**Kubernetes**

**Kubernetes is a container orchestration tool**

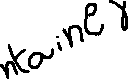
It is a platform for managing containerized application

Its responsibilities include container deployment, scaling and descaling of containers and container load balancing

Kubernetes is replacement for Docker Swarm

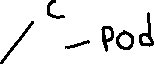


* Kubernetes started by google in 2014
* Base on Borg
* First release in July 2015
* Donated to cloud native computing foundation
* 100% open source



Kubernetes Architecture

In Kubernetes we have master and worker nodes



* **Master**

API Server

API server

Control manager

Scheduler

Etcd

Container run time

Master machine

Controller

kubeproxy

kubelet

Scheduler

Container runtime

worker

Etcd

kubectl

* **Workers**

**Kubelet**

**Kub proxy**

kubectl can be installed on remote server or master

kubectl uses .kube/config file to interact and deploy any object in Kubernetes cluster

**API server**

will persist(store) the information in ETCD

Etcd have all the information of cluster

**Scheduler**

Scheduler will schedule the unscheduled using kubelet

It will check unscheduled apps /pods from etcd

Controller

We have different types of controllers

Node controller

Replica controller etc.

Controller manager responsible for noticing and responding when node or pod or any control component goes down

Kubelet

Kubelet is a node agent which is present in Every node, it talks with container run time and make sure that containers are running and ensure container are healthy and manages the containers

Kub proxy

The end user response will be processed via kube proxy, kube proxy will forward the request to appropriate app(pod)

Pods

Pod is a smallest deployable Unit called pod

Small group of tightly coupled containers

Shared network and data volumes

Routable IP address

Mortal

**Static pods**

Static Pods are managed directly by the kubelet and the API server does not have control over these Pods

**Pod lifecycle**

Make a Pod request to API server using a local pod definition file

The API server saves the information for the pod in ETCD

The scheduler finds the unscheduled pods and schedules it to node

Kubelet running on the node sees the pod scheduled and fires up docker

Docker runs the container

The entire lifecycle of pod stored in ETCD

**Replicaset** Run x copies of a pod My app replicaset replicas = 3

My app Relicaset replicas = 2

Start or kill pods if necessary

My app pod

handle pod failures

My app pod

Health checks

**Kubernetes installation**

Take three instances one for master and two for worker nodes

Here I am taking 3 ubuntu servers

One for master and two for workers t2.medium instance type for master and t2.micro for workers

Open required Ports in AWS Security Groups

Here we have few step common for master and the workers

sudo apt update

sudo apt -y install curl apt-transport-https

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add - echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt update

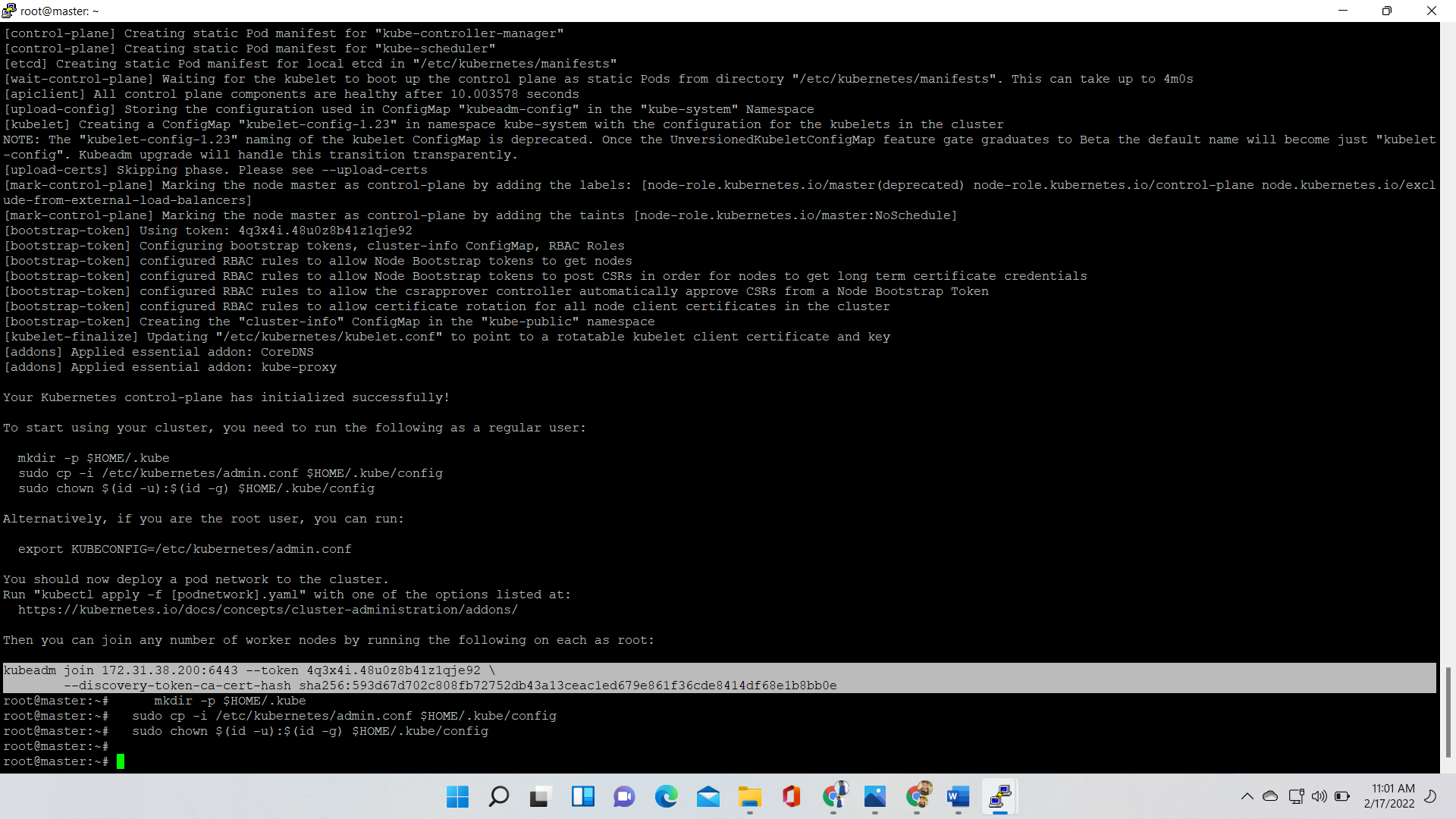
apt-get install -y kubelet kubeadm kubectl kubernetes-cni

|  |
| --- |
| cat > /etc/docker/daemon.json <<EOF |
|  | { |
|  | "exec-opts": ["native.cgroupdriver=systemd"], |
|  | "log-driver": "json-file", |
|  | "log-opts": { |
|  | "max-size": "100m" |
|  | }, |
|  | "storage-driver": "overlay2", |
|  | "storage-opts": [ |
|  | "overlay2.override\_kernel\_check=true" |
|  | ] |
|  | } |
|  | EOF |
|  |  |
|  | mkdir -p /etc/systemd/system/docker.service.d |
|  |  |
|  | systemctl daemon-reload |
|  | systemctl restart docker |

######In Master###########

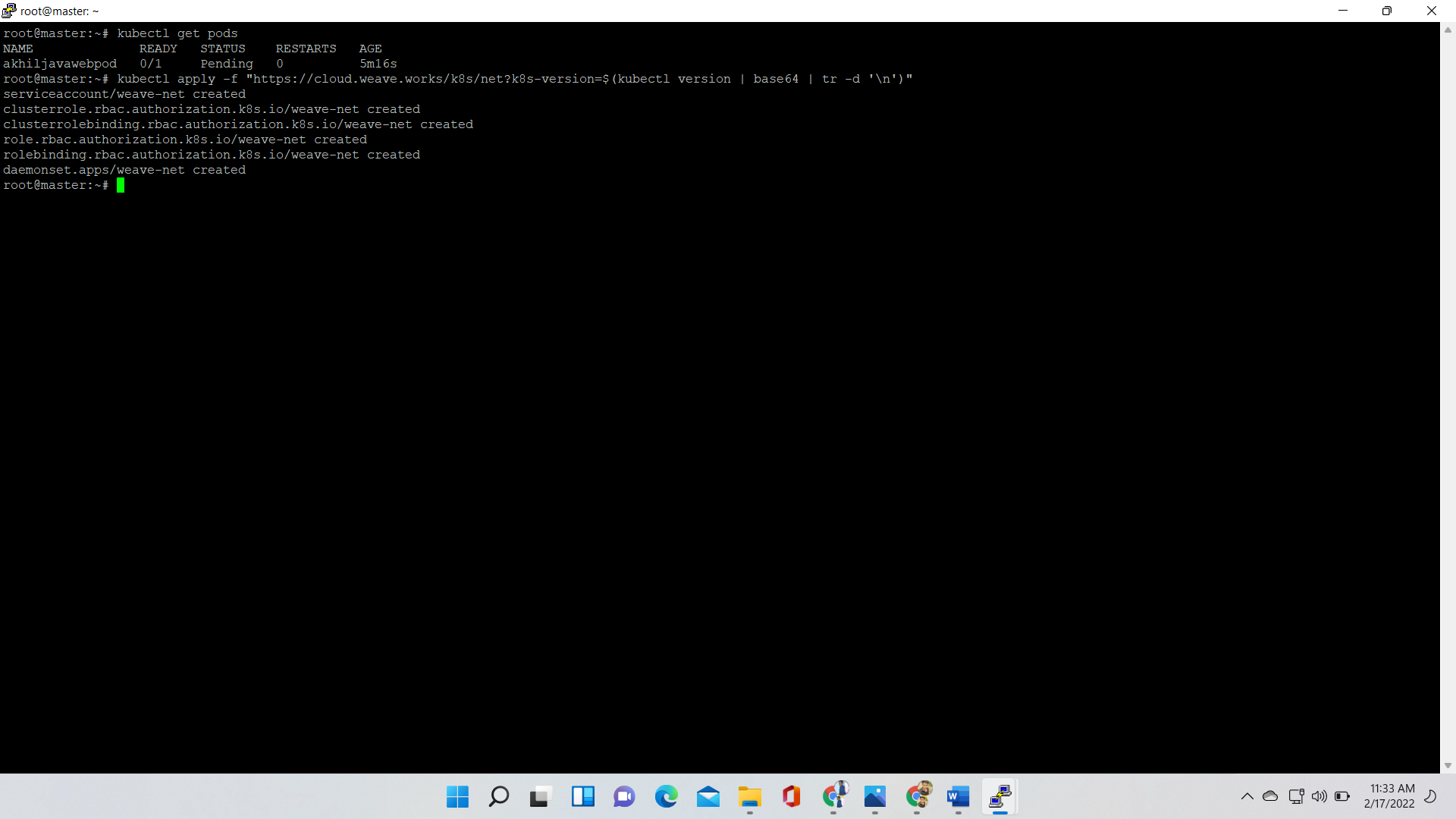
|  |
| --- |
|  |
| **kubedeaminit** |
|  |  |
|  | mkdir -p $HOME/.kube |
|  | sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config |
|  | sudo chown $(id -u):$(id -g) $HOME/.kube/config |
|  |  |

After completion off all those it will generate a token copy that token and paste it on workers

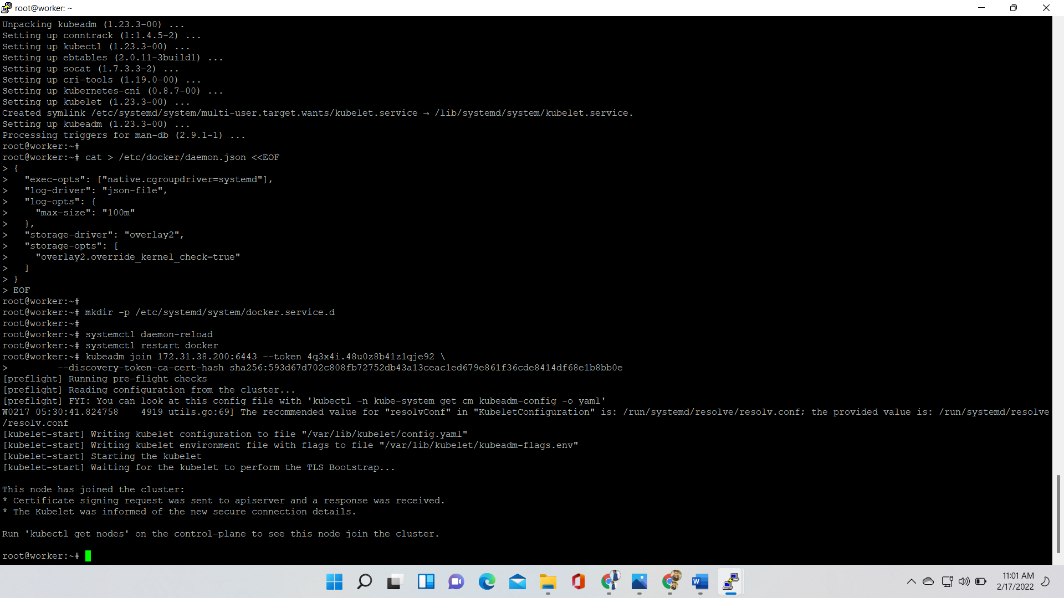


Create a network in your Kubernetes using command

kubectl apply -f [https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')](https://cloud.weave.works/k8s/net?k8s-version=$(kubectl%20version%20|%20base64%20|%20tr%20-d%20'\n'))



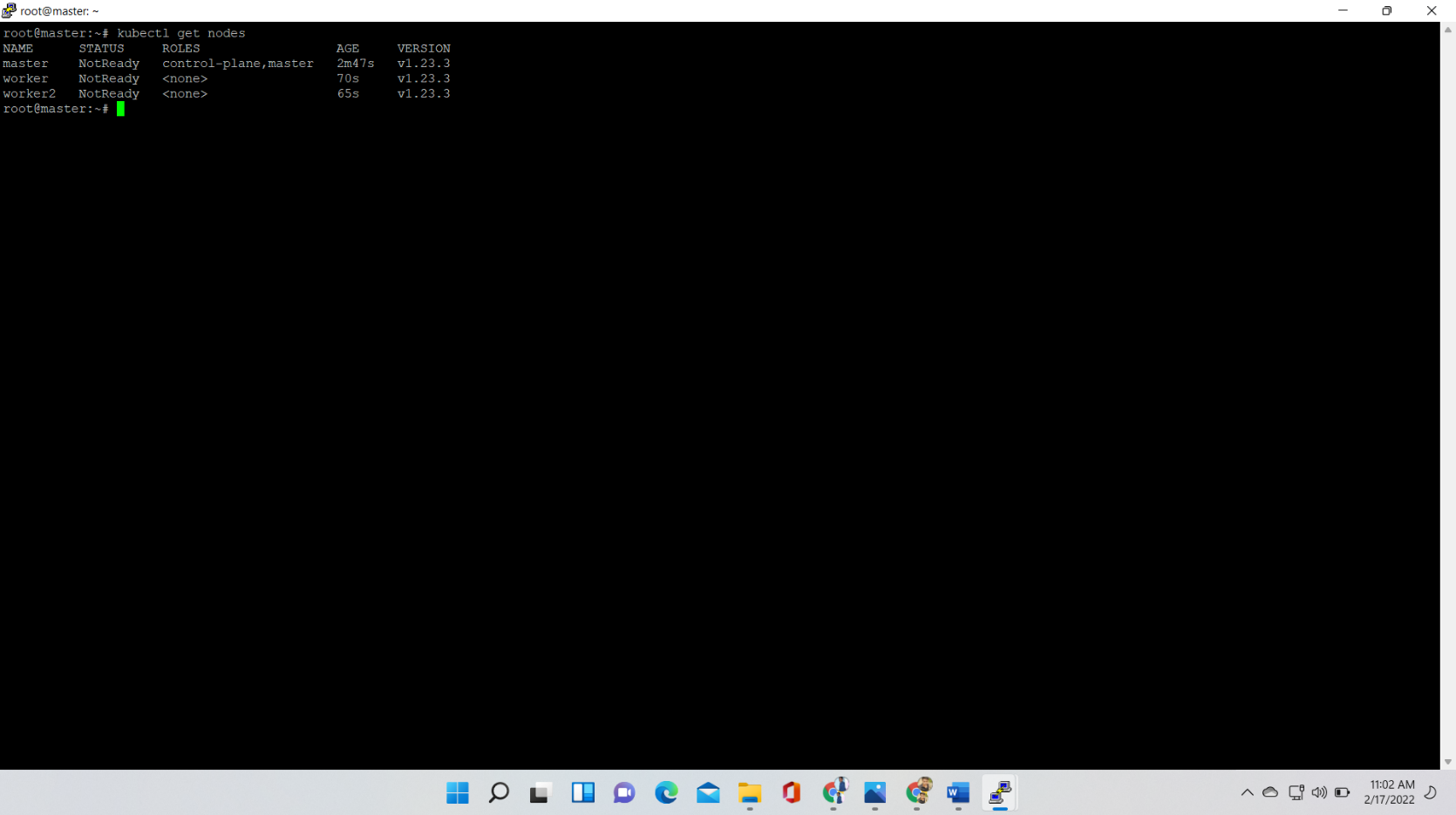
Copy the token from master and paster it on workers



Open your master machine in terminal and check nodes are ready are not

Using below command

Kubectl get nodes



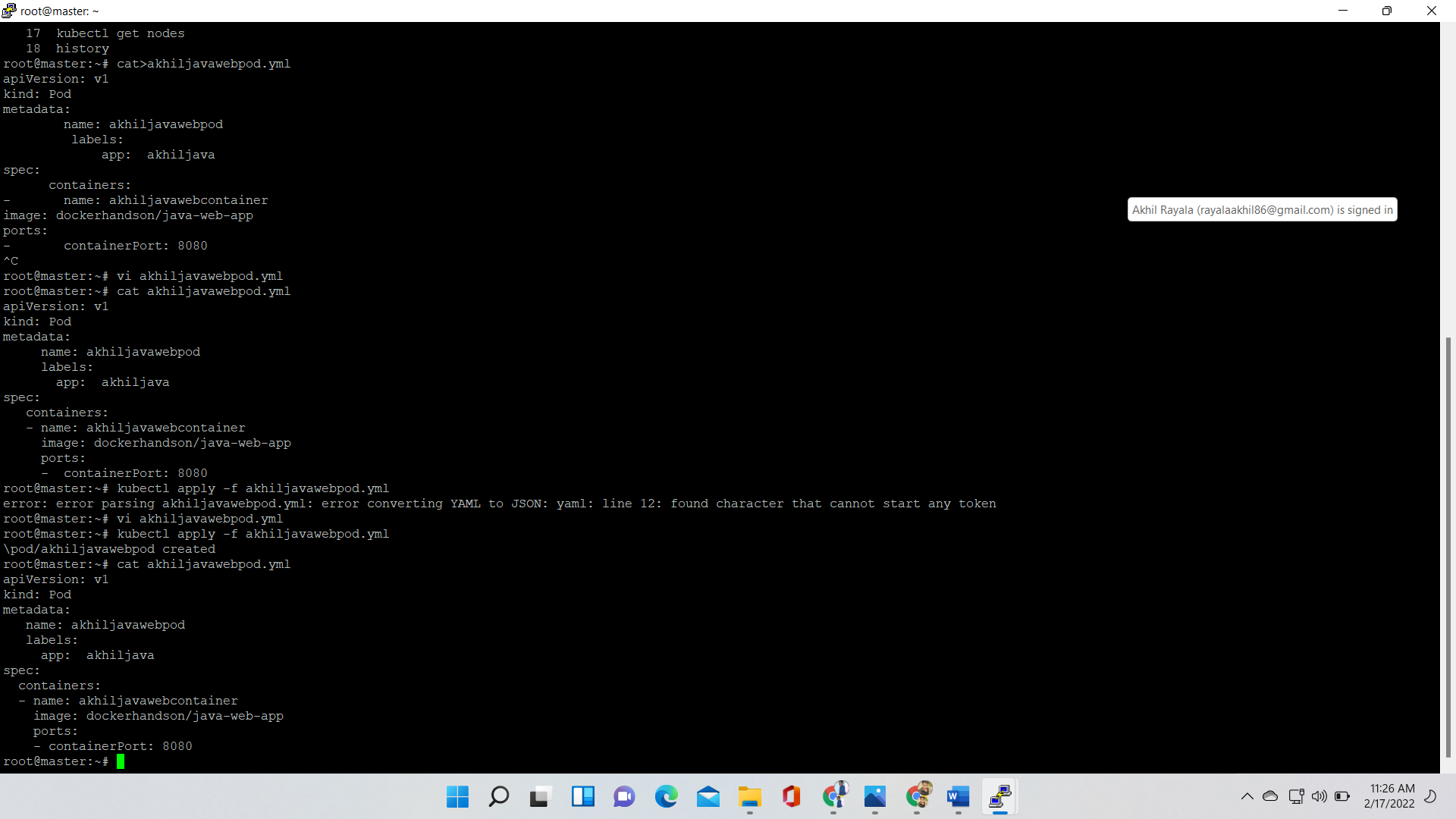
Here I am showing how to create a pod in Kubernetes

In master machine create one file

**Indentation is very Important**

Cat >akhiljavawebpod.yml

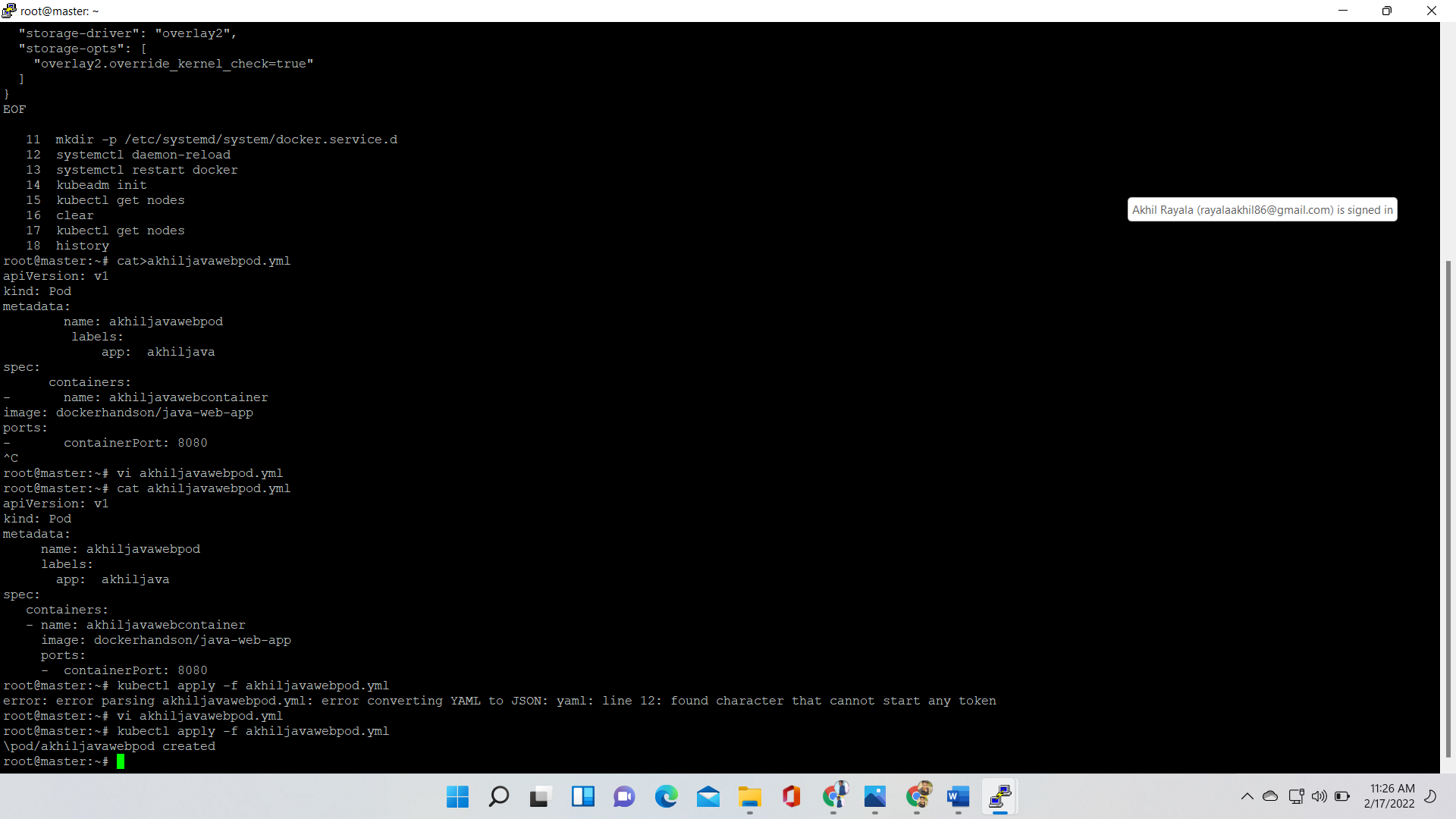
Here we have content of yaml file



Here I am creating the pod

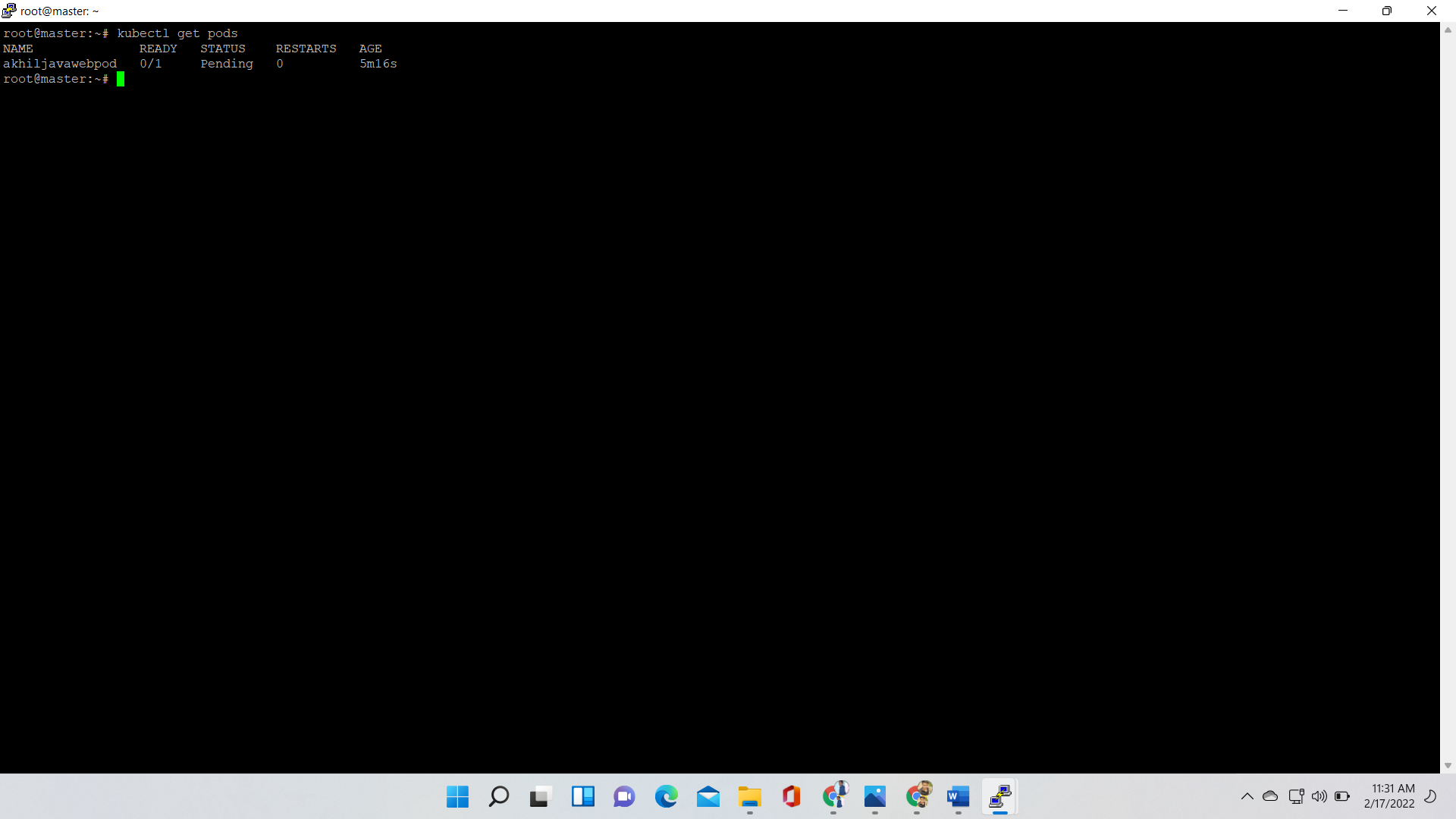
Cat>akhiljavawebpod.yml

Kubectl apply -r akhiljavawebpod.yml



Here check pod is created are not using command

Kubectl get Pods



**service**

In Kubernetes service is for communication

Service is responsible for making our Pods discoverable inside the network or exposing them to the internet. A service identifies pods by its label selector types

**services**

cluster IP

Node Port

Load Balancer

To communicate within the cluster cluster IP will be used

To communicate within in the cluster and out side of cluster we use Node Port and Load balancer

Here I am creating the Service

Cat>javawebappsvc.yml

apiVersion v1

kind: Service

metadata:

name: javawebappsvc

spec

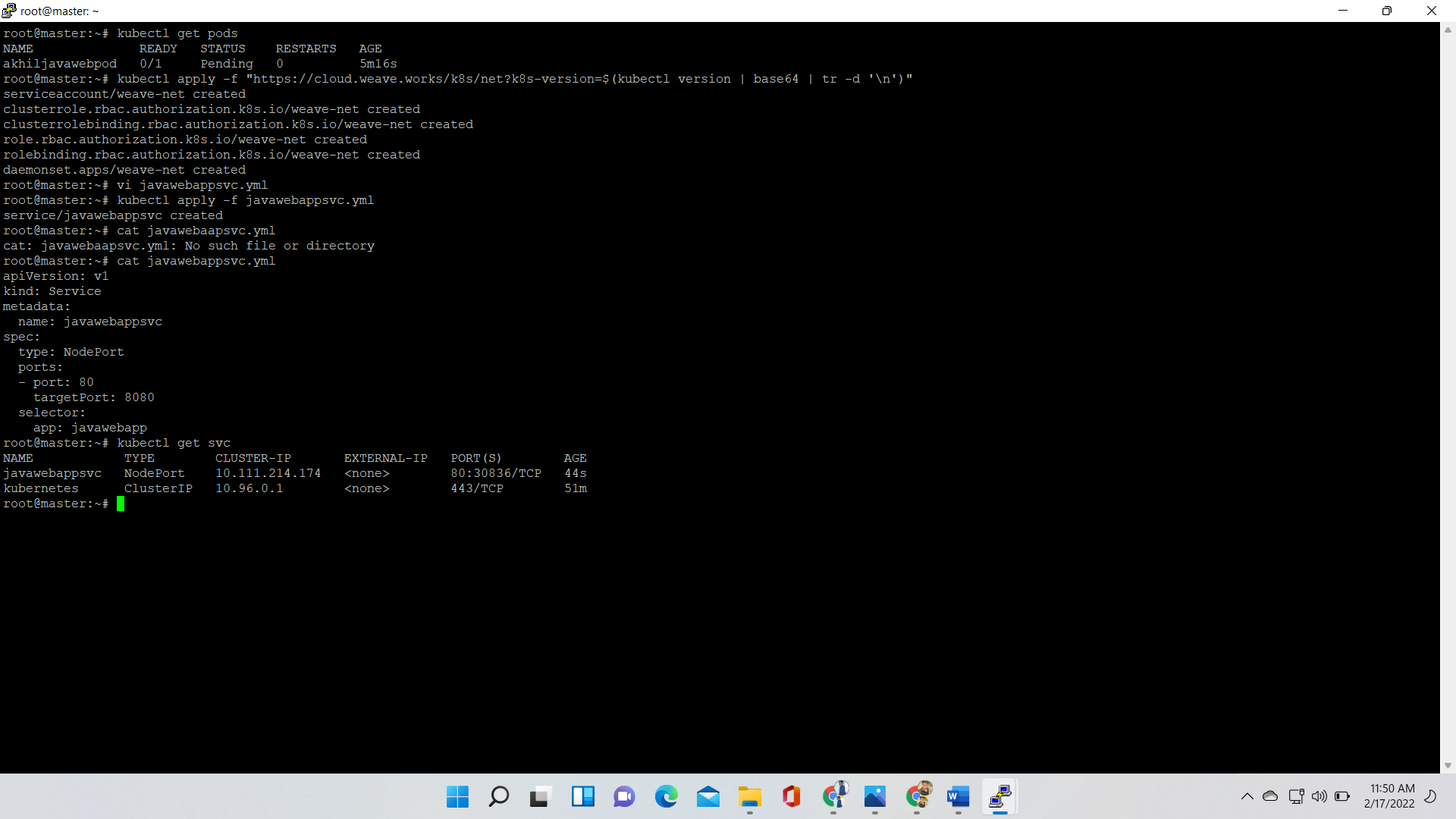
type: NodePort  
  
 Ports:

-port: 80

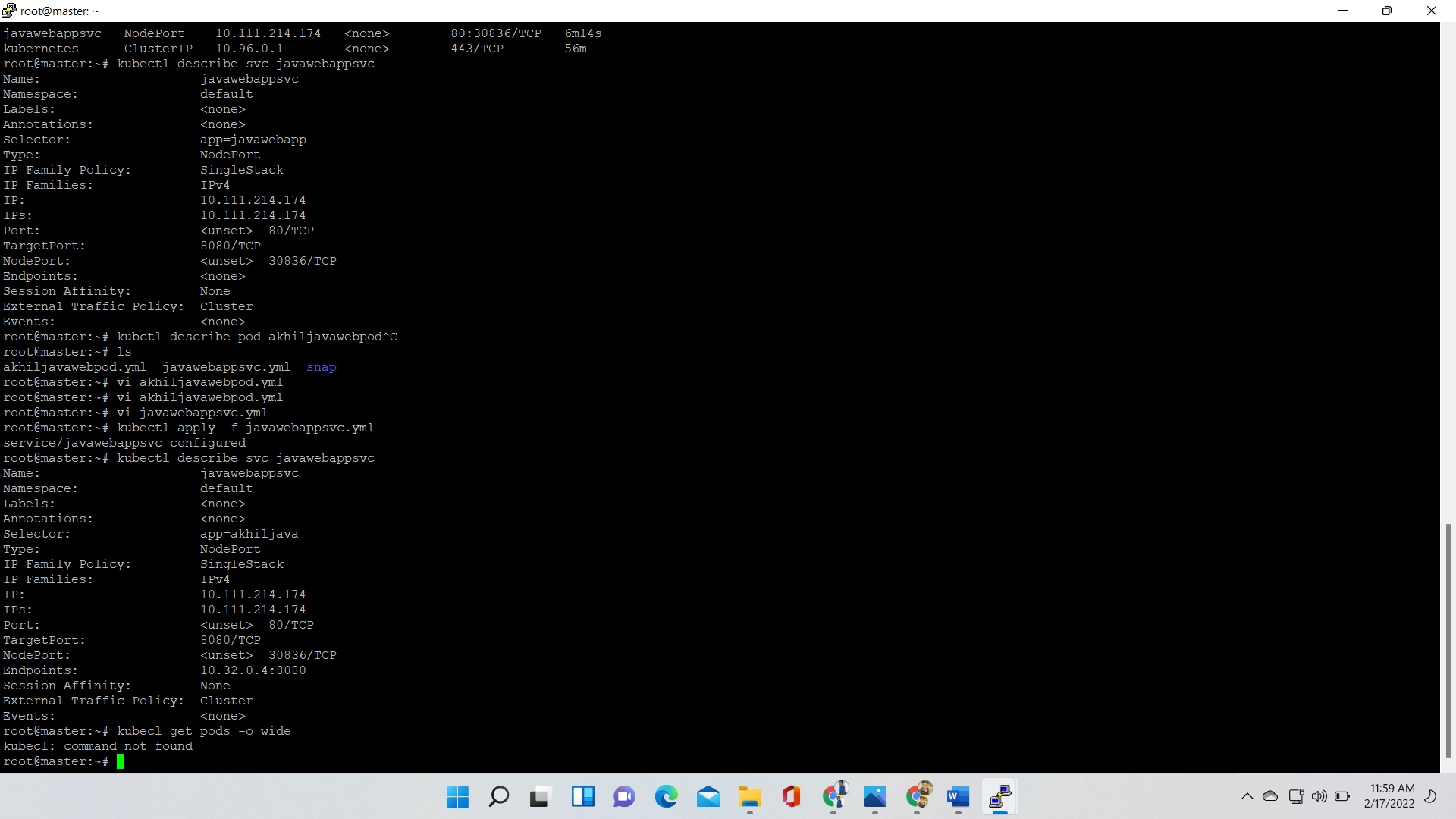
targetPort: 80

selector:

app: akhiljava



Here I am connected Node Port service to the Pod



Take instance Ip address and search in browser

