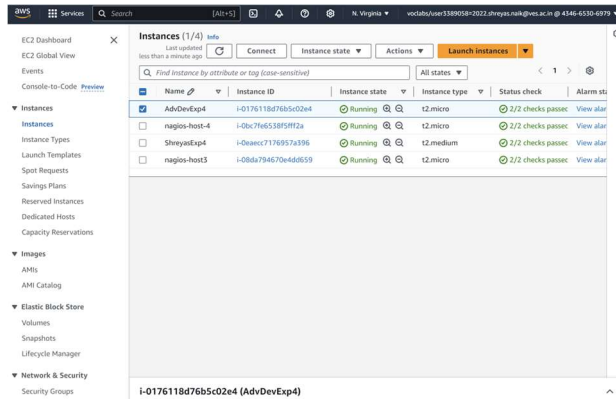
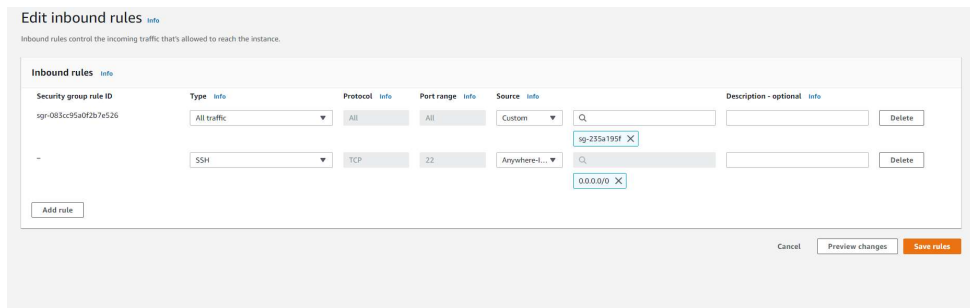


## Steps:

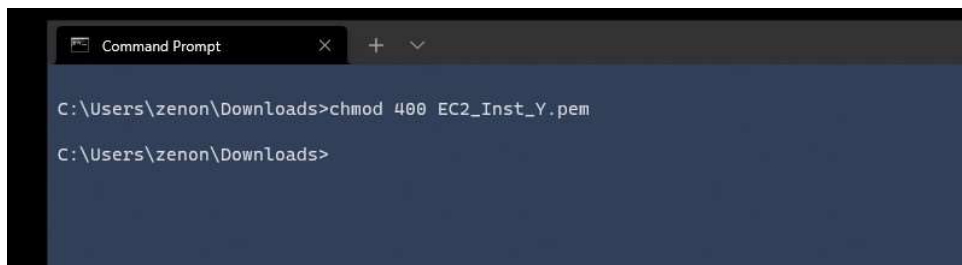
1. Create an EC2 Ubuntu Instance on AWS.



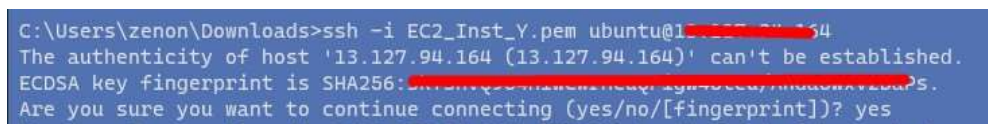
2. Edit the Security Group Inbound Rules to allow SSH



3. SSH into the machine

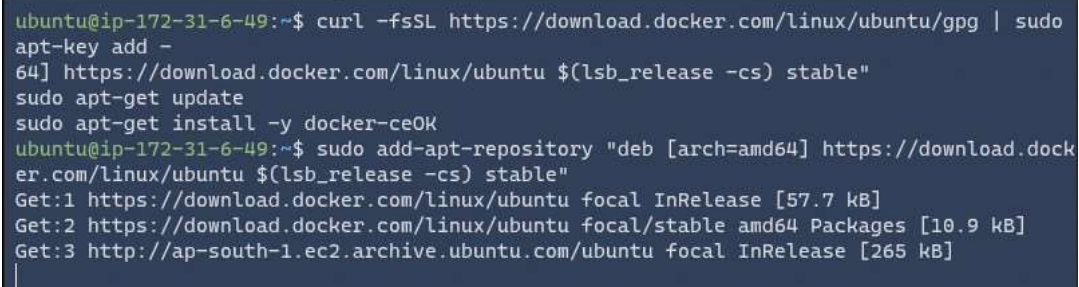


**ssh -i <keyname>.pem ubuntu@<public\_ip\_address>**



#### 4. Install Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key  
add -  
sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"  
sudo apt-get update  
sudo apt-get install -y docker-ce
```



```
ubuntu@ip-172-31-6-49:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo  
apt-key add -  
64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"  
sudo apt-get update  
sudo apt-get install -y docker-ceOK  
ubuntu@ip-172-31-6-49:~$ sudo add-apt-repository "deb [arch=amd64] https://download.dock  
er.com/linux/ubuntu $(lsb_release -cs) stable"  
Get:1 https://download.docker.com/linux/ubuntu focal InRelease [57.7 kB]  
Get:2 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages [10.9 kB]  
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease [265 kB]  
|
```

Then, configure cgroup in a daemon.json file.

```
cd /etc/docker  
cat <<EOF | sudo tee /etc/docker/daemon.json  
{  
  "exec-opts": ["native.cgroupdriver=systemd"]  
}  
EOF  
sudo systemctl enable docker  
sudo systemctl daemon-reload  
sudo systemctl restart docker
```

#### 5. Install Kubernetes

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

EOF

```
sudo apt-get update
```

```
sudo apt-get install -y kubelet kubeadm kubectl
```

```
ubuntu@ip-172-31-6-49:~$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
  sudo apt-key add -
s.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectlOK
ubuntu@ip-172-31-6-49:~$ cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
> deb https://apt.kubernetes.io/ kubernetes-xenial main
> EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
ubuntu@ip-172-31-6-49:~$ sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu focal InRelease
0% [Waiting for headers] [Connecting to security.ubuntu.com (91.189.91.39)] [Waiting fo
```

After installing Kubernetes, we need to configure internet options to allow bridging.

```
sudo swapoff -a
```

```
echo "net.bridge.bridge-nf-call-iptables=1" | sudo tee -a
/etc/sysctl.conf
```

```
sudo sysctl -p
```

## 6. Initialize the Kubecluster

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

```
ubuntu@ip-172-31-6-49:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:  
<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join 172.31.6.49:6443 --token 2dtk88-1m7gn4stx2v509ui \
--discovery-token-ca-cert-hash sha256:ab3fb9f05059f2f4829be48caa1830254e6c49218c9aeffc4
```

Copy the mkdir and chown commands from the top and execute them

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

Then, add a common networking plugin called flannel as mentioned in the code.

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k
ube-flannel.yml
```

```
ubuntu@ip-172-31-4-0:/etc/docker$ kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

7. Now that the cluster is up and running, we can deploy our nginx server on this cluster.

Apply this deployment file using this command to create a deployment

```
kubectl apply -f https://k8s.io/examples/application/deployment.yaml
ubuntu@ip-172-31-4-0:/etc/docker$ kubectl apply -f https://k8s.io/examples/application/deployment.yaml
deployment.apps/nginx-deployment created
```

Use 'kubectl get pods' to verify if the deployment was properly created and the pod is working correctly.

Next up, create a name alias for this pod.

```
POD_NAME=$(kubectl get pods -l app=nginx -o
jsonpath="{.items[0].metadata.name}")
```

8. Lastly, port forward the deployment to your localhost so that you can view it.

```
kubectl port-forward $POD_NAME 8080:80
```

9. Verify your deployment

Open up a new terminal and ssh to your EC2 instance.

Then, use this curl command to check if the Nginx server is running.

```
curl --head http://127.0.0.1:8080
```

```
ubuntu@ip-172-31-4-0:~$ curl --head http://127.0.0.1:8080
HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sat, 02 Oct 2021 16:07:48 GMT
Content-Type: text/html
Content-Length: 612
Last-Modified: Tue, 04 Dec 2018 14:44:49 GMT
Connection: keep-alive
ETag: "5c0692e1-264"
Accept-Ranges: bytes
```

If the response is 200 OK and you can see the Nginx server name, your deployment was successful.

We have successfully deployed our Nginx server on our EC2 instance.