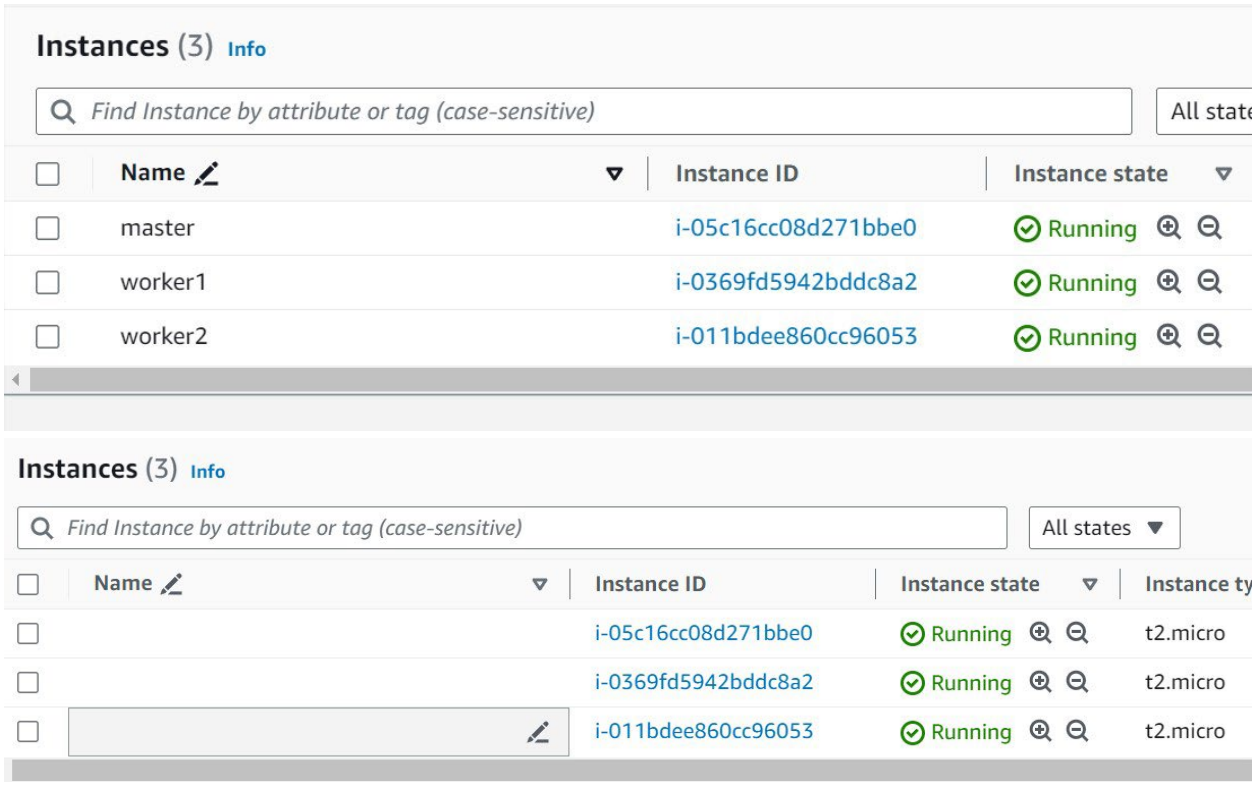
HOW TO INSTALL CLUSTERS ON LINUX OPERATING SYSTEM (WITH KUBEADM)

This lab will allow you to practice the process of creating a new Kubernetes cluster. You will be given a set of Linux machines and the ability to configure them into a working Kubernetes cluster. This will help you develop the skills required to build your own Kubernetes clusters in the real world.

STEP1

NOTES(Lets say we want to build a cluster with one master and two worker nodes on a Linux server without the help of amazon EKS and cloud formation)

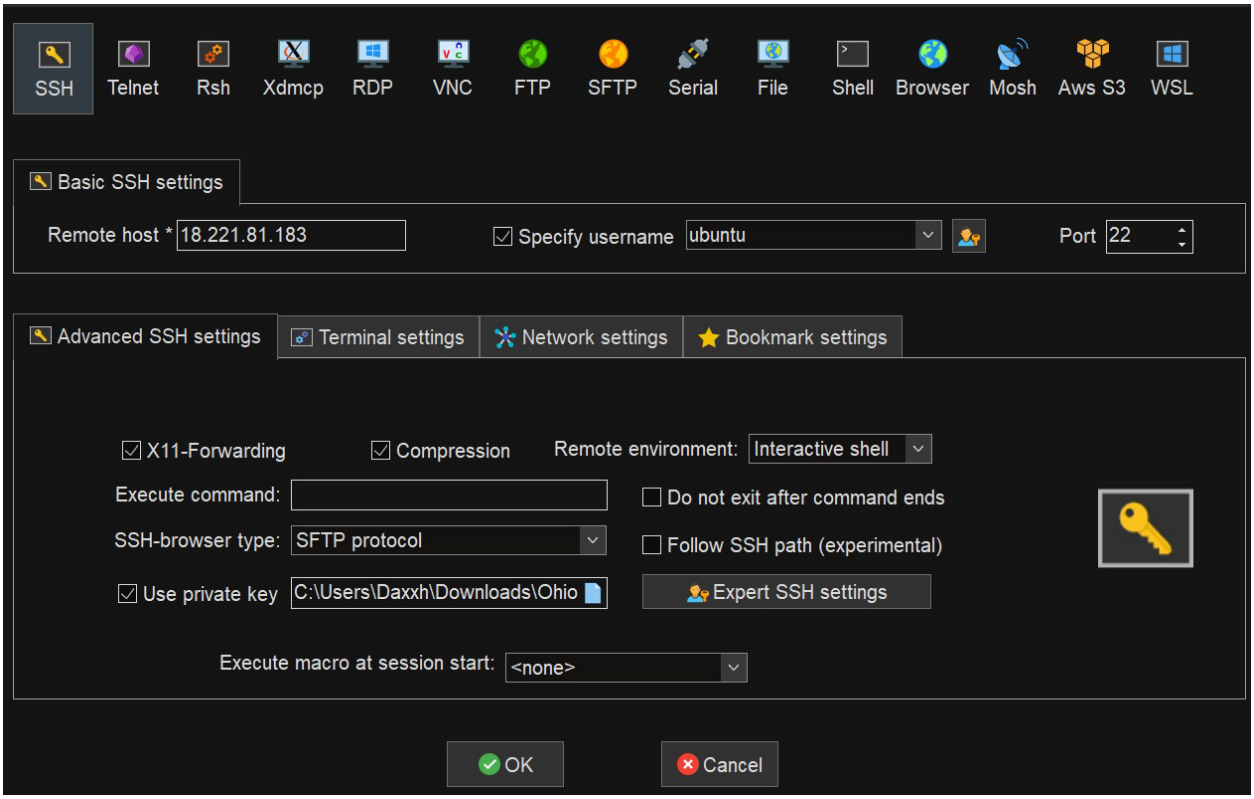
- Lets go to our aws console and create our EC2 instance(Number of Instances -3 , Machin Image – Ubuntu ,Instance Type – T2 Micro) also create a keypair since you would need it to SSH.



After creating your instance name them as I have done above , In the real world these 3 instances will be 3 physical machines at an on prem location where one would be acting as a master and two others as worker nodes, I don't have Physical machines that's why I borrowed servers from amazon for this project .

STEP 2

Now that we have our instances we need to connect with them securely we can do this with https://mobaxterm.mobatek.net/download.html Download mobaxterm since we are going to use it to ssh our Instance.

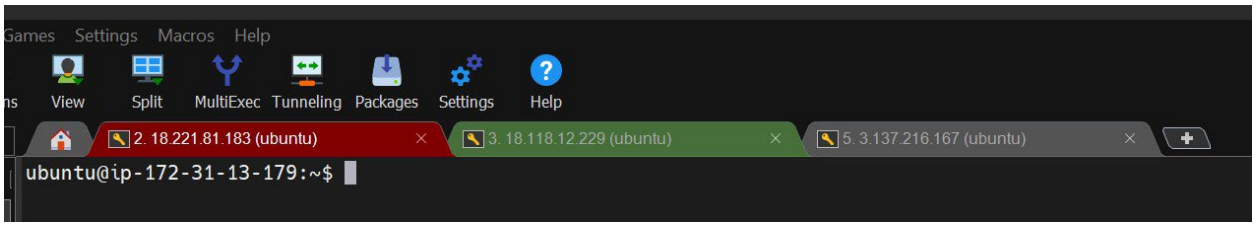
- Copy The Public Ip address of your master Instance
- Go your Mobaxterm Choose new Session and set the configuration as I have done below
- 

Remote Host – Public IP of your master EC2 Instance

Specify username – Ubuntu

With Advanced SSH setting this is where you use your key pair to connect your instance

Repeat this for all , you can distinguish between master and worker nodes by setting the color of your tabs so you don't get confused when doing your configurations .



Master(control plane) – Red Tab Color

Worker Node 1- Green Tab Color

Worker Node 2- White Tab Color

STEP 3

Lets Install all our packages .

- Follow the commands Below

```
INSTALL PACKAGES ON EACH OF THE NODES


Create the configuration file for containerd:

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

overlay

br_netfilter

EOF
```

Load the modules:

```
sudo modprobe overlay

sudo modprobe br_netfilter
```

Set the system configurations for Kubernetes networking:

```
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf

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

net.ipv4.ip_forward = 1

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

EOF
```

Apply the new settings:

```
sudo sysctl --system
```

Install containerd:

```
sudo apt update

sudo apt install -y docker.io
```

Create the default configuration file for containerd:

```
sudo mkdir -p /etc/containerd
```

Generate the default containerd configuration, and save it to the newly created default file:

```
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

Restart containerd to ensure the new configuration file is used:

```
sudo systemctl restart containerd
```

Verify that containerd is running:

```
sudo systemctl status containerd
```

Disable swap:

```
sudo swapoff -a
```

Install the dependency packages:

```
sudo apt-get update && sudo apt-get install -y apt-transport-https curl
```

Download and add the GPG key:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.27/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

Add Kubernetes to the repository list:

```
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.27/deb/ /

EOF
```

Update the package listings:

sudo apt-get update

Install Kubernetes packages:

Note: If you get a dpkg lock message, just wait a minute or two before trying the command again.

sudo apt-get install -y kubelet kubeadm kubectl

Turn off automatic updates:

sudo apt-mark hold kubelet kubeadm kubectl

STEP 4

INTIALIZE THE CLUSTER

Open Port 6443 on your control plane server

- On Your Ec2 Instance click your master Ec2
- Scroll down to security
- Click on security and click the security groups link provided
- Edit Inbound Rules
- Add rule
- Add custom TCP 6443 anywhere on the internet

After let's continue with configurations

```
sudo kubeadm init --pod-network-cidr 192.168.0.0/16 --kubernetes-version 1.27.11
```

Set kubectl access:

```
mkdir -p $HOME/.kube

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

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

Test access to the cluster:

```
kubectl get nodes
```

Install the Calico Network Add-On

On the control plane node, install Calico Networking:

```
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml
```

Check the status of the control plane node:

```
kubectl get nodes
```

Join the Worker Nodes to the Cluster

In the control plane node, create the token and copy the kubeadm join command:

```
kubeadm token create --print-join-command
```

Note: This output will be used as the next command for the worker nodes.

Copy the full output from the previous command used in the control plane node. This command starts with kubeadm join.

In both worker nodes, paste the full kubeadm join command to join the cluster. Use sudo to run it as root:

sudo kubeadm join...

In the control plane node, view the cluster status:

kubectl get nodes

Note: You may have to wait a few moments to allow all nodes to become ready.