**Creation of k8s cluster:-**

* **Installing the pre requisites**
* **Setting up the container run time**
* **Installing the kubeadm**
* **Initializing the cluster**

**Installing the pre requisites:-**

Forwarding IPv4 and letting iptables see bridged traffic Execute the below mentioned instructions:

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

*# sysctl params required by setup, params persist across reboots*

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

net.ipv4.ip\_forward = 1

EOF

*# Apply sysctl params without reboot*

sudo sysctl --system

Verify that the br\_netfilter, overlay modules are loaded by running below instructions:

lsmod | grep br\_netfilter

lsmod | grep overlay

Verify that the net.bridge.bridge-nf-call-iptables, net.bridge.bridge-nf-call-ip6tables, net.ipv4.ip\_forward system variables are set to 1 in your sysctl config by running below instruction

sysctl net.bridge.bridge-nf-call-iptables net.bridge.bridge-nf-call-ip6tables net.ipv4.ip\_forward

**Setting up the container run time :-**

Install the necessary dependencies with:

**sudo apt install curl gnupg2 software-properties-common apt-transport-https ca-certificates -y**

Add the GPG key with:

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –**

Add the required repository with:

**sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"**

Install containerd with the commands:+

**sudo apt update**

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

Change to the root user with:

**sudo su -**

Create a new directory for containerd with:

**mkdir -p /etc/containerd**

Generate the configuration file with:

**containerd config default>/etc/containerd/config.toml**

Exit from the root user with:

**Exit**

#### **Configuring the systemd cgroup driver**

To use the systemd cgroup driver in /etc/containerd/config.toml with runc, set

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]

...

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]

SystemdCgroup = true

Note :- the above change must be done

Restart containerd with the command:

**sudo systemctl restart containerd**

Enable containerd to run at startup with:

**sudo systemctl enable containerd**

**Installing the kubeadm:-**

1. Update the apt package index and install packages needed to use the Kubernetes apt repository:
2. sudo apt-get update
3. sudo apt-get install -y apt-transport-https ca-certificates curl
4. Download the Google Cloud public signing key:
5. sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
6. Add the Kubernetes apt repository:
7. echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
8. Update apt package index, install kubelet, kubeadm and kubectl, and pin their version:
9. sudo apt-get update
10. sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

**Initializing the cluster**

**On Master:**

**kubeadm init --pod-network-cidr=192.168.0.0/16**

**>Copy the token and paste it into the worker node.**

**exit**

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

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

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

**kubectl apply -f** [**https://docs.projectcalico.org/manifests/calico.yaml**](https://docs.projectcalico.org/manifests/calico.yaml)

## **Deploying the Dashboard UI**

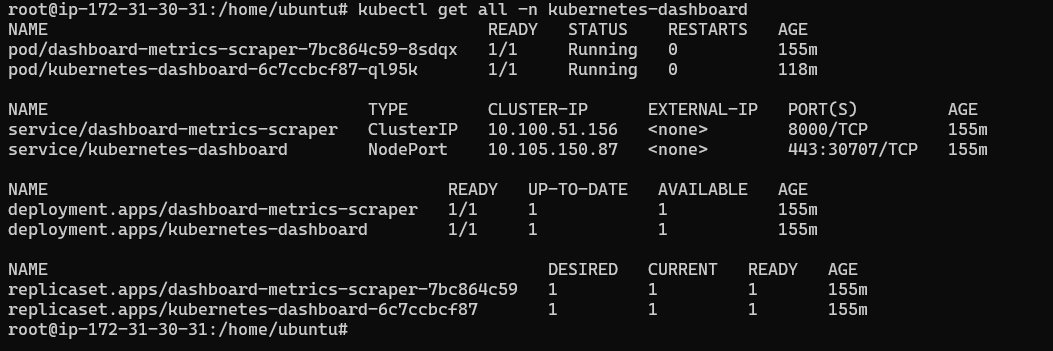
The Dashboard UI is not deployed by default. To deploy it, run the following command:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

Now, verify all of the resources were installed successfully by running the [**kubectl get**](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) command. The command below fetches information about all resources on the cluster created in the **kubernetes-dashboard** (**-n**) namespace.

kubectl get all -n kubernetes-dashboard

As you see below, all the resources inside the Kubernetes dashboard, such as service, deployment, replica set, pods, are deployed successfully in the cluster.



 Edit the Kubernetes dashboard service created in the previous section using the [**kubectl edit**](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#edit) command, as shown below. Running the below command will open an editable service configuration file displaying the service configuration

kubectl edit service/kubernetes-dashboard -n kubernetes-dashboard

Once the file is opened, change the type of service from **ClusterIP** to **NodePort** and save the file as shown below. By default, the service is only available internally to the cluster (**ClusterIP**) but changing to **NodePort** exposes the service to the outside.

Setting the service type to **NodePort** allows all IPs (inside or outside of) the cluster to access the service.

In the below code snippet, the Kubernetes dashboard service is listening on TCP port 443 and maps TCP port 8443 from port 443 to the dashboard pod port TCP/8443.

**Text

Description automatically generated**



Find the name of each pod that step two in the previous section created using the **kubectl get pods** command enumerating all pods across all namespaces with the **--all-namespaces** parameter.

You should see a pod that starts with **kubernetes-dashboard**.

kubectl get pods --all-namespaces

Next, delete the Kubernetes dashboard pod using the name found in step three using the [kubectl delete](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#delete) command. For this tutorial, the name of the pod is **kubernetes-dashboard-6c7ccbcf87-ql95k**.

**Text

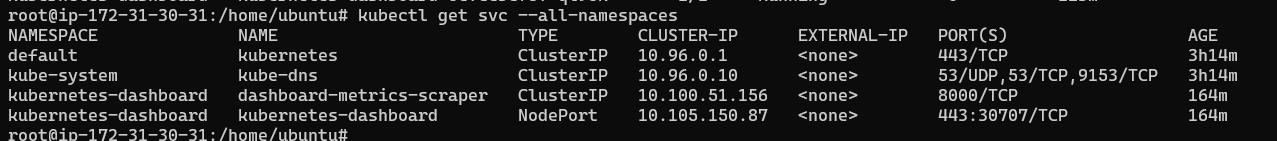
Description automatically generated**

Whenever you modify the service type, you must delete the pod. Once deleted, Kubernetes will create a new one for you with the updated service type to access the entire network.

kubectl delete pod kubernetes-dashboard- 7bc864c59-8sdqx -n kubernetes-dashboard

 Verify the **kubernetes-dashboard** service has the correct type by running the [kubectl get svc](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get)**--all-namespace** command. You will now notice that the service type has changed to **NodePort,** and the service exposes the pod’s internal TCP port 30265 using the outside TCP port of 443.

kubectl get svc --all-namespaces



kubectl create serviceaccount dashboard -n kubernetes-dashboard

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

kubectl create clusterrolebinding dashboard-admin -n kubernetes-dashboard --clusterrole=cluster-admin --serviceaccount=default:dashboard

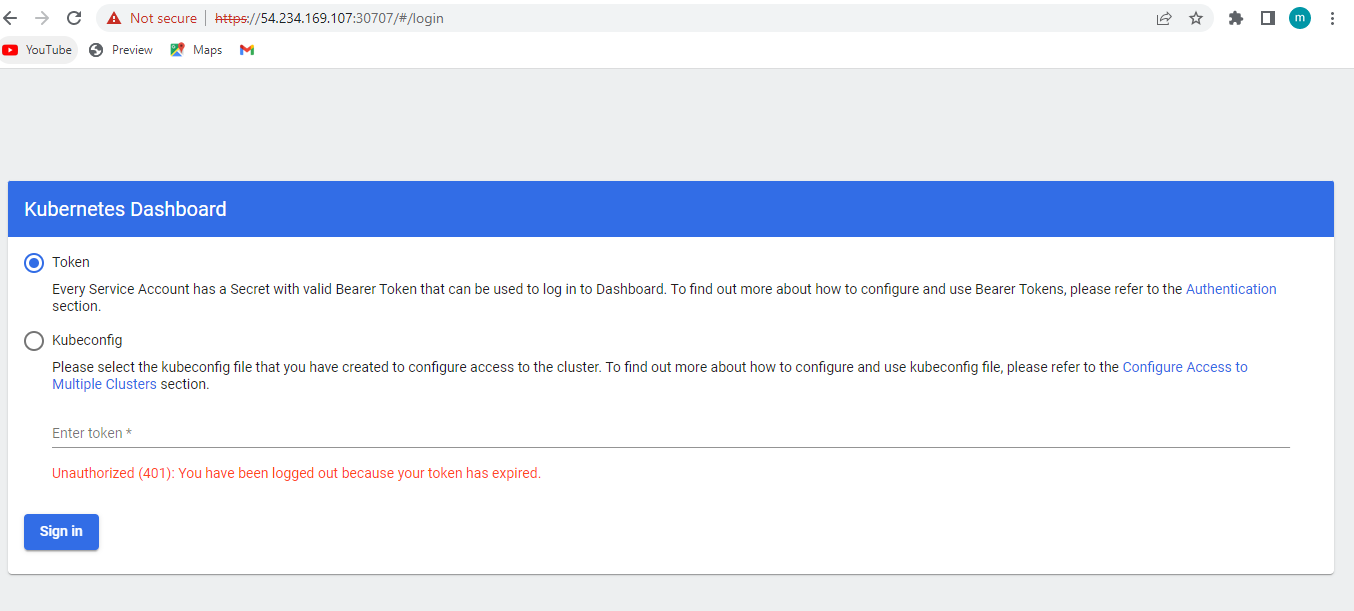
## Accessing the Kubernetes Dashboard

Now that you’ve installed and set up the Kubernetes dashboard, the only thing left to do is enjoy its functionality!

Open your favorite browser and navigate to [*https://kuberntes-master-node*](https://kuberntes-master-node/):NodePort/#/login to access the Kubernetes dashboard.

The Kubernetes master node is the host you’ve installed the dashboard onto, while the node port is the node port found in step five of the previous section.

The main Kubernetes Dashboard page requires you to authenticate either via a valid bearer token or with a pre-existing [kubeconfig file](https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/" \t "_blank). For this tutorial, you’ll be using the token generated in the previous section to access the Kubernetes dashboard.

****

**Enter the token you get in above steps**

**If you can not see the tocken then follow these steps**

#### **Create The Dashboard Service Account**

Run the following command to create a service account

kubectl create serviceaccount kubernetes-dashboard-admin-sa -n kube-system

The command will create a service account in the namespace of kube-system. replace your namespace instate of kube-system

#### 2. Bind The Service Account To The Cluster-Admin Role

kubectl create clusterrolebinding kubernetes-dashboard-admin-sa --clusterrole=cluster-admin --serviceaccount=kube-system:kubernetes-dashboard-admin-sa

#### 

#### 3. List Secretes

kubectl get secrets -n kube-system

if you using kubernetes 1.23 or above please use following to get secret

kubectl -n kube-system create token kubernetes-dashboard-admin-sa

then you dont have to follow step 4.

#### 4. Get The Token From Secret

kubectl describe secret kubernetes-dashboard-admin-sa-token-lj8cc -n kube-system

Here your secret name can be different. Now copy the token and use it to login kubernetes dashboard.

**Reffrence:-----**

[**https://thenewstack.io/how-to-deploy-kubernetes-with-kubeadm-and-contai**](https://thenewstack.io/how-to-deploy-kubernetes-with-kubeadm-and-contai)

[**https://kubernetes.io/docs/setup/production-environment/container-runtimes**](https://kubernetes.io/docs/setup/production-environment/container-runtimes)

[**https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/**](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/)

[**https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/**](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/)

**for dash boards**

[**https://adamtheautomator.com/kubernetes-dashboard/#Accessing\_the\_Kubernetes\_Dashboard**](https://adamtheautomator.com/kubernetes-dashboard/#Accessing_the_Kubernetes_Dashboard)

[**https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/**](https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/)