
- Identity Reporting

Generating Certificates for OSP

Perform the following steps to generate the certificates:

1. Log in to the iManager container.

   ```
   docker exec -it <container name> bash
   ```

   For example,

   ```
   docker exec -it imanager bash
   ```

2. Ensure that you set the Java path. For example, run the following command:

   ```
   export PATH=<java installed location>/bin:$PATH
   ```

   For example,

   ```
   export PATH=/opt/netiq/common/jre/bin/:$PATH
   ```

   NOTE: Ensure that the Java version installed is Azul Zulu 1.80_252 or later.

3. Generate the PKCS keystore:

   ```
   keytool -genkey -alias osp -keyalg RSA -storetype pkcs12 -keystore /config/
   tomcat-osp.ks -validity 3650 -keysize 2048 -dname "CN=osp.example.com" -keypass
   <password> -storepass <password>
   ```

4. Generate a certificate signing request:

   ```
   keytool -certreq -v -alias osp -file /config/osp.csr -keypass <password> -
   keystore /config/tomcat-osp.ks -storepass <password>
   ```

5. Generate a self-signed certificate:

   5a Launch iManager from Kubernetes worker node and log in as an administrator.

   5b Navigate to **Roles and Tasks** > **NetIQ Certificate Server** > **Issue Certificate**.

   5c Browse to the `.csr` file created in step 4. For example, `osp.csr`.

   5d Click **Next**.

   5e Specify the key usage and click **Next**.

   5f For the certificate type, select **Unspecified**.

   5g Click **Next**.

   5h Specify the validity of the certificate and click **Next**.

   5i Select the **File in binary DER format radio** button.

   5j Click **Next**.

   5k Click **Finish**.

   5l Download the certificate and copy the downloaded certificate to the `/data` directory.

6 Export the root certificate in `.der` format:

   6a Launch iManager from Kubernetes worker node and log in as an administrator.

   6b Navigate to **Roles and Tasks** > **NetIQ Certificate Access** > **Server Certificates**.

   6c Select the **SSL CertificateDNS** check box and click **Export**.

   6d In the **Certificates** drop-down list, select the Organizational CA.

   6e In the **Export Format** drop-down list, select `DER`.

   6f Click **Next**.

   6g Download the certificate and copy the downloaded certificate to the `/data` directory.

7 Import the certificates into the PKCS keystore you created in step 2:

```
keytool -import -trustcacerts -alias root -keystore /config/tomcat-osp.ks -file
/config/cert.der -storepass <password> -noprompt
```

```
keytool -import -alias osp -keystore /config/tomcat-osp.ks -file /config/
osp.der -storepass <password> -noprompt
```

NOTE: Ensure that the keystore is available in the path that was specified as an input for deployment.

Generating Certificates for Identity Applications

Perform the following steps to generate the certificates:

1 Log in to the iManager container.

```
docker exec -it <pod name> bash
```

For example,

```
docker exec -it imanager bash
```

2 Ensure that you set the Java path. For example, run the following command:

```
export PATH=<java installed location>/bin:$PATH
```

For example,

```
export PATH=/opt/netiq/common/jre/bin/:$PATH
```

NOTE: Ensure that the Java version installed is Azul Zulu 1.80_252 or later.

---

3 Generate the PKCS keystore:

```
keytool -genkey -alias ua -keyalg RSA -storetype pkcs12 -keystore /config/
tomcat-ua.ks -validity 3650 -keysize 2048 -dname "CN=identityapps.example.com"
-keypass <password> -storepass <password>
```

4 Generate a certificate signing request:

```
keytool -certreq -v -alias ua -file /config/ua.csr -keypass <password> -
keystore /config/tomcat-ua.ks -storepass <password>
```

5 Generate a self-signed certificate:

   5a Log in to iManager as an administrator.

   5b Navigate to **Roles and Tasks** > **NetIQ Certificate Server** > **Issue Certificate**.

   5c Browse to the `.csr` file created in step 4. For example, `ua.csr`.

   5d Click **Next**.

   5e Specify the key usage and click **Next**.

5f For the certificate type, select **Unspecified**.

5g Click **Next**.

5h Specify the validity of the certificate and click **Next**.

5i Select the **File in binary DER format radio** button.

5j Click **Next**.

5k Click **Finish**.

5l Download the certificate and copy the downloaded certificate to the `/data` directory.

6 Export the root certificate in `.der` format:

6a Log in to iManager as an administrator.

6b Navigate to **Roles and Tasks** > **NetIQ Certificate Access** > **Server Certificates**.

6c Select the **SSL CertificateDNS** check box and click **Export**.

6d In the **Certificates** drop-down list, select the Organizational CA.

6e In the Export Format drop-down list, select `DER`.

6f Click **Next**.

6g Download the certificate and copy the downloaded certificate to the `/data` directory.

7 Import the certificates into the PKCS keystore in step 2:

```
keytool -import -trustcacerts -alias root -keystore /config/tomcat-ua.ks -file
/config/cert.der -storepass <password> -noprompt
```

```
keytool -import -alias ua -keystore /config/tomcat-ua.ks -file /config/ua.der -
storepass <password> -noprompt
```

NOTE: Ensure that the certificates are available in the path that was specified as an input for deployment.

Generating Certificates for Identity Reporting

Perform the following steps to generate the certificates:

1 Log in to the iManager container.

```
docker exec -it <pod name> bash
```

For example,

```
docker exec -it imanager bash
```

2 Ensure that you set the Java path. For example, run the following command:

```
export PATH=<java installed location>/bin:$PATH
```

For example,

```
export PATH=/opt/netiq/common/jre/bin/:$PATH
```

NOTE: Ensure that the Java version installed is Azul Zulu 1.80_252 or later.

---

3 Generate the PKCS keystore:

```
keytool -genkey -alias rpt -keyalg RSA -storetype pkcs12 -keystore /config/
tomcat-rpt.ks -validity 3650 -keysize 2048 -dname
"CN=identityreporting.example.com" -keypass <password> -storepass <password>
```

4 Generate a certificate signing request:

```
keytool -certreq -v -alias rpt -file /config/rpt.csr -keypass <password> -
keystore /config/tomcat-rpt.ks -storepass <password>
```

5 Generate a self-signed certificate:

   5a Log in to iManager as an administrator.

   5b Navigate to **Roles and Tasks** > **NetIQ Certificate Server** > **Issue Certificate**.

   5c Browse to the `.csr` file created in step 4. For example, `rpt.csr`.

   5d Click **Next**.

   5e Specify the key usage and click **Next**.

   5f For the certificate type, select **Unspecified**.

   5g Click **Next**.

   5h Specify the validity of the certificate and click **Next**.

   5i Select the **File in binary DER format radio** button.

   5j Click **Next**.

   5k Click **Finish**.

   5l Download the certificate and copy the downloaded certificate to the `/data` directory.

6 Export the root certificate in `.der` format:

   6a Log in to iManager as an administrator.

   6b Navigate to **Roles and Tasks** > **NetIQ Certificate Access** > **Server Certificates**.

   6c Select the **SSL CertificateDNS** check box and click **Export**.

   6d In the **Certificates** drop-down list, select the Organizational CA.

   6e In the Export Format drop-down list, select `DER`.

   6f Click **Next**.

   6g Download the certificate and copy the downloaded certificate to the `/data` directory.

7 Import the certificates into the PKCS keystore you created in step 2:

```
keytool -import -trustcacerts -alias root -keystore /config/tomcat-rpt.ks -file
/config/cert.der -storepass <password> -noprompt
```

```
keytool -import -alias rpt -keystore /config/tomcat-rpt.ks -file /config/
rpt.der -storepass <password> -noprompt
```

NOTE: Ensure that the certificates are available in the path that was specified as an input for deployment.

Deploying OSP Container

1 Log in to the Kubernetes master node.

2 Copy the `secret.properties` file and the OSP Tomcat keystore (`tomcat-osp.ks`) into the data volume path that you specified during the configuration. For example, `/data`.

3 Run the following command to deploy the container:

```
kubectl create -f <location of the YAML file>
```

For example:

```
kubectl create -f /data/OSP_AE.yaml
```

4 To verify whether the pod was successfully deployed, check the log file by running the following command:

```
kubectl exec <pod name> -- /usr/bin/tail -f /var/opt/netiq/idm/log/
idmconfigure.log
```

For example,

```
kubectl exec osp -- /usr/bin/tail -f /var/opt/netiq/idm/log/idmconfigure.log
```

5 (Conditional) Run the following command to verify the status of the deployed pod:

```
kubectl get pods -o=wide
```

6 To check the details of the deployed pod, run the following command:

```
kubectl describe pod <pod name>
```

For example,

```
kubectl describe pod osp
```

7 To log in to the pod, run the following command:

```
kubectl exec -it <pod name> -- /bin/bash
```

For example,

```
kubectl exec -it osp -- /bin/bash
```

Deploying Identity Applications Container

1 Ensure that the Identity Applications database and WorkFlow Engine database are created with proper privileges (by default, `idmadmin` user as owner for both databases) and is accessible from the Identity Application container running remotely on Kubernetes worker node.

2 Log in to the Kubernetes master node.

3 Copy the `secret.properties` file and the Identity Applications Tomcat keystore (`tomcat-ua.ks`) into the data volume path that you specified during the configuration. For example, `/data`.

4 Run the following command to deploy the container:

```
kubectl create -f <location of the YAML file>
```

For example:

```
kubectl create -f /data/UserApplication_AE.yaml
```

5 To verify whether the pod was successfully deployed, check the log file by running the following command:

```
kubectl exec <pod name> -- /usr/bin/tail -f /var/opt/netiq/idm/log/
idmconfigure.log
```

For example,

```
kubectl exec identityapps -- /usr/bin/tail -f /var/opt/netiq/idm/log/
idmconfigure.log
```

6 (Conditional) Run the following command to verify the status of the deployed pod:

```
kubectl get pods -o=wide
```

7 To check the details of the deployed pod, run the following command:

```
kubectl describe pod <pod name>
```

For example,

```
kubectl describe pod identityapps
```

8 Run the following command to log in to the pod:

```
kubectl exec -it <pod name> -- /bin/bash
```

For example,

```
kubectl exec -it identityapps -- /bin/bash
```

9 Run the following command inside the pod:

NOTE: Before performing this step, ensure that the pod is deployed successfully.

```
/opt/netiq/common/jre/bin/keytool -importkeystore -srckeystore /config/tomcat-
osp.ks -srcstorepass novell -destkeystore /opt/netiq/idm/apps/tomcat/conf/
idm.jks -deststorepass novell
```

10 Run the following command to exit the pod.

```
exit
```

11 Restart the pod using the following command:

```
docker restart identityapps
```

Deploying Form Renderer Container

1 Log in to the Kubernetes master node.

2 Copy the `secret.properties` file into the data volume path that you specified during the configuration. For example, `/data`.

3 Run the following command to deploy the container:

```
kubectl create -f <location of the YAML file>
```

For example:

```
kubectl create -f /data/FormRender.yaml
```

4 (Conditional) Run the following command to verify the status of the deployed pod:

```
kubectl get pods -o=wide
```

5 To check the details of the deployed pod, run the following command:

```
kubectl describe pod <pod name>
```

For example,

```
kubectl describe pod formrenderer
```

6 To log in to the pod, run the following command:

```
kubectl exec -it <pod name> -- /bin/bash
```

For example,

```
kubectl exec -it formrenderer -- /bin/bash
```

Deploying ActiveMQ Container

1 Log in to the Kubernetes master node.

2 Run the following command to deploy the container:

```
kubectl create -f <location of the YAML file>
```

For example:

```
kubectl create -f /data/ActiveMQ.yaml
```

3 (Conditional) Run the following command to verify the status of the deployed pod:

```
kubectl get pods -o=wide
```

4 To check the details of the deployed pod, run the following command:

```
kubectl describe pod <pod name>
```

For example,

```
kubectl describe pod activemq
```

5 To log in to the pod, run the following command:

```
kubectl exec -it <pod name> -- /bin/bash
```

For example,

```
kubectl exec -it activemq -- /bin/bash
```

Deploying Identity Reporting Container

1 Ensure that the Identity Reporting database is created with proper privileges (by default, `postgres` user as owner for Identity Reporting database) and is accessible from the Identity Reporting container running remotely on Kubernetes worker node.

2 Log in to the Kubernetes master node.

3 Copy the `secret.properties` file and the Identity Reporting Tomcat keystore (`tomcat-rpt.ks`) into the data volume path that you specified during the configuration. For example, `/data`.

4 Run the following command to deploy the container:

```
kubectl create -f <location of the YAML file>
```

For example:

```
kubectl create -f /data/Reporting_AE.yaml
```

5 To verify whether the pod was successfully deployed, check the log file by running the following command:

```
kubectl exec <pod> -- /usr/bin/tail -f /var/opt/netiq/idm/log/idmconfigure.log
```

For example,

```
kubectl exec identityreporting -- /usr/bin/tail -f /var/opt/netiq/idm/log/
idmconfigure.log
```

6 (Conditional) Run the following command to verify the status of the deployed pod:

```
kubectl get pods -o=wide
```

7 To check the details of the deployed pod, run the following command:

```
kubectl describe pod <pod name>
```

For example,

```
kubectl describe pod identityreporting
```

8 To log in to the pod, run the following command:

```
kubectl exec -it <pod name> -- /bin/bash
```

For example,

```
kubectl exec -it identityreporting -- /bin/bash
```

9 Run the following command:

NOTE: Before performing this step, ensure that the container is deployed successfully.

```
/opt/netiq/common/jre/bin/keytool -importkeystore -srckeystore /config/tomcat-
osp.ks -srcstorepass <password> -destkeystore /opt/netiq/idm/apps/tomcat/conf/
idm.jks -deststorepass <password>
```

10 Run the following command to exit the container.

```
exit
```

11 Restart the pod using the following command:

```
docker restart identityreporting
```

4 Completing the Deployment

After deploying the containers in Kubernetes, you must perform certain tasks to make the containers work properly in your environment.

Configuring an NGINX Proxy Service to Expose HTTPS Outside the Cluster

<TBD>