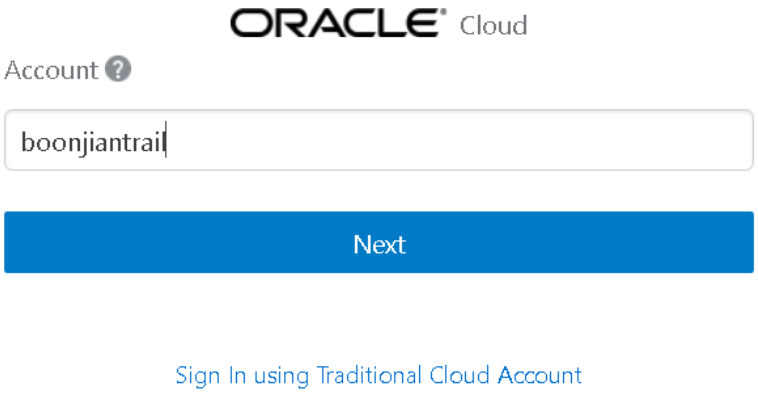
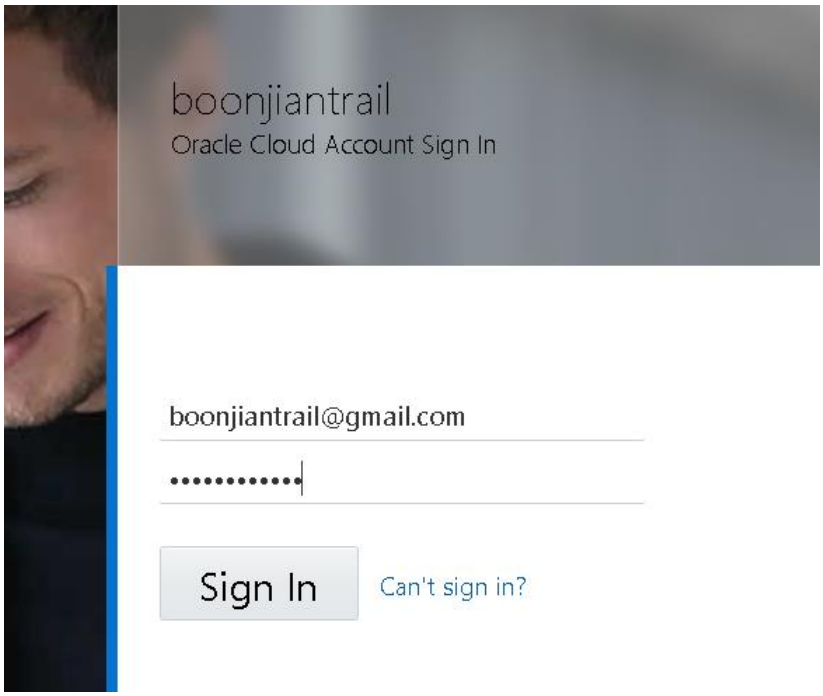
Contents

Login to cloud trail

1. Sign in to https://cloud.oracle.com/en_US/sign-in



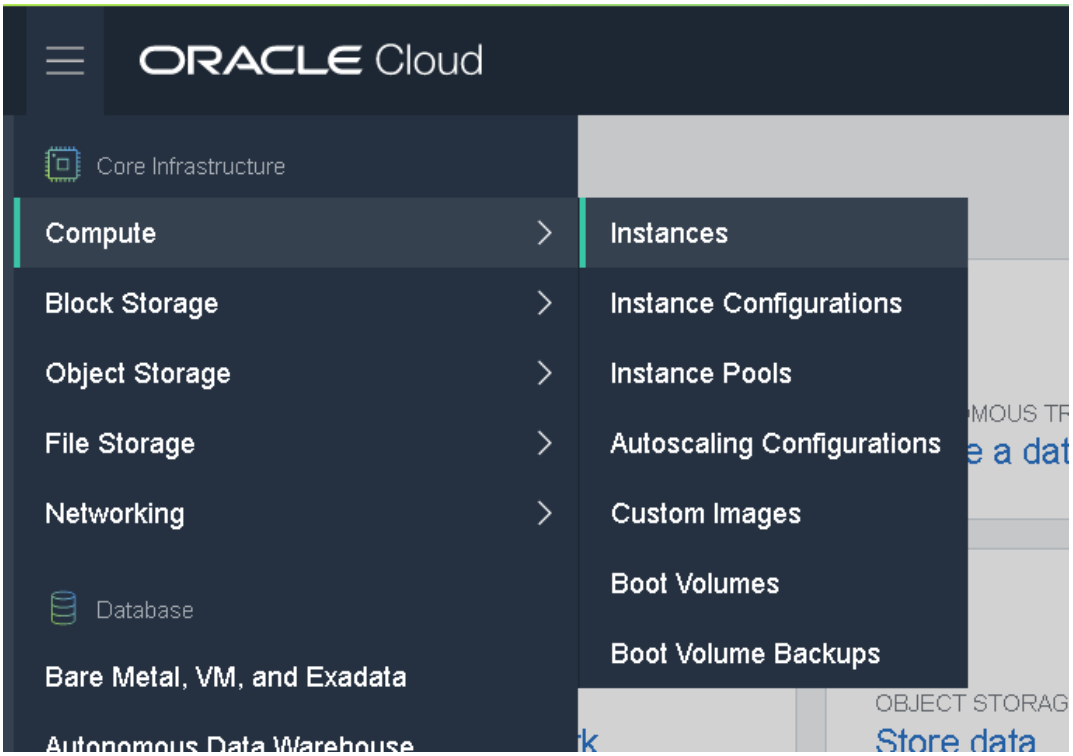
2. Login with user name and password



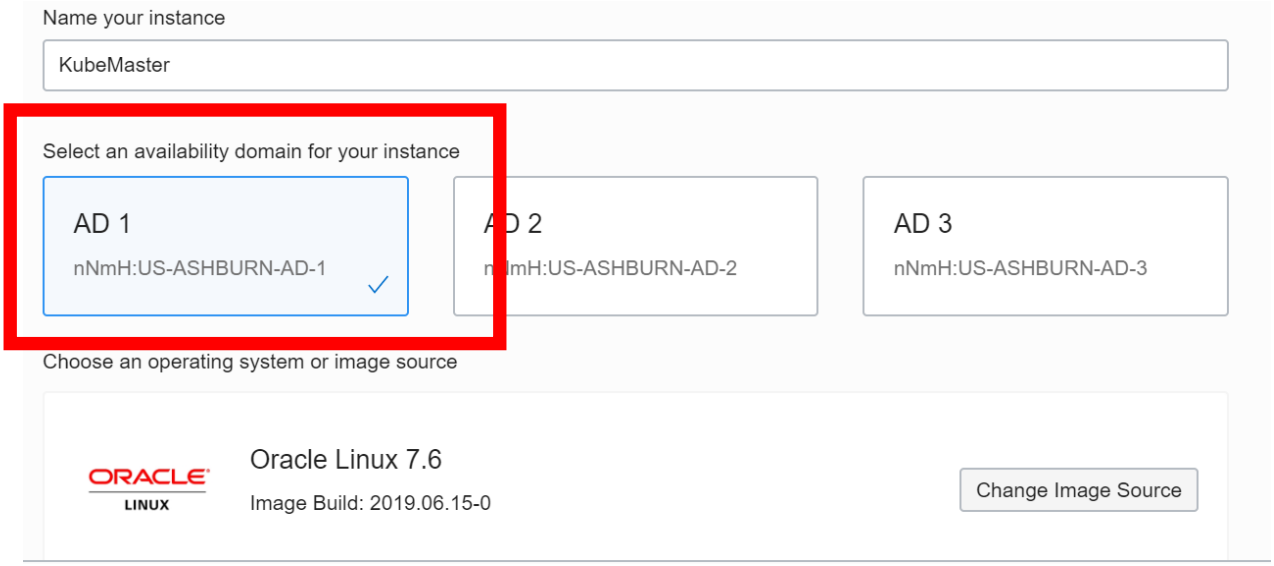
Create kubernetes on compute instance

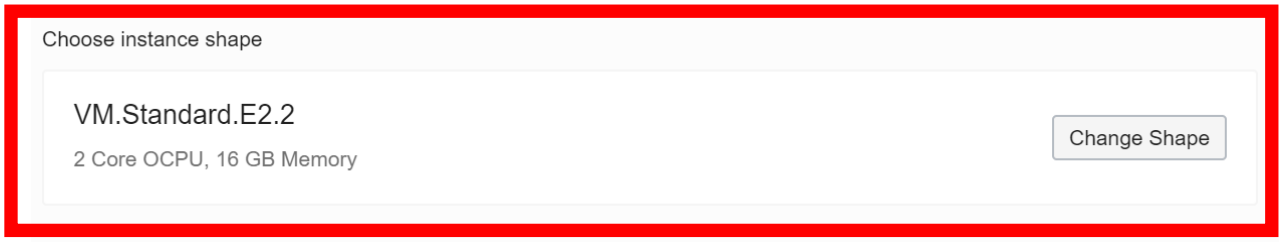
Creating the Master Node

1. Access the compute menu

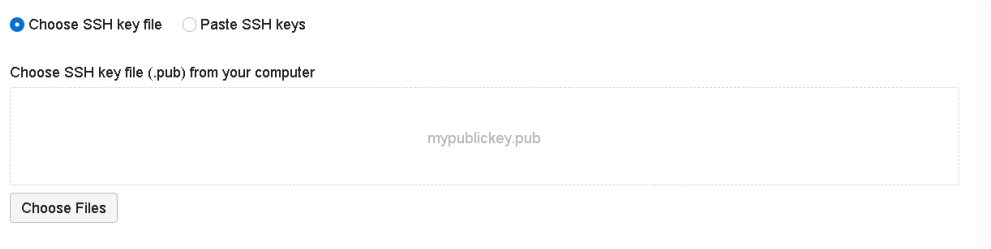


2. Go to Compute instance and create a virtual machine with name “KubeMaster” with the following configuration.
   AD: AD 1
   Instance Shape: VM.Standard.E2.2

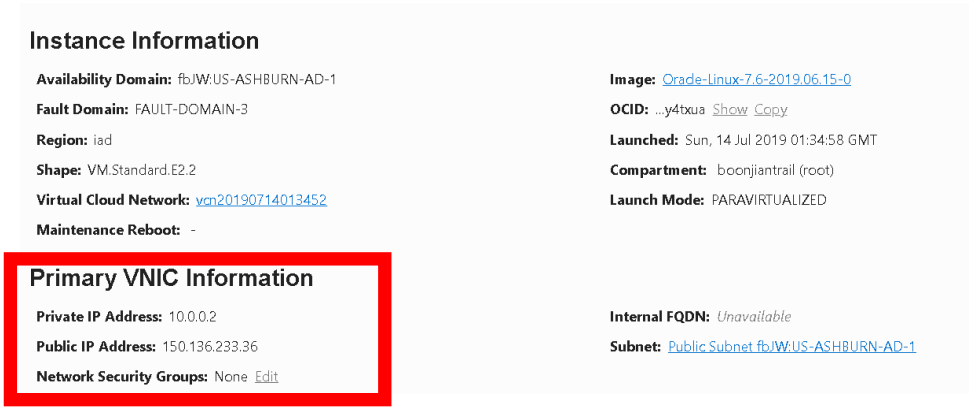




3. Choose the public key which is create for you.

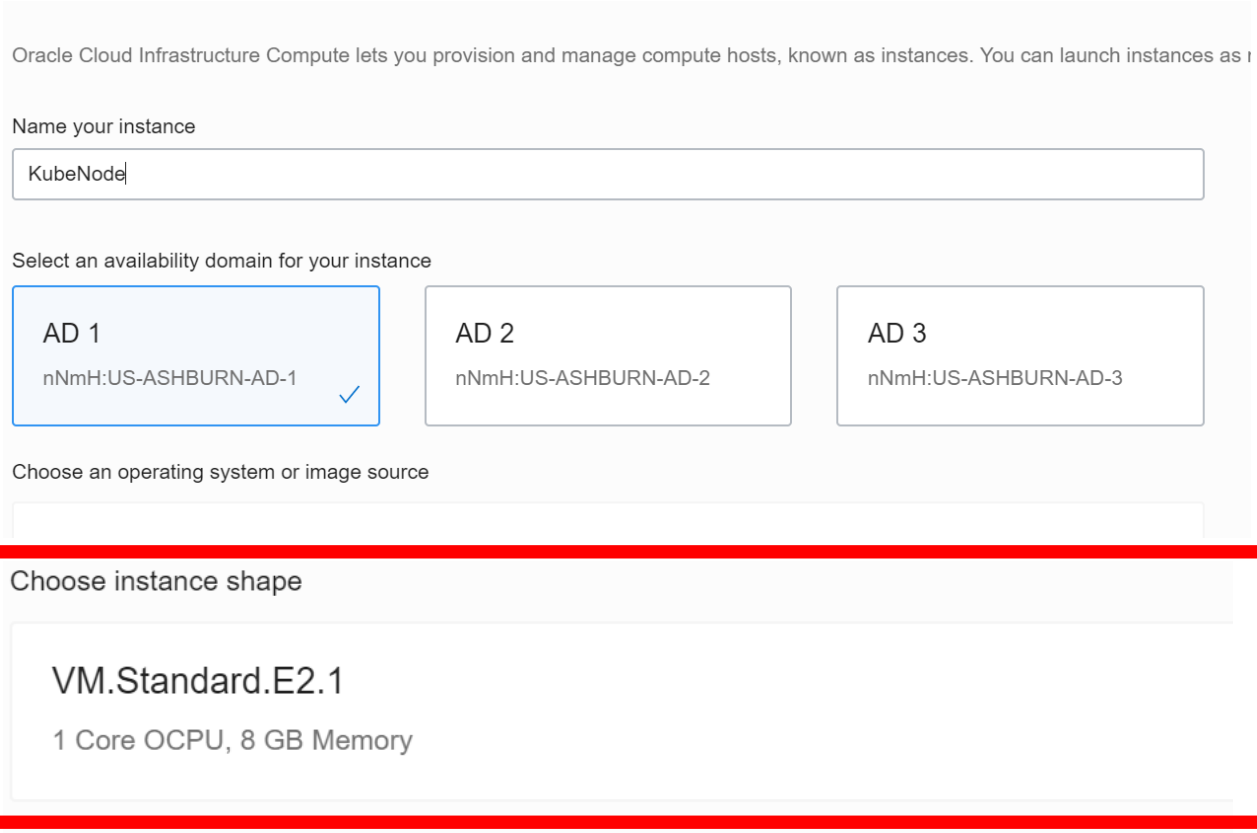


4. After the instance is created, note down the public ip. You will use this ip to ftp the materials over.

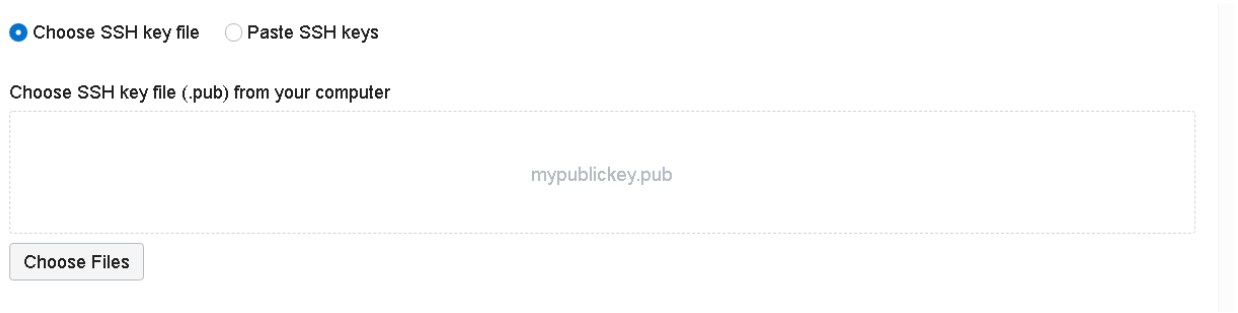


Creating the Worker Node

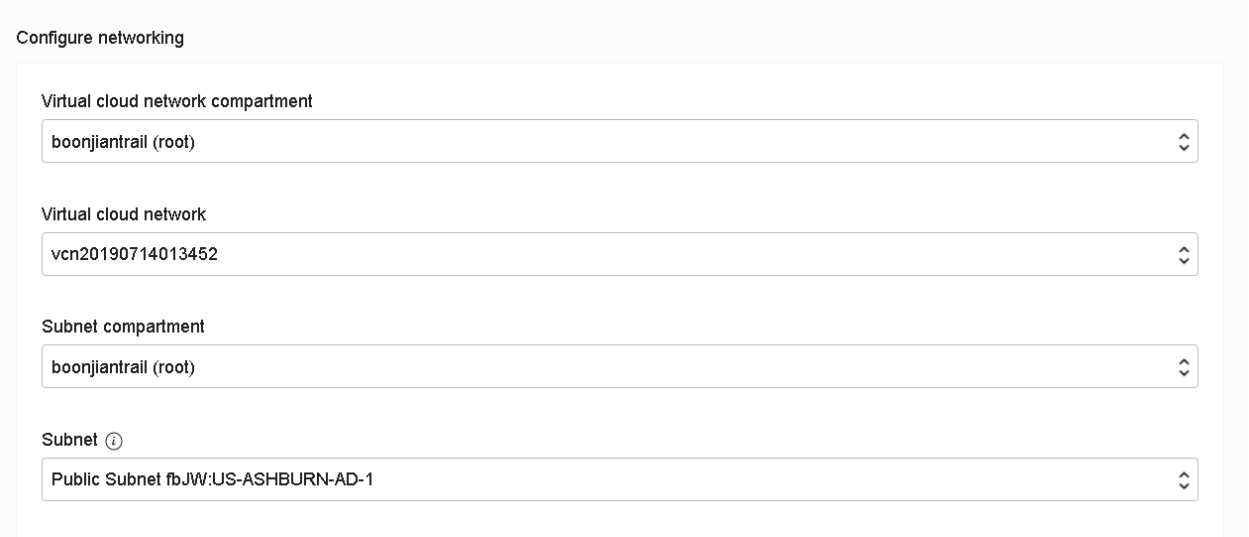
1. Go to Compute instance and create a virtual machine with name “KubeNode” with the following configuration.
   AD: AD 1
   Instance Shape: VM.Standard.E2.1



2. Choose the public key which is create for you.



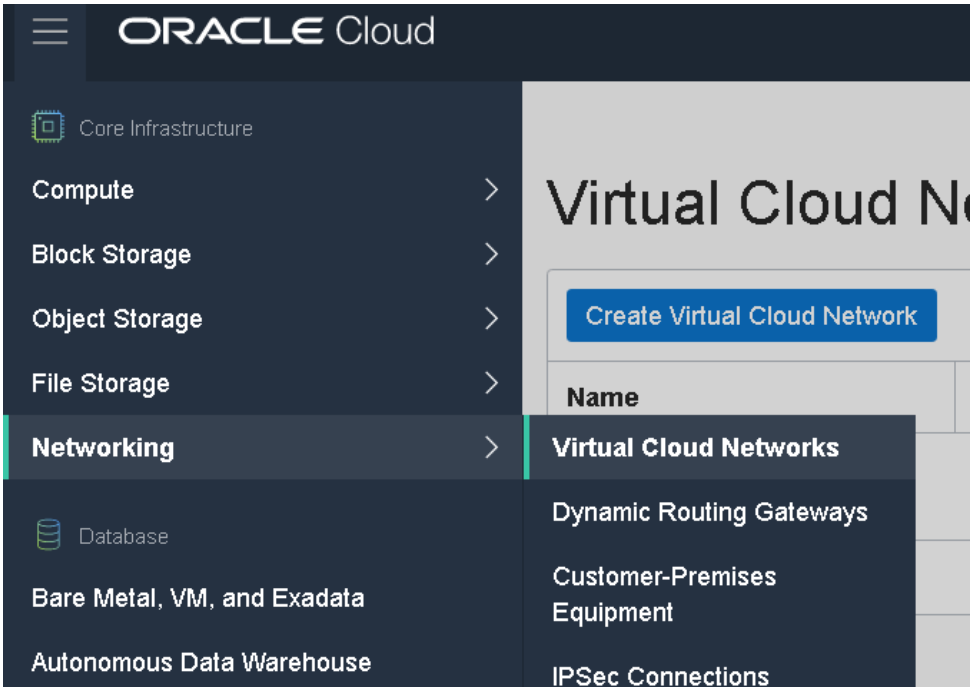
3. The networking will be using the one that is created earlier. Do not change the default values that is already selected for you.



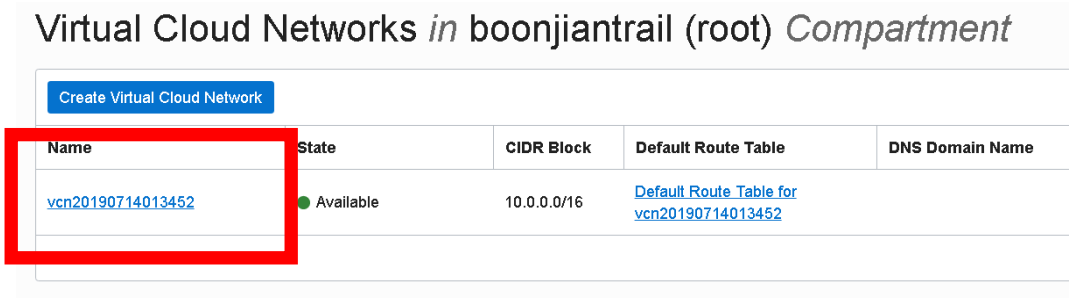
4. Note down the public ip after the instance is created.

VCN firewall rules

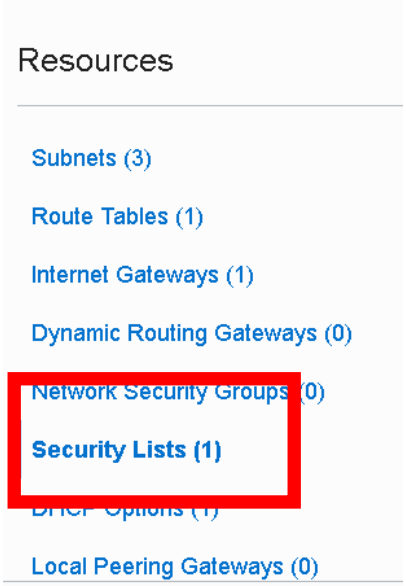
1. Select networking -> Virtual Cloud Networks



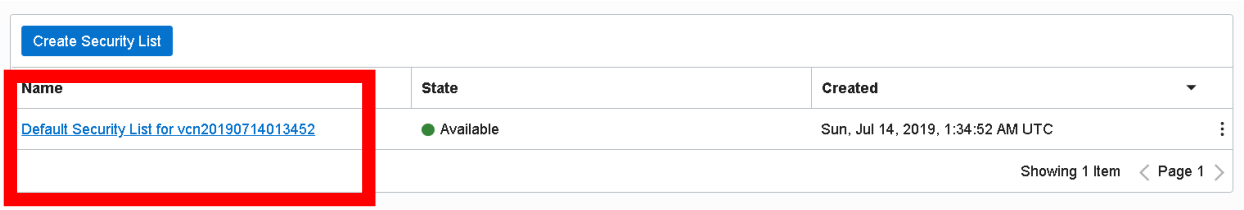
2. Click on the vcn name that is created.



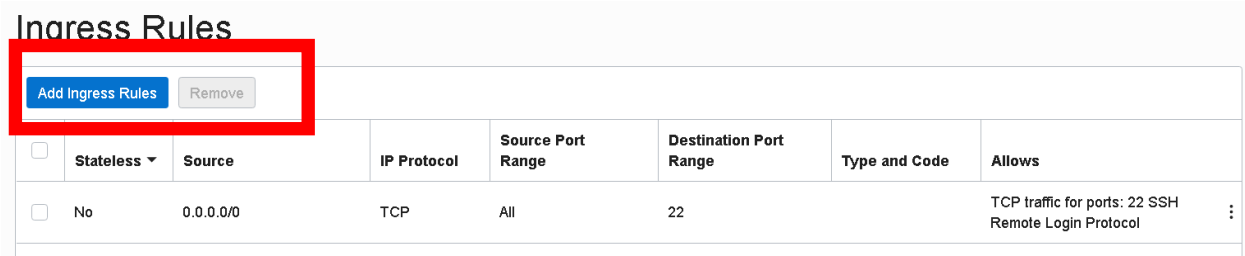
3. Select security list.



4. Click on Default security list

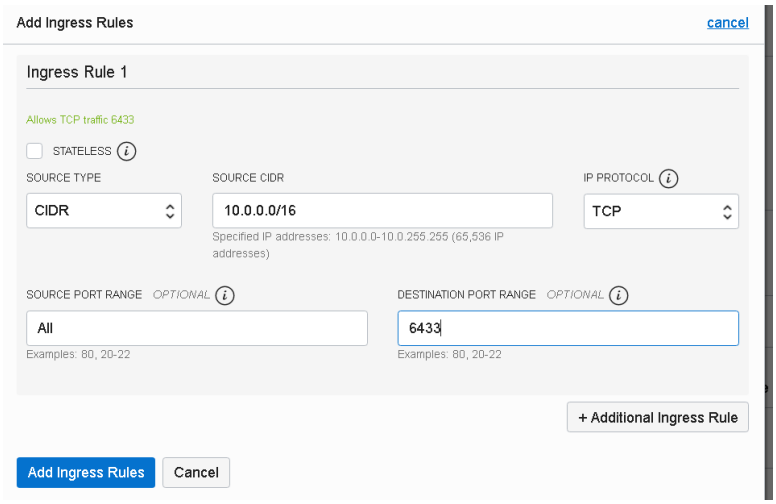


5. Go to the VCN and set the security Ingress rules to the default security list. Click on Add ingress rules. And enter the inputs below as screen shot.

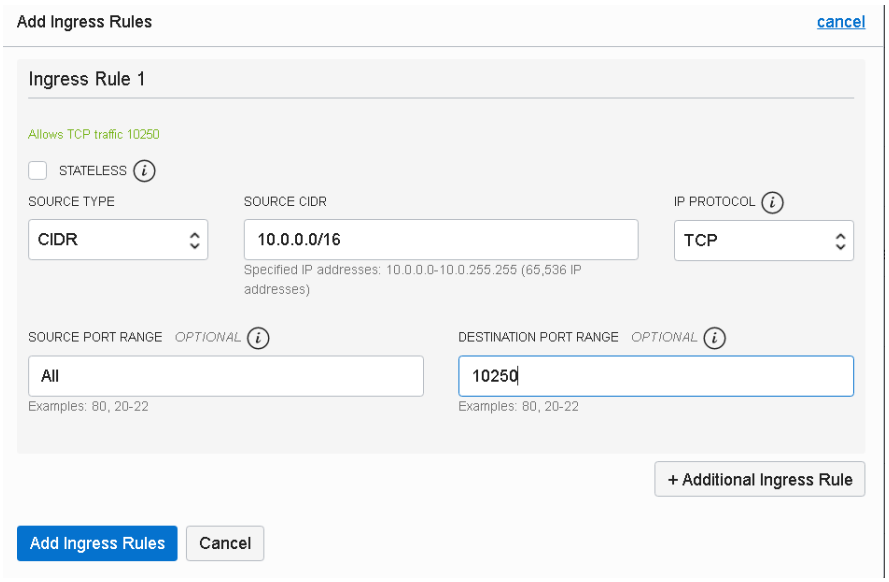


The documentation documented the pre-req for install kubernetes in linux.
https://docs.oracle.com/cd/E52668_01/E88884/html/requirements-bmc.html

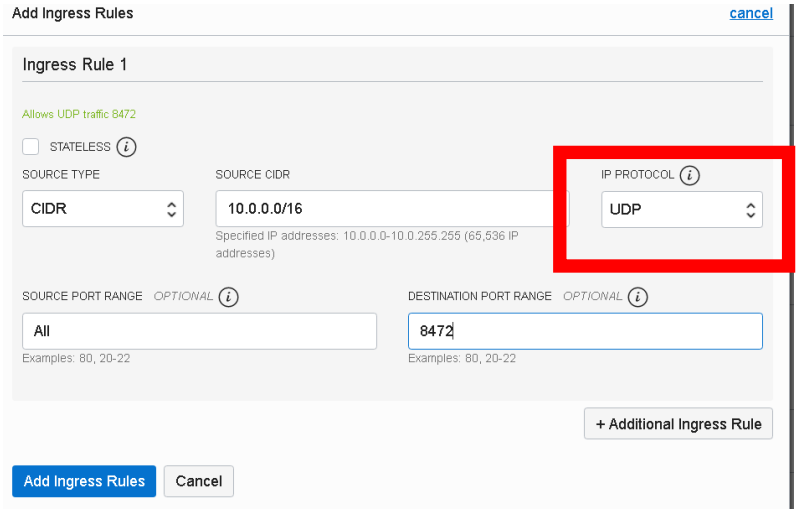
1. Allow 6443/TCP.
   - STATELESS: Unchecked
   - SOURCE CIDR: *10.0.0.0/16*
   - IP PROTOCOL: TCP
   - SOURCE PORT RANGE: All
   - DESTINATION PORT RANGE: 6443



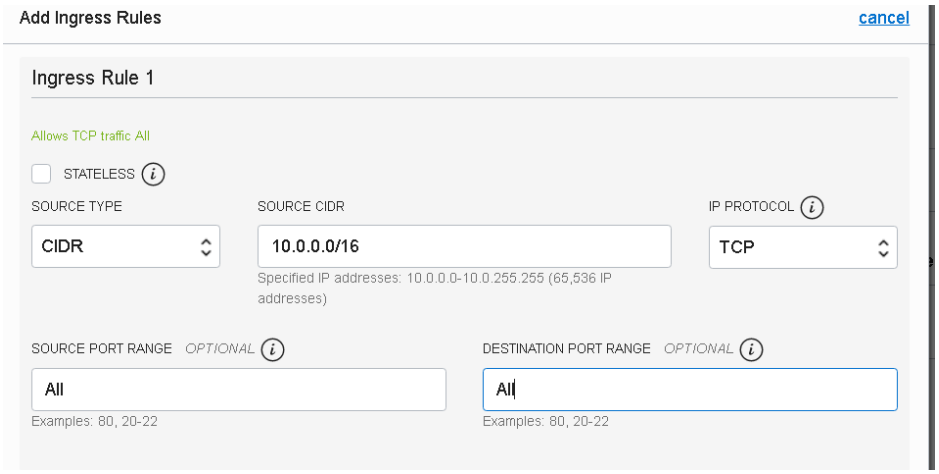
2. Allow 10250/TCP.
   - STATELESS: Unchecked
   - SOURCE CIDR: *10.0.0.0/16*
   - IP PROTOCOL: TCP
   - SOURCE PORT RANGE: All
   - DESTINATION PORT RANGE: 10250

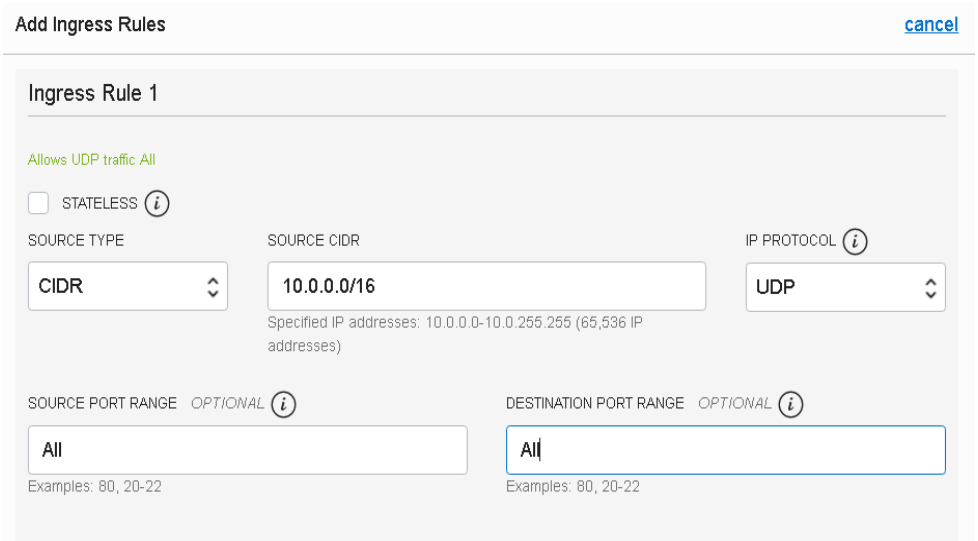


3. Allow 8472/UDP.
   - STATELESS: Unchecked
   - SOURCE CIDR: *10.0.0.0/16*
   - IP PROTOCOL: UDP
   - SOURCE PORT RANGE: All
   - DESTINATION PORT RANGE: 8472



4. Enable Any to Any for tcp and udp.





Add Ingress Rules

cancel

Ingress Rule 1

Allows TCP traffic All

STATELESS

SOURCE TYPE

CIDR

SOURCE CIDR

0.0.0.0/0

Specified IP addresses: 0.0.0.0-255.255.255.255 (4,294,967,296 IP addresses)

IP PROTOCOL

TCP

SOURCE PORT RANGE *OPTIONAL*

All

Examples: 80, 20-22

DESTINATION PORT RANGE *OPTIONAL*

All

Examples: 80, 20-22

+ Additional Ingress Rule

Add Ingress Rules

Cancel

Add Ingress Rules

cancel

Ingress Rule 1

Allows UDP traffic for ports: all

STATELESS

SOURCE TYPE

CIDR

SOURCE CIDR

0.0.0.0/0

Specified IP addresses: 0.0.0.0-255.255.255.255 (4,294,967,296 IP addresses)

IP PROTOCOL

UDP

SOURCE PORT RANGE *OPTIONAL*

All

Examples: 80, 20-22

DESTINATION PORT RANGE *OPTIONAL*

All

Examples: 80, 20-22

+ Additional Ingress Rule

Add Ingress Rules

Cancel

5. You ingress rules will look like the belows.

Ingress Rules

Add Ingress Rules | Remove

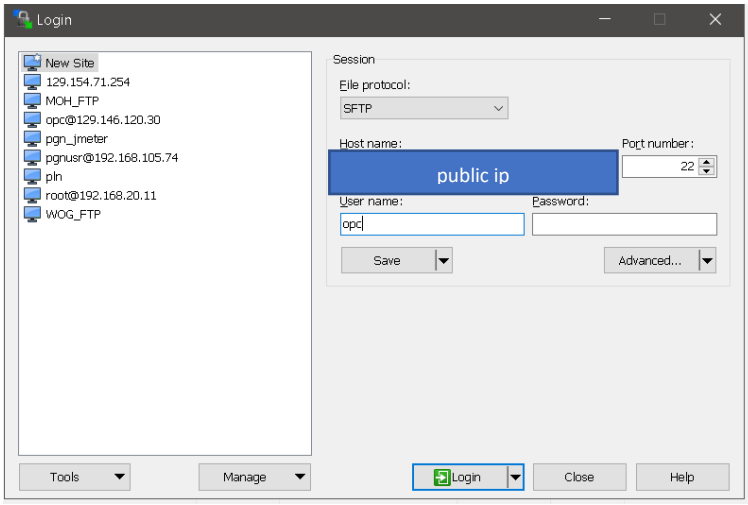
Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	0.0.0.0/0	TCP	All	22		TCP traffic for ports: 22 SSH Remote Login Protocol
No	0.0.0.0/0	ICMP			3, 4	ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set
No	10.0.0.0/16	ICMP			3	ICMP traffic for: 3 Destination Unreachable
No	10.0.0.0/16	TCP	All	6433		TCP traffic for ports: 6433
No	10.0.0.0/16	TCP	All	10250		TCP traffic for ports: 10250
No	10.0.0.0/16	UDP	All	8472		UDP traffic for ports: 8472
No	10.0.0.0/16	TCP	All	All		TCP traffic for ports: All
No	10.0.0.0/16	UDP	All	All		UDP traffic for ports: All
No	0.0.0.0/0	TCP	All	All		TCP traffic for ports: All
No	0.0.0.0/0	UDP	All	All		UDP traffic for ports: All

0 Selected

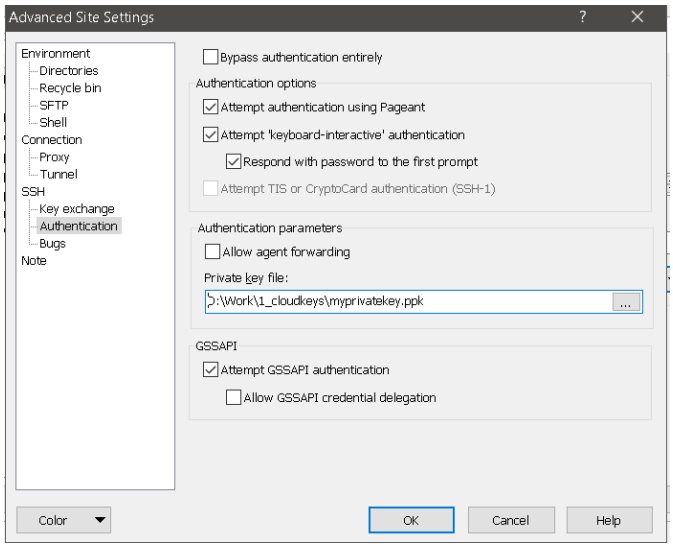
Showing 10 Items < Page 1 >

Copy script [Copy to both master and worker Node]

1. Using winscp and login to master node and worker node. You hostname will be your master node ip/worker node ip and username is opc.



2. You will also need to specify the private key.



3. Copy the kube_lab folder to /home/opc.
4. Run the below command, the change user rights and fix any special character issues.

```
chmod -R 777 /home/opc/kube_lab
sed -i -e "s/^M//" /home/opc/kube_lab/*.sh
sed -i -e 's/\r$//' /home/opc/kube_lab/*.sh
sed -i -e "s/^M//" /home/opc/kube_lab/*.yaml
sed -i -e 's/\r$//' /home/opc/kube_lab/*.yaml
```

Installing kubernetes

Initial Configuration on both the machines [Run on Master and Worker Node]

5. SHH to the both worker and master node
6. Sudo as root

```
sudo su - root
```

7. Execute the Script 0_InitialMachine_Config in both master and worker machine.

Install and configure docker on both the machines [Run on Master and Worker Node]

1. Execute the Script 1_Docker_Config.sh
   With two parameters
   Username for container-registry.oracle.com and Password

Configure Firewall on KubeMaster [Run on Master Node only]

1. Execute the Script 2_KubeMaster_Firewall_Config.sh

Configure Firewall on KubeNode [Run on Worker Node only]

1. Execute the 3_KubeNode_Firewall_Config.sh

Configure Kubernetes on KubeMaster [Run Master Node only]

1. Execute the Script 4_KubeMaster_Kubernetes_Config.sh
2. Take the Token and SSL Value for adding nodes to the cluster.
3. Take note of the command to for worker to join the node
4. Copy the whole string and amend the 5_KubeNode_Kubernetes_Config.sh script in next steps.

```
  sudo chown $(id -u):$(id -g) $HOME/.kube/config


You can now join any number of machines by running the following on each node
as root:

  kubeadm-setup.sh join 10.0.0.2:6443 --token jjpy7j.lce9vmq7zu0o1cvr --discovery-token-ca-cert-hash sha256:ca1dae5ef6330ad181c149bdbdd31319495a8dbd928

coredns-b7df4d4c4-lftbm   0/1     Pending    0          4s
coredns-b7df4d4c4-p46sr   0/1     Pending    0          4s
kube-flannel-ds-f7bhs     0/1     Init:0/1   0          1s
kube-proxy-25qhw          1/1     Running    0          4s
TOKEN jjpy7j.lce9vmq7zu0o1cvr
SSL : ca1dae5ef6330ad181c149bdbdd31319495a8dbd92872d2b75ca85b1dcca4d32
[root@master kube_lab]#
```

Configure Kubernetes on KubeNode [Run on Worker Node only]

1) Replace the parameters in the 5_KubeNode_Kubernetes_Config.sh

- Replace the IP address and port, *`192.0.2.10:6443`*, with the IP address and port that is used by the API Server (the master node). Note that the default port is 6443
- Replace the `--token` value, *`8tipwo.tst0nvf7wcaqjcj0`*, with a valid token for the master node.
- Replace the `--discovery-token-ca-cert-hash` value, *`f2a5b22b658683c3634459c8e7617c9d6c080c72dd149f3eb903445efe9d8346`*, with the correct SHA256 CA certificate hash that is used to sign the token certificate for the master node.

```
# Replace the IP address and port, 192.0.2.10:6443, with the IP address and port that is used by the API Server (the master node). Note that the default port is 6443
# Replace the --token value, 8tipwo.tst0nvf7wcaqjcj0, with a valid token for the master node.
# Replace the --discovery-token-ca-cert-hash value, f2a5b22b658683c3634459c8e7617c9d6c080c72dd149f3eb903445efe9d8346, with the correct SHA256 CA certificate hash that is used to sign the token certificate for the master node.

kubeadm-setup.sh join 10.0.0.2:6443 --token 01guf8.v05ie94748q9ymtj --discovery-token-ca-cert-hash sha256:161dc4ead2037f3ae6637f1c59dcf5fbf3913ab9b7110d79bd8e8dcb9a749143
```

2) Execute the Script 5_KubeNode_Kubernetes_Config.sh

Verify the cluster [Run on master node only]

1. Execute the command on the master node (kubemaster) and verify the out put

kubectl get node

Make sure that worker node is ready. It will wait a while

```
[opc@master ~]$ sudo su
[root@master opc]# kubectl get node
NAME      STATUS   ROLES    AGE    VERSION
master    Ready    master   112m   v1.12.7+1.1.2.el7
worker    Ready    <none>   101m   v1.12.7+1.1.2.el7
[root@master opc]#
```

2. Execute the command 6_Check_Kubedns.sh on master node to check kubernetest networking.

```
[root@master kube_lab]# ./6_Check_Kubedns.sh
pod/busybox unchanged
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      kubernetes.default
Address 1: 10.96.0.1 kubernetes.default.svc.cluster.local
[root@master kube_lab]#
```

Deploy sample application to kubnernetes

Deploy a sample hello world app application

kubectl apply -f /home/opc/kube_lab/helloworld.yaml

```
[root@master kube_lab]# kubectl apply -f /home/opc/kube_lab/helloworld.yaml
service/helloworld created
deployment.apps/helloworld created
[root@master kube_lab]#
```

kubectl get po

```
[root@master kube_lab]# kubectl get po
NAME                          READY   STATUS    RESTARTS   AGE
busybox                       1/1     Running   1          101m
helloworld-5f4dbcdfc9-nrcwc   1/1     Running   0          48s
[root@master kube_lab]#
```

kubectl logs -f helloworld-5f4dbcdfc9-nrcwc

```
[root@master kube_lab]# kubectl logs -f helloworld-5f4dbcdfc9-nrcwc
2019/07/14 04:32:52 Server listening on port 8080
```

kubectl get deployment

```
[root@master kube_lab]# kubectl get deployment
NAME         DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
helloworld   1         1         1            1           2m10s
[root@master kube_lab]#
```

kubectl get services

```
[root@master kube_lab]# kubectl get services
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
helloworld   NodePort    10.96.119.229   <none>        8080:31111/TCP   2m37s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          123m
[root@master kube_lab]#
```

kubectl describe deployment helloworld

kubectl describe service helloworld

```
[root@master kube_lab]# kubectl describe deployment helloworld
Name:                   helloworld
Namespace:              default
CreationTimestamp:      Sun, 14 Jul 2019 04:32:51 +0000
Labels:                 app=helloworld
Annotations:            deployment.kubernetes.io/revision: 1
                        kubectl.kubernetes.io/last-applied-configuration:
                          {"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"helloworld"},"
Selector:               app=helloworld
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=helloworld
           version=v1
  Containers:
   helloworld:
    Image:        bjlim80/bjlim_oracle:hello-app
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   helloworld-5f4dbcdfc9 (1/1 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  3m49s  deployment-controller  Scaled up replica set helloworld-5f4dbcdfc9 to 1
[root@master kube_lab]#
```

```
[root@master kube_lab]# kubectl describe service helloworld
Name:                     helloworld
Namespace:                default
Labels:                   app=helloworld
Annotations:              kubectl.kubernetes.io/last-applied-configuration:
                            {"apiVersion":"v1","kind":"Service","metadata":{"a
Selector:                 app=helloworld
Type:                     NodePort
IP:                       10.96.119.229
Port:                     http  8080/TCP
TargetPort:               8080/TCP
NodePort:                 http  31111/TCP
Endpoints:                10.244.1.13:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Test the application

Access the cluster ip of the service name

curl http://10.96.119.229:8080

curl http://[worker node public ip]:31111

```
[root@master kube_lab]# kubectl getsvc
Error: unknown command "getsvc" for "kubectl"
Run 'kubectl --help' for usage.
unknown command "getsvc" for "kubectl"
[root@master kube_lab]# kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
helloworld   NodePort    10.96.119.229   <none>        8080:31111/TCP   6m28s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP          127m
[root@master kube_lab]# curl http://10.96.119.229:8080
Hello, world!
Version: 1.0.0
Hostname: helloworld-5f4dbcdfc9-mbxgp
[root@master kube_lab]# curl http://132.145.206.79:31111
Hello, world!
Version: 1.0.0
Hostname: helloworld-5f4dbcdfc9-nrcwc
[root@master kube_lab]#
```

Scale the application

kubectl scale deployment --replicas=3 helloworld

kubectl get po

```
[root@master kube_lab]# kubectl scale deployment --replicas=3 helloworld
deployment.extensions/helloworld scaled
[root@master kube_lab]# kubectl get po
NAME                          READY   STATUS    RESTARTS   AGE
busybox                       1/1     Running   1          105m
helloworld-5f4dbcdfc9-mbxgp   1/1     Running   0          14s
helloworld-5f4dbcdfc9-nrcwc   1/1     Running   0          4m59s
helloworld-5f4dbcdfc9-xwtn9   1/1     Running   0          14s
[root@master kube_lab]#
```

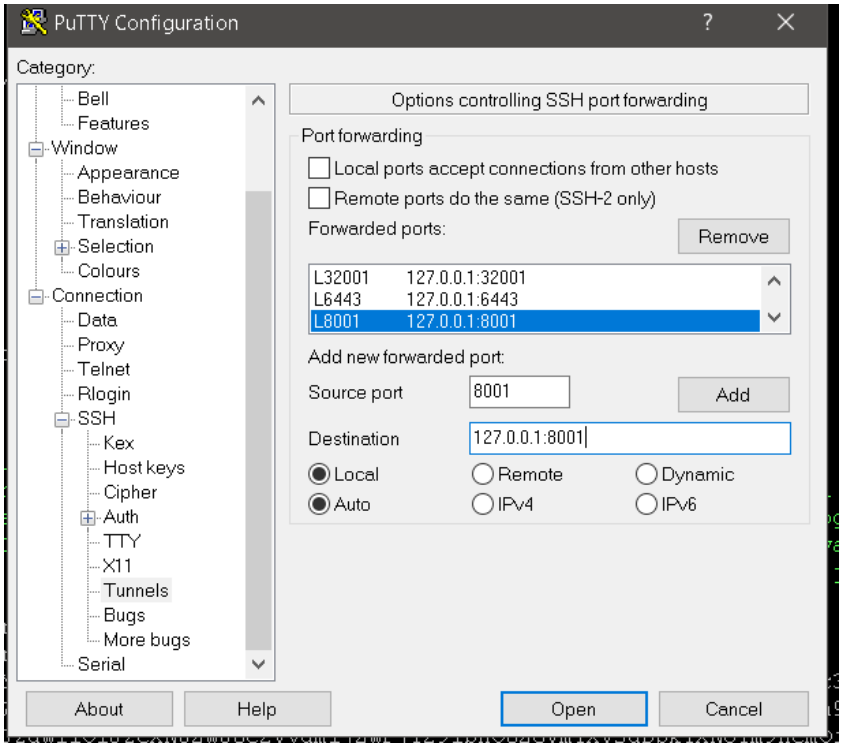
Delete deployment

kubectl delete -f /home/opc/kube_lab/helloworld.yaml

```
[root@master kube_lab]# kubectl delete -f /home/opc/kube_lab/helloworld.yaml
service "helloworld" deleted
deployment.apps "helloworld" deleted
[root@master kube_lab]#
```

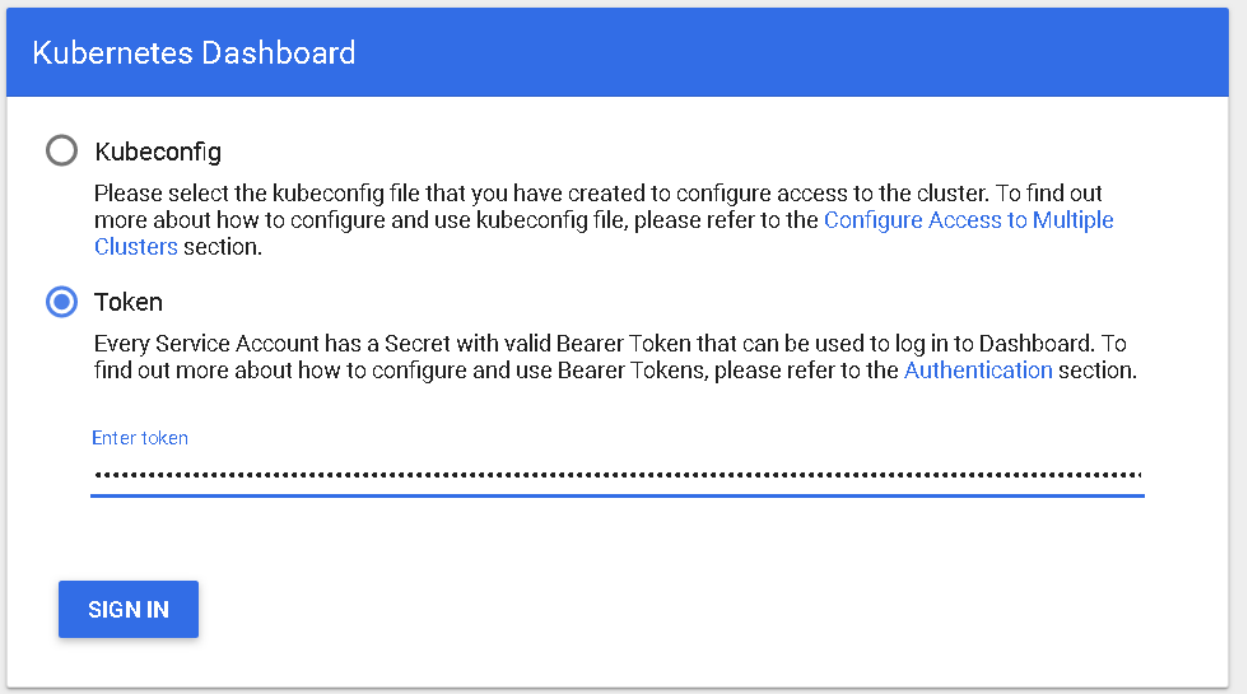
Setup kubernetes dashboard [Optional Lab]

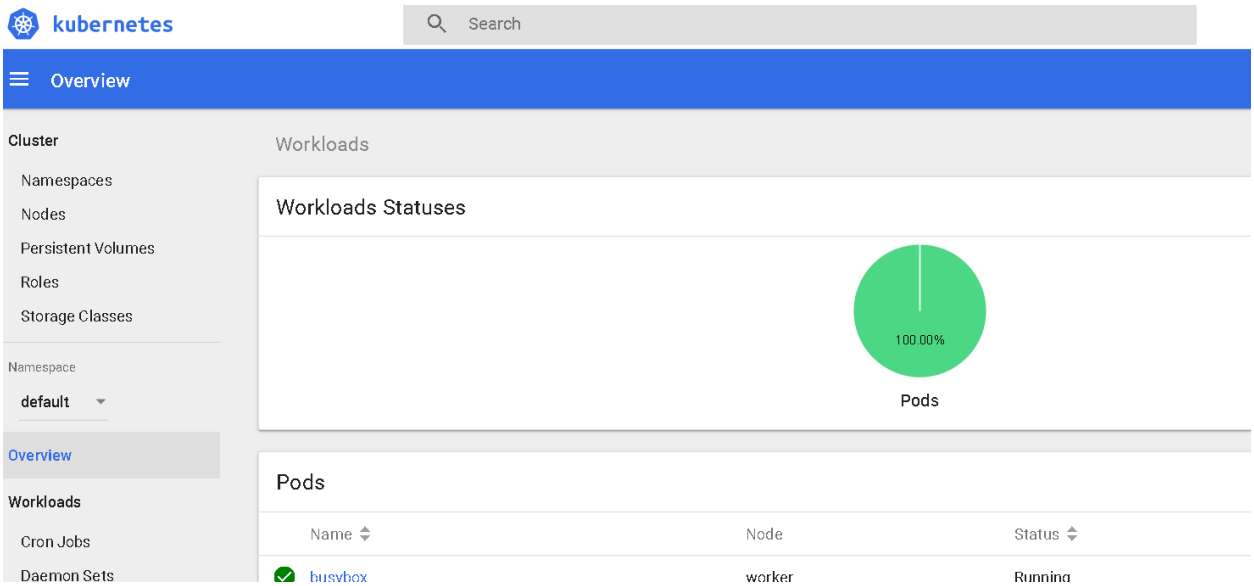
1) Run 7_Kube_proxy.sh on master node
2) Allow tunnel in putty



3) Access the below in local machine and use the token above to authenticate.

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


Kubernetes Dashboard
Kubeconfig
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.
Token
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.
Enter token
SIGN IN


kubernetes
Search
Overview
Cluster
Namespaces
Nodes
Persistent Volumes
Roles
Storage Classes
Namespace
default
Overview
Workloads
Cron Jobs
Daemon Sets
Workloads
Workloads Statuses
100.00%
Pods
Pods
Name
Node
Status
busybox
worker
Running