Lab 13. Setting up Container Clusters in OC

Objectives:

- Create a Kubernetes Clusters and access the dashboard from local system.

Pre-Requisite

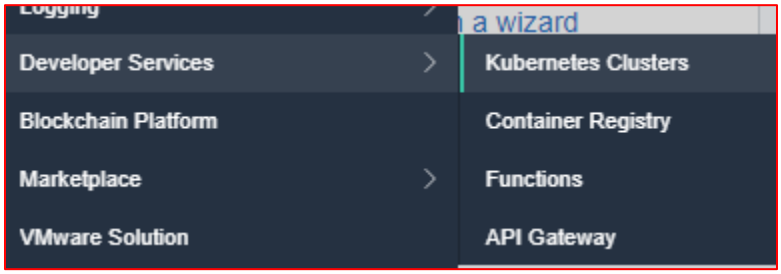
- A Trial Account on OCI

Sequence 1. Create a Kubernetes Clusters and Access the Dashboard

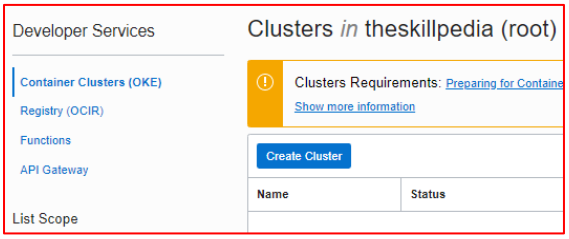
Anyone with a trial account for Oracle Cloud can use Oracle Cloud Infrastructure (OCI) to get herself a three-node Kubernetes Cluster instance, running on Oracle's managed Kubernetes Engine Cloud Service called Kubernetes Clusters (Subject to Service Limits).

Prerequisite:

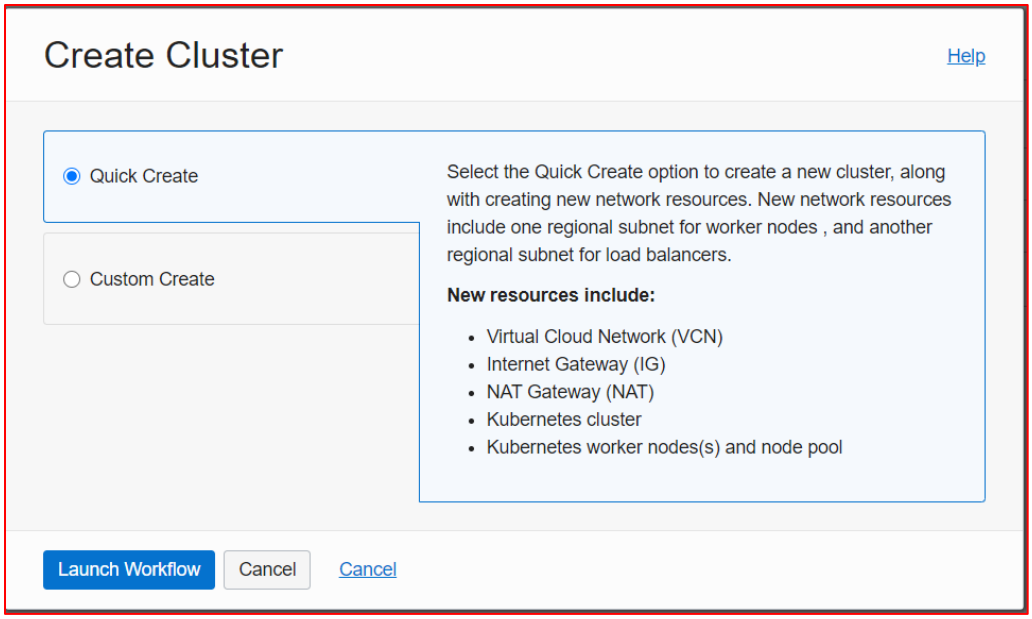
1. From the menu shown on the left side of the screen select Developer Services | Kubernetes Clusters.



2. From the page that open, click on Create Cluster.



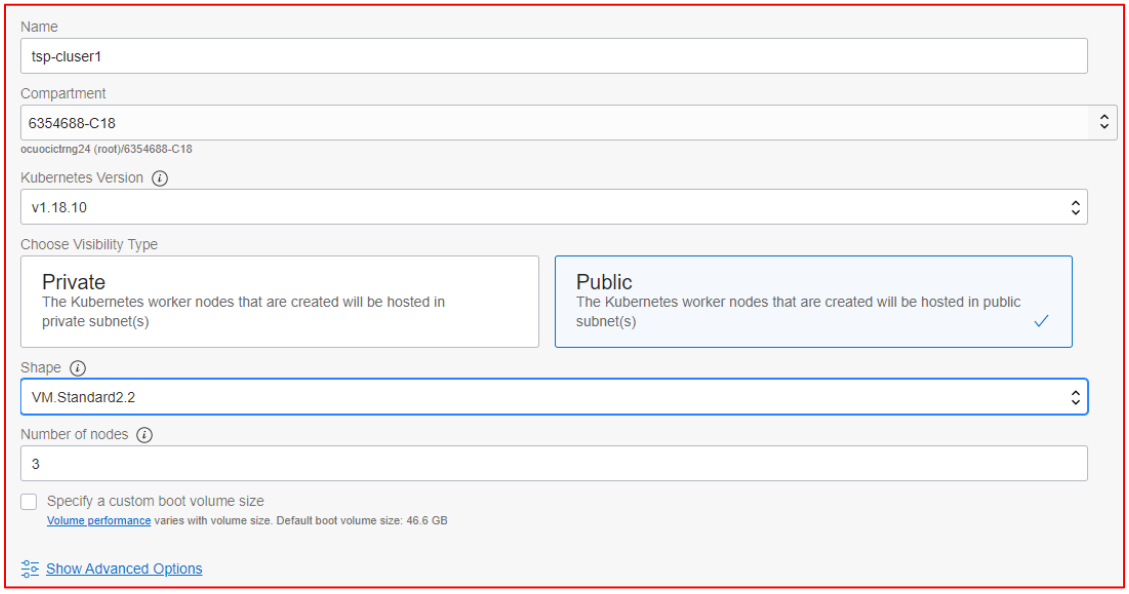
3. You will see a popup with two options. Keep default option "Quick Start" and click on Launch Workflow.



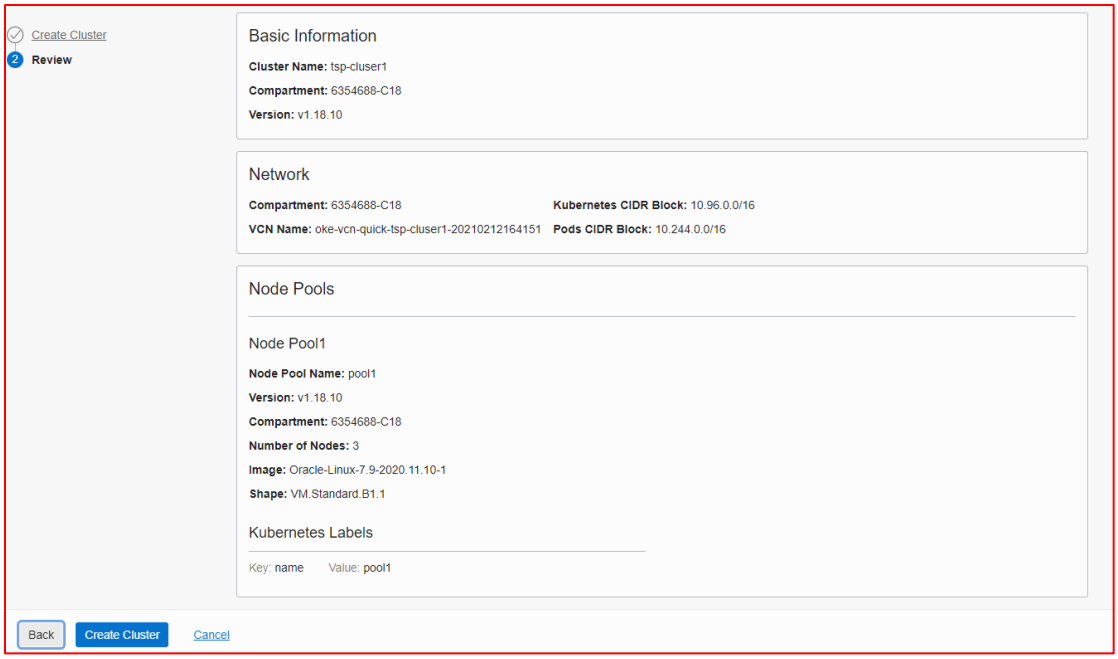
4. Provide the details such as cluster Name and Version and Click on ***Next***.

   Enter Following Details :

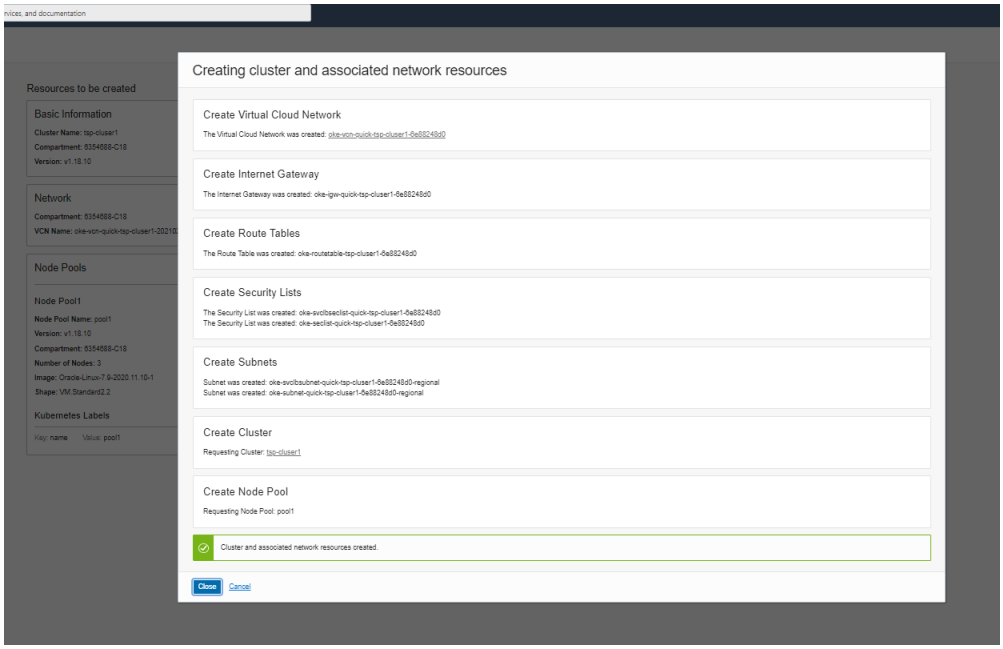
   - Name: tsp-cluser1
   - Compartment: root
   - Kubernetes Version: latest
   - Visibility Type: Public
   - Shape: VM Standard2.2
   - Number of Nodes: 3



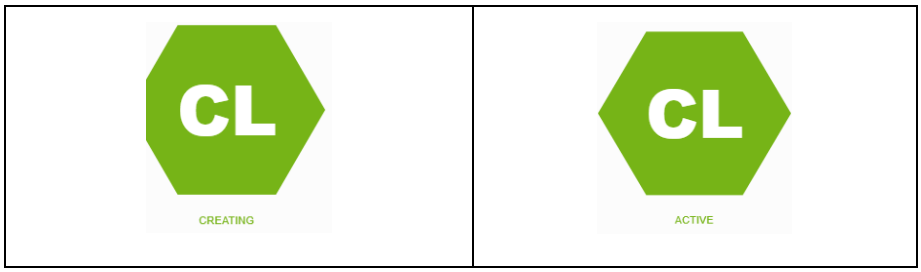
5. Click on Next and Review the Details. Click on ***Create Cluster***



6. You will see a Request processing popup. Click on Close once the request is complete.



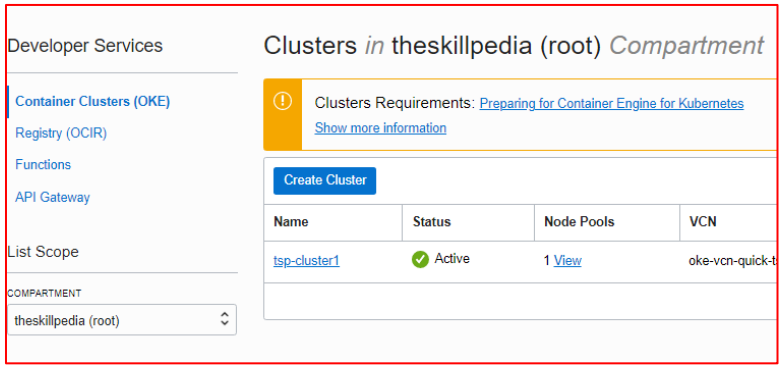
7. You should see the status ***Creating***. After some time the cluster will be ready and status will change to ***Active***.



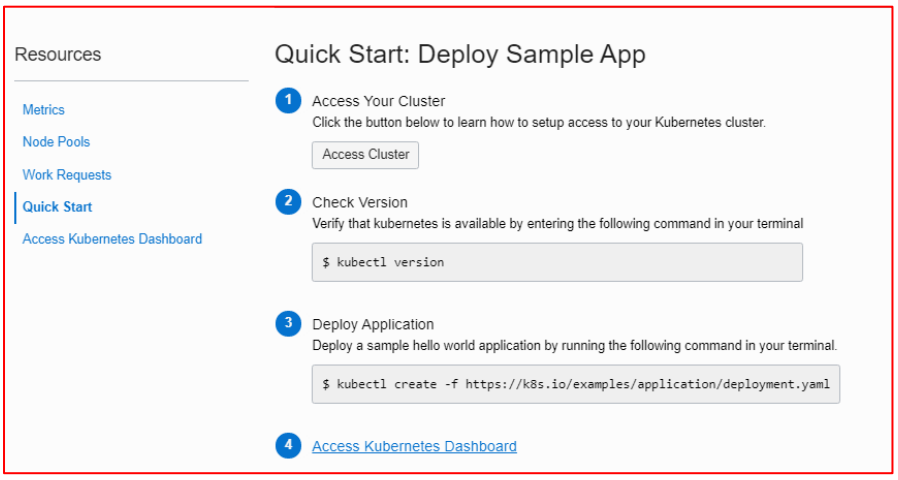
8. To access the Cluster from your Virtual Machine,

   - Create another instance ***master*** in your root compartment, if you are using trial account.
   - Or use existing master if working as a normal user.
   - Once the instance it ready, come back to OKE console.
   - You can also use your VM, ***server***.

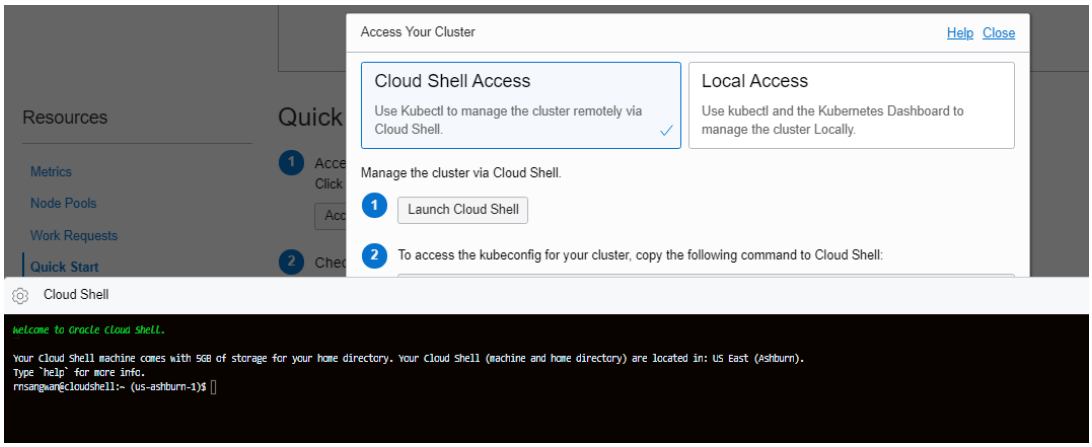
9. Click on Cluster Name “tsp-cluster1” as shown below.



10. Click on Quick Start from sidebar f



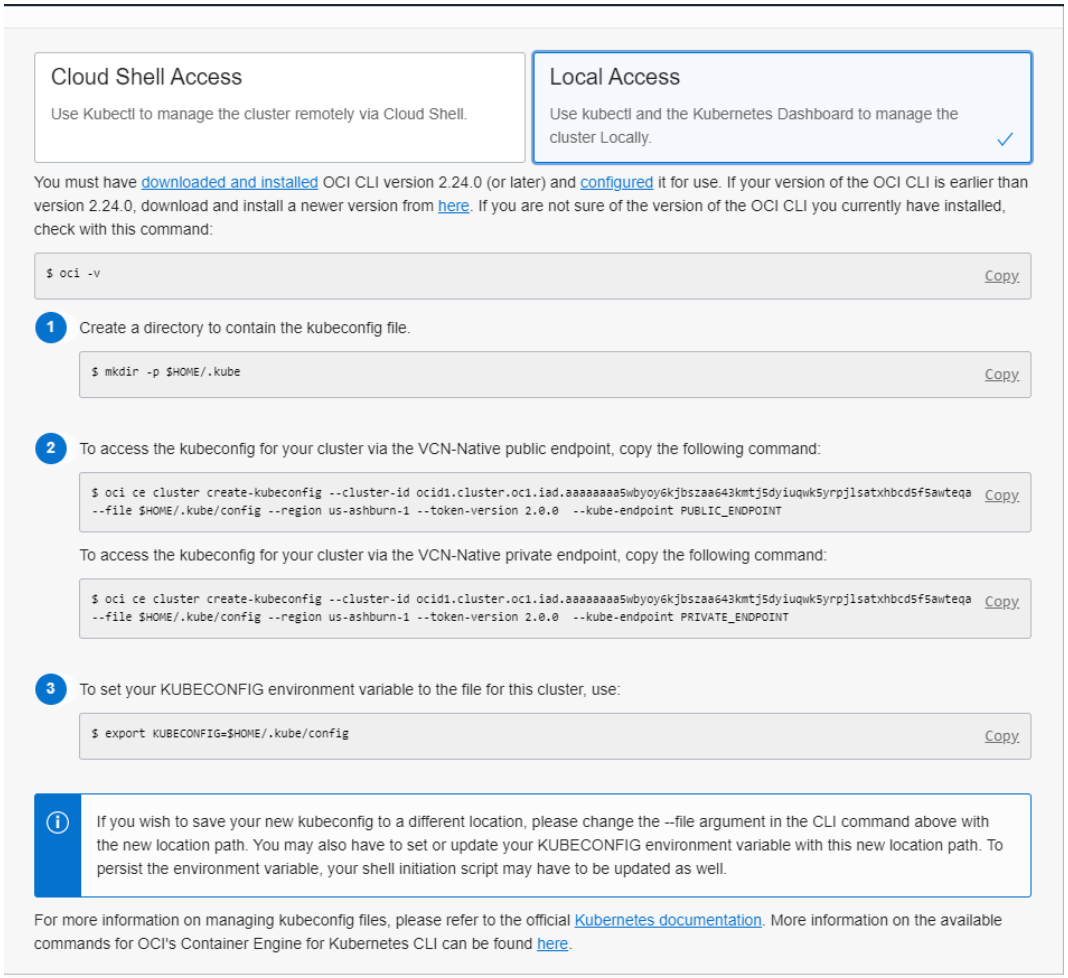
11. Follow the Instruction given to Access Cluster using ***Cloud Shell***.



Copy paste the command as shown in screen shot to access the cluster.

OPTION 2.

12. To get started click on **Access Cluster**. Execute the commands on the VM, ***server*** (Assuming that you have already completed the OCI CLI Lab).



Sequence 1. Install and Configure OCI CLI on Linux.

The Lab outcome will be used in other labs. So please be careful and complete all steps.

1. Login to Linux Machine ***server,*** as root user.
2. To install OCI CLI on the compute instance, Enter Command:

   # bash -c "$(curl -L https://raw.githubusercontent.com/oracle/oci-cli/master/scripts/install/install.sh)"

```
[root@server ~]# bash -c "$(curl –L https://raw.githubusercontent.com/oracle/oci-cli/master/scri
pts/install/install.sh)"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0curl: (6) Could no
t resolve host: xn--l-5gn; Unknown error
100 14738  100 14738    0     0  18237      0 --:--:-- --:--:-- --:--:--  299k

    ********************************************************************************
    You have started the OCI CLI Installer in interactive mode. If you do not wish
    to run this in interactive mode, please include the --accept-all-defaults option.
    If you have the script locally and would like to know more about
    input options for this script, then you can run:
    ./install.sh -h
    If you would like to know more about input options for this script, refer to:
    https://github.com/oracle/oci-cli/blob/master/scripts/install/README.rst
    ********************************************************************************
Downloading Oracle Cloud Infrastructure CLI install script from https://raw.githubusercontent.co
m/oracle/oci-cli/v2.9.3/scripts/install/install.py to /tmp/oci_cli_install_tmp_1oHO.
############################################################################ 100.0%
Python3 not found on system PATH
Running install script.
```

   If you are using Oracle Linux, you can simply use ***yum install python36-oci-cli***

3. When prompted for Install directory, Press Enter (choose default)
4. When prompted for ‘oci’ directory, Press Enter (choose default)
5. When prompted for ‘Y/N’ for $Path, Enter Y, when prompted for path for rc file Press Enter (choose default)

```
===> Modify profile to update your $PATH and enable shell/tab completion now? (Y/n): y

===> Enter a path to an rc file to update (leave blank to use '/root/.bashrc'):
-- Backed up '/root/.bashrc' to '/root/.bashrc.backup'
-- Tab completion set up complete.
-- If tab completion is not activated, verify that '/root/.bashrc' is sourced by your shell.
--
-- ** Run `exec -l $SHELL` to restart your shell. **
--
-- Installation successful.
-- Run the CLI with /root/bin/oci --help
[root@server ~]#
```

6. Check oci CLI installed version, Enter command:

   # oci -v

```
[root@server ~]# oci -v
2.12.0
[root@server ~]#
```

```
[root@server ~]#  oci -v
2.12.0
[root@server ~]#  mkdir -p $HOME/.kube
[root@server ~]#  oci ce cluster create-kubeconfig --cluster-id ocid1.cluste
a3t --file $HOME/.kube/config --region us-ashburn-1 --token-version 2.0.0
New config written to the Kubeconfig file /root/.kube/config
[root@server ~]# export KUBECONFIG=$HOME/.kube/config
```

13. Setup OCI CLI

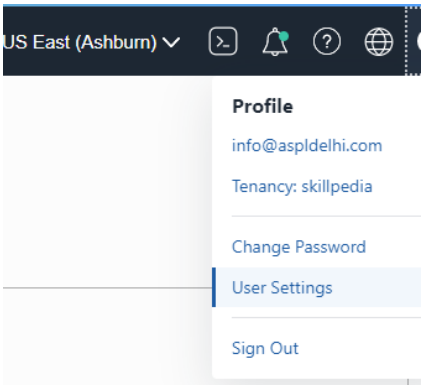
    ```
    # oci setup config
    ```

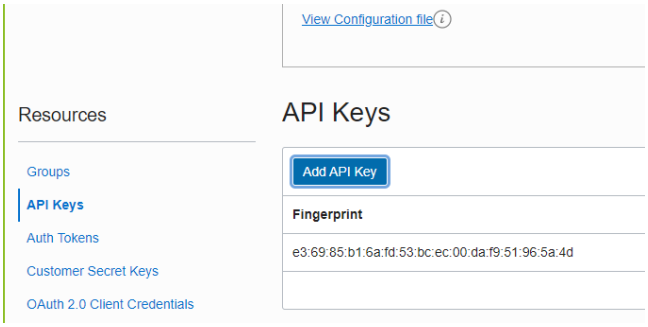
    Accpet all Defaults and Provide

    ```
    User OCID

    Tenancy OCID

    Select the Region name and asked for.
    ```

14. This will also generate a Public Key in PEM format located under .oci folder in your home directory. Add this key under API Key in User Settings on OCI

    ```
    # cat .oci/oci_api_key_public.pem
    ```





15. Verify kubectl

    a. Confirm that you've already installed kubectl.
    b. Verify that you can use kubectl to connect to the new cluster you've created. In a terminal window, enter the following command:

    ```
    # kubectl get nodes
    ```

```
[root@server ~]# kubectl get nodes
NAME         STATUS   ROLES    AGE     VERSION
10.0.10.2    Ready    node     6m44s   v1.16.8
10.0.10.3    Ready    node     6m36s   v1.16.8
10.0.10.4    Ready    node     6m10s   v1.16.8
[root@server ~]#
```

Sequence 2 Kubernetes Dashboard.

 # kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml

```
[root@server ~]# kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.3/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

1. In a text editor, create a file called ***oke-admin-service-account.yaml*** with following content:

   # vi ***oke-admin-service-account.yaml***

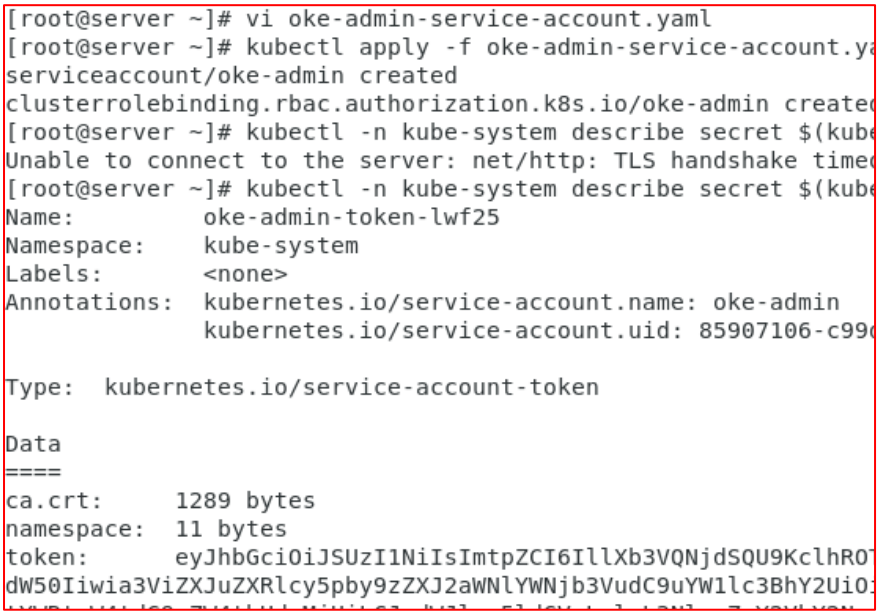
   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: oke-admin
     namespace: kube-system
   ---
   apiVersion: rbac.authorization.k8s.io/v1beta1
   kind: ClusterRoleBinding
   metadata:
     name: oke-admin
   roleRef:
     apiGroup: rbac.authorization.k8s.io
     kind: ClusterRole
     name: cluster-admin
   subjects:
   - kind: ServiceAccount
     name: oke-admin
     namespace: kube-system
   ```

   The file defines an administrator service account and a clusterrolebinding, both called oke-admin.
2. Create the service account and the clusterrolebinding in the cluster by entering:

 # kubectl apply -f oke-admin-service-account.yaml

serviceaccount "oke-admin" created

clusterrolebinding.rbac.authorization.k8s.io "oke-admin" created



You can now use the oke-admin service account to view and control the cluster, and to connect to the Kubernetes dashboard.

3. Obtain an authentication token for the oke-admin service account by entering:

   $ kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep oke-admin | awk '{print $1}')

   The output from the above command includes an authentication token as:

   token: eyJh_____px1Q

4. Copy the value of the token: element from the output. You will use this token to connect to the dashboard.

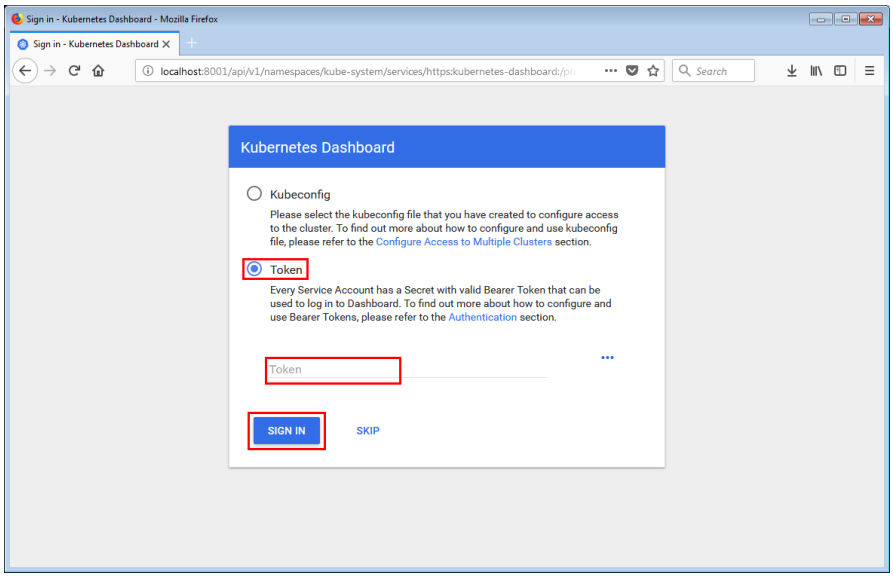
5. In a terminal window, enter following command to start the Kubernetes Dashboard

   $ kubectl proxy .

6. Open a browser and go to

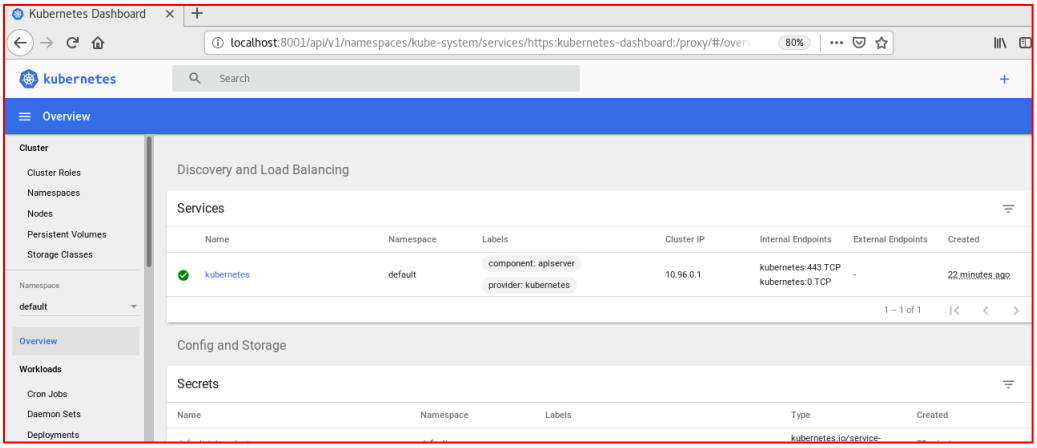
   http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

7. It will display the Kubernetes Login Page like this



8. In the Kubernetes Dashboard as shown above, select **Token** and paste the value of the ***token***: element you copied earlier into the **Token** field.

9. In the Kubernetes Dashboard, click **Sign In**, and then click **Overview** to see the applications deployed on the cluster.



Sequence 3 Deploy an Application

1. Deploy nginx Web Server on your Cluster by executing

   # kubectl create deployment nginx --image=nginx

2. Verify the Deployment

   # kubectl get deployments

```
[root@server ~]# kubectl create deployment nginx --image=nginx
deployment.apps/nginx created
[root@server ~]# kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           72s
[root@server ~]#
```

3. If you'd like to see more detail about your deployment, run the describe command.

   # kubectl describe deployment nginx

```
[root@server ~]# kubectl describe deployment nginx
Name:                   nginx
Namespace:              default
CreationTimestamp:      Sat, 15 Aug 2020 02:09:32 -0400
Labels:                 app=nginx
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=nginx
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx
```

4. Kubernetes offers several options when exposing your service based on a feature called Kubernetes Service-types:

   - **ClusterIP** – To exposes the service on an internal IP, reachable only within the cluster.
   - **NodePort** – To expose your service to be accessible outside of your cluster, on a specific port (called the NodePort) on every node in the cluster.
   - **LoadBalancer** – Leverages on external Load-Balancing services offered by various providers to allow access to your service.

   Create a Service Type NodePort for Nginx

   # kubectl create service nodeport nginx --tcp=80:80

```
[root@server ~]# kubectl create service nodeport nginx --tcp=80:80
service/nginx created
[root@server ~]#
```

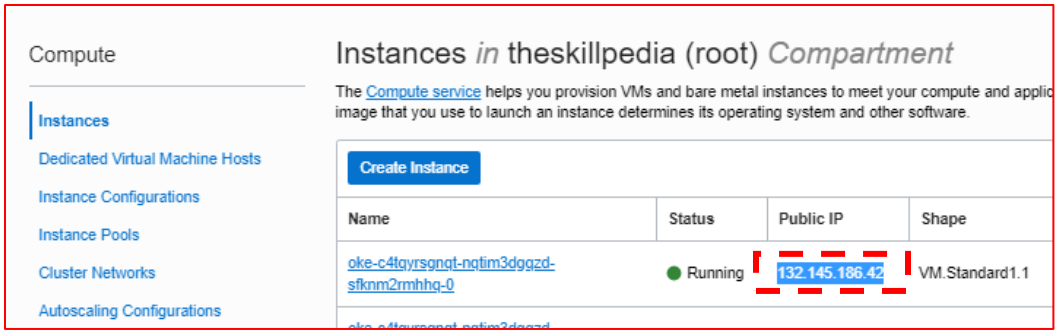
5. Run the ***get svc*** command to see a summary of the service and the ports exposed.

   # kubectl get svc

```
[root@server ~]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP        81m
nginx        NodePort    10.96.198.225   <none>        80:31054/TCP   51s
```

   Take note of the Port corresponding to port 80

6. Visit Compute -> Instances page and take note of public IP of any of your node



7. Now Open the Browser and Access the Website with Public IP and Port Noted above

8. You have successfully deployed an application on your OKE cluster