Oracle® Communications

Cloud Native Environment (OC-CNE) Upgrade Guide

Release 1.4

F29076-01

March 2020

Oracle Communications Cloud Native Environment (OC-CNE) Upgrade Guide, Release 1.4

F29076-01

Contents

List of Figures

List of Tables

1
What's New in This Guide

This document is new in Release 1.4.

2 Introduction

This document details the procedure for upgrading Oracle Communications Signaling, Network Function Cloud Native Environment, referred to in these procedures simply as OCCNE. The intended audiences for this document are Oracle engineers who work with customers to maintain a Cloud Native Environment (CNE) on-site at customer facilities.

Limitations

This document is only used to upgrade from release 1.3.2 to release 1.4.0.

Prerequisites

The customer central repository should be updated with the current OCCNE Images for 1.4.0 and any RPMs and binaries should be updated to the latest versions.

Acronyms

Table 2-1 Acronyms

Term	Definition
CD	Continuous Delivery
OCCNE	Oracle Communications Cloud Native Environment

How to use this document

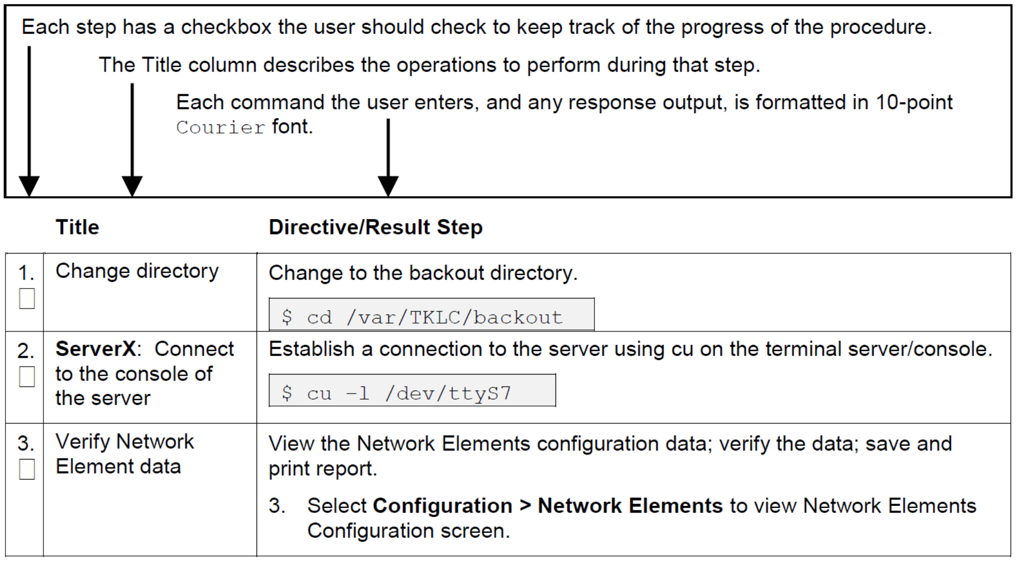
Although this document is primarily to be used as an initial installation guide, its secondary purpose is to be used as a reference for Disaster Recovery procedures.

When executing this document for either purpose, there are a few points which help to ensure that the user understands the author's intent. These points are as follows:

1. Before beginning a procedure, completely read the instructional text (it will appear immediately after the Section heading for each procedure) and all associated procedural WARNINGS or NOTES.
2. Before execution of a STEP within a procedure, completely read the left and right columns including any STEP specific WARNINGS or NOTES.

If a procedural STEP fails to execute successfully, STOP and contact Oracle's Customer Service for assistance before attempting to continue. My Oracle Support for information on contacting Oracle Customer Support.

Figure 2-1 Example of a Procedure Steps Used in This Document



Each step has a checkbox the user should check to keep track of the progress of the procedure.

The Title column describes the operations to perform during that step.

Each command the user enters, and any response output, is formatted in 10-point `Courier` font.

	Title	Directive/Result Step
1. □	Change directory	Change to the backout directory. `$ cd /var/TKLC/backout`
2. □	**ServerX**: Connect to the console of the server	Establish a connection to the server using cu on the terminal server/console. `$ cu –l /dev/ttyS7`
3. □	Verify Network Element data	View the Network Elements configuration data; verify the data; save and print report. 3. Select **Configuration > Network Elements** to view Network Elements Configuration screen.

Documentation Admonishments

Admonishments are icons and text throughout this manual that alert the reader to assure personal safety, to minimize possible service interruptions, and to warn of the potential for equipment damage.

Table 2-2 Admonishments

Icon	Description
DANGER	Danger: (This icon and text indicate the possibility of personal injury.)
WARNING	Warning: (This icon and text indicate the possibility of equipment damage.)
CAUTION	Caution: (This icon and text indicate the possibility of service interruption.)

Locate Product Documentation on the Oracle Help Center Site

Oracle Communications customer documentation is available on the web at the Oracle Help Center site, http://docs.oracle.com. You do not have to register to access these documents. Viewing these files requires Adobe Acrobat Reader, which can be downloaded at http://www.adobe.com.

1. Access the Oracle Help Center site at http://docs.oracle.com.
2. Click **Industries**.
3. Under the Oracle Communications subheading, click **Oracle Communications documentation** link.

   The Communications Documentation page displays.
4. Click on your product and then the release number.

   A list of the documentation set for the selected product and release displays.
5. To download a file to your location, right-click the **PDF** link, select **Save target as** (or similar command based on your browser), and save to a local folder.

Customer Training

Oracle University offers training for service providers and enterprises. Visit our web site to view, and register for, Oracle Communications training at http://education.oracle.com/communication.

To obtain contact phone numbers for countries or regions, visit the Oracle University Education web site at www.oracle.com/education/contacts.

My Oracle Support

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown below on the Support telephone menu:

1. Select **2** for New Service Request.
2. Select **3** for Hardware, Networking and Solaris Operating System Support.
3. Select one of the following options:
   - For Technical issues such as creating a new Service Request (SR), select **1**.
   - For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Emergency Response

In the event of a critical service situation, emergency response is offered by the Customer Access Support (CAS) main number at 1-800-223-1711 (toll-free in the US), or by calling the Oracle Support hotline for your local country from the list at http://

www.oracle.com/us/support/contact/index.html. The emergency response provides immediate coverage, automatic escalation, and other features to ensure that the critical situation is resolved as rapidly as possible.

A critical situation is defined as a problem with the installed equipment that severely affects service, traffic, or maintenance capabilities, and requires immediate corrective action. Critical situations affect service and/or system operation resulting in one or several of these situations:

- A total system failure that results in loss of all transaction processing capability
- Significant reduction in system capacity or traffic handling capability
- Loss of system ability to perform automatic system reconfiguration
- Inability to restart a processor or the system
- Corruption of system databases that requires service affecting corrective actions
- Loss of access for maintenance or recovery operations
- Loss of the system ability to provide any required critical or major trouble notification

Any other problem severely affecting service, capacity/traffic, billing, and maintenance capabilities may be defined as critical by prior discussion and agreement with Oracle.

3
Upgrade Procedures

Overview

The upgrade procedures in this document explain how to setup and perform an upgrade on the Oracle Communications Signaling, Network Function Cloud Native Environment (OCCNE) environment. The upgrade includes the OL7 base image, kubernetes, and the common services.

Pre-upgrade Procedures

Following is the pre-upgrade procedures for OCCNE upgrade:

1. SSH to Bastion Host and start docker using command below after adding the required parameters:

   ```
   $ docker run -u root -d --restart=always -p 8080:8080 -p 50000:50000 -v
   jenkins-data:/var/jenkins_home -v /var/occne/cluster/$
   {OCCNE_CLUSTER}:/var/occne/cluster/${OCCNE_CLUSTER} -v /var/run/
   docker.sock:/var/run/docker.sock ${CENTRAL_REPO}:$
   {CENTRAL_REPO_DOCKER_PORT/jenkinsci/blueocean:1.19.0
   ```

2. Get the administrator password from the Jenkins container to log into the Jenkins user interface running on the Bastion Host.

   a. SSH to the Bastion host and run following command to get the Jenkins docker container ID:

      ```
      $ docker ps | grep 'jenkins' | awk '{print $1}'

      Example output-
      19f6e8d5639d
      ```

   b. Get the admin password from the Jenkins container running as bash. Execute the following command run the container in bash mode:

      ```
      $ docker exec -it <container id from above command> bash
      ```

   c. Run the following command from the Jenkins container while in bash mode. Once complete, capture the password for use later with user-name: admin to log in to the Jenkins GUI.

      ```
      $ cat /var/jenkins_home/secrets/initialAdminPassword

      Example output -
      e1b3bd78a88946f9a0a4c5bfb0e74015
      ```

   d. Execute the following ssh command from the Jenkins container in bash mode after getting the bastion host ip address.

Note: The Bare Metal user is admusr and the vCNE user is cloud-user

```
ssh -t -t -i /var/occne/cluster/<cluster_name>/.ssh/occne_id_rsa
<user>@<bastion_host_ip_address>
```

e. After executing the SSH command the following prompt appears: 'The authenticity of host can't be established. Are you sure you want to continue connecting (yes/no)' enter yes.

f. Exit from bash mode of the Jenkins container (ie. enter exit at the command line).

3. Open the Jenkins GUI in a browser window using url, <bastion-host-ip>:8080 and login using the password from step 3c with user admusr

4. Create a job with an appropriate name after clicking New Item from Jenkins home page. Follow the steps below:

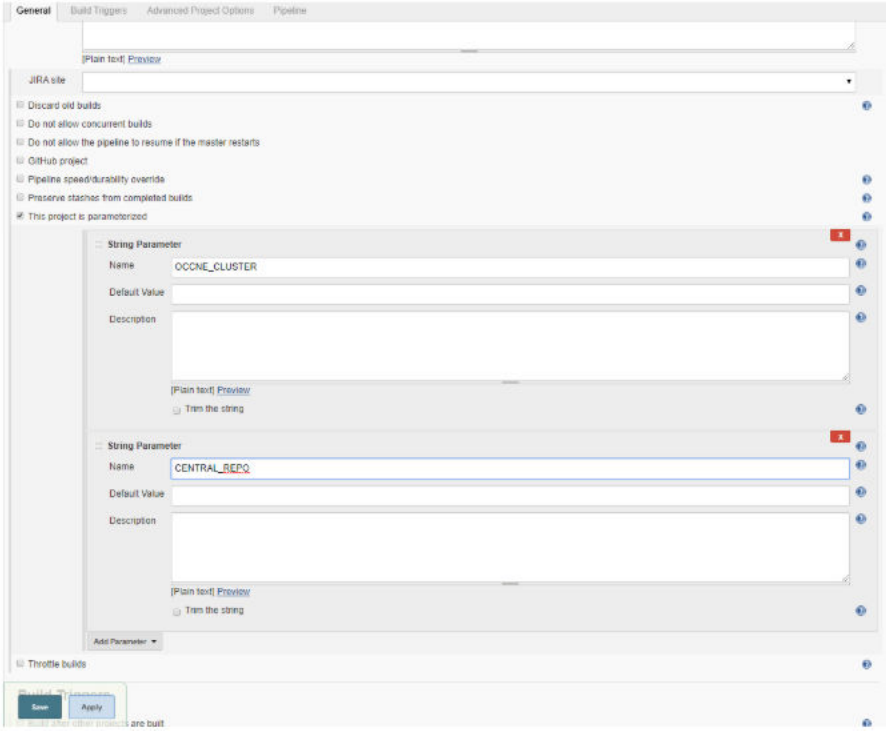
   a. Select **New Item** on the Jenkins home page.

   b. Add a name and select the **Pipeline** option for creating the job

   c. Once the job is created and visible on the Jenkins home page, select **Job**. Select **Configure**.

   d. Add parameters OCCNE_CLUSTER and CENTRAL_REPO from configure screen. Two String Parameter dialogs will appear, one for OCCNE_CLUSTER and one for OCCNE_REPO. Add default value for the Default Value field in the OCCNE_CLUSTER dialog and the Default Value field in theCENTRAL_REPO dialog.

Figure 3-1   Jenkins UI



JIRA site

□ Discard old builds
□ Do not allow concurrent builds
□ Do not allow the pipeline to resume if the master restarts
□ GitHub project
□ Pipeline speed/durability override
□ Preserve stashes from completed builds
☑ This project is parameterized

String Parameter
Name: OCCNE_CLUSTER
Default Value:
Description:
□ Trim the string

String Parameter
Name: CENTRAL_REPO
Default Value:
Description:
□ Trim the string

□ Throttle builds

□ Build after other projects are built

   e. Copy the following configuration to the pipeline script section in the Configure page of the Jenkins job with substituting values for upgrade-image-version, OCCNE_CLUSTER , CENTRAL_REPO, and CENTRAL_REPO_DOCKER_PORT. All of these values should be known to the user.

```
node ('master') {

    sh "docker run -i --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e ANSIBLE_NOCOLOR=1 ${CENTRAL_REPO}:$
{CENTRAL_REPO_DOCKER_PORT}/occne/provision:<upgrade-image-version>
cp deploy_upgrade/JenkinsFile /host/artifacts"
    sh "docker run -i --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e ANSIBLE_NOCOLOR=1 ${CENTRAL_REPO}:$
{CENTRAL_REPO_DOCKER_PORT}/occne/provision:<upgrade-image-version>
cp deploy_upgrade/upgrade_services.py /host/artifacts"
    load '/var/occne/cluster/<cluster-name>/artifacts/JenkinsFile'
}
```

f. Select **Apply** and **Save**

g. Go back to the **Job** page and select **Build with Parameter** for the new pipeline script to be enabled for the job, this job will be aborted.

h. Select **Build with Parameters** option to see latest Jenkins file parameters in the GUI

Setup admin.conf

Execute the pre-upgrade admin.conf setup:

> **Note:**
>
> Only for upgrade from 1.3.2 to 1.4.0.

1. Add following entries to /etc/hosts file on Bastion Host, master and worker nodes. Add all the master internal ip's with name lb-apiserver.kubernetes.local:

```
<master-internal-ip-1> lb-apiserver.kubernetes.local
<master-internal-ip-2> lb-apiserver.kubernetes.local
<master-internal-ip-3> lb-apiserver.kubernetes.local


Example for 3 master nodes-

172.16.5.6 lb-apiserver.kubernetes.local
172.16.5.7 lb-apiserver.kubernetes.local
172.16.5.8 lb-apiserver.kubernetes.local
```

2. Login to bastion host and get dependencies for 1.4.0 latest pipeline:

```
$ docker run -it --rm -v /var/occne/cluster/<cluster-name>:/host -e
ANSIBLE_NOCOLOR=1 -e 'OCCNEARGS= '
<central_repo>:<central_repo_docker_port>/occne/
provision:<upgrade_image_version> /getdeps/getdeps
```

```
Example:
$ docker run -it --rm -v /var/occne/cluster/delta:/host -e
ANSIBLE_NOCOLOR=1 -e 'OCCNEARGS= ' winterfell:5000/occne/provision:
1.4.0 /getdeps/getdeps
```

a. Execute below command on Bare Metal Cluster:

   ```
   $ docker run -it --rm --cap-add=NET_ADMIN --network host -v /var/
   occne/cluster/<cluster-name>:/host -v /var/occne:/var/occne:rw -e
   'OCCNEARGS= --tags=pre_upgrade  '
   <central_repo>:<central_repo_docker_port>/occne/provision:1.4.0

   Example:
   $ docker run -it --rm --cap-add=NET_ADMIN --network host -v /var/
   occne/cluster/delta:/host -v /var/occne:/var/occne:rw -e
   'OCCNEARGS= --tags=pre_upgrade  ' winterfell:5000/occne/provision:
   1.4.0
   ```

b. Execute below command on VCNE Cluster:

   ```
   $ docker run -it --rm --cap-add=NET_ADMIN --network host -v /var/
   occne/cluster/<cluster-name>:/host -v /var/occne:/var/occne:rw -e
   OCCNEINV=/host/terraform/hosts -e 'OCCNEARGS= --tags=pre_upgrade  --
   extra-
   vars={"occne_vcne":"1","occne_cluster_name":"<occne_cluster_name>","
   occne_repo_host":"<occne_repo_host_name>","occne_repo_host_address":
   "<occne_repo_host_address>"} '
   <central_repo>:<central_repo_port_name>/occne/
   provision:<upgrade_image_version>

   Example:
   $ docker run -it --rm --cap-add=NET_ADMIN --network host -v /var/
   occne/cluster/delta:/host -v /var/occne:/var/occne:rw -e OCCNEINV=/
   host/terraform/hosts -e 'OCCNEARGS= --tags=pre_upgrade  --extra-
   vars={"occne_vcne":"1","occne_cluster_name":"ankit-
   upgrade-3","occne_repo_host":"ankit-upgrade-3-
   bastion-1","occne_repo_host_address":"192.168.200.9"} ' winterfell:
   5000/occne/provision:1.4.0
   ```

   Run kubectl get nodes command to verify above changes were applied correctly.

c. Run following command on all the master/ worker nodes:

   ```
   yum clean all
   ```

Upgrading K8s container engine from Docker to Containerd

This section explains the procedure to upgrade K8s container engine from docker to container.
Note: This step is only for execution from 1.3.2 to 1.4.0 where kube version is same but there is a change to container engine for cluster, this step should be removed for future upgrade procedure.

1. Get k8s dependencies for 1.4.0 k8s upgrade for containerd on Bastion Host

   ```
   Example-
   ANSIBLE_NOCOLOR=1 OCCNE_VERSION= K8S_IMAGE=winterfell:5000/occne/
   k8s_install:1.4.0 CENTRAL_REPO=winterfell K8S_ARGS="" K8S_SKIP_TEST=1
   K8S_SKIP_DEPLOY=1  /var/occne/cluster/<cluster-name>/artifacts/
   pipeline.sh
   ```

2. Create upgrade_container.yml in /var/occne/cluster/<cluster_name> directory with contents below:

   ```
   - hosts: k8s-cluster
     tasks:
     - name: Switch Docker container runtime to containerd
       shell: "{{ item }}"
       with_items:
         - "sudo cp /etc/cni/net.d/calico.conflist.template 10-containerd-
   net.conflist"
         - "systemctl daemon-reload"
         - "systemctl enable containerd"
         - "systemctl restart containerd"
         - "systemctl stop docker"
         - "systemctl daemon-reload"
         - "systemctl restart kubelet"
         - "sudo yum remove -y docker-ce"
       ignore_errors: yes
   ```

3. Run k8s install in bash mode to update container engine from docker to containerd
   Bare Metal Clusters

   ```
   docker run -it --rm --cap-add=NET_ADMIN --network host -v /var/occne/
   cluster/<cluster-name>:/host -v /var/occne:/var/occne:rw -e
   ANSIBLE_NOCOLOR=1 -e 'OCCNEARGS=       ' winterfell:5000/occne/
   k8s_install:1.4.0 bash
   ```

   VCNE Clusters

   ```
   // Get Values from Cloud Config
   Example-
   docker run -it --rm --cap-add=NET_ADMIN --network host -v /var/occne/
   cluster/<cluster-name>:/host -v /var/occne:/var/occne:rw -e OCCNEINV=/
   host/terraform/hosts -e 'OCCNEARGS=--extra-
   vars={"occne_vcne":"1","occne_cluster_name":"ankit-
   upgrade-3","occne_repo_host":"ankit-upgrade-3-
   bastion-1","occne_repo_host_address":"192.168.200.9"} --extra-
   vars={"openstack_username":"ankit.misra","openstack_password":"{Cloud-
   Password}","openstack_auth_url":"http://thundercloud.us.oracle.com:5000/
   v3","openstack_region":"RegionOne","openstack_tenant_id":"811ef89b5f154a
   b0847be2f7e41117c0","openstack_domain_name":"LDAP","openstack_lbaas_subn
   et_id":"2787146b-56fe-4c58-
   bd87-086856de24a9","openstack_lbaas_floating_network_id":"e4351e3e-81e3-
   4a83-bdc1-
   dde1296690e3","openstack_lbaas_use_octavia":"true","openstack_lbaas_meth
   ```

```
od":"ROUND_ROBIN","openstack_lbaas_enabled":true}      ' winterfell:
5000/occne/k8s_install:<image_tag> bash
```

Below steps are common once in bash docker mode for both vcne and bare metal:

```
sed -i /kubespray/roles/bootstrap-os/tasks/bootstrap-oracle.yml -re '2,
16d'
sed -i /kubespray/roles/kubernetes-apps/ingress_controller/cert_manager/
tasks/main.yml -re '3, 58d'
//  The command runs the playbook to add configuration files for
containerd

/copyHosts.sh ${OCCNEINV} && ansible-playbook -i /kubespray/inventory/
occne/hosts \
    --become \
    --become-user=root \
    --private-key /host/.ssh/occne_id_rsa \
    /kubespray/cluster.yml ${OCCNEARGS}


// Once done run the upgrade_container in bash mode below.
// Around a 2 -3 minute timeout for some services may occur depending
on how quickly the next command is executed.


/copyHosts.sh ${OCCNEINV} && ansible-playbook -i /kubespray/inventory/
occne/hosts \
    --become \
    --become-user=root \
    --private-key /host/.ssh/occne_id_rsa \
    /host/upgrade_container.yml

// Note : There will be a prompt during running above task on vcne that
calico.conflist.template does not exist, this is because flannel is
used rather then calico. Prompt will be skipped for vcne
```

Wait for all pods to become ready with 1/1 and status as running. This can be done by executing kubectl get pods. Run next steps after confirming all pods are ready , running.

4. Test to check all containers are managed by containerd:

```
// Login into any node of the cluster to see all the containers are
managed by crictl
sudo /usr/local/bin/crictl ps
```

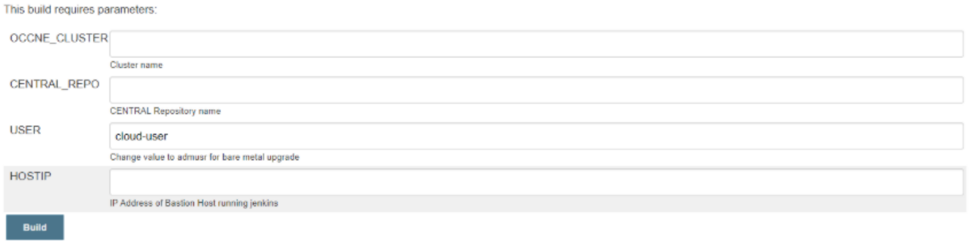
Upgrading OCCNE

Following is the procedure to upgrade OCCNE.

1. Click the job name created in the previous step. Select the **Build with Parameters** option on the left top corner panel in the Jenkins GUI as shown below:

2. On selecting the **Build with Parameters** option, there will be a list of parameters with a description describing which values need to be used for the Bare-Metal upgrade vs the vCNE upgrade.
   Note: Change **USER** to admusr before upgrading bare-metal cluster

   Figure 3-2   Jenkins UI: Build with Parameters

   This build requires parameters:

   OCCNE_CLUSTER
   Cluster name

   CENTRAL_REPO
   CENTRAL Repository name

   USER
   cloud-user
   Change value to admusr for bare metal upgrade

   HOSTIP
   IP Address of Bastion Host running jenkins

3. After entering correct values for parameters, select Build to start upgrade.
4. Once the build has started, go to the job home page to see the live console for upgrade. This can be done in two ways: Either select **console output** or **Open Blue Ocean** (Recommended is blue ocean as it will show each stage of upgrade)
5. Check the job progress from **Blue Ocean** link in the job to see each stage being executed, once upgrade is complete all the stages will be in **Green**.
   Make sure before running next step that docker registry name and host name are same, if not run:

   ```
   vi /var/occne/cluster/<cluster-name>/artifacts/upgrade_services.py
   and change hostname= os.environ['HOSTNAME'] in the script to
   hostname= '<bastion-registry-name>'
   ```

6. After a successful upgrade, run following commands to upgrade the major version for the common services:

   ```
   cd /var/occne/cluster/<cluster-name>/artifacts
   python -c "execfile('upgrade_services.py');
   patch_random_named_pods_image()"
   // Wait for all the pods to be patched, run kubectl get pods to verify
   all the pods are running and ready flag is 1/1
   python -c "execfile('upgrade_services.py'); patch_pods_image()"
   // Wait for all the pods to be patched, run kubectl get pods to verify
   all the pods are running and ready flag is 1/1
   ```

7. To check if all the elastic search data pods have restarted, run command below, sometimes it can take time for a single pod to restart:

   ```
   kubectl get pods -n occne-infra | grep data
   Sample output:
   NAME
   READY   STATUS      RESTARTS   AGE
   occne-elastic-elasticsearch-data-0
   1/1     Running     0          2d1h
   ```

8. Make sure that restarts value has changed to 1 from 0 after running the python command from code block above.

9. Wait till data pods restart is done, if not get information for node running the data pods and stop elastic search data containers using the following command:

   ```
   $ sudo /usr/local/bin/crictl stop containerid
   ```

   Verify all elasticsearch-data-<count> pods are with ready 1/1 , status = running and Restarts =1.

10. Once all the elastic search data pods have restarted, run the command mentioned below:

    ```
    $ python -c "execfile('upgrade_services.py');
    patch_deployment_image_chart()"
    ```

4

Post Upgrade Health Checks

Perform Post Upgrade Health Checks

The health check procedure is as follows:

Run command below and verify all the pods in namespace **occne-infra** are in running status. All the pods from the list should have status as Running and READY value set to 1/1.

```
kubectl get pods -n occne-infra
Sample output:
NAME                                                          READY
STATUS      RESTARTS   AGE
occne-elastic-elasticsearch-data-0                            1/1
Running     0          2d1h



All the pods from the list should have status as Running and READY value
set to 1/1
```