**Setting up Kubernetes Cluster on Ubuntu**

Contents

[Set hostname and add entries in /etc/hosts file 1](#_Toc60933350)

[Install and enable Docker service on all computers (master + nodes) in the cluster 2](#_Toc60933351)

[Disable swap on all computers 2](#_Toc60933352)

[Enable IP forwarding on all computers (master + nodes) in the cluster 2](#_Toc60933353)

[Install Kubectl, kubelet and kubeadm on all nodes 3](#_Toc60933354)

[Initialize Kubernetes Cluster using kubeadm (from kubemaster) 3](#_Toc60933355)

[Deploy Calico Pod Network Add-on (Master Node) 4](#_Toc60933356)

[Bash Completion on Master node 5](#_Toc60933357)

[Test and Verify Kubernetes Cluster by deploying nginx-web 5](#_Toc60933358)

[Web UI (Dashboard) 6](#_Toc60933359)

[Creating sample user 6](#_Toc60933360)

[Creating a Service Account 6](#_Toc60933361)

[Creating a ClusterRoleBinding 7](#_Toc60933362)

[Getting a Bearer Token 7](#_Toc60933363)

[Command line proxy 8](#_Toc60933364)

[Troubleshooting Scenarios 10](#_Toc60933365)

[References: 10](#_Toc60933366)

[Version History 10](#_Toc60933367)

Create and Prepare the VMs – KubeMaster and Worker nodes

* Machine 1 (Ubuntu 20.04 LTS Server) – K8s-master – 192.168.56.104
* Machine 2 (Ubuntu 20.04 LTS Server) – K8s-node-0 – 192.168.56.105
* Machine 3 (Ubuntu 20.04 LTS Server) – K8s-node-1 – 192.168.56.106

# Set hostname and add entries in /etc/hosts file

// Run this command on master node which sets master hostname

**sudo hostnamectl set-hostname "k8s-master"**

// Run this command on node-1 which sets workernode 01

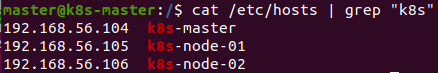
**sudo hostnamectl set-hostname "k8s-node-0"**

// Run this command on node-2 which sets workernode 02 and so on

**sudo hostnamectl set-hostname "k8s-node-1"**



Configure the correct IP Addresses and ensure that the entries are added to /etc/hosts file in all nodes including kubemaster. When looked at the hosts file, it should show similar output for all computers in the cluster.



# Install and enable Docker service on all computers (master + nodes) in the cluster

We are using Docker as Container Runtime. Install it on all the nodes including kubemaster

**sudo apt update**

**sudo apt install -y docker.io**

alternatively, you could combine the above commands using &&

**sudo apt update && sudo apt install -y docker.io**

after installing, enable docker service on all computers

**sudo systemctl enable docker.service --now**

Verify the status of docker service and it’s version

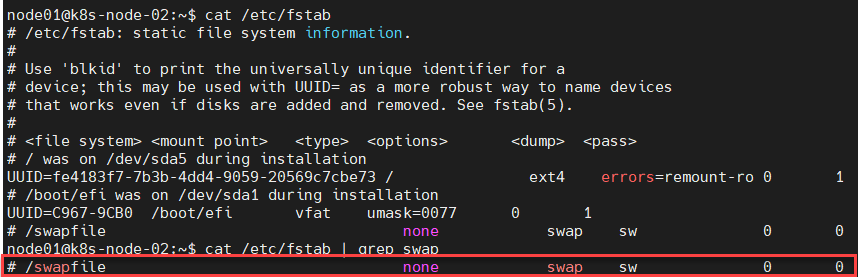
**systemctl status docker**

**docker --version**

## Disable swap on all computers

**sudo nano /etc/fstab**

comment the swapfile like below

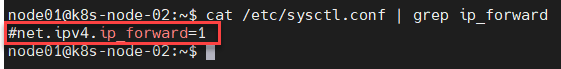


Run swapoff command to disable the swap on the fly on all the computers

**sudo swapoff -a**

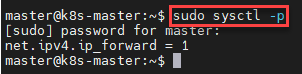
## Enable IP forwarding on all computers (master + nodes) in the cluster

make sure to comment out (#) the line net.ipv4.ip\_forward=1 in the file /etc/sysctl.conf



After commenting out, run the command

sudo sysctl -p



# Install Kubectl, kubelet and kubeadm on all nodes

**sudo apt install -y apt-transport-https curl**

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

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

**sudo apt update**

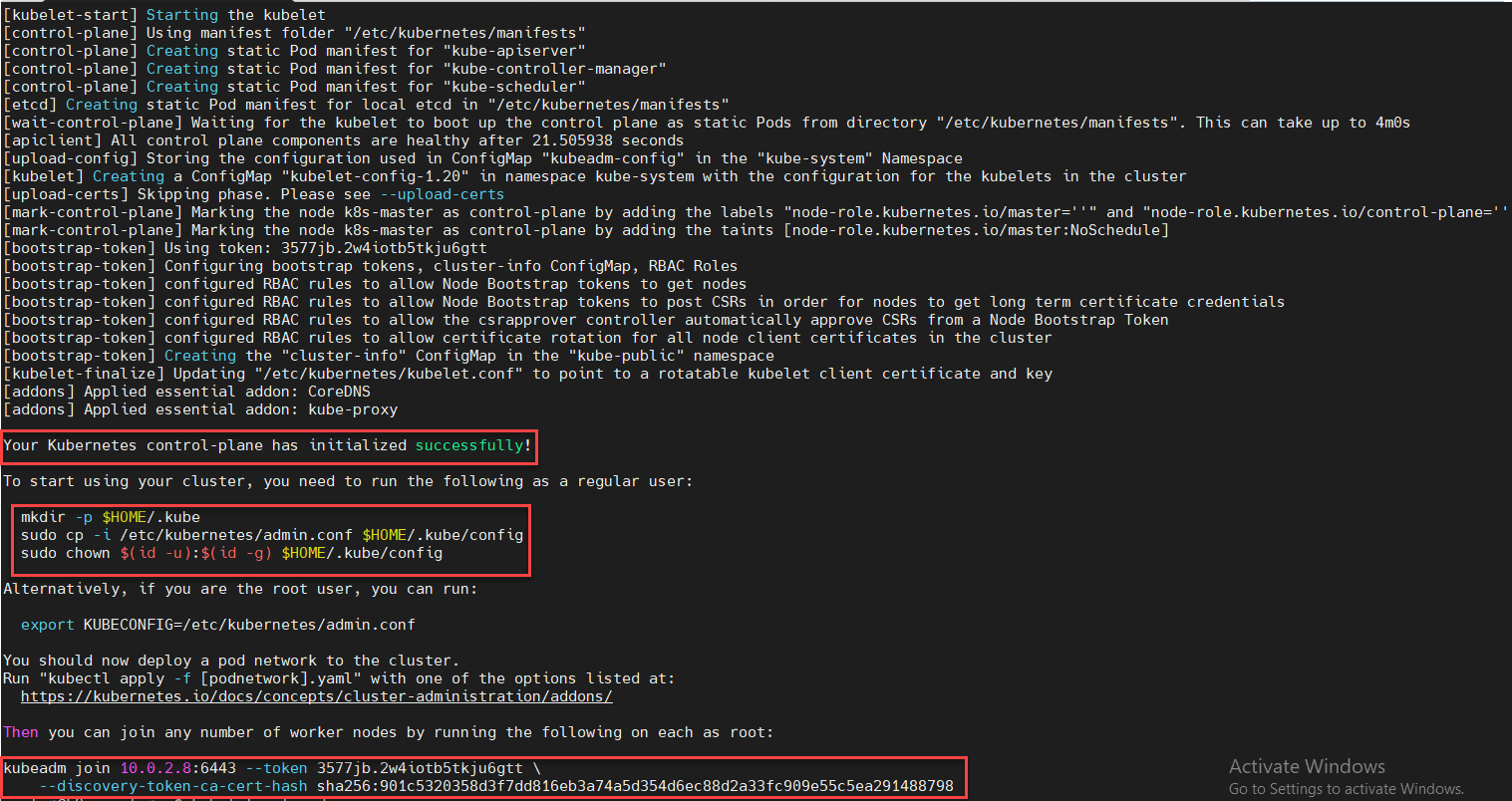
**sudo apt install -y kubelet kubeadm kubectl**

alternatively, you could run all the commands in one go like below:

**sudo apt install -y apt-transport-https curl && curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add && sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main" && sudo apt update && sudo apt install -y kubelet kubeadm kubectl**

## Initialize Kubernetes Cluster using kubeadm (from kubemaster)

Once the cluster is initialized successfully, we will get the following output



**mkdir -p $HOME/.kube**

**sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

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

To Join the worker nodes to the kubemaster and form a cluster, Copy “kubeadm join” command and paste it on both nodes (worker nodes). Connect to k8s-node-01 and k8s-node-02 and run following command.)

**kubeadm join 10.0.2.8:6443 --token y74yko.dcfa0jsnarx0p376 --discovery-token-ca-cert-hash sha256:a62eab24dd31c500175e856a7c4233dacec9b09b13bbcbd9cfd906eb725ea09ah**

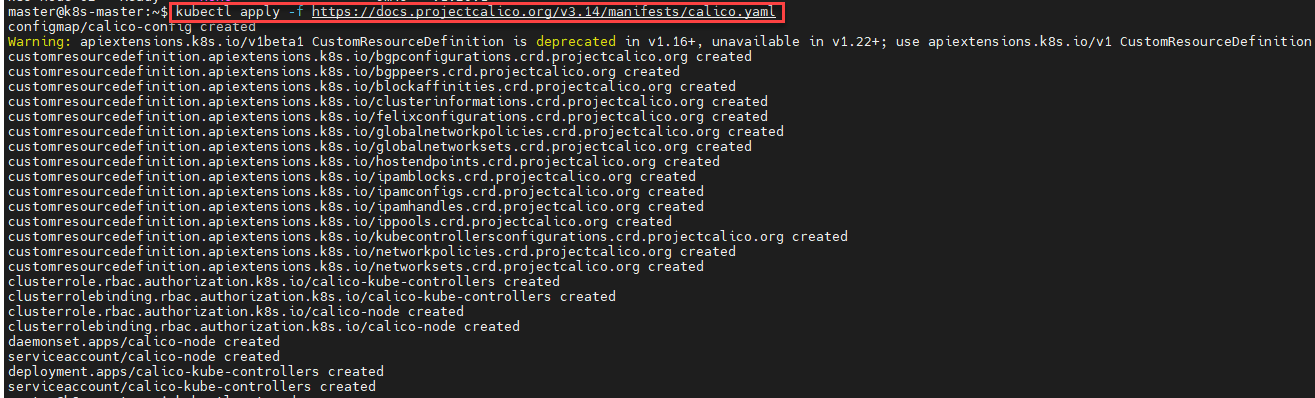
Remember the above is for sample. It will be different when you are running kubeadm init which could be captured from the above screenshot (for reference only

From the master node run “kubectl get nodes” command to verify nodes status

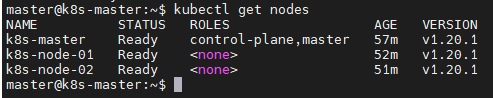
# Deploy Calico Pod Network Add-on (Master Node)

From the master node, run the following command to install Calico pod network add-on,

**kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml**

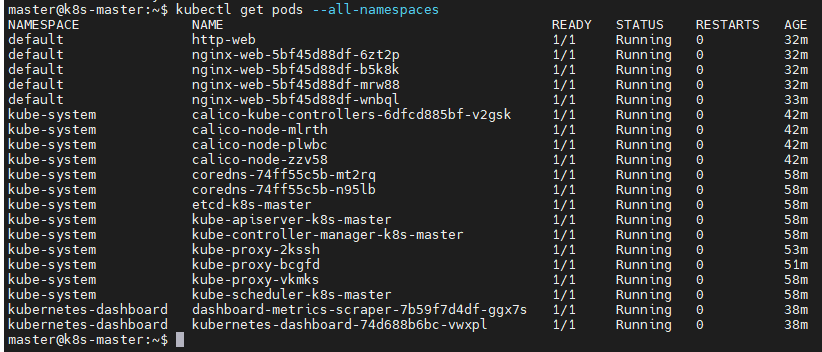


Once it has been deployed successfully then nodes status will become ready, let’s re-run kubectl command to verify nodes status

****

Run below command to verify status of pods from all namespaces

**kubectl get pods –all-namespaces**



## Bash Completion on Master node

To enable bash completion feature on your master node, execute the followings

**echo 'source <(kubectl completion bash)' >>~/.bashrc**

**source .bashrc**

****

# Test and Verify Kubernetes Cluster by deploying nginx-web

Let’s create a deployment named nginx-web with nginx container image in the default namespace, run the following kubectl command from the master node.

**kubectl create deployment nginx-web --image=nginx**

**kubectl get deployments.apps**

**kubectl get deployments.apps -o wide**

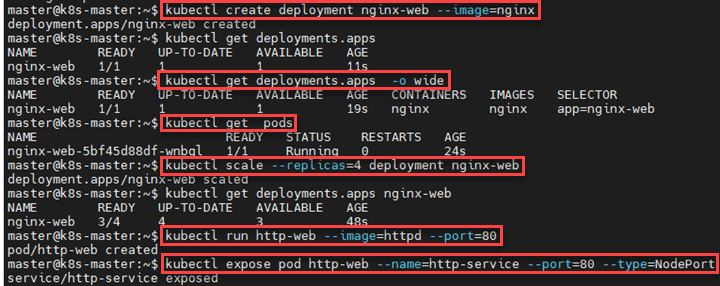
**kubectl get pods**

**kubectl scale --replicas=4 deployment nginx-web**

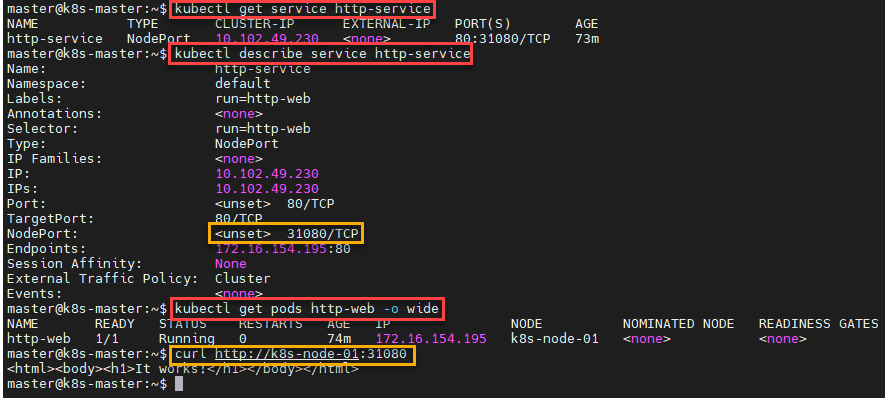
**kubectl get deployments.apps nginx-web**

**kubectl expose pod http-web --name=http-service --port=80 --type=NodePort**

**kubectl get service http-service**



Now verify status of your deployment using following commands,



# Web UI (Dashboard)

The Dashboard UI is not deployed by default. To deploy it, run the following command:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml

## Creating sample user

### Creating a Service Account

We are creating Service Account with name admin-user in namespace kubernetes-dashboard first.

cat <<EOF | kubectl apply -f -

apiVersion: v1

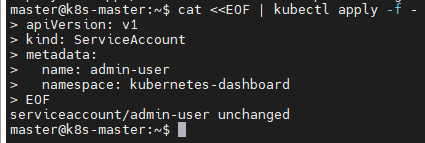
kind: ServiceAccount

metadata:

name: admin-user

namespace: kubernetes-dashboard

EOF



### Creating a ClusterRoleBinding

In most cases after provisioning cluster using kops, kubeadm or any other popular tool, the ClusterRole cluster-admin already exists in the cluster. We can use it and create only ClusterRoleBinding for our ServiceAccount. If it does not exist then you need to create this role first and grant required privileges manually.

cat <<EOF | kubectl apply -f -

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: admin-user

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

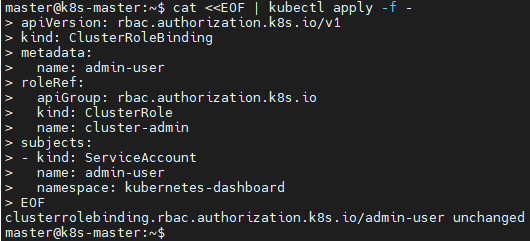
subjects:

- kind: ServiceAccount

name: admin-user

namespace: kubernetes-dashboard

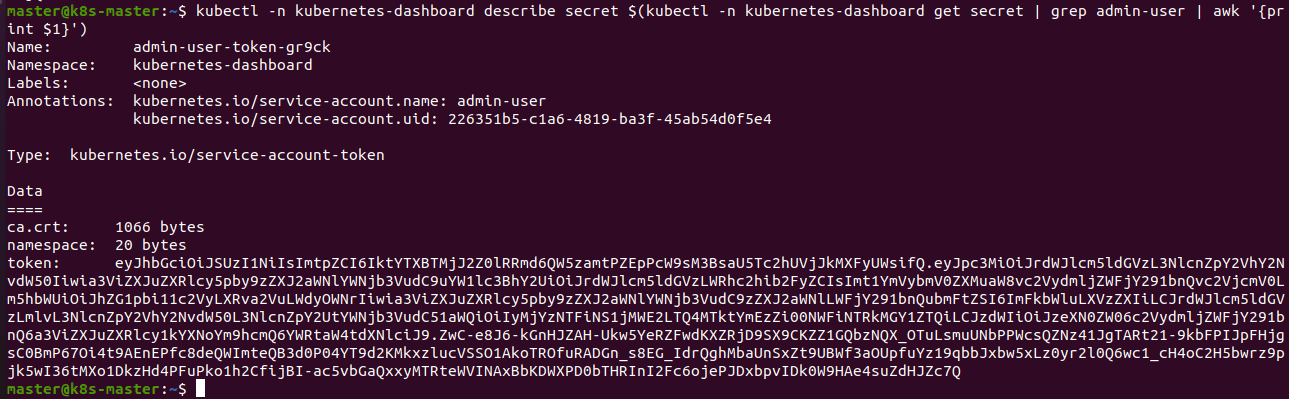
EOF



### Getting a Bearer Token

From the master node console, run the following command to get the bearer token

kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')



Copy the token from the above command

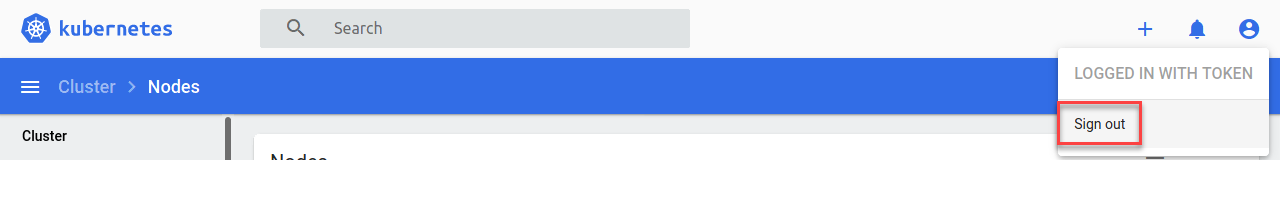
### Command line proxy

You can access Dashboard using the kubectl command-line tool by running the following command:

kubectl proxy

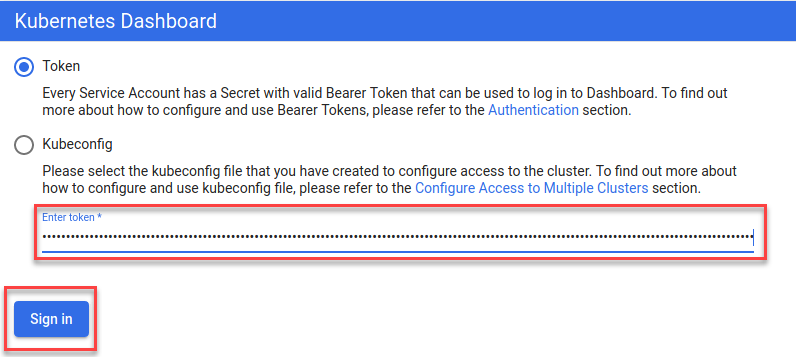
Kubectl will make Dashboard available at <http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/>.

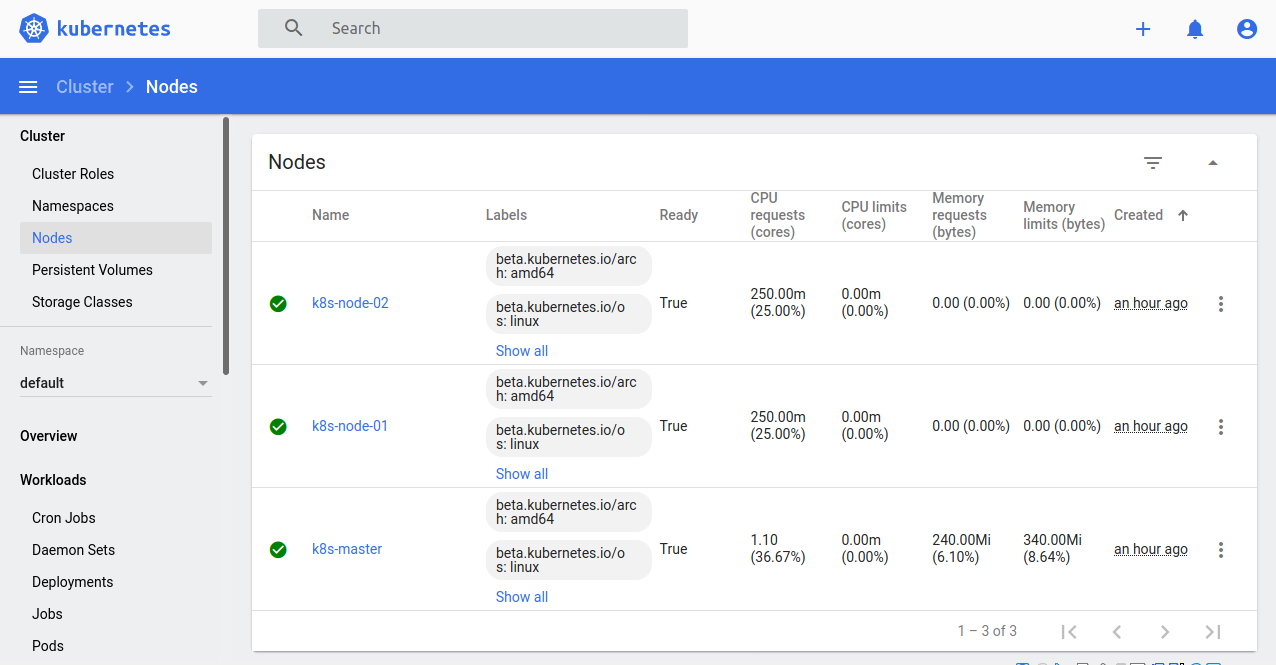
The UI can *only* be accessed from the machine where the command is executed. See kubectl proxy --help for more options.

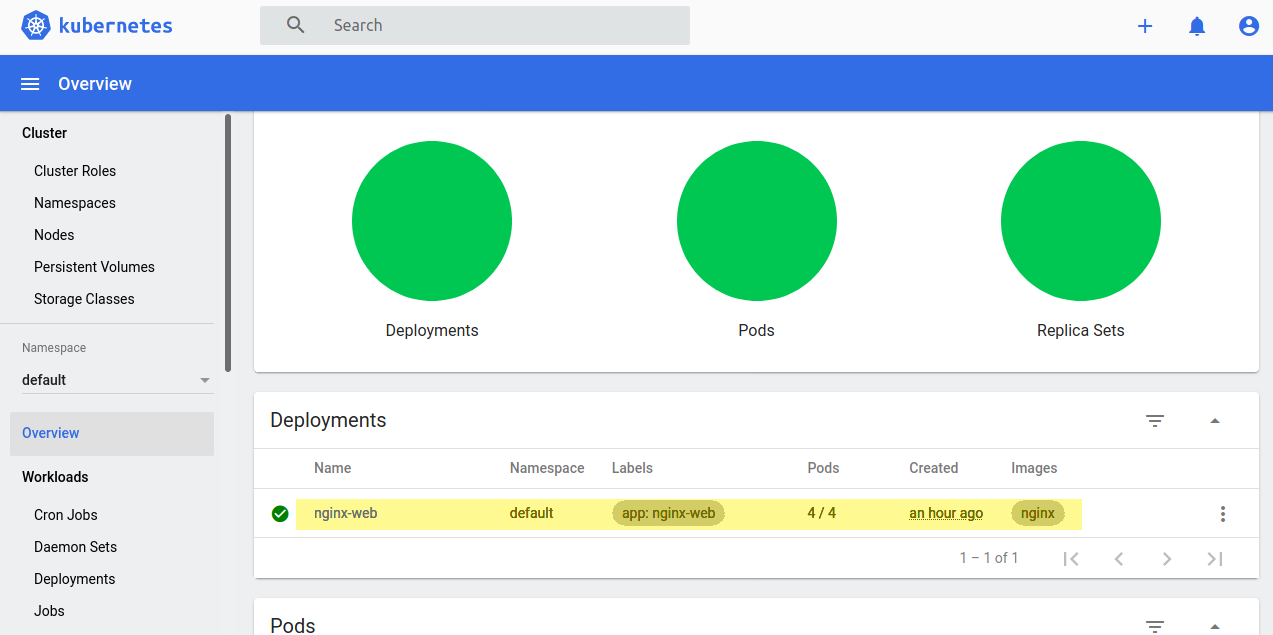


After you sign out, copy the bearer token from the command and authenticate with it

kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')



****



# Troubleshooting Scenarios

# References:

<https://www.linuxtechi.com/install-kubernetes-k8s-on-ubuntu-20-04>

<https://github.com/kubernetes/dashboard/blob/master/docs/user/access-control/creating-sample-user.md>

<https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/>

# Version History

|  |  |  |
| --- | --- | --- |
| Date of Change | Changed By | Summary of Changes |
| 1/7/2021 | Murali Krishna T G | Original document |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |