# **How To Install Kubernetes Cluster On Ubuntu 16.04**

Link: <https://www.edureka.co/blog/install-kubernetes-on-ubuntu>

The first step towards [*Kubernetes Course*](https://www.edureka.co/kubernetes-certification) is installing Kubernetes. This blog is a step by step guide to install Kubernetes on top of Ubuntu VMs (Virtual Machines). Here, one VM will act as the master and the other VM will be the node. You can then replicate the same steps to deploy the Kubernetes cluster onto your prod.

*Note: For this installation, we recommend a fresh Ubuntu 16.04 image since Kubernetes can take up a lot of resources. If your installation fails at any time, then execute all the steps mentioned from the very beginning in a fresh VM, because debugging would take longer.*

To install Kubernetes, you have to diligently follow the 3 phases that come as part of the installation process:

1. [Pre-requisites to install Kubernetes](https://www.edureka.co/blog/install-kubernetes-on-ubuntu#KubernetesInstallationPreRequisites)
2. [Setting up Kubernetes environment](https://www.edureka.co/blog/install-kubernetes-on-ubuntu#SetupKubernetesEnvironment)
3. [Installing Kubeadm, Kubelet, Kubectl](https://www.edureka.co/blog/install-kubernetes-on-ubuntu#InstallingKubeadmKubeletKubectl)
4. [Starting the Kubernetes cluster from master](https://www.edureka.co/blog/install-kubernetes-on-ubuntu#StartingKubernetesCluster)
5. [Getting the nodes to join the cluster](https://www.edureka.co/blog/install-kubernetes-on-ubuntu#NodesJoiningKubeCluster)

## Pre-requisites To Install Kubernetes

Since we are dealing with VMs, we recommend the following settings for the VMs:-

*Master*:

* 2 GB RAM
* 2 Cores of CPU

*Slave/ Node*:

* 1 GB RAM
* 1 Core of CPU

By this point of time, I have assumed you have 2 plain Ubuntu VMs imported onto your Oracle Virtual Box. So, I’l just get along with the installation process.

## Pre-Installation Steps On Both Master & Slave (To Install Kubernetes)

The following steps have to be executed on both the master and node machines. Let’s call the the master as ‘*kmaster*‘ and node as ‘*knode*‘.

First, login as ‘sudo’ user because the following set of commands need to be executed with ‘sudo’ permissions. Then, update your ‘apt-get’ repository.

$ sudo su

# apt-get update

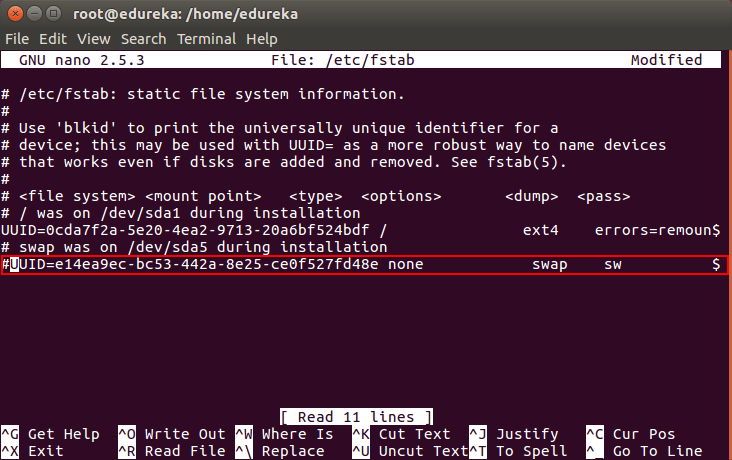
Note: After logging-in as ‘sudo’ user, note that your shell symbol will change to ‘#’ from ‘$’.

### Turn Off Swap Space

Next, we have to turn off the swap space because Kubernetes will start throwing random errors otherwise. After that you need to open the ‘fstab’ file and comment out the line which has mention of swap partition.

# swapoff -a

# nano /etc/fstab

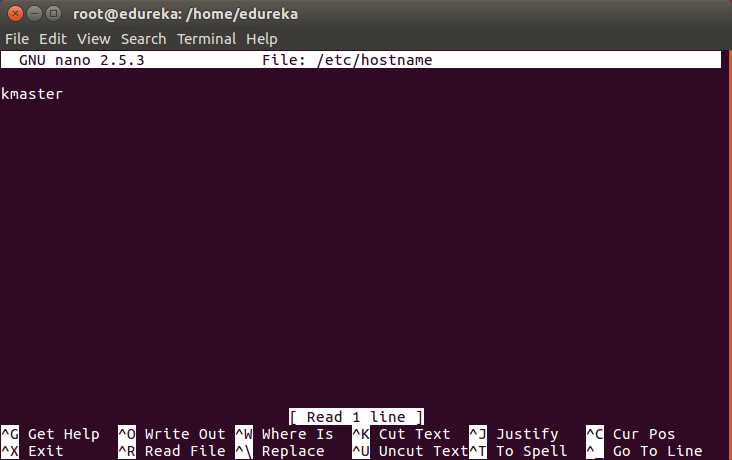


Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

### Update The Hostnames

To change the hostname of both machines, run the below command to open the file and subsequently rename the master machine to ‘kmaster’ and your node machine to ‘knode’.

# nano /etc/hostname



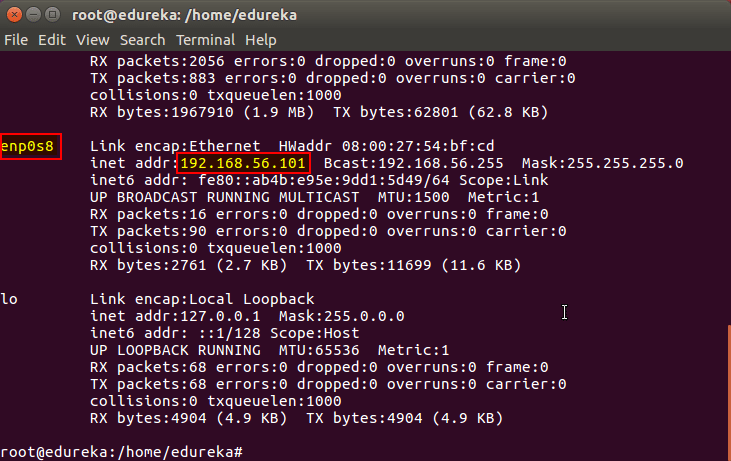
Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

### Update The Hosts File With IPs Of Master & Node

Run the following command on both machines to note the IP addresses of each.

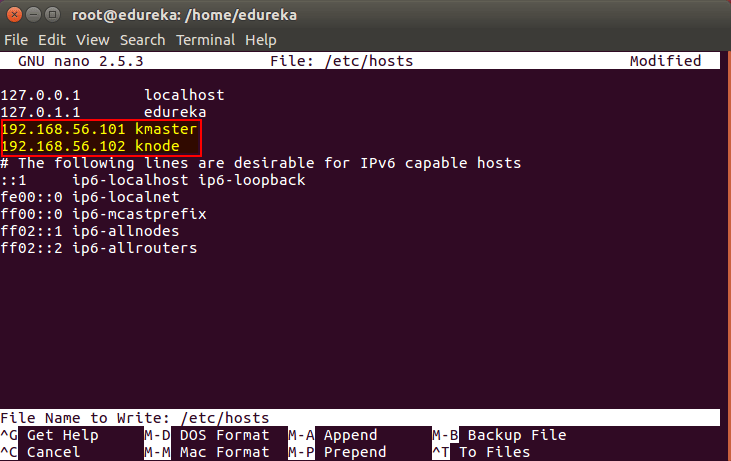
# ifconfig

Make a note of the IP address from the output of the above command. The IP address which has to be copied should be under “enp0s8”, as shown in the screenshot below.



Now go to the ‘hosts’ file on both the master and node and add an entry specifying their respective IP addresses along with their names ‘kmaster’ and ‘knode’. This is used for referencing them in the cluster. It should look like the below screenshot on both the machines.

# nano /etc/hosts



Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

### Setting Static IP Addresses

Next, we will make the IP addresses used above, static for the VMs. We can do that by modifying the network interfaces file. Run the following command to open the file:[**ning**](https://www.edureka.co/kubernetes-certification)

# nano /etc/network/interfaces

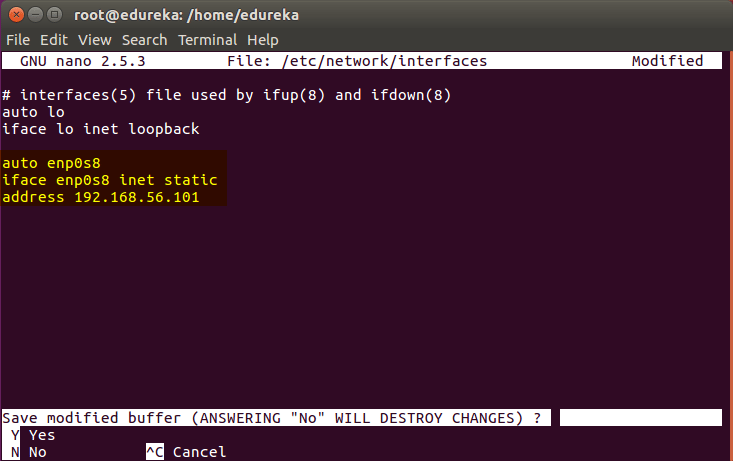
Now enter the following lines in the file.

auto enp0s8

iface enp0s8 inet static

address <*IP-Address-Of-VM*>

It will look something like the below screenshot.



Then press ‘Ctrl+X’, then press ‘Y’ and then press ‘Enter’ to Save the file.

After this, restart your machine(s).

### Install OpenSSH-Server

Now we have to install openshh-server. Run the following command:

# sudo apt-get install openssh-server

### Install Docker

Now we have to install Docker because Docker images will be used for managing the containers in the cluster. Run the following commands:

# sudo su

# apt-get update

# apt-get install -y docker.io

Next we have to install these 3 essential components for setting up Kubernetes environment: kubeadm, kubectl, and kubelet.

Run the following commands before installing the Kubernetes environment.

# apt-get update && apt-get install -y apt-transport-https curl

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

# cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

# apt-get update

## Install kubeadm, Kubelet And Kubectl

Now its time to install the 3 essential components. *Kubelet* is the lowest level component in Kubernetes. It’s responsible for what’s running on an individual machine. *Kuebadm* is used for administrating the Kubernetes cluster. *Kubectl* is used for controlling the configurations on various nodes inside the cluster.

# apt-get install -y kubelet kubeadm kubectl

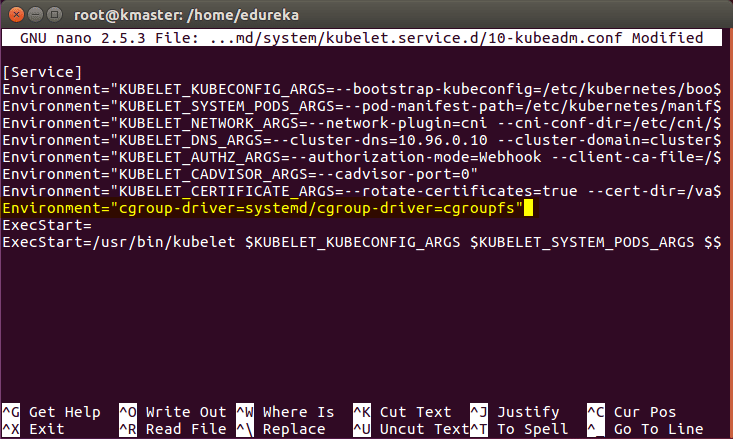
### Updating Kubernetes Configuration

Next, we will change the configuration file of Kubernetes. Run the following command:

# nano /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

This will open a text editor, enter the following line after the last “Environment Variable”:

Environment=”cgroup-driver=systemd/cgroup-driver=cgroupfs”



Now press Ctrl+X, then press Y, and then press Enter to Save.

*Voila!* You have successfully installed Kubernetes on both the machines now!

As of now, only the Kubernetes environment has been setup. But now, it is time to install Kubernetes completely, by moving onto the next 2 phases, where we will individually set the configurations in both machines.

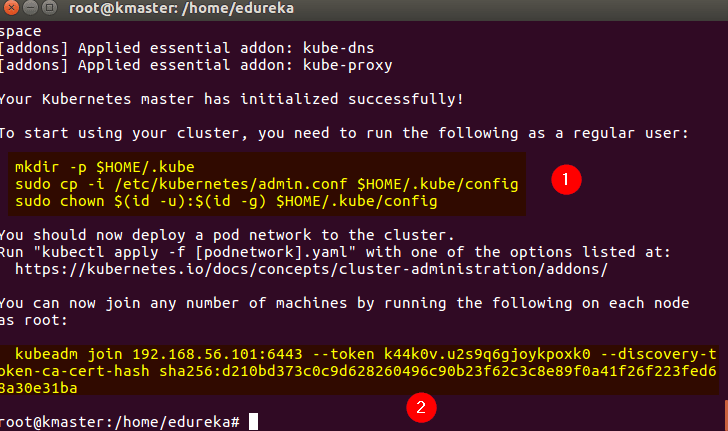
## Steps Only For Kubernetes Master VM (kmaster)

*Note*: These steps will only be executed on the master node (kmaster VM).

Step 1: We will now start our Kubernetes cluster from the master’s machine. Run the following command:

# kubeadm init --apiserver-advertise-address=192.168.1.202 --pod-network-cidr=10.244.0.0/16

1. You will get the below output. The commands marked as (1), execute them as a non-root user. This will enable you to use kubectl from the CLI
2. The command marked as (2) should also be saved for future. This will be used to join nodes to your cluster



If output like above then everything is good to go and skip the error solution step and proceed.

**Error Solution: —--------------start—--------------------------**  
**If an error occur after running the above command like blow**

[kubelet-check] It seems like the kubelet isn't running or healthy:

Then the solution is given blow:

The problem was cgroup driver. Kubernetes cgroup driver was set to systems but docker was set to systemd. So I created **/etc/docker/daemon.json** and added below:

If daemon.json file is present at the location /etc/docker then add the blow line init and save if the file is not present at the given location then create the file at that location with same name and add the following line init and save using the following command.

**Solution:**

sudo nano /etc/docker/daemon.json

After that past the blow line init then press ctrl+x then press y then press enter

{ "exec-opts": ["native.cgroupdriver=systemd"] }

Then run the following commands after it

sudo systemctl daemon-reload

sudo systemctl restart docker

sudo systemctl restart kubelet

After that run the blow command

sudo kubeadm reset

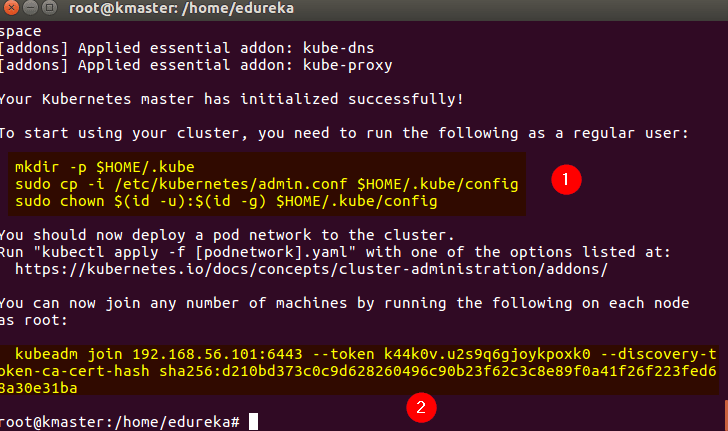
sudo rm -r /etc/cni/net.d

sudo rm -r $HOME/.kube/config

sudo -i

# kubeadm init --apiserver-advertise-address=192.168.1.202 --pod-network-cidr=10.244.0.0/16

1. You will get the below output. The commands marked as (1), execute them as a non-root user. This will enable you to use kubectl from the CLI
2. The command marked as (2) should also be saved for future. This will be used to join nodes to your cluster



**Error solution —-----------------------------End—----------------------------------**

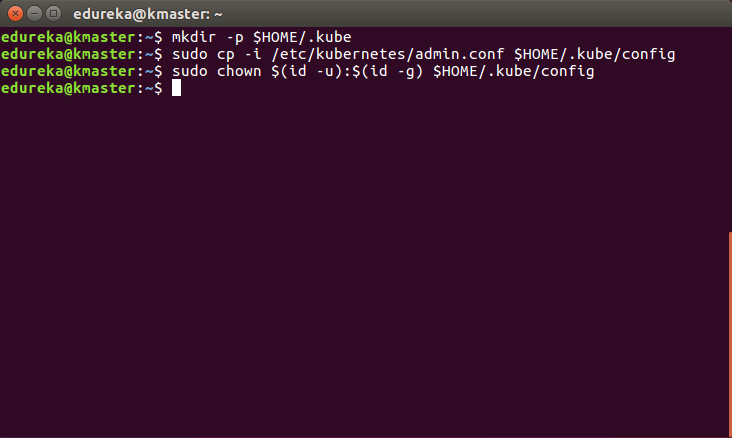
Step 2: As mentioned before, run the commands from the above output as a non-root user

$ mkdir -p $HOME/.kube

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

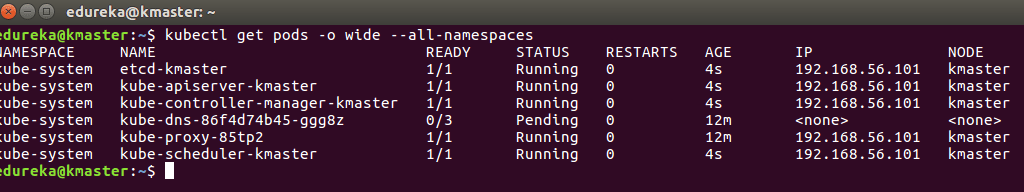
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

It should look like this:



To verify, if kubectl is working or not, run the following command:

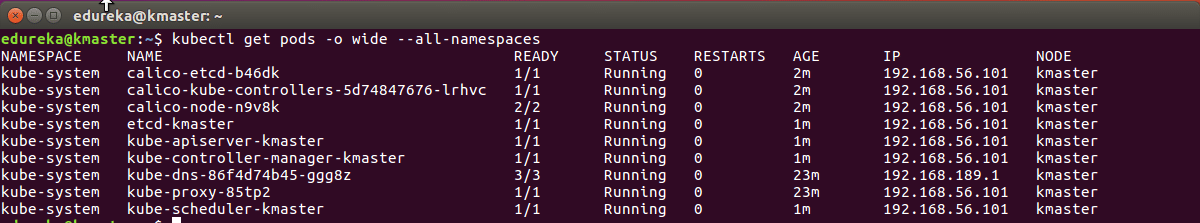
$ kubectl get pods -o wide --all-namespaces



Download calico ymal file from that website link given blow and then apply as (kubectl apply -f calico.ymal)

Link: <https://projectcalico.docs.tigera.io/getting-started/kubernetes/self-managed-public-cloud/gce>

After some time, you will notice that all pods shift to the running state

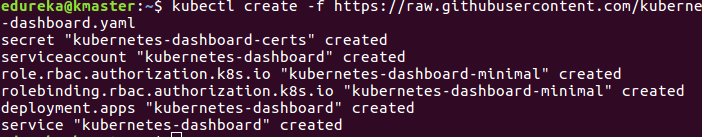


Step 4: Next, we will install the dashboard. To install the Dashboard, run the following command:

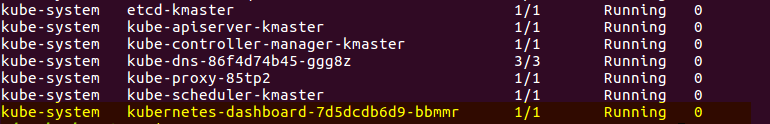
$ kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/recommended.yaml>

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

It will look something like this:



Step 5: Your dashboard is now ready with it’s the pod in the running state.



Step 6: By default dashboard will not be visible on the Master VM. Run the following command in the command line:

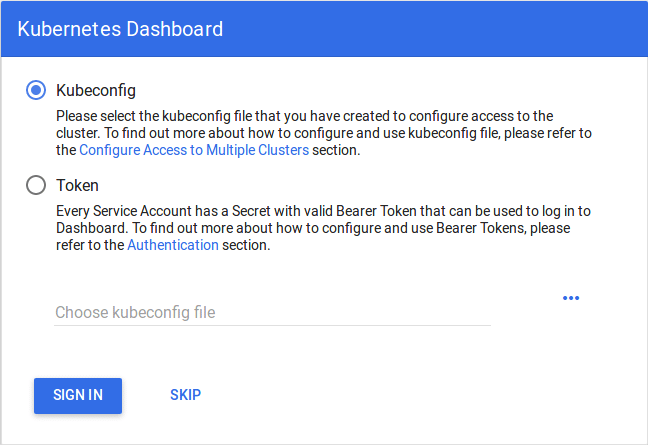
$ kubectl proxy

Then you will get something like this:

kubectl proxy - install kubernetes - edureka

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

ou will then be prompted with this page, to enter the credentials:



Step 7: In this step, we will create the service account for the dashboard and get it’s credentials.

Note: Run all these commands in a new terminal, or your kubectl proxy command will stop.

Run the following commands:

1. This command will create a service account for dashboard in the default namespace

$ kubectl create serviceaccount dashboard -n default

2. This command will add the cluster binding rules to your dashboard account

$ kubectl create clusterrolebinding dashboard-admin -n default

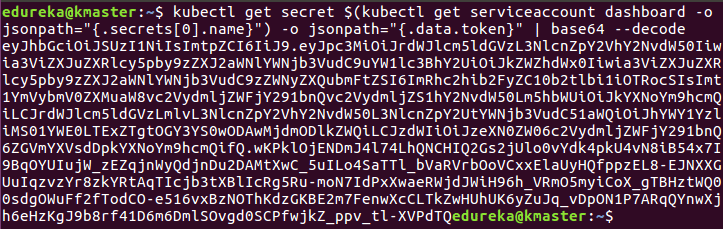
--clusterrole=cluster-admin

--serviceaccount=default:dashboard

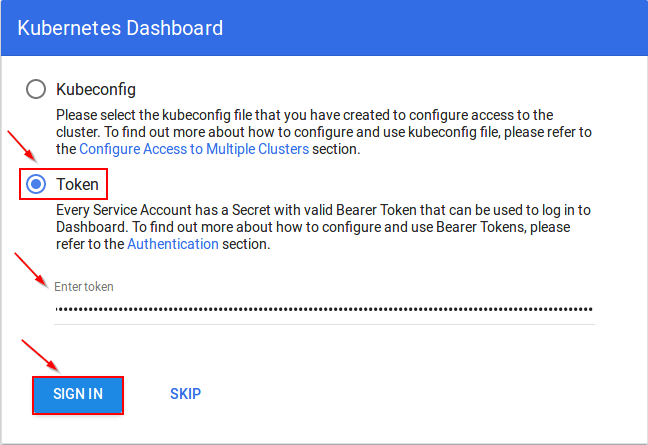
3. This command will give you the token required for your dashboard login

$ kubectl get secret $(kubectl get serviceaccount dashboard -o jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode

You should get the token like this:



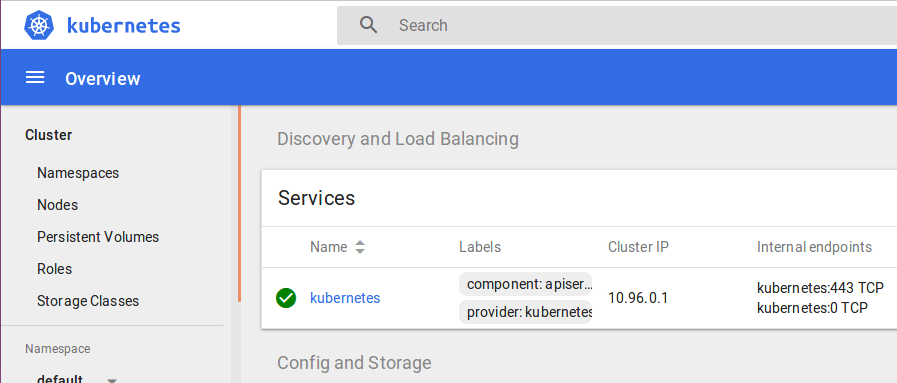
4. Copy this token and paste it in Dashboard Login Page, by selecting token option



5. You have successfully logged into your dashboard!

### **[Kubernetes Certification Training Course: Administrator (CKA)](https://www.edureka.co/kubernetes-certification)**

[Weekday / Weekend Batc](https://www.edureka.co/kubernetes-certification)

[](https://www.edureka.co/kubernetes-certification)

## Steps For Only Kubernetes Node VM (knode)

It is time to get your node, to join the cluster! This is probably the only step that you will be doing on the node, after installing kubernetes on it.

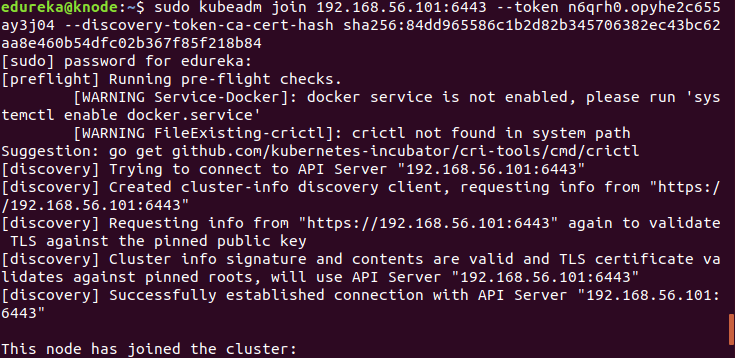
Run the join command that you saved, when you ran ‘kubeadm init’ command on the master.

**Do the following steps to all the worker machines**:

Go to worker machine using ssh as given blow [ssh hostname@ipaddress].  
**ssh worker@192.168.1.201**

Note: Run this command with “sudo”.

sudo kubeadm join --apiserver-advertise-address=<ip-address-of-the master> --pod-network-cidr=192.168.0.0/16



*Bingo!* Your Kubernetes Cluster is ready if you get something similar to the above screenshot.

**Error solution:----------------------------------start—------------------------------------**

If blow error accor

error execution phase preflight: [preflight] Some fatal errors occurred

<https://jhooq.com/kubernetes-error-execution-phase-preflight-preflight/#file-already-exist>

**Solution:**

sudo kubeadm reset

sudo rm -r /etc/cni/net.d

sudo rm -r $HOME/.kube/config

**After adding the worker nodes to the cluster by following the above steps run the blow command on the master.**

watch kubectl get pods -o wide --all-namespaces

And check node status if it is 0/1 then the cluster is not ok and that error solution is given blow.

**Error solution:--------------------------start:-----------------------------------------------**

If kubectl get po -o wide –all-namespaces give node status 0/1 then it can be fix given blow

<https://github.com/projectcalico/calico/issues/2561>

Change the calico.yaml file by overriding that ip to etho-ip by using the following steps.

**Solution:**

# Specify interface

- name: IP\_AUTODETECTION\_METHOD

value: "interface=eth1"

**like :**

containers:

# Runs calico-node container on each Kubernetes node. This

# container programs network policy and routes on each

# host.

- name: calico-node

image: calico/node:v3.8.2

env:

# Use Kubernetes API as the backing datastore.

- name: DATASTORE\_TYPE

value: "kubernetes"

# Wait for the datastore.

- name: WAIT\_FOR\_DATASTORE

value: "true"

# Set based on the k8s node name.

- name: NODENAME

valueFrom:

fieldRef:

fieldPath: spec.nodeName

# Choose the backend to use.

- name: CALICO\_NETWORKING\_BACKEND

valueFrom:

configMapKeyRef:

name: calico-config

key: calico\_backend

# Cluster type to identify the deployment type

- name: CLUSTER\_TYPE

value: "k8s,bgp"

# Specify interface

- name: IP\_AUTODETECTION\_METHOD

value: "interface=eth1"

# Auto-detect the BGP IP address.

- name: IP

value: "autodetect"

# Enable IPIP

- name: CALICO\_IPV4POOL\_IPIP

value: "Always"

# Set MTU for tunnel device used if ipip is enabled

- name: FELIX\_IPINIPMTU

Update the calico.ymal file and apply it again on the master machine using following command.

kubectl apply -f calico.ymal

After that again check nodes status using the following command

watch kubectl get pods -o wide --all-namespaces

After some time the status of all the nodes will be 1/1.

**Error Solution:--------------------------------End—--------------------------------------------------------**

After that deploy your pods on the cluster if it work fine then good. If some pods not start then

Check log of the pod that is not starting using the following command

Kubect logs -f

# If it give the blow output then the solution is given blow

# kubernetes: Error from server: Get //IPaddress:10250/containerLogs/default/l: dial tcp 10.19.0.41:10250: getsockopt: no route to host

**Error Solution:**

[**https://www.techrunnr.com/kubernetes-error-from-server-get-ipaddress10250-containerlogs-default-l-dial-tcp-10-19-0-4110250-getsockopt-no-route-to-host/**](https://www.techrunnr.com/kubernetes-error-from-server-get-ipaddress10250-containerlogs-default-l-dial-tcp-10-19-0-4110250-getsockopt-no-route-to-host/)

# kubernetes: Error from server: Get //IPaddress:10250/containerLogs/default/l: dial tcp 10.19.0.41:10250: getsockopt: no route to host

**Solution**

Disable firewall on all the machines using the following commands

systemctl status firewalld

systemctl stop firewalld

systemctl disable firewalld

How to disable firewall:

<https://www.tecmint.com/start-stop-disable-enable-firewalld-iptables-firewall/>

Note:

If you don’t want to disable firewall then the solution is on the blow link.

[**https://www.techrunnr.com/kubernetes-error-from-server-get-ipaddress10250-containerlogs-default-l-dial-tcp-10-19-0-4110250-getsockopt-no-route-to-host/**](https://www.techrunnr.com/kubernetes-error-from-server-get-ipaddress10250-containerlogs-default-l-dial-tcp-10-19-0-4110250-getsockopt-no-route-to-host/)