Name: Hrishikesh Kumbhar

Div: D15A

Roll no: 32

Sub: Advanced DevOps

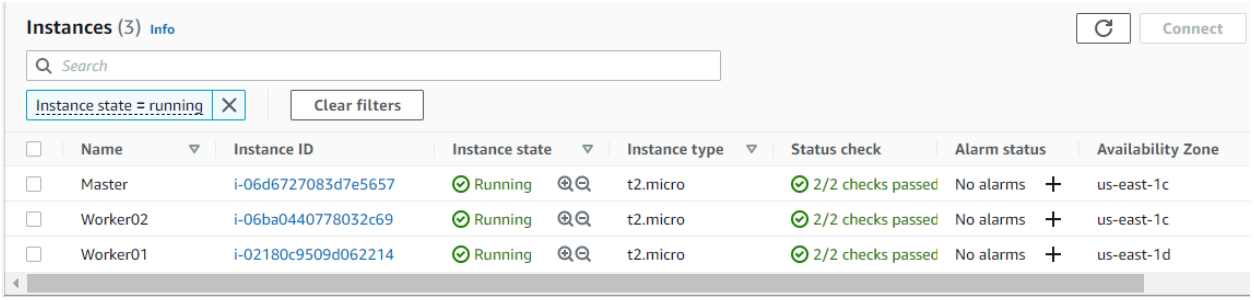
Experiment No: 3

Date: 30/08/2022

Aim: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

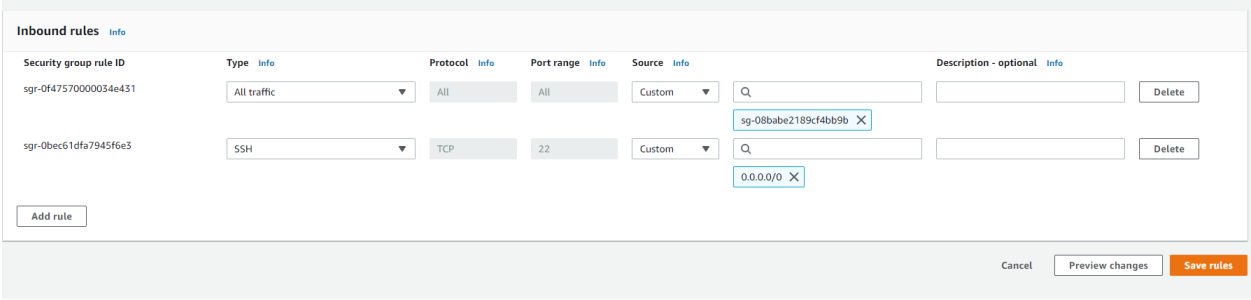
Steps:

Create three EC2 instances of linux 18.04 named Master, Worker1, Worker2



Follow all the commands until instructed on all 3 instances i.e. Master, Worker1, Worker2.

Edit Inbound rules of security group of each node to allow SSH and all traffic:



Set up Docker

Step 1: Install Docker

Kubernetes requires an existing Docker installation. If you already have Docker installed, skip ahead to Step 2.
If you do not have Kubernetes, install it by following these steps:

1. Update the package list with the command:

```
$sudo apt-get update
```

2. Next, install Docker with the command:

```
$sudo apt-get install docker.io
```

3. Repeat the process on each server that will act as a node.

4. Check the installation (and version) by entering the following:

```
$docker --version
```

Master:

```
ubuntu@ip-172-31-81-104: ~
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-81-104:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
ubuntu@ip-172-31-81-104:~$
```

Worker01:

```
ubuntu@ip-172-31-29-0: ~
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-29-0:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
ubuntu@ip-172-31-29-0:~$
```

Worker02:

```
Select ubuntu@ip-172-31-92-247: ~
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-92-247:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
ubuntu@ip-172-31-92-247:~$
```

Step 2: Start and Enable Docker

1. Set Docker to launch at boot by entering the following:

```
$sudo systemctl enable docker
```

2. Verify Docker is running:

```
$sudo systemctl status docker
```

To start Docker if it's not running:

```
$sudo systemctl start docker
```

3. Repeat on all the other nodes.

On Master:

```
ubuntu@ip-172-31-81-104: ~
Processing triggers for ureadahead (0.100.0-21) ...
ubuntu@ip-172-31-81-104:~$ docker --version
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
ubuntu@ip-172-31-81-104:~$ sudo systemctl enable docker
ubuntu@ip-172-31-81-104:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Wed 2022-08-31 13:16:46 UTC; 5min ago
     Docs: https://docs.docker.com
 Main PID: 3025 (dockerd)
    Tasks: 8
   CGroup: /system.slice/docker.service
           └─3025 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/conta

Aug 31 13:16:45 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:45.7627
Aug 31 13:16:45 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:45.7628
Aug 31 13:16:45 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:45.7630
Aug 31 13:16:45 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:45.7633
Aug 31 13:16:45 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:45.9351
Aug 31 13:16:46 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:46.0102
Aug 31 13:16:46 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:46.1240
Aug 31 13:16:46 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:46.1245
Aug 31 13:16:46 ip-172-31-81-104 systemd[1]: Started Docker Application Contai
Aug 31 13:16:46 ip-172-31-81-104 dockerd[3025]: time="2022-08-31T13:16:46.1901
lines 1-19/19 (END)
```

On Worker01:

```
ubuntu@ip-172-31-29-0: ~
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-08-31 13:16:45 UTC; 6min ago
     Docs: https://docs.docker.com
 Main PID: 3051 (dockerd)
    Tasks: 8
   CGroup: /system.slice/docker.service
           └─3051 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Aug 31 13:16:44 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:44.720393716Z" level=wa
Aug 31 13:16:44 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:44.720540790Z" level=wa
Aug 31 13:16:44 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:44.720703698Z" level=wa
Aug 31 13:16:44 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:44.721147074Z" level=in
Aug 31 13:16:44 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:44.944393721Z" level=in
Aug 31 13:16:45 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:45.025196286Z" level=in
Aug 31 13:16:45 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:45.142095818Z" level=in
Aug 31 13:16:45 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:45.142724221Z" level=in
Aug 31 13:16:45 ip-172-31-29-0 systemd[1]: Started Docker Application Container Engine.
Aug 31 13:16:45 ip-172-31-29-0 dockerd[3051]: time="2022-08-31T13:16:45.200302502Z" level=in
~
```

On Worker02:

```
ubuntu@ip-172-31-92-247: ~
ubuntu@ip-172-31-92-247:~$ sudo systemctl enable docker
ubuntu@ip-172-31-92-247:~$ sudo systemctl enable docker
ubuntu@ip-172-31-92-247:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset:
   Active: active (running) since Wed 2022-08-31 13:16:49 UTC; 6min ago
     Docs: https://docs.docker.com
 Main PID: 3067 (dockerd)
    Tasks: 8
   CGroup: /system.slice/docker.service
           └─3067 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/conta

Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.4028
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.4030
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.4031
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.4034
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.6909
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.7653
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.9910
Aug 31 13:16:48 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:48.9916
Aug 31 13:16:49 ip-172-31-92-247 systemd[1]: Started Docker Application Contai
Aug 31 13:16:49 ip-172-31-92-247 dockerd[3067]: time="2022-08-31T13:16:49.0525
lines 1-19/19 (END)
```

Install Kubernetes

Step 3: Add Kubernetes Signing Key

Since you are downloading Kubernetes from a non-standard repository, it is essential to ensure that the software is authentic. This is done by adding a signing key.

1. Enter the following to add a signing key:

```
$curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add
```

```
ubuntu@ip-172-31-81-104:~$ curl -s https://packages.cloud.google.com/apt/doc/a
pt-key.gpg | sudo apt-key add
OK
ubuntu@ip-172-31-81-104:~$
```

```
ubuntu@ip-172-31-29-0:~$ curl -s https://packages.cloud.google.com/apt/doc/apt
-key.gpg | sudo apt-key add
OK
ubuntu@ip-172-31-29-0:~$
```

```
ubuntu@ip-172-31-92-247:~$ curl -s https://packages.cloud.google.com/apt/doc/a
pt-key.gpg | sudo apt-key add
OK
ubuntu@ip-172-31-92-247:~$
```

If you get an error that curl is not installed, install it with:

```
$sudo apt-get install curl
```

2. Then repeat the previous command to install the signing keys. Repeat for each server node.

Step 4: Add Software Repositories

Kubernetes is not included in the default repositories. To add them, enter the following:

```
$sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial
main"
```

Repeat on each server node.

```
ubuntu@ip-172-31-81-104:~$ sudo apt-add-repository "deb http://apt.kubernetes.
io/ kubernetes-xenial main"
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelea
e
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383
B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packa
ges [58.4 kB]
Fetched 67.8 kB in 0s (139 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-81-104:~$
```

```
ubuntu@ip-172-31-29-0:~$ sudo apt-add-repository "deb http://apt.kubernetes.io
/ kubernetes-xenial main"
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelea
e
Hit:4 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383
B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packa
ges [58.4 kB]
Fetched 67.8 kB in 0s (138 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-29-0:~$
```

```
ubuntu@ip-172-31-92-247:~$ sudo apt-add-repository "deb http://apt.kubernetes.
io/ kubernetes-xenial main"
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-backports InRelea
e
Hit:5 http://security.ubuntu.com/ubuntu bionic-security InRelease
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [9383
B]
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packa
ges [58.4 kB]
Fetched 67.8 kB in 0s (140 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-92-247:~$
```

Step 5: Kubernetes Installation Tools

Kubeadm (Kubernetes Admin) is a tool that helps initialize a cluster. It fast-tracks setup by using community-sourced best practices. Kubelet is the work package, which runs on every node and start containers. The tool gives you command-line access to clusters.

1. Install Kubernetes tools with the command:

```
$sudo apt-get install kubeadm kubelet kubectl -y
$sudo apt-mark hold kubeadm kubelet kubectl
```

Allow the process to complete.

2. Verify the installation with:

```
$kubeadm version
```

on Master:

```
ubuntu@ip-172-31-81-104:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.0", Gi
tCommit:"a866cbe2e5bbaa01cfd5e969aa3e033f3282a8a2", GitTreeState:"clean", Buil
dDate:"2022-08-23T17:43:25Z", GoVersion:"go1.19", Compiler:"gc", Platform:"lin
ux/amd64"}
```

on Worker1:

```
ubuntu@ip-172-31-29-0:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.0", Gi
tCommit:"a866cbe2e5bbaa01cfd5e969aa3e033f3282a8a2", GitTreeState:"clean", Buil
dDate:"2022-08-23T17:43:25Z", GoVersion:"go1.19", Compiler:"gc", Platform:"lin
ux/amd64"}
ubuntu@ip-172-31-29-0:~$
```

on Worker2:

```
ubuntu@ip-172-31-92-247:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.0", Gi
tCommit:"a866cbe2e5bbaa01cfd5e969aa3e033f3282a8a2", GitTreeState:"clean", Buil
dDate:"2022-08-23T17:43:25Z", GoVersion:"go1.19", Compiler:"gc", Platform:"lin
ux/amd64"}
```

3. Repeat for each server node.

Kubernetes Deployment

Step 6: Begin Kubernetes Deployment

Start by disabling the swap memory on each server:

```
$sudo swapoff --a
```

Step 7: Assign Unique Hostname for Each Server Node

Decide which server to set as the master node. Then enter the command:

```
$sudo hostnamectl set-hostname master-node
```

Next, set a worker node hostname by entering the following on the worker server:

```
$sudo hostnamectl set-hostname worker01
```

```
$sudo hostnamectl set-hostname worker02
```

If you have additional worker nodes, use this process to set a unique hostname on each.

Step 8: Initialize Kubernetes on Master Node

Switch to the master server node, and enter the following:

```
$sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

If you are trying to run this on EC2 you'll get an error message saying less cpu and memory to override the error run the above command with --ignore-preflight-errors=all

```
$sudo kubeadm init --pod-network-cidr=10.244.0.0/16
--ignore-preflight-errors=all
```

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each
as root:

kubeadm join 172.31.81.104:6443 --token kxnbuc.o7366yxom4l7tw49 \
        --discovery-token-ca-cert-hash sha256:44dcec291b8cf26218630f6f0e0a8a25
e28c7b4332ed55d5afcb55c1728de093
```

```
kubeadm join 172.31.81.104:6443 --token kxnbuc.o7366yxom4l7tw49 \
--discovery-token-ca-cert-hash
sha256:44dcec291b8cf26218630f6f0e0a8a25e28c7b4332ed55d5afcb55c1728de093
```

Once this command finishes, it will display a kubeadm join message at the end. Make a note of the whole entry. This will be used to join the worker nodes to the cluster.

Next, enter the following to create a directory for the cluster:

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
ubuntu@ip-172-31-81-104:~$  mkdir -p $HOME/.kube
ubuntu@ip-172-31-81-104:~$  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/
config
ubuntu@ip-172-31-81-104:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Step 9: Deploy Pod Network to Cluster

A Pod Network is a way to allow communication between different nodes in the cluster. This tutorial uses the flannel virtual network.
Enter the following:

```
$ sudo kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml
```

Allow the process to complete.

```
ubuntu@ip-172-31-81-104:~$ sudo kubectl apply -f  https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
namespace/kube-flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
serviceaccount/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
ubuntu@ip-172-31-81-104:~$
```

Verify that everything is running and communicating:

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

```
ubuntu@ip-172-31-81-104:~$ kubectl get pods --all-namespaces
NAMESPACE      NAME                                     READY   STATUS    RESTART
S   AGE
kube-flannel   kube-flannel-ds-jdqws                    1/1     Running   0
    41s
kube-system    coredns-565d847f94-sfndt                 1/1     Running   0
    3m57s
kube-system    coredns-565d847f94-v6wsj                 1/1     Running   0
    3m57s
kube-system    etcd-master-node                         1/1     Running   0
    4m10s
kube-system    kube-apiserver-master-node               1/1     Running   0
    4m10s
kube-system    kube-controller-manager-master-node      1/1     Running   0
    4m10s
kube-system    kube-proxy-qs694                         1/1     Running   0
    3m58s
kube-system    kube-scheduler-master-node               1/1     Running   0
    4m10s
ubuntu@ip-172-31-81-104:~$
```

Step 10: Join Worker Node to Cluster

As indicated in Step 7, you can enter the kubeadm join command on each worker node to connect it to the cluster.

Switch to the worker01 system and enter the command you noted from Step 7:

First do command in other two worker nodes

```
sudo su
```

and then paste:

```
kubeadm join 172.31.81.104:6443 --token kxnbuc.o7366yxom4l7tw49 \
--discovery-token-ca-cert-hash
sha256:44dcec291b8cf26218630f6f0e0a8a25e28c7b4332ed55d5afcb55c1728de093
```

```
root@worker01:/home/ubuntu# kubeadm join 172.31.81.104:6443 --token kxnbuc.o73
66yxom4l7tw49 \
>         --discovery-token-ca-cert-hash sha256:44dcec291b8cf26218630f6f0e0a8a
25e28c7b4332ed55d5afcb55c1728de093
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system
 get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config
.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/
kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
```

```
ubuntu@ip-172-31-92-247:~$ sudo su
root@worker02:/home/ubuntu# kubeadm join 172.31.81.104:6443 --token kxnbuc.o73
66yxom4l7tw49 \
>         --discovery-token-ca-cert-hash sha256:44dcec291b8cf26218630f6f0e0a8a
25e28c7b4332ed55d5afcb55c1728de093
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system
 get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config
.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/
kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
```

ON EC2 make sure you open the port in security group ADVERTISED HERE:

Replace the alphanumeric codes with those from your master server. Repeat for each worker node on the cluster. Wait a few minutes; then you can check the status of the nodes.

Switch to the master server, and enter:

```
$ kubectl get nodes
```

The system should display the worker nodes that you joined to the cluster.

```
ubuntu@ip-172-31-81-104:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
master-node    Ready    control-plane   19m     v1.25.0
worker01       Ready    <none>          3m29s   v1.25.0
worker02       Ready    <none>          2m50s   v1.25.0
ubuntu@ip-172-31-81-104:~$
```

If all of your nodes have the value Ready for STATUS, it means that they're part of the cluster and ready to run workloads.
If, however, a few of the nodes have NotReady as the STATUS, it could mean that the worker nodes haven't finished their setup yet. Wait for around five to ten minutes before re-running kubectl get node and inspecting the new output. If a few nodes still have NotReady as the status, you might have to verify and re-run the commands in the previous steps.
Now that your cluster is verified successfully, let's schedule an example Nginx application on the cluster.