# **Set Up a Two-Nodes (Master node and Worker node) Kubernetes Cluster on Ubuntu 20.04**

# Set Up a Two-Nodes (Master node and Worker node) Kubernetes Cluster on Ubuntu 20.04. Kubernetes is an open-source system originally designed by Google for deploying, managing, scaling, and automating containerized applications. The open-source platform works with various containers and creates a framework to run these containers in clusters with images built via Docker.Kubernetes provides relevant services covering all these features to help deploy cloud-native applications anywhere. Kubernetes has become one of the best and fast-growing solutions for development.

### **Ojbectives :**

### 1) Install and configure redies masters node and worker node.

### 2) Setting up Kubernetes Cluster .

### 

### **Use Case:**

### Kubernetes can work virtually with any infrastructure, including the public cloud, a private cloud, hybrid, or an on-premises server.

### Using Kubernetes Can Improve Developer Productivity

### Affordable Solution

### Self-Healing Service.

### Multi-Cloud capability.

**Steps to be followed Install and configure Kubernetes cluster (master and worker node) :**

## **Prerequisites :**

* Two servers running an Ubuntu 20.04 operating system on the aws Cloud Platform
* Minimum 2 GB RAM and 2 Core CPUs on each node
* A root password configured on each server

**Step1: install and configure master and worker node :**

sudo apt-get update

**Step 2:Disable Swap**

# Disable Swap

# swapoff -a

**Step 3: edit the /etc/sysctl.conf file to enable the IP forwarding:**

#vim /etc/sysctl.conf

and uncommnet following line:

net.ipv4.ip\_forward = 1

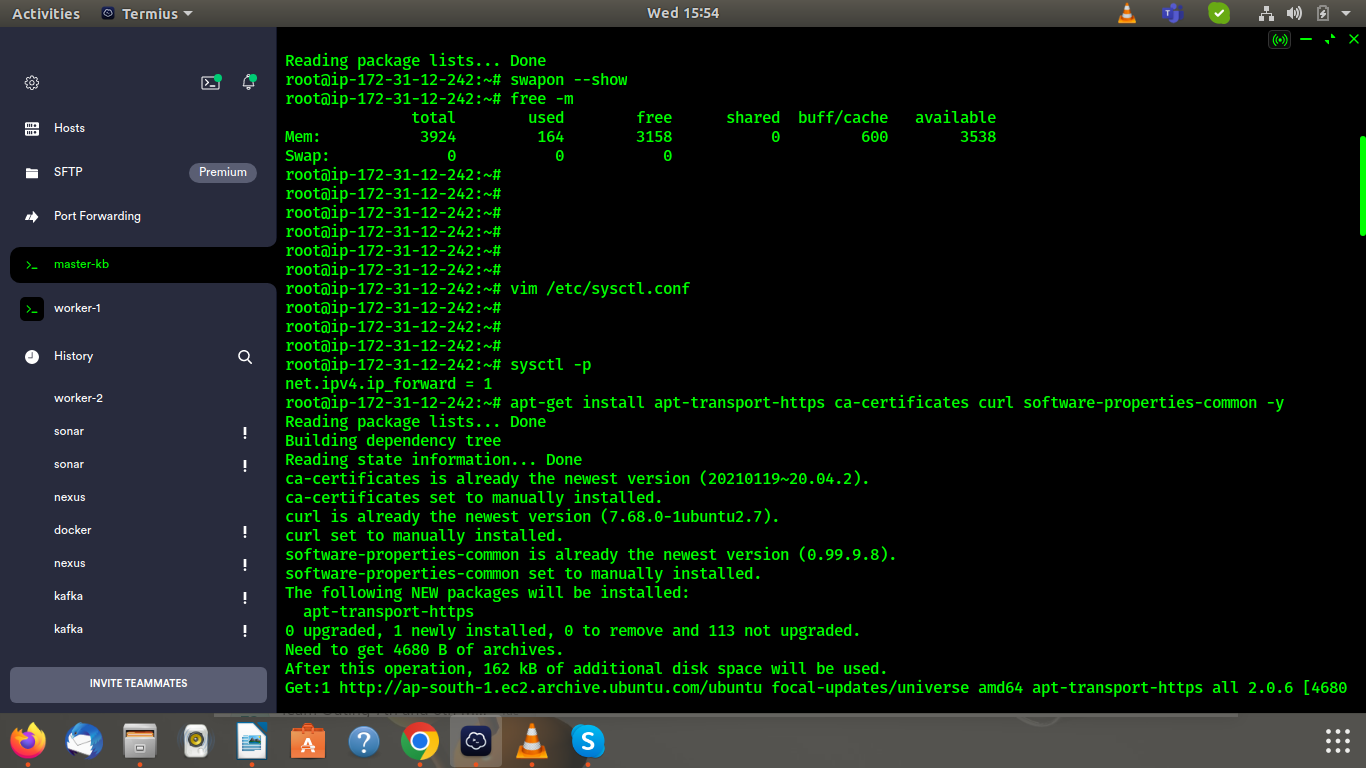
apply the configuration changes:

#sysctl -p

**Step 4: Install Docker CE :**

**A) First, install the required dependencies to access Docker repositories over HTTPS:**

# apt-get install apt-transport-https ca-certificates curl software-properties-common -y

****

**B) run the curl command to download and add Docker’s GPG key:**

# curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> | apt-key add -

**C) add Docker’s official repository to the APT:**

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

**D) Run the following command to install Docker CE:**

# apt-get install docker-ce -y

**Step 4: Add Kubernetes Repository**

**A) Add the Kubernetes GPG key:**

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

**B) Add the Kubernetes repository to APT:**

# apt-add-repository "deb <http://apt.kubernetes.io/> kubernetes-xenial main"

**C) Once the repository is added, update the APT cache using the command below:**

# apt-get update -y

## **Step 5 : Install Kubernetes Components (Kubectl, kubelet and kubeadm) :**

## **A) Run the following command on all nodes to install all Kubernetes components:**

## # apt-get install kubelet kubeadm kubectl -y

## 

## **B)Need to update the cgroupdriver on all nodes. You can do it by creating the following file:**

## # vim /etc/docker/daemon.json

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

"log-driver": "json-file",

"log-opts":

{ "max-size": "100m" },

"storage-driver": "overlay2”

}

## **C) Now reload the systemd daemon and restart the Docker service with the following command:**

## #systemctl daemon-reload

## # systemctl restart docker

## # systemctl enable docker

## 

## **Step 6 – Initialize Kubernetes Master Node :**

## **A) Run the kubeadm command-line tool to initialize the Kubernetes cluster:**

## **#** kubeadm init

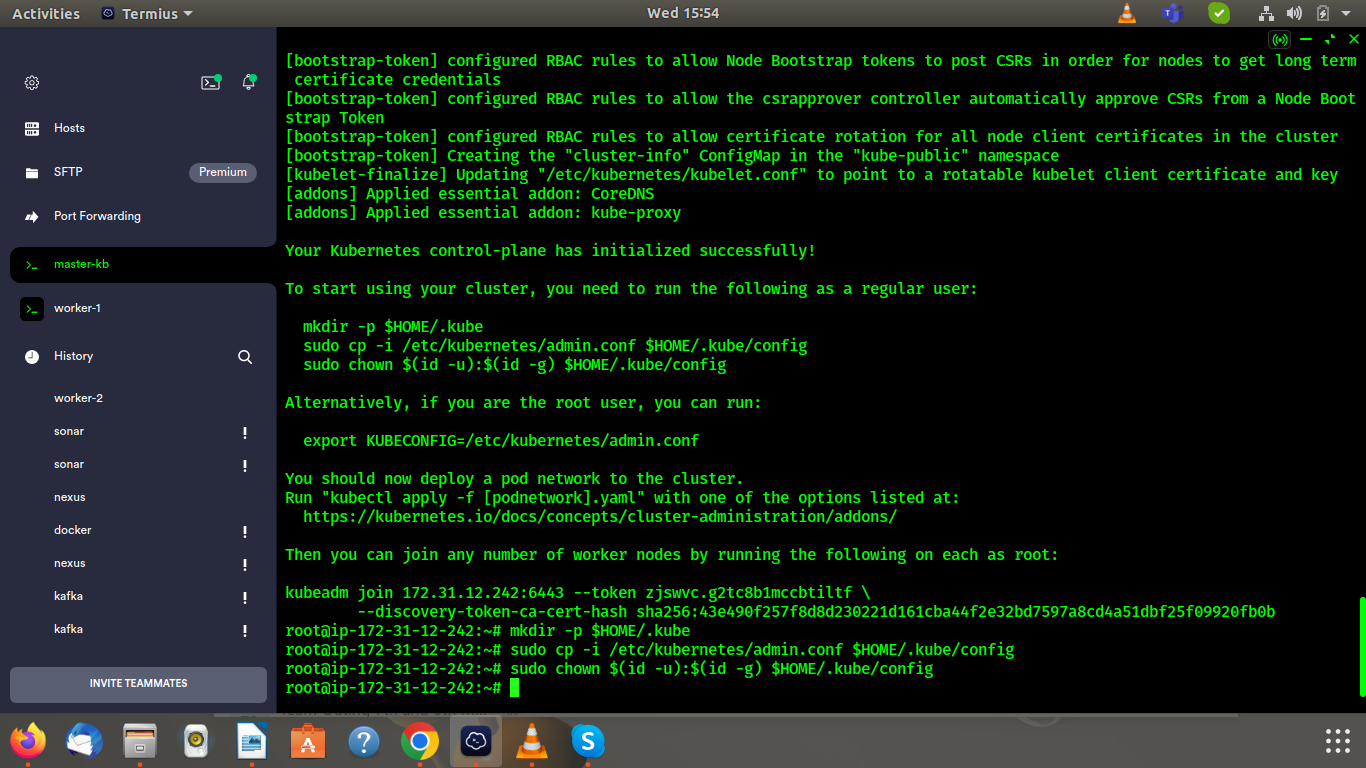
## 

## **B) If you are logged in as a regular user then run the following command to start using your cluster:**

mkdir -p $HOME/.kube

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

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



## **Step 7 – Deploy a Pod Network :**

## A) Run the following command on the Master node to deploy a Flannel pod network:

## # kubectl apply -f calico.yaml

## The yaml file url : https://projectcalico.docs.tigera.io/manifests/calico.yaml

## 

## B) run the following command to see the status of all pods:

## # kubectl get pods –all-namespaces

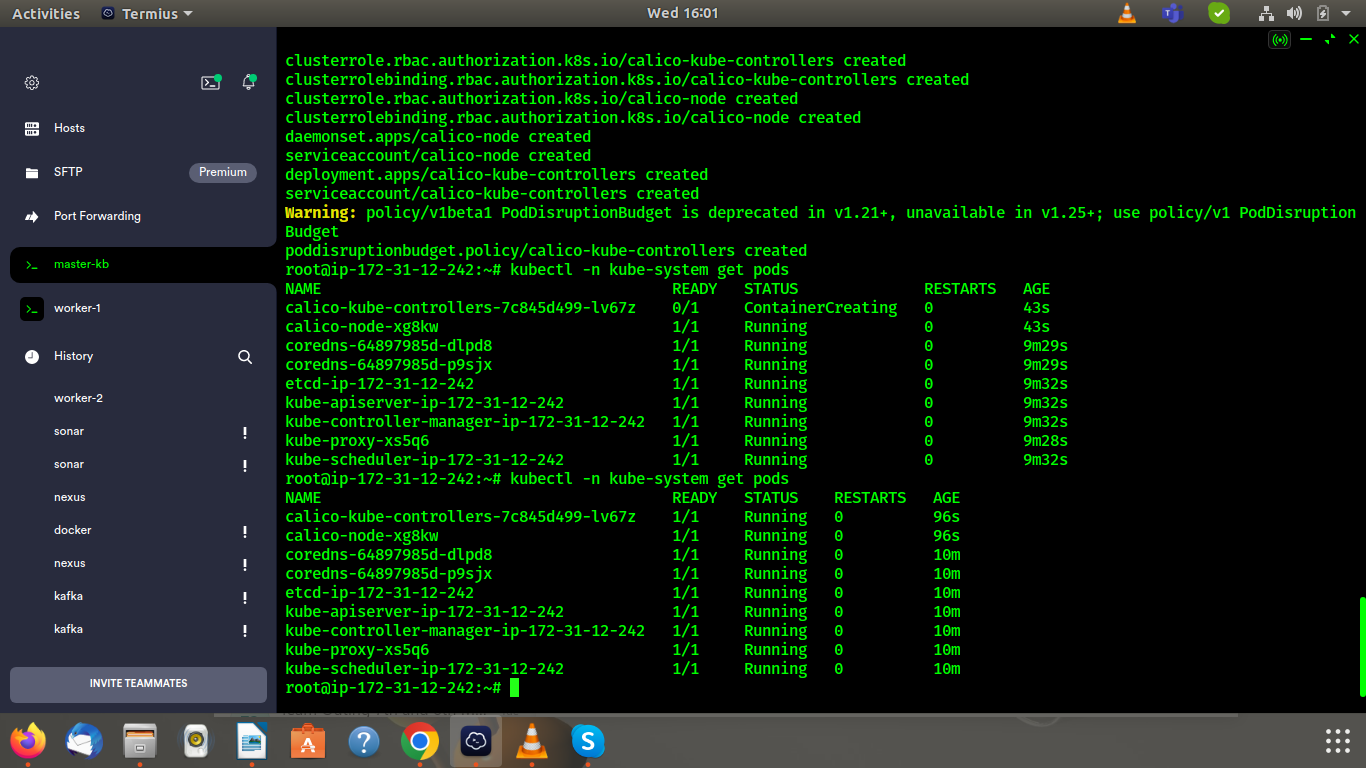
## **Step 8 – Join Worker Nodes to the Kubernetes Cluster :**

A) After the successful pod network initialization, the Kubernetes cluster is ready to join the worker nodes. In this section, we will show you how to add both worker nodes to the Kubernetes cluster.

You can use the kubeadm join command on each worker node to join them to the Kubernetes cluster.

## Use generated token when pod network initialized :

kubeadm join 69.28.88.236:6443 --token alfisa.guuc5t2f66cpqz8e --discovery-token-ca-cert-hash sha256:1db0bb5317ae1007c1f7774d5281d22b2189b239ffabecaedcd605613a9b10cd



## **B) go to the master node and run the following command to verify that both worker nodes have joined the cluster:**

# kubectl get nodes

## 

## **C) You can also get the cluster information using the following command:**

**#** kubectl cluster-info

## **Step 10 – Verify the Kubernetes Cluster :**

A ) After setting up the Kubernetes cluster, you can deploy any containerized application to your cluster. In this section, we will deploy an Nginx service on the cluster and see how it works.

To test the Kubernetes cluster, we will use the Nginx image and create a deployment called nginx-web:

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

## **B) Wait for some time, then run the following command to verify the status of deployment:**

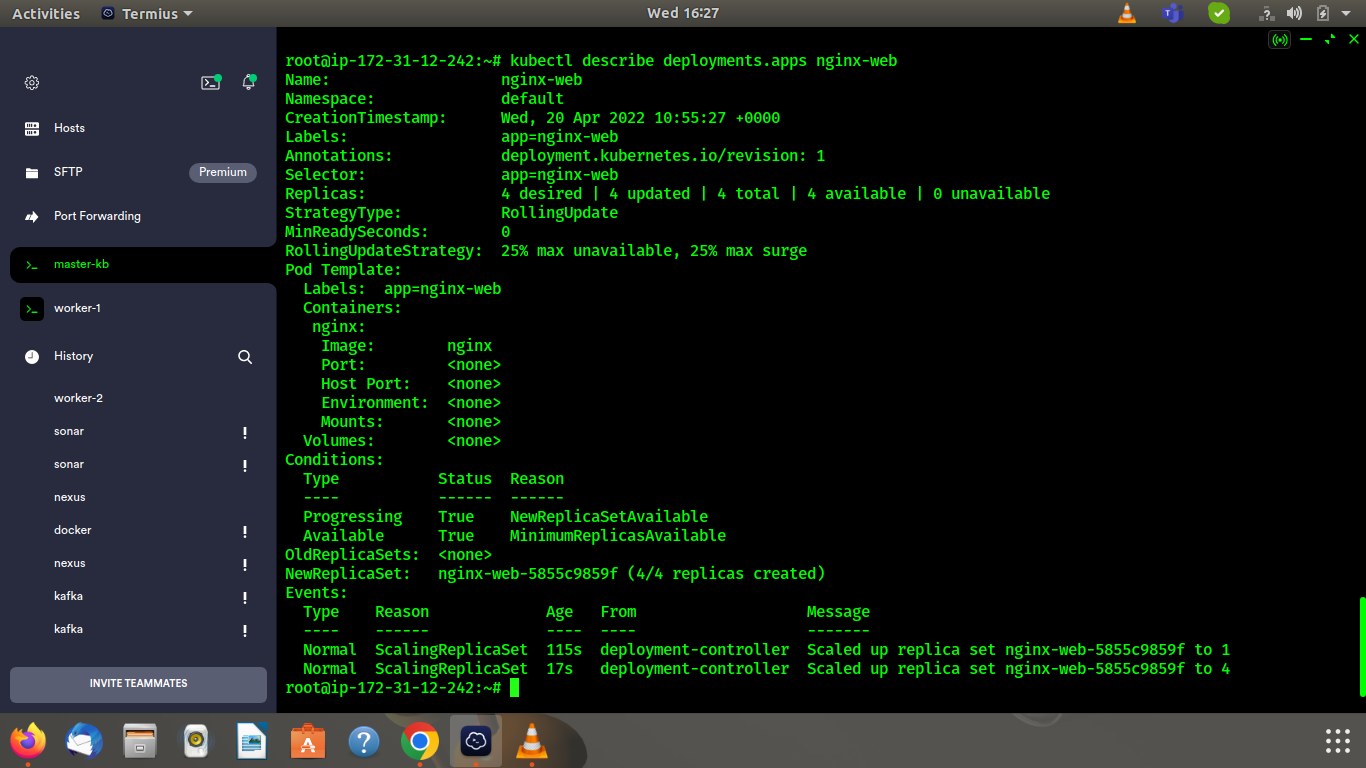
# kubectl get deployments.apps

**C) Next, scale the Nginx deployment with 4 replicas using the following command:**

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

**D) Wait for some time, then run the following command to verify the status of Nginx replicas:**

# kubectl get deployments.apps nginx-web



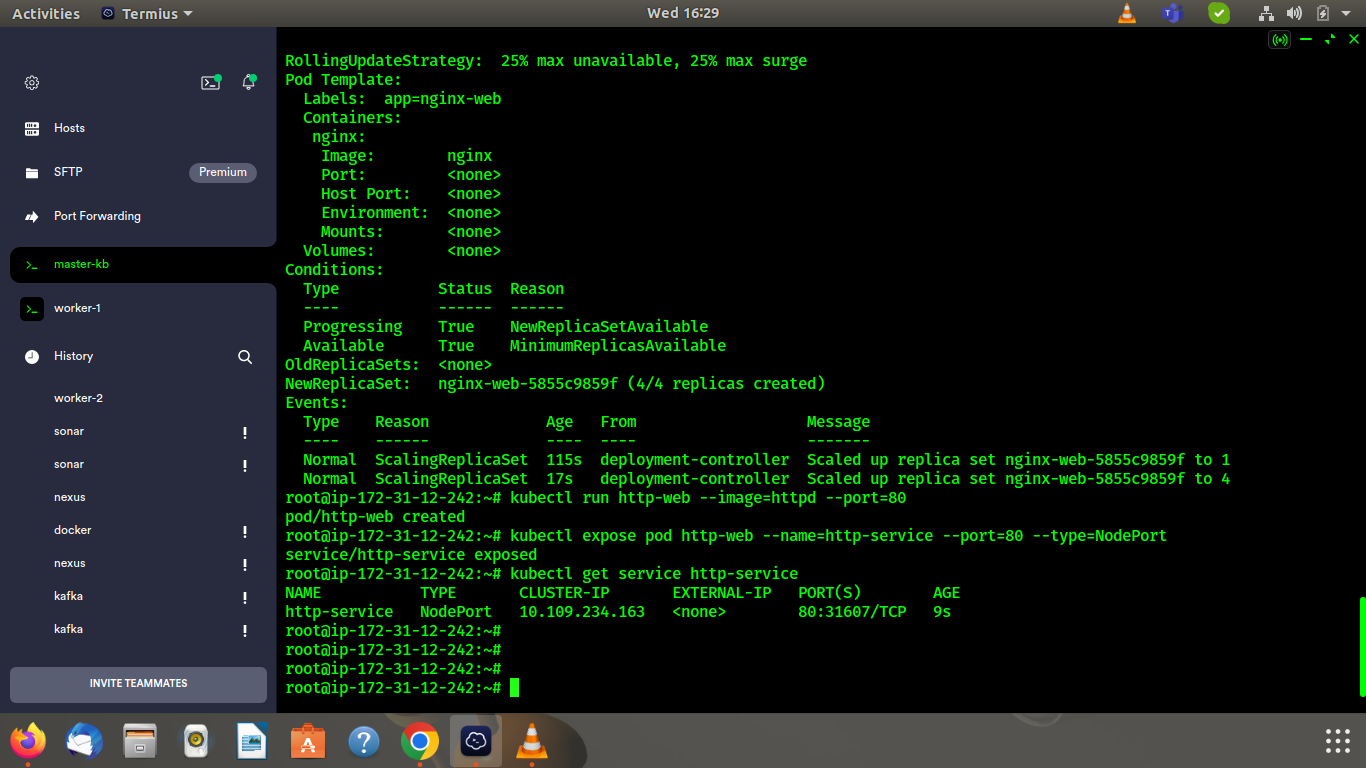
**E) To see the detailed information of your deployment, run:**

# kubectl describe deployments.apps nginx-web

**F) Next, run the following command to expose the above pod on port 80:**

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

# kubectl get service http-service

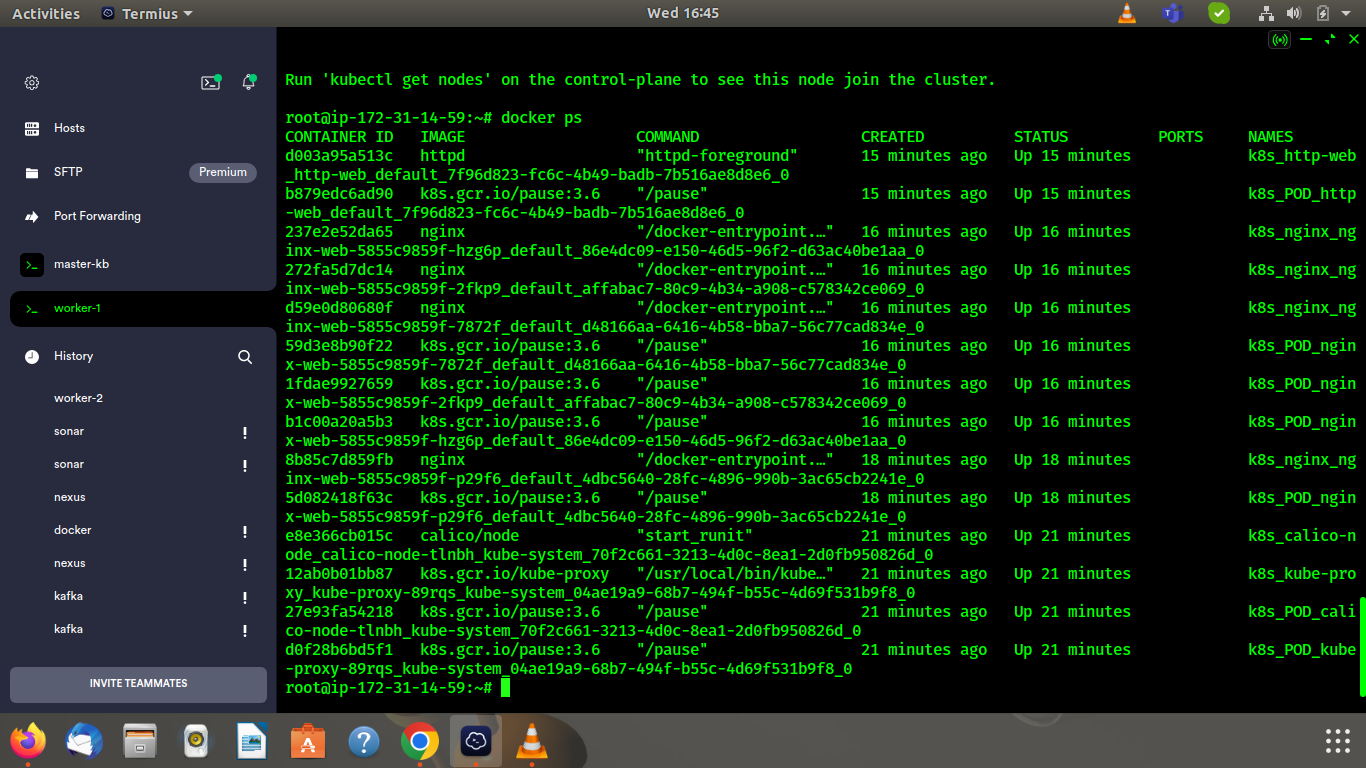


**G) You can now use the curl command to verify the webserver using port 80:**

curl [http://10.244.1.4:80](http://10.244.1.4/)

Output : <html><body><h1>It works!</h1></body></html>

## 

****

**Reference :**

**1)** <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-rbac.yml>

**2 )** https://www.atlantic.net/dedicated-server-hosting/how-to-set-up-three-node-kubernetes-cluster-on-ubuntu-20-04/