Oracle® Communications
Cloud Native Environment (OC-CNE)
Upgrade Guide

Release 1.6.0
F33966-01
September 2020

Oracle Communications Cloud Native Environment (OC-CNE) Upgrade Guide, Release 1.6.0

F33966-01

Contents

List of Figures

List of Tables

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.

What's New in This Guide

This section introduces the documentation updates for Release 1.6.x in Oracle Communications Cloud Native Environment (OCCNE) Upgrade Guide.

New and Updates Features in Release 1.6.0

For Release 1.6.0, the following changes are performed in the document:

- Updated Pre-upgrade Procedures section to be executed prior to upgrade.
- Added the following sections:
  - Post-Upgrade Procedures
  - Reference Procedures

1 Introduction

Oracle Communications Cloud Native Environment (OCCNE) is in essence infrastructure code that provisions, configures, and manages reference cloud native environments.

This document details the procedure for upgrading the 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.

Acronyms

Table 1-1 Acronyms

Term	Definition
CD	Continuous Delivery
OCCNE	Oracle Communications Cloud Native Environment

2
Upgrading OCCNE

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

Prerequisites

Following are the prerequisites for upgrading OCCNE:

- The Preserving MetalLB Configuration During Upgrade procedure must be executed prior to upgrade.
- While upgrading MYSQL NDB on the second site the Mate Site DB Replication Service Load Balancer IP must be provided as the configuration parameter for the geo-replication process to continue. Login to Bastion Host of the first site and execute the following command to retrieve DB Replication Service Load Balancer IP.

  ```
  $ kubectl get svc --namespace=occne-infra | grep replication
  ```

  Example:

  ```
  $ kubectl get svc --namespace=occne-infra | grep replication
  occne-db-replication-svc     LoadBalancer   10.233.3.117
  10.75.182.88     80:32496/TCP   2m8s
  ```

  In the above example IPv4: 10.75.182.88 is the Mate Site DB Replication Service Load Balancer IP.
- The customer central repository should be updated with the current OCCNE Images for 1.6.0 and any RPMs and binaries should be updated to the latest versions.
- Copy V987059-01.zip (MySQL Cluster Manager 1.4.8+Cluster TAR for Oracle Linux / RHEL 7 x86 (64bit), 557.6 MB downloaded from OSDC) to "/var/occne" directory on bastion host.
- Ensure that cluster is in healthy state by checking that all the pods are ready and running. Execute the following command and verify that all pods are in completed/ running status. The pods from the list should have status as **Running** and **READY** value set to **x/x** (or) as **Completed** and **READY** value set to **0/x**.

  ```
  kubectl get pods --all-namespaces
  ```

Example:

```
NAMESPACE      NAME                                       READY
STATUS    RESTARTS   AGE
cert-manager   cert-manager-77fb98dc45-7ch6b              1/1
Running     0        8d
kube-system    calico-kube-controllers-7df59b474d-r4f7z 1/1
Running     0        8d
kube-system    calico-node-6cvhp                          1/1
Running     1        8d
...
...
...
occne-infra    occne-elastic-elasticsearch-client-0     1/1
Running     0        6d8h
occne-infra    occne-elastic-elasticsearch-client-1     1/1
Running     0        6d8h
occne-infra    occne-elastic-elasticsearch-client-2     1/1
Running     0        6d8h

...
...
...
```

Pre-upgrade Procedures

Following is the pre-upgrade procedure for OCCNE:

1. All NFs must be upgraded before OCCNE upgrade. Execute Installing Network Functions procedure for all NFs that have upgrades available. This procedure includes steps to update NF-specific alerts.
2. The below procedure needs to be executed to track and save the changes which can be reapplied after upgrade to keep the SNMP running after upgrade:
   a. If trap receiver (snmp.destination) for occne-snmp-notifier is modified after installation then the IP details must be saved so that after upgrade it can be reassigned.

      ```
      $ kubectl get  deployment occne-snmp-notifier -n occne-infra -o
      yaml
      ```

      Search for `- --snmp.destination=<trap receiver ip address>:162`. Copy the IP and save it for future reference.

   b. Execute the following command to determine whether multiple SNMP notifiers are configured or not:

      ```
      $ kubectl get pods --all-namespaces | grep snmp
      occne-infra occne-snmp-notifier-1-f4d4876c7-hxnkb  1/1  Running
      0   44h
      occne-infra occne-snmp-notifier-6b99997bfd-r59t7   1/1  Running
      0   43h
      ```

c. If multiple SNMP notifiers are created then alert manager configmap must be saved before upgrade, so that the config can be reapplied after upgrade by following the post upgrade steps:

```
$ kubectl get configmap occne-prometheus-alertmanager -n occne-
infra -o yaml
apiVersion: v1
data:
  alertmanager.yml: |
    global: {}
    receivers:
    - name: default-receiver
      webhook_configs:
      - url: http://occne-snmp-notifier:9464/alerts
    - name: test-receiver-1
      webhook_configs:
      - url: http://occne-snmp-notifier-1:9465/alerts
    route:
      group_interval: 5m
      group_wait: 10s
      receiver: default-receiver
      repeat_interval: 3h
      routes:
      - receiver: default-receiver
        group_interval: 1m
        group_wait: 10s
        repeat_interval: 9y
        group_by: [instance, alertname, severity]
        continue: true
      - receiver: test-receiver-1
        group_interval: 1m
        group_wait: 10s
        repeat_interval: 9y
        group_by: [instance, alertname, severity]
        continue: true
kind: ConfigMap
```

3. Any existing customer specific dashboad(s) must be saved to a local directory so that it can be restored after the upgrade. Log into the Grafana GUI to backup dashboard:
   a. Select the dashboard to be saved.
   b. Go to **Shared Dashboard** option on the top-right side of the dashboard that needs to be saved.
   c. Click **Export**. Click **Save to file** to save the file in the local repository.
   d. Repeat these steps until all customer specific dashboards have been saved.
4. Run k8s install pre-upgrade script to switch network plugin from flannel to calico ***(only applicable for vCNE cluster)***
   a. Execute below command on bastion to run k8s getdeps:

```
$ docker run -it --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e 'OCCNEARGS=--extra-vars={"occne_vcne":"1"}   '
winterfell:5000/occne/k8s_install:<image_tag> /getdeps/getdeps
```

```
Example:
$ docker run -it --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e 'OCCNEARGS=--extra-vars={"occne_vcne":"1"}   '
winterfell:5000/occne/k8s_install:1.6.0 /getdeps/getdeps
```

b. Execute below commands on bastion to fetch k8s related binaries and docker images:

> **Note:**
>
> These commands can be executed as written below since the indicated environment variables have already been set on initial deployment. They can be verified by using the linux command: `echo $<variable_name>`.

```
$ /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/
k8s_retrieve_bin.sh http://${CENTRAL_REPO}/occne/
binaries /var/www/html/occne
$ /var/occne/cluster/${OCCNE_CLUSTER}/artifacts/
retrieve_docker.sh winterfell:5000 ${HOSTNAME%%.*}:5000 < /var/
occne/cluster/${OCCNE_CLUSTER}/artifacts/k8s_docker_images.txt
```

c. Execute below command on bastion to trigger the network plugin upgrade from flannel to calico:

> **Note:**
>
> Make sure the *openstack_lbaas_floating_network_id* field is set to the floating IP network ID and the *openstack_lbaas_subnet_id* field is set to the user specific internal network subnet ID. Openstack values can be obtained by executing the command: **openstack configuration show** from the openstack client.

```
// Get values from Cloud config
$ 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":"<occne_cluster_name>
","occne_repo_host":"<occne_repo_host_name>","occne_repo_host_add
ress":"<occne_repo_host_address>"} --extra-
vars={"openstack_username":"<user.name>","openstack_password":"<o
penstack-cloud-
password>","openstack_auth_url":"<openstack_auth_url>","openstack
_region":"RegionOne","openstack_tenant_id":"<openstack_tenant_id>
","openstack_domain_name":"LDAP","openstack_lbaas_subnet_id":"<op
enstack_lbaas_subnet_id>","openstack_lbaas_floating_network_id":"
<openstack_lbaas_floating_network_id>","openstack_lbaas_use_octav
ia":"true","openstack_lbaas_method":"ROUND_ROBIN","openstack_lbaa
s_enabled":true}      ' winterfell:5000/occne/
```

```
k8s_install:<image_tag> /upgrade/pre-upgrade.sh

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":"{C
loud-Password}","openstack_auth_url":"http://
thundercloud.us.oracle.com:5000/
v3","openstack_region":"RegionOne","openstack_tenant_id":"811ef89
b5f154ab0847be2f7e41117c0","openstack_domain_name":"LDAP","openst
ack_lbaas_subnet_id":"2787146b-56fe-4c58-
bd87-086856de24a9","openstack_lbaas_floating_network_id":"e4351e3
e-81e3-4a83-bdc1-
dde1296690e3","openstack_lbaas_use_octavia":"true","openstack_lba
as_method":"ROUND_ROBIN","openstack_lbaas_enabled":true}      '
winterfell:5000/occne/k8s_install:1.6.0 /upgrade/pre-upgrade.sh
```

d. Wait for all pods to become ready with 1/1 and status as running. This can be done by executing the following command from the Bastion Host:

```
$ kubectl get pod -A
```

5. Execute the following commands from the Bastion Host to pass the OCCNE_CLUSTER Bastion Host environment variable to the Jenkins container. This applies to both vCNE and Bare Metal:

```
$ docker stop occne_jenkins
$ docker rm -f occne_jenkins
$ docker run -u root -d --name occne_jenkins
--restart=always -p 8080:8080 -p 50000:50000
-e OCCNE_CLUSTER=${OCCNE_CLUSTER} -v jenkins-data:/var/
jenkins_home -v {{ cluster_dir }}:{{ cluster_dir }} -v /var/run/
docker.sock:/var/run/docker.sock {{ central_repo_hostname }}:
{{ central_repo_docker_port}}/jenkinsci/blueocean:{{ jenkins_tag }}
```

Example:

```
$ docker run -u root -d --name occne_jenkins --restart=always
-p 8080:8080 -p 50000:50000 -e OCCNE_CLUSTER=${OCCNE_CLUSTER} -v
jenkins-data:/var/jenkins_home -v /var/occne/cluster/john-doe:/var/
occne/cluster/john-doe -v /var/run/docker.sock:/var/run/docker.sock
winterfell:5000/jenkinsci/blueocean:1.19.0
```

6. Execute the below command on bastion host to run provision getdeps for generating script pipeline.sh in artifacts directory:

a. For bare metal cluster:

```
$ docker run -it --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e ANSIBLE_NOCOLOR=1 -e OCCNEARGS='' winterfell:5000/occne/
provision:<image-tag> /getdeps/getdeps
```

Example:

```
$ docker run -it --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host  -e ANSIBLE_NOCOLOR=1 -e OCCNEARGS='' winterfell:5000/occne/
provision:1.6.0 /getdeps/getdeps
```

b. For vCNE cluster:

```
$ docker run -it --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e 'OCCNEARGS=--extra-vars={"occne_vcne":"1"}   '
winterfell:5000/occne/provision:<image-tag> /getdeps/getdeps
```

Example:

```
$ docker run -it --rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/
host -e 'OCCNEARGS=--extra-vars={"occne_vcne":"1"}   '
winterfell:5000/occne/provision:1.6.0 /getdeps/getdeps
```

7. 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 to 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 later use 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 internal address:
Note: The Bare Metal user is admusr and the vCNE user is cloud-user.

```
$ ssh -t -t -i /var/occne/cluster/${OCCNE_CLUSTER}/.ssh/
occne_id_rsa <user>@<bastion_host_ip_address>

Example (for vCNE):
$  ssh -t -t -i /var/occne/cluster/${OCCNE_CLUSTER}/.ssh/
occne_id_rsa cloud-user@192.168.200.17
```

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 (that is, enter exit at the command line).

8. Open the Jenkins GUI in a browser window using url, <bastion-host-ip>:8080. Login using the password from step 7c with admin.

> **Note:**
>
> You may be prompted to enter proxy configurations or skip the plugin configurations. Select Skip Plugin Configuration". You will then be prompted to configure the first admin user. Creating a new admin user here is optional. Use credentials: username admin, password:admin, Full name: admin, and email address: admin@<domain.com>. The Full name and email address do not have to be valid values.
> You may also get the following Unlock Jenkins page. If this page is displayed, enter the password from step 7c.

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

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

b. Add a name (such as **upgrade**) and select the **Pipeline** option for creating the job.

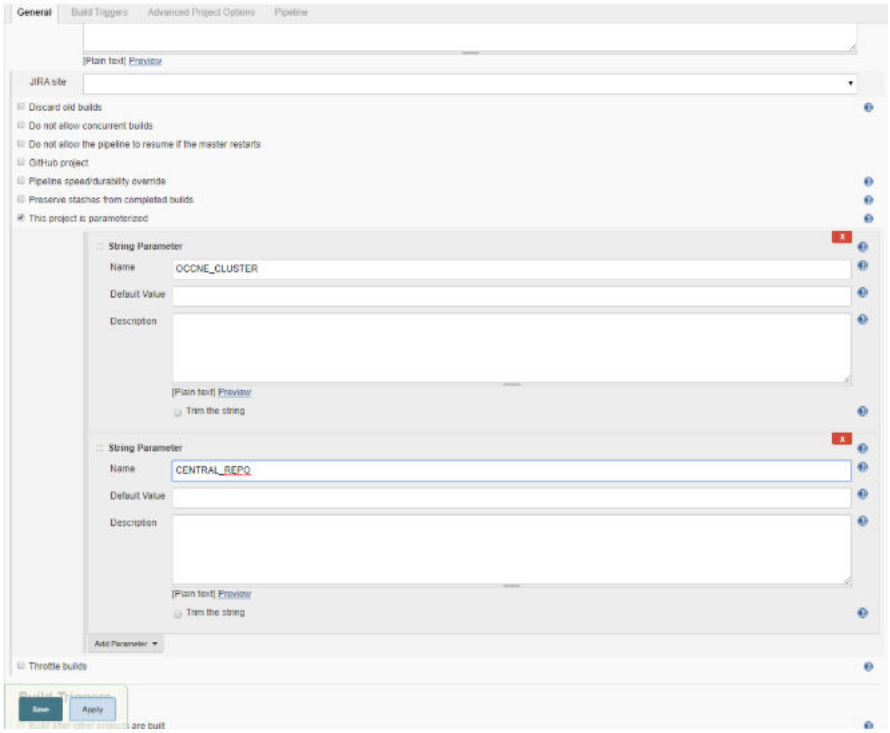
c. This brings up the **Configure** page (as displayed in step e below) with the **General** tab selected. If the **Configure** page is displayed, skip the next step and go to step e.

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

e. This brings up the Configure page and allows the user to add parameters to the configuration. Select **This project is parameterized** checkbox. This will display the following screen with the **Add Parameter** button visible. Select the **Add Parameter** button twice (after the first string parameter is added, the Add Parameter button will be just below the first parameter dialog box) and add two string parameters using the **Select String Parameter** menu item.

f. Add parameters **OCCNE_CLUSTER** and **CENTRAL_REPO** from configure screen. Two **String Parameter** dialogs will appear, one for **OCCNE_CLUSTER** and one for **CENTRAL_REPO**. Enter values for the **Default Value** fields in the **OCCNE_CLUSTER** dialog and **CENTRAL_REPO** dialog.

Figure 2-1 Jenkins UI



g. Copy the following configuration to the pipeline script section in the **Configure** page of the Jenkins job by manually substituting values for <upgrade_image_version> and <central_repo_docker_port>.

> **Note:**
>
> There are two examples below. Use the appropriate version depending on whether your Openstack Provider uses credentials or certificates.

i. For deployment on openstack non certificate authentication environment:

```
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"
    load '/var/occne/cluster/<cluster-name>/artifacts/
JenkinsFile'
}
```

Example:

```
node ('master') {
    sh "docker run -i --rm -v /var/occne/
cluster/${OCCNE_CLUSTER}:/host -e ANSIBLE_NOCOLOR=1 $
{CENTRAL_REPO}:5000/occne/provision:1.6.0 cp deploy_upgrade/
JenkinsFile /host/artifacts"
    load '/var/occne/cluster/occne3-john-doe/artifacts/
JenkinsFile'
}
```

ii. For deployment on openstack certificate authentication environment:

```
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>
sed -i 's/\${env.openstack_domain_name}\\\\\
\\\\\\\"}/\${env.openstack_domain_name}\\\\\\\\\\\\\\",\\\\\\
\\\\\\"openstack_cacert\\\\\\\\\\\\\":\\\\\\\\\\\\\"\\/host\
\/openstack-cacert.pem\\\\\\\\\\\\\"}/g' /host/artifacts/
JenkinsFile"
    load '/var/occne/cluster/<cluster-name>/artifacts/
JenkinsFile'
}
```

Example:

```
node ('master') {
    sh "docker run -i --rm -v /var/occne/
cluster/${OCCNE_CLUSTER}:/host -e ANSIBLE_NOCOLOR=1 $
{CENTRAL_REPO}:5000/occne/provision:1.6.0 cp deploy_upgrade/
JenkinsFile /host/artifacts"
    sh "docker run -i --
rm -v /var/occne/cluster/${OCCNE_CLUSTER}:/host
-e ANSIBLE_NOCOLOR=1 ${CENTRAL_REPO}:5000/occne/
provision:1.6.0 sed -i 's/\${env.openstack_domain_name}\\\\\
\\\\\\\"}/\${env.openstack_domain_name}\\\\\\\\\\\\\\",\\\\\\
\\\\\\"openstack_cacert\\\\\\\\\\\\\":\\\\\\\\\\\\\"\\/host\
\/openstack-cacert.pem\\\\\\\\\\\\\"}/g' /host/artifacts/
JenkinsFile"
    load '/var/occne/cluster/occne3-john-doe/artifacts/
JenkinsFile'
}
```

h. Select **Apply** and **Save**.

i. Go back to the **Job** page and select **Build with Parameters** to see the new parameters added from the Jenkins desktop.

j. Select **Build** to execute the pipeline script for this job.

> **Note:**
>
> This job will be aborted because it writes the Jenkins file to the Bastion Host /var/occne/cluster/<cluster_name>/artifacts directory which is not there initially.

k. Execute following command on Bastion host (execute this step ONLY if upgrading to a RC build. Example: upgrading from 1.5.0 to 1.6.0-rc.5)

```
$ sed -i 's/1.6.0/1.6.0-<rc_version>/g' /var/occne/cluster/
<cluster_name>/artifacts/JenkinsFile
```

Example:

```
$ sed -i 's/1.6.0/1.6.0-rc.5/g' /var/occne/cluster/occne3-user1/
artifacts/JenkinsFile
```

l. Select **Build with Parameters** option to see latest Jenkins file parameters in the Jenkins desktop.

m. Select **Configure** from the same menu and edit the **Pipeline** section again. Remove or comment out the line in the script that does the copy of the JenkinsFile: *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"*
This must be completed for use of certificates or it should look like the following:

```
node ('master') {    load '/var/occne/cluster/<cluster-name>/
artifacts/JenkinsFile'}
```

Example:

```
node ('master') {    load '/var/occne/cluster/occne3-user1/
artifacts/JenkinsFile'
```

n. Click the **Apply** button and then **Save** button.

Upgrade Procedure

This section describes how to upgrade the 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.
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.

Notes on adding parameters:

a. If default values are not displayed they should be manually entered.

b. The below are the input parameter values required for upgrading both bare metal cluster and vCNE cluster:

- CENTRAL_REPO: The name of central repository. For example: 'winterfell'
- USER: The value should be set to admusr for upgrading a bare-metal cluster and cloud-user for upgrading a vCNE cluster.
- HOSTIP: External IP address of Bastion Host
- OCCNE_REPO_HOST: Name of the Bastion Host
- OCCNE_REPO_HOST_ADDRESS: Internal IP address of Bastion Host
- DBTIER_NDB_CLUSTER_ID: is the previous cluster_id which was used during the previous dbtier installation.
- DBTIER_REPLICATION_SVC_IP: must be set to the EXTERNAL IP of the dbtier replication service of site 1 if replication is enabled between site 1 and site 2 with the assumption that site 1 was installed first. If you are upgrading site 1 then DBTIER_REPLICATION_SVC_IP should be left blank. If you are upgrading site 2 then replication service ip address of site 1 can be obtained by executing the following command from the site 1 bastion host.

```
$ kubectl gete svc -n occne-infra
NAME                          TYPE           CLUSTER-IP
EXTERNAL-IP      PORT(S)          AGE
.
.
.
occne-db-replication-svc   LoadBalancer   10.233.43.244
10.75.235.61     80:32416/TCP     23h
.
.
```

c. Additional input parameter values that are required for upgrading vCNE cluster alone. The openstack specific values can be obtained by using the openstack configuration show command on a shell that supports the openstack client (cli).

- OPENSTACK_AUTH_URL: Openstack authentication URL. For example: 'http://thundercloud.us.oracle.com:5000/v3'
- OPENSTACK_USERNAME: Openstack user name
- OPENSTACK_PROJECT_ID: Openstack project ID
- OPENSTACK_USER_DOMAIN_NAME: Openstack user domain name
- OPENSTACK_PASSWORD: Openstack password
- OPENSTACK_REGION_NAME: Openstack region name

3. After entering correct values for parameters, select **Build** to start the 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 as shown in the image below (Blue Ocean is recommended as it will show each stage of upgrade).

5. Re-trigger the Jenkins build by repeating procedure from step 2, when build gets aborted with the below message (applicable only for bare metal upgrade)

```
Reboot is required to ensure that your system benefits from these
updates.
+ echo 'Node being restarted due to updates, returns 255'
Node being restarted due to updates, returns 255
+ nohup sudo -b bash -c 'sleep 2; reboot'
+ echo 'restart queued'
restart queued
+ exit 255
```

6. 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** .

Post-Upgrade Procedures

This section describes the post-upgrade procedure.

1. The below procedure needs to be executed to revert back the changes so that SNMP runs smoothly:
   a. Edit SNMP notifier to add (snmp.destination) IP back:

```
$ kubectl edit deployment occne-snmp-notifier -n occne-infra
 Move cursor to the line:
        - --snmp.destination=127.0.0.1:162
Modify to the first trap receiver ip:
        - --snmp.destination=<trap receiver ip address>:162
  The editor is vi, use the vi command :x or :wq to same the
change and exit.
```

   b. If multiple SNMP notifiers were created before upgrade then alert manager configmap needs to be reloaded with previous configuration:

```
$ kubectl edit configmap occne-prometheus-alertmanager -n occne-
infra

apiVersion: v1
data:
  alertmanager.yml: |
    global: {}
    receivers:
    - name: default-receiver
      webhook_configs:
      - url: http://occne-snmp-notifier:9464/alerts
    - name: test-receiver-1
      webhook_configs:
      - url: http://occne-snmp-notifier-1:9465/alerts
    route:
      group_interval: 5m
      group_wait: 10s
```

```
    receiver: default-receiver
    repeat_interval: 3h
    routes:
    - receiver: default-receiver
      group_interval: 1m
      group_wait: 10s
      repeat_interval: 9y
      group_by: [instance, alertname, severity]
      continue: true
    - receiver: test-receiver-1
      group_interval: 1m
      group_wait: 10s
      repeat_interval: 9y
      group_by: [instance, alertname, severity]
      continue: true
```

c. Restart Alert Manager pods for the configmap changes to take effect:

   i. Execute the below command to make sure both the Alert Manager pods are running:

   ```
   $kubectl get pods -n occne-infra | grep alert
   occne-prometheus-alertmanager-0    2/2     Running
   0          5h
   occne-prometheus-alertmanager-1    2/2     Running
   0          5h
   ```

   ii. Execute the below command to delete the first Alert manager Pod. The pod will be recreated automatically after delete:

   ```
   $kubectl delete pod occne-prometheus-alertmanager-0 -n occne-
   infra
   pod "occne-prometheus-alertmanager-0" deleted
   ```

   iii. Execute the below command to make sure the new Alert manager pod is up and running:

   ```
   $ kubectl get pods -n occne-infra | grep alert
   occne-prometheus-alertmanager-0  2/2     Running
   0          5h
   occne-prometheus-alertmanager-1  2/2     Running
   0          50s
   ```

   iv. Execute the below command to delete the second Alert manager Pod. The pod will be recreated automatically after delete.

   ```
   $kubectl delete pod occne-prometheus-alertmanager-1 -n occne-
   infra
   pod "occne-prometheus-alertmanager-1" deleted
   ```

   v. Execute the below command to make sure the new Alert manager pod is up and running:

   ```
   $ kubectl get pods -n occne-infra | grep alert
   occne-prometheus-alertmanager-0  2/2     Running
   ```

```
0          5h
occne-prometheus-alertmanager-1  2/2     Running
0          50s
```

2. Execute the Configuring ZIPKIN Support in Jaeger Collector to restore Zipkin compatibility in Jaeger.
3. Encrypt the Kubernetes secrets using the following command. Since secrets are encrypted on write, performing an update on a secret will encrypt that content:

   ```
   $ kubectl get secrets --all-namespaces -o json | kubectl replace -f
   -
   ```

Post Upgrade Health Checks

Encrypt the Kubernetes secrets using the following command. Since secrets are encrypted on write, performing an update on a secret will encrypt that content.

```
kubectl get secrets --all-namespaces -o json | kubectl replace -f -
```

The health check procedure is as follows:

1. 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
   ```

2. Load previously configured Grafana dashboard:
   a. Click **+** icon on the left panel, click **Import**.
   b. Once in ***Import*** panel, click **Upload .json file**. Choose the dashboard file which is saved locally.
   c. Repeat the above steps for all previously configured dashboards.

A
Reference Procedures

Preserving MetalLB Configuration During Upgrade

This purpose of this procedure is to verify the MetalLB configuration changes that were made in a non-standard manner, such as by editing the MetalLB ConfigMap using *kubectl edit*. This procedure should be performed prior to upgrade.

> **Note:**
>
> All commands in this procedure are executed from the Bastion Host.

Verifying current MetalLB configuration

This section describes the procedure to determine whether or not the MetalLB config file needs to be updated.

Following the procedure to verify the existing MetalLB configuration:

1. Ensure that the MetalLB ConfigMap exists in the occne-infra namespace:

   ```
   $ kubectl get configmap -n occne-infra | grep occne-metallb
   ```

   Expected output:

   ```
   NAME            DATA
   occne-metallb    1
   ```

2. Retrieve the current configuration stored in the MetalLB ConfigMap using the following command:

   ```
   $ kubectl get configmap occne-metallb -n occne-infra -o yaml
   ```

   Sample configuration:

   ```
   apiVersion: v1
   data:
     config: |
       address-pools:
       - addresses:
         - 10.75.182.88/29
         auto-assign: false
         name: oam
   ```

```
      protocol: bgp
    peers:
    - my-asn: 64512
      peer-address: 172.16.7.3
      peer-asn: 64501
    - my-asn: 64512
      peer-address: 172.16.7.2
      peer-asn: 64501
kind: ConfigMap
metadata:
  creationTimestamp: "2020-08-14T22:32:27Z"
  labels:
    app: metallb
    chart: metallb-0.12.0
    heritage: Helm
    release: occne-metallb
  managedFields:
  - apiVersion: v1
    fieldsType: FieldsV1
    fieldsV1:
      f:data:
        .: {}
        f:config: {}
      f:metadata:
        f:labels:
          .: {}
          f:app: {}
          f:chart: {}
          f:heritage: {}
          f:release: {}
    manager: Go-http-client
    operation: Update
    time: "2020-08-14T22:32:27Z"
  name: occne-metallb
  namespace: occne-infra
  resourceVersion: "2442"
  selfLink: /api/v1/namespaces/occne-infra/configmaps/occne-metallb
  uid: 130e0464-856d-48d5-9a6a-626adc0234ee
```

> **Note:**
>
> The output is similar as in sample, but as per your configuration the IP addresses and ASN numbers might be different.

3. Display the MetalLB config file using the below command:

```
$ cat /var/occne/cluster/<cluster_name>/mb_configmap.yaml
```

Sample file:

```
configInline:
    peers:
```

```
- peer-address: 172.16.7.3
  peer-asn: 64501
  my-asn: 64512
- peer-address: 172.16.7.2
  peer-asn: 64501
  my-asn: 64512
address-pools:
- name: oam
  protocol: bgp
  auto-assign: false
  addresses:
  - '10.75.182.88/29'
```

4. Compare the current configuration against the configuration stored in the MetalLB config file. Compare the "peers" and "address-pools" sections from the ConfigMap output and the config file. Check the differences in these sections.

> **Note:**
>
> The ConfigMap output may show the contents of the "peers" and "address-pools" sections in a different order than the config file. This might be as per your configuration, and does not constitute a "difference".

5. If there no differences in these sections, then this procedure is complete. If the command output shows differences, such as in the sample output shown above, update the file as mentioned in Updating MetalLB configuration file.

Updating MetalLB configuration file

This section describes the procedure to update the current MetalLB configuration to the MetalLB config file, so that this configuration is preserved during upgrade.

Following the procedure to update the MetalLB configuration file:

1. Take backup of the current configuration file:

```
$ cp /var/occne/cluster/<cluster_name>/mb_configmap.yaml /var/occne/
cluster/<cluster_name>/mb_configmap_backup.yaml
```

2. Copy the contents of the ConfigMap to the new config file.
Using a text editor such as *vi*, overwrite the "peers" and "address-pools" sections in */var/occne/cluster/<cluster_name>/mb_configmap.yaml* with the same sections from the ConfigMap output derived from Verifying current MetalLB configuration.
Save your changes and exit when the file is updated completely.

Installing Network Functions

This section describes the procedure to install Network Functions (NF) on the OCCNE.

> **Note:**
>
> It is assumed that the NF has docker images located on a docker registry that is reachable by the cluster's bastion, and associated helm charts located at a URL also accessible by the bastion.
> Run the following commands from the cluster bastion.

Follow the procedure to install the NF:

1. Place docker images into the bastion-host docker registry:
   a. Create a file **docker_images.txt** listing the required docker images and tags:

      ```
      dockerRepo:5000/serviceNameImage1:1.0.1
      dockerRepo:5000/serviceNameImage2:1.2.2
      ```

   b. Load these images into the cluster-local docker registry by running (the parameters are the central-docker-registry-repo:port, and the local registry:port):

      ```
      $ /var/occne/cluster/<cluster>/artifacts/retrieve_docker.sh
      <central-repo>:<central-repo-docker-port> ${HOSTNAME}:5000 <
      docker_images.txt
      ```

2. Place the Helm chart into the bastion-host helm chart repository:
   a. Create a file **helm_charts.txt** listing the Helm chart and version:

      ```
      helmRepoName/chart_name 2.7.6
      ```

   b. Add the source helm repository to the cluster-local helm configuration (this need only be done once for each repo):

      ```
      $ helm repo add helmRepoName https://someUrl.oracle.com
      ```

   c. Load the chart into the cluster-local helm chart repository by running (the parameters are the source repo URL, the target directory, and the cluster directory (where helm binary exists)):

      ```
      $ /var/occne/cluster/<cluster>/artifacts/retrieve_helm.sh
      http://<central-repo>/occne/charts /var/www/html/occne /var/
      occne/cluster/${OCCNE_CLUSTER}/artifacts/ < helm_charts.txt
      ```

3. Install the NF:
   a. Create a **values.yaml** file on the Bastion Host that contains the values needed by the Helm chart and run the helm install command as follows:

      ```
      $ helm install --name <release-name> --namespace <service-
      namespace> -f values.yaml <chart_name>
      ```

   b. After successful install, follow section Storing NF Alerts File to configure the NF alerts.

4. Updating the previously installed NF:

   ```
   $ helm upgrade -f values.yaml <release-name> <chart_name>
   ```

   After successful upgrade, follow section Storing NF Alerts File to configure the NF alerts.

5. Removing the previously installed NF:

   ```
   $ helm del <release-name> --purge
   ```

Storing NF Alerts File

After the NF install/upgrade, the alert file must be copied to a pre-defined location in bastion host before upgrading from OCCNE 1.5 to OCCNE 1.6.

> **Note:**
>
> A YAML file containing the alerting rules, hereby referenced as *NFAlertRule.yaml* will be added . This must be in a valid YAML format, and must have the extension *.yaml*.

Steps to follow on Bastion host:

1. Verify the environment variable **OCCNE_CLUSTER** is set correctly. Check if below directory exists, if not then create it using below command:

   ```
   echo $OCCNE_CLUSTER
   mkdir /var/occne/cluster/$OCCNE_CLUSTER/artifacts/alerts
   chmod 755 /var/occne/cluster/$OCCNE_CLUSTER/artifacts/alerts
   ```

2. Set the permissions on the *NFAlertRule.yaml* file as below:

   ```
   chmod 644 NFAlertRule.yaml
   ```

3. Copy the *NFAlertRule.yaml* file to the newly created alerts directory as above using below command:

   ```
   cp NFAlertRule.yaml /var/occne/cluster/$OCCNE_CLUSTER/artifacts/
   alerts
   ```

4. Here is an example for occne_alerts.yaml with the expected permissions:

   ```
   Directory:
   $ ls -lrt -d  /var/occne/cluster/$OCCNE_CLUSTER/artifacts/alerts
   drwxr-xr-x. 2 cloud-user cloud-user 4096 Sep  2 14:46 /var/occne/
   cluster/occne-cluster-name/artifacts/alerts

   File:
   $ ls -lrt  /var/occne/cluster/$OCCNE_CLUSTER/artifacts/alerts
   total 16
   ```

```
-rw-r--r--. 1 cloud-user cloud-user 12991 Sep  2 14:46
occne_alerts.yaml
```

Configuring ZIPKIN Support in Jaeger Collector

This section describes how to configure Jaeger to support ZIPKIN after fresh installation of OCCNE or upgrade from older versions of OCCNE.

> **Note:**
>
> This procedure must be applied after upgrading to OCCNE 1.4, 1.5, or 1.6. It must be re-applied after each upgrade, because upgrade overwrites these changes.

Follow the below procedure to configure Jaeger:

1. From bastion host, edit deployment of jaeger-collector as:

   ```
   $ kubectl edit deployment occne-tracer-jaeger-collector --namespace
   occne-infra
   ```

2. Add environment variable **COLLECTOR_ZIPKIN_HTTP_PORT** with value "**9411"** above SPAN_STORAGE_TYPE in jaeger-collector deployment. Sample file:

   ```
   spec:
     containers:
     - env:
       - name: COLLECTOR_ZIPKIN_HTTP_PORT
         value: "9411"
       - name: SPAN_STORAGE_TYPE
         value: elasticsearch
   ```

3. Save the file to enable ZIPKIN support in Jaeger Collector.