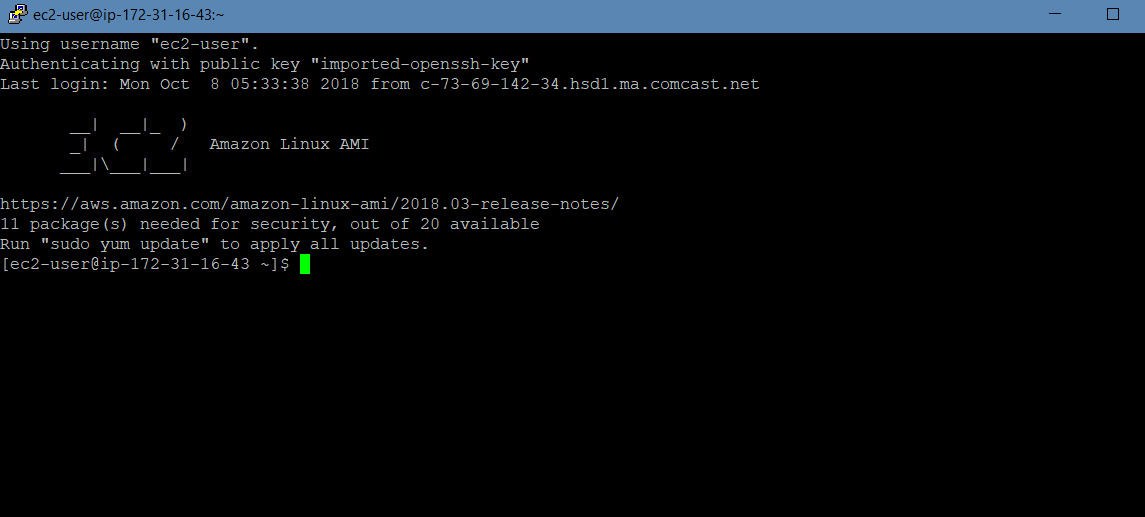
**Installing Kubernetes on AWS EC2**

**Prerequisites:**

– Ubuntu instance (You may use other linux instance as well)  
– AWS-cli setup  
– S3 bucket



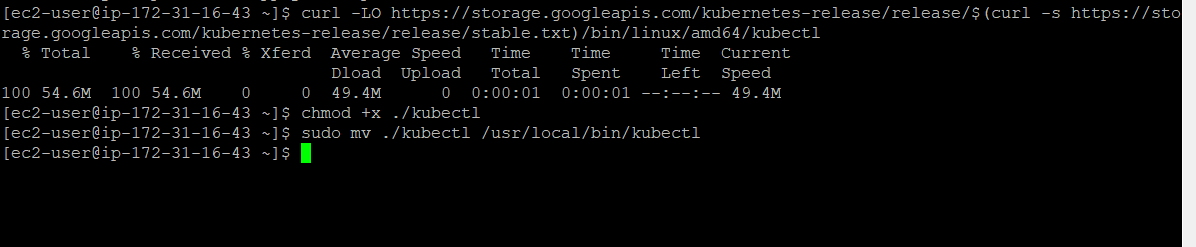
**Install kubectl**

On ubuntu instance, make sure you have AWS cli and KOPS setup. We shall also need **kubectl** (Kubernetes cli)  
– Install Kubectl on Linux:

curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/kubectl

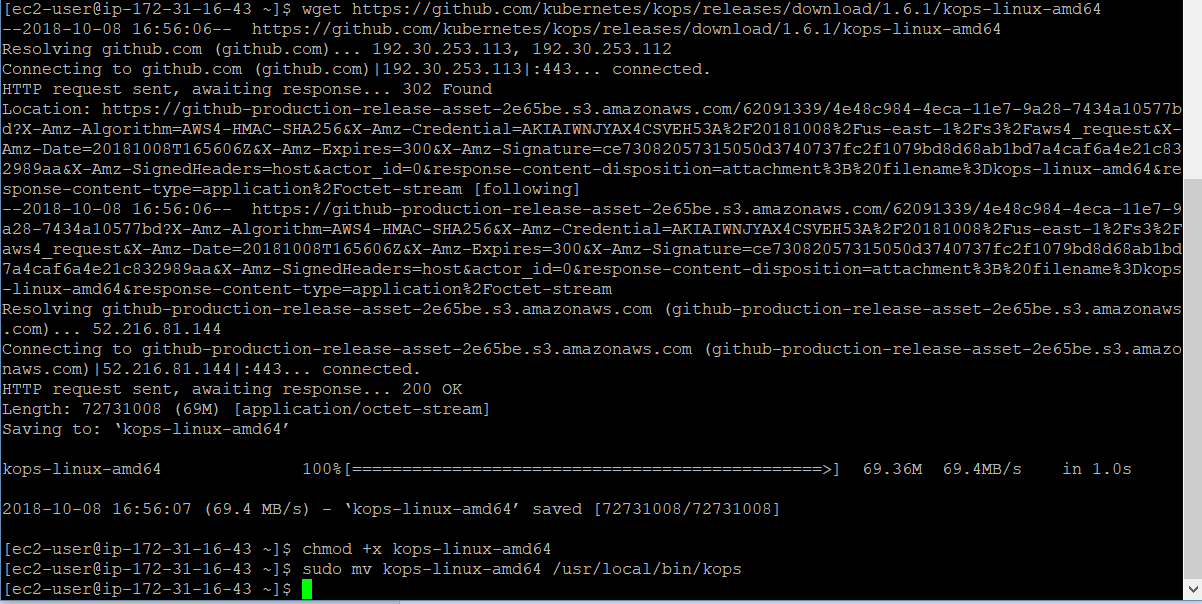


Now, let’s install kops on ubuntu box:

wget <https://github.com/kubernetes/kops/releases/download/1.6.1/kops-linux-amd64>

chmod +x kops-linux-amd64

sudo mv kops-linux-amd64 /usr/local/bin/kops



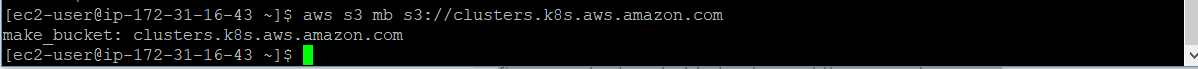
**Create Route53 domain for the cluster**

kubernetes make use of DNS for discovery within the cluster so that you can reach out kubernetes-API-server from clients.  
Create a hosted zone on Route53, say, **k8s.appychip.vpc**. The API server endpoint will then be **api.k8s.appychip.vpc**

**Create a S3 bucket**

Now, create a S3 bucket to store the configuration for the cluster. Make sure the instance have right role to access S3 and Route53:

$ aws s3 mb s3://clusters.k8s.appychip.vpc



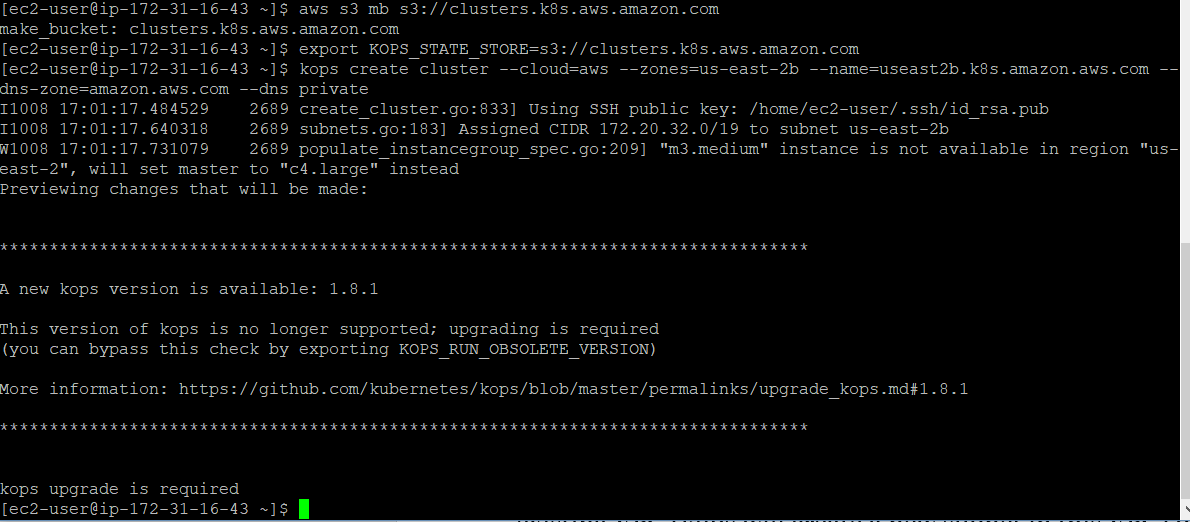
Expose environment variable:

$ export KOPS\_STATE\_STORE=s3://clusters.k8s.appychip.vpc

**Create Kubernetes Cluster**

Now comes the interesting part to create the cluster. You can reuse existing VPC (kops will create a new subnet in this VPC) by providing the **vpc-id**option. The following command will give you details what all things are going to happen:

$ kops create cluster --cloud=aws --zones=us-east-1d --name=useast1.k8s.appychip.vpc --dns-zone=appychip.vpc --dns private

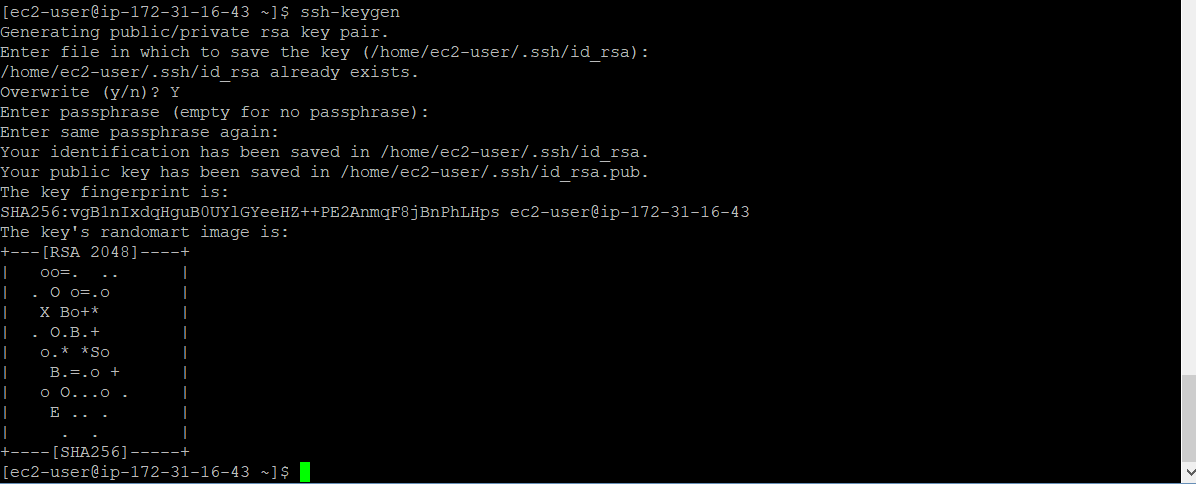


UPGRADE KOPS

wget -O kops <https://github.com/kubernetes/kops/releases/download/$(curl> -s <https://api.github.com/repos/kubernetes/kops/releases/latest> | grep tag\_name | cut -d '"' -f 4)/kops-linux-amd64

**NOTE:** Make sure you have ssh keys already generated otherwise it will throw an error.

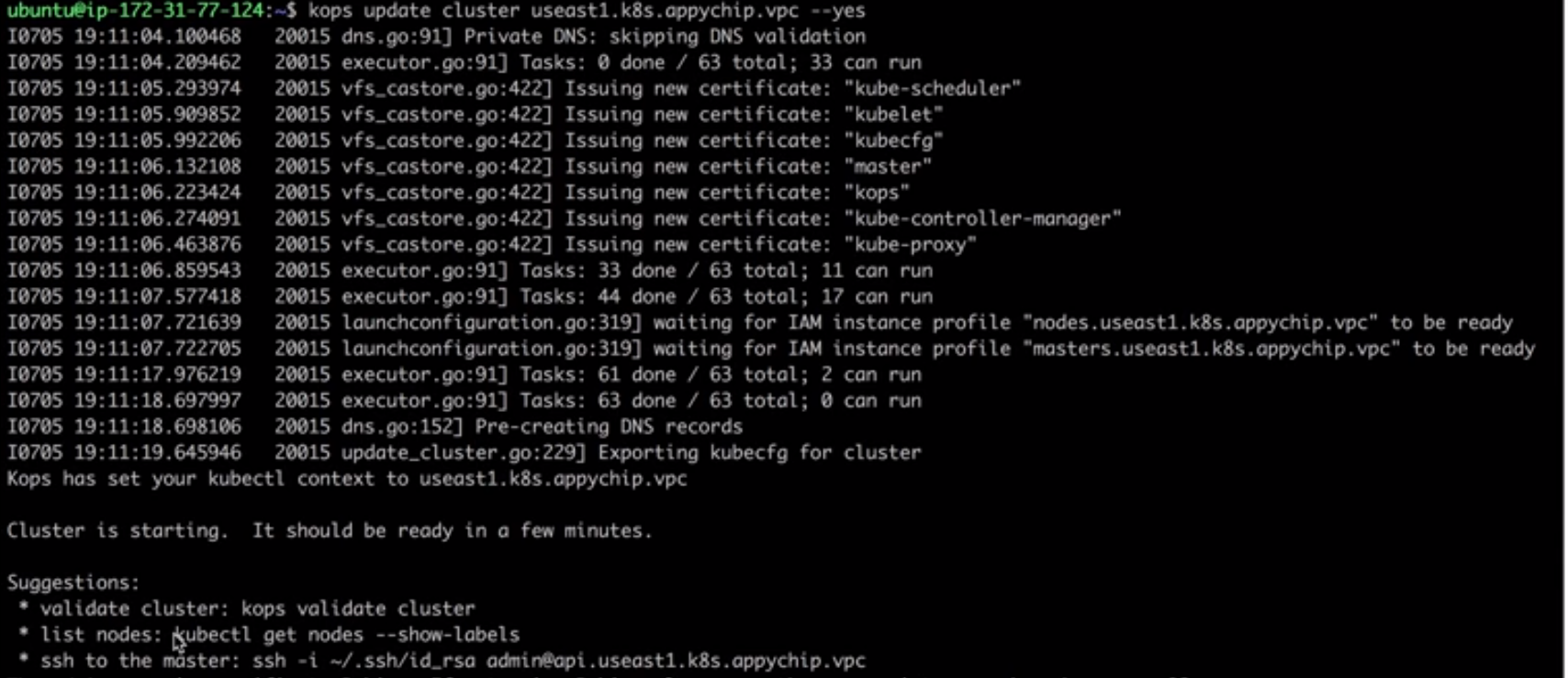
Generating SSH key

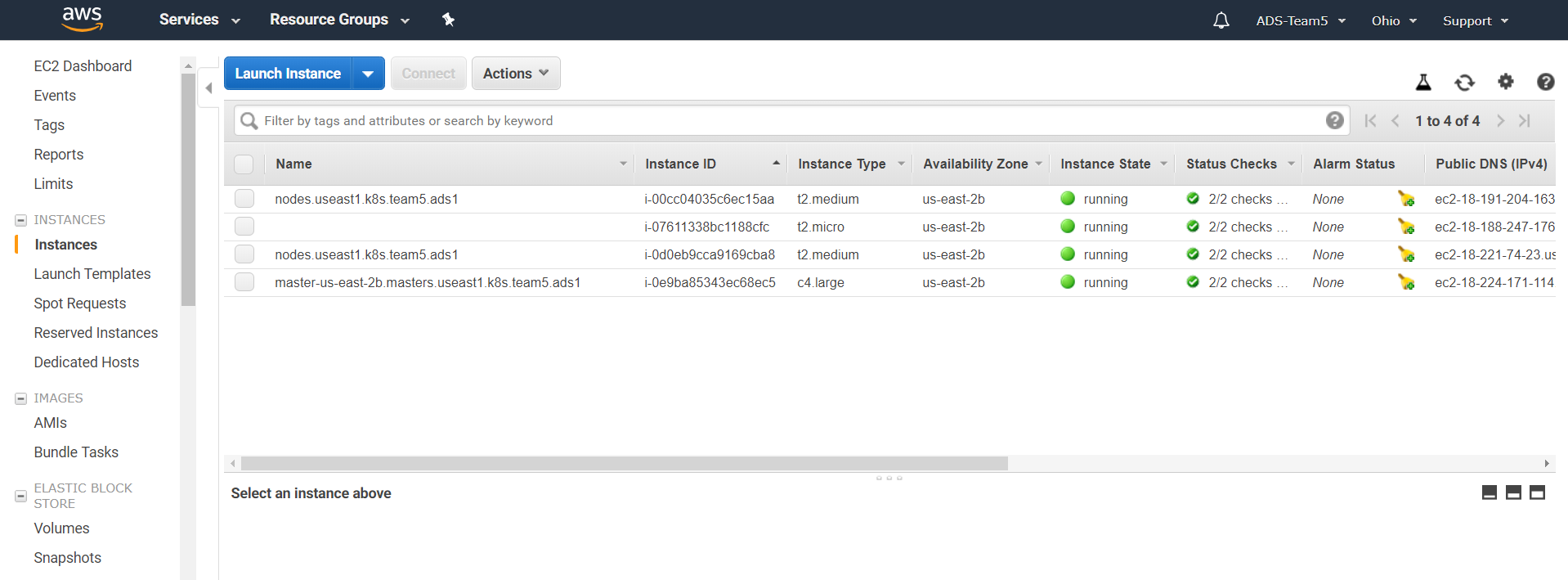


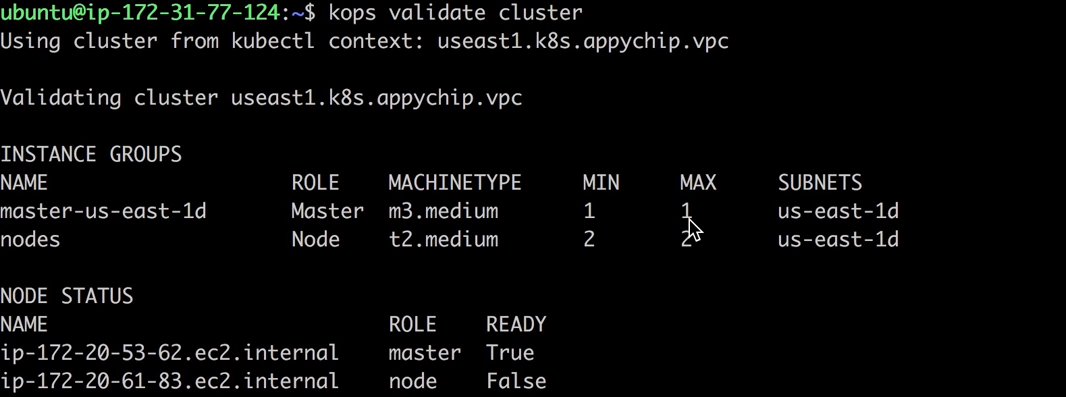
Now to actually create cluster run the following command:

kops update cluster useast1.k8s.appychip.vpc --yes

Now after the update cluster command has been executed the clusters will be created in AWS EC2







To get nodes

Run—kubectl get nodes



**Deploying Nginx Container**

Let’s deploy a simple service made up of some nginx containers:

**Create an nginx deployment:**

$ kubectl run sample-nginx --image=nginx --replicas=2 --port=80  
$ kubectl get pods

NAME READY STATUS RESTARTS AGE  
sample-nginx-379829228-xb9y3 1/1 Running 0 10s  
sample-nginx-379829228-yhd25 1/1 Running 0 10s

$ kubectl get deployments

NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE  
sample-nginx 2 2 2 2 29s

Expose the deployment as service. This will create an ELB in front of those 2 containers and allow us to publicly access them:

$ kubectl expose deployment sample-nginx --port=80 --type=LoadBalancer

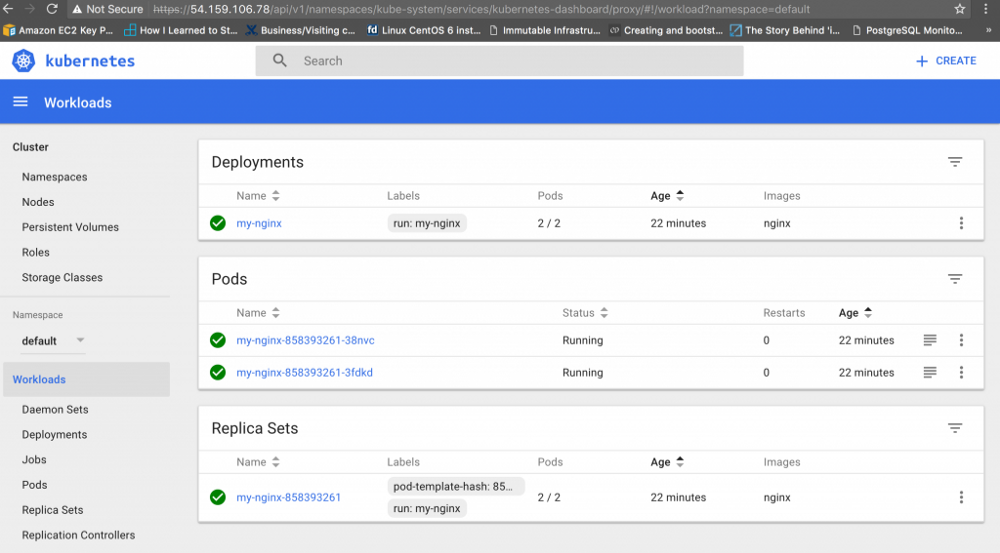
$ kubectl get services -o wide

NAME CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR  
kubernetes 100.64.0.1 <none> 443/TCP 25m <none>  
sample-nginx 100.70.129.69 adca6650a60e611e7a66612ae64874d4-175711331.us-east-1.elb.amazonaws.com/ 80/TCP 19m run=sample-nginx

There is an ELB running on [http://adca6650a60e611e7a66612ae64874d4-175711331.us-east-1.elb.amazonaws.com](http://adca6650a60e611e7a66612ae64874d4-175711331.us-east-1.elb.amazonaws.com/) with our nginx containers behind it:



You can also view the UI by accessing master node. Hit master node’s IP/Domain in browser, it will ask for credentials. Run command ***kubectl config view*** to see the credentials.



To delete the cluster and remove all AWS resources with, run the following command:

$ kops delete cluster --name=useast1.k8s.appychip.vpc –yes

That’s All is required to setup kubernetes on AWS EC2.