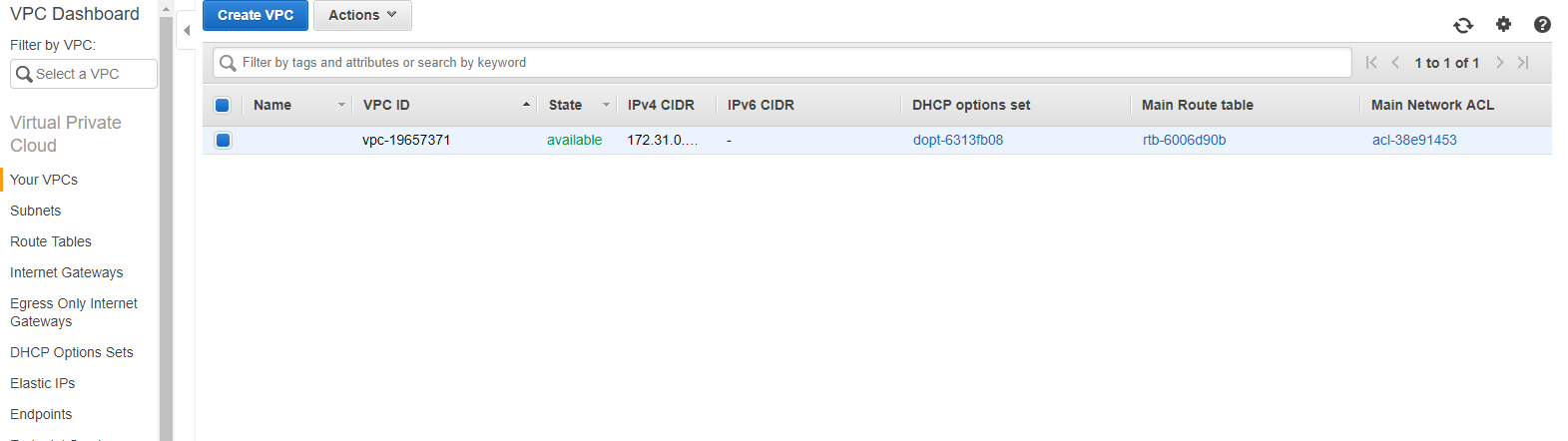
**Kubernetes Test**

Step 1: As per the task first we have to create the VPC in AWS Console.

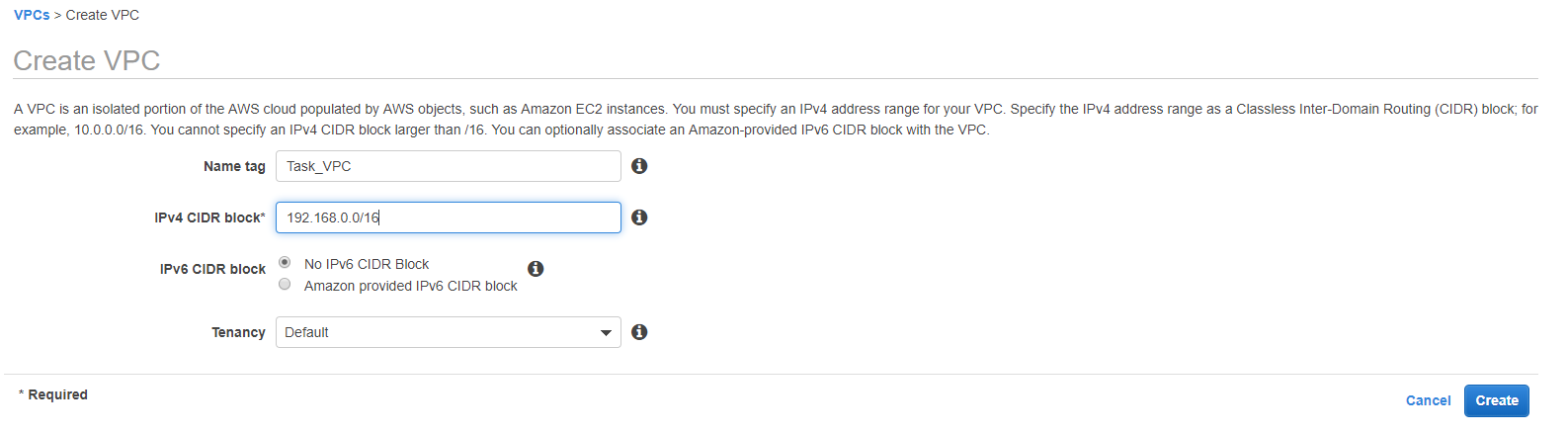
Step 2: Now Login to AWS Management Console

Step 3: From the services select the VPC



The above diagram will shows the VPC and VPC navigation pane

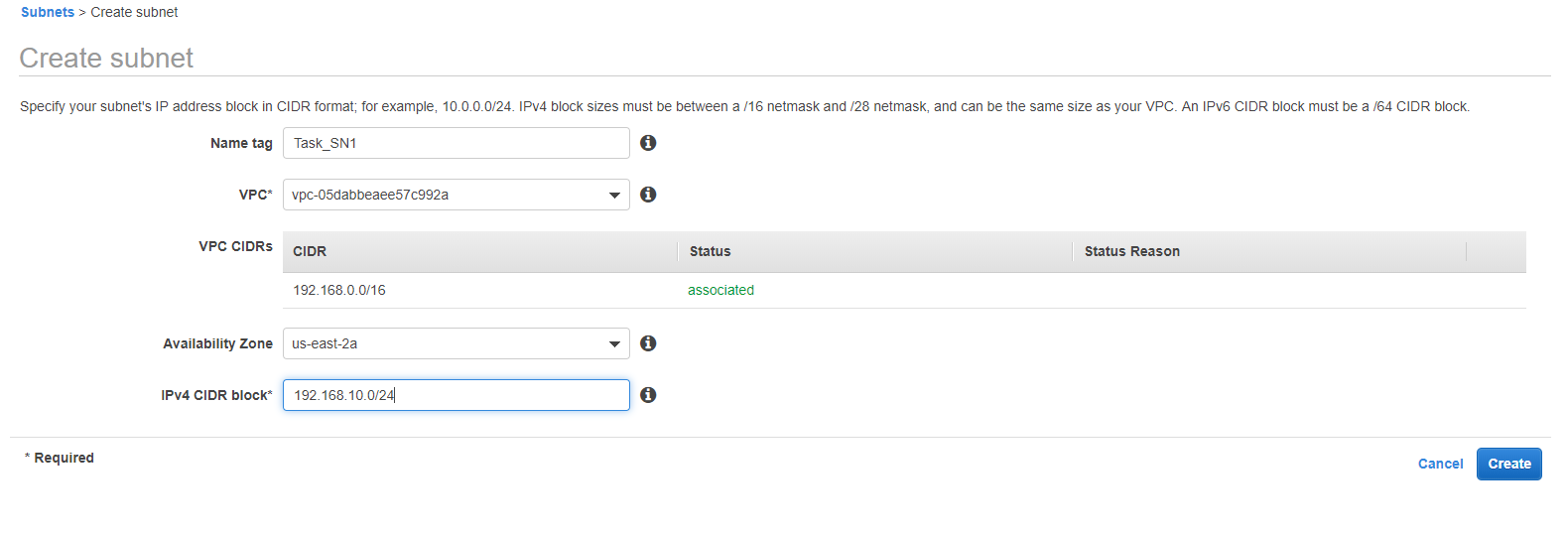
Step 4: Now click on “Create VPC” from the VPC navigation pane



Fill it with the required fields as shown in the above figure

Now click on “create” option to create the VPC

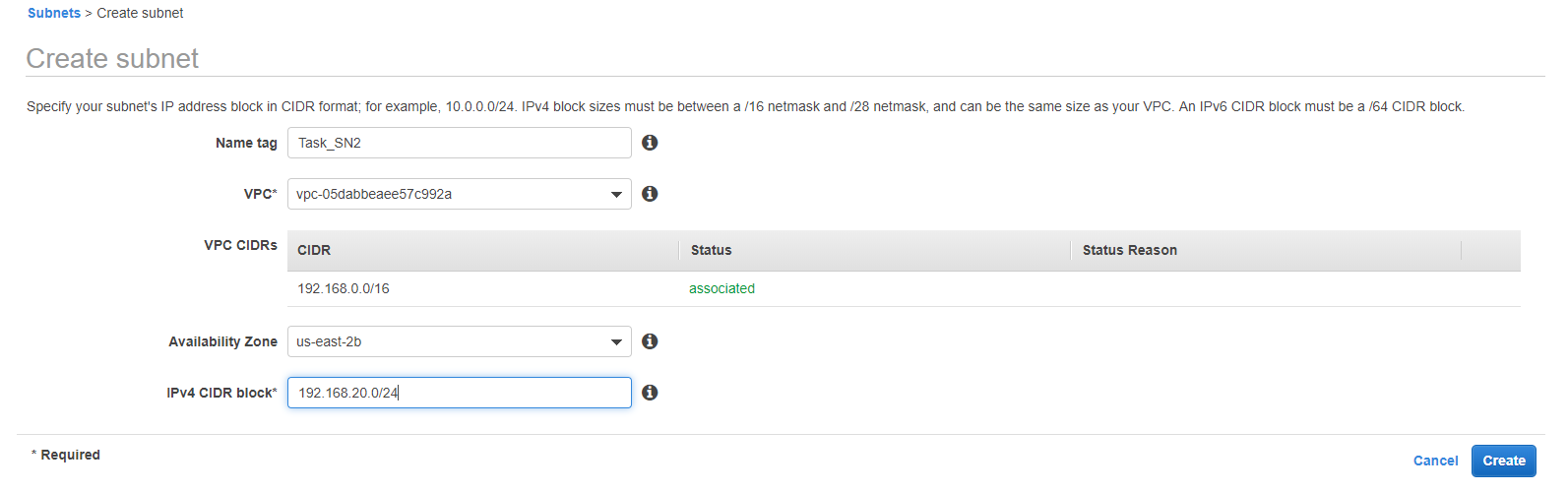
Step 5: Click on “Subnets” from the navigation pane and click on the “Create Subnet” option



Fill the fields with required details

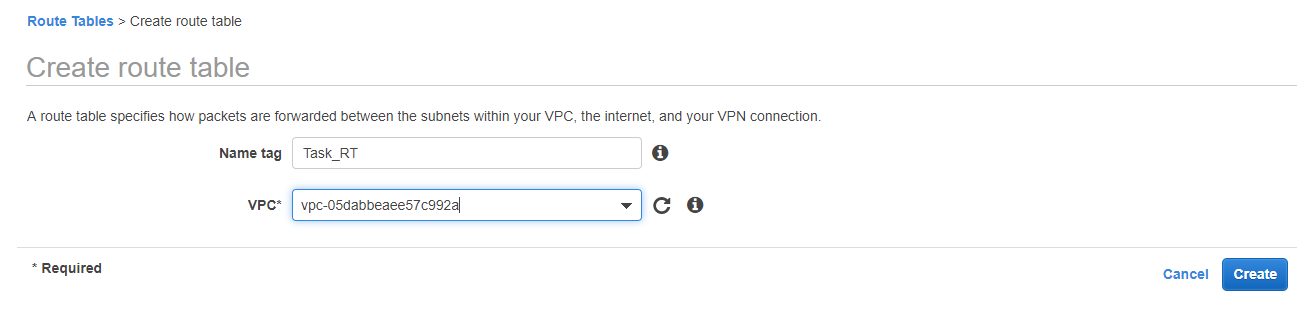
Now click on “create” option then the subnet will be created.

Step 6: Again click on “subnet” to create one more subnet



Fill the fields and click on create.

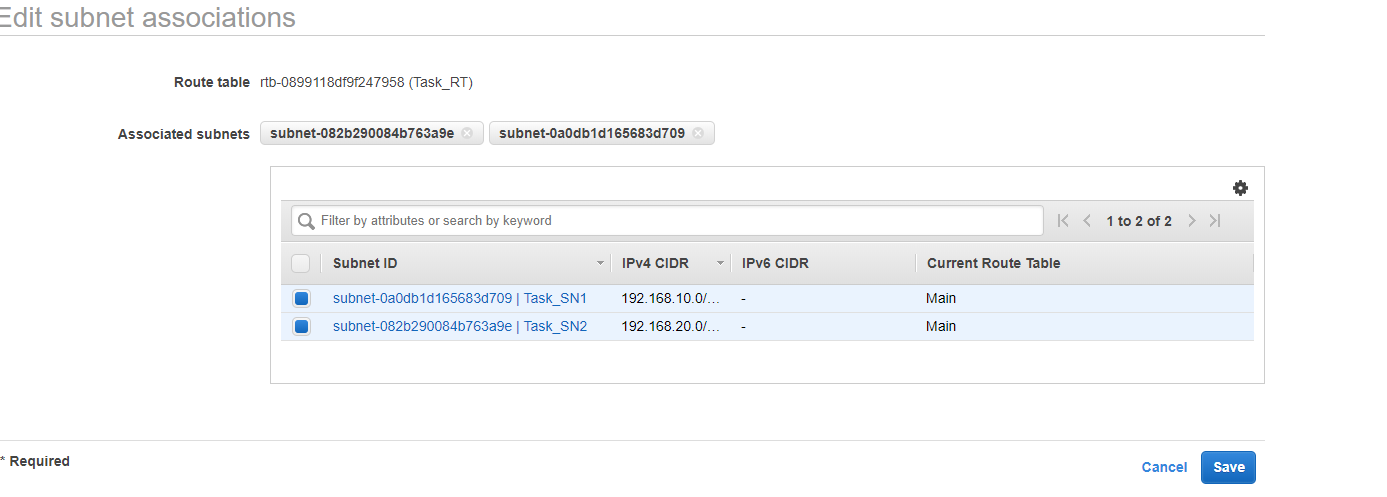
Step 7: Click on Route Table form the navigation pane



Fill the required fields as shown above

Select the VPC “Task\_VPC” which we have created.

Once after creating Route Table now click on “subnet associatios”

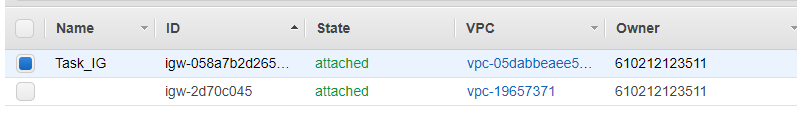


Select the two subnets which we have created and click on “save”

Step 8: Click on “Internet Gateway”

Now click on “Create internet gateway”

Give name as “Task\_IG” and click on create



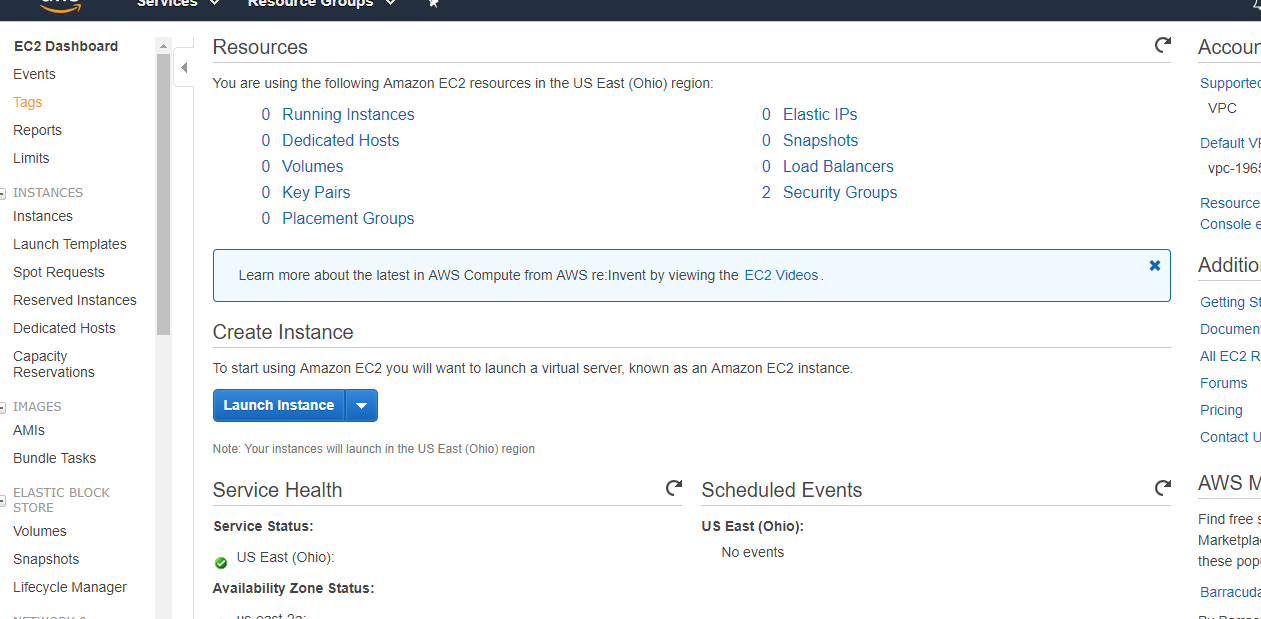
Attach the internet gateway to the created VPC “Task\_VPC”

Now click on “Route Table” and add the Route by giving “0.0.0.0/0” and select the Task\_IG and click on save.

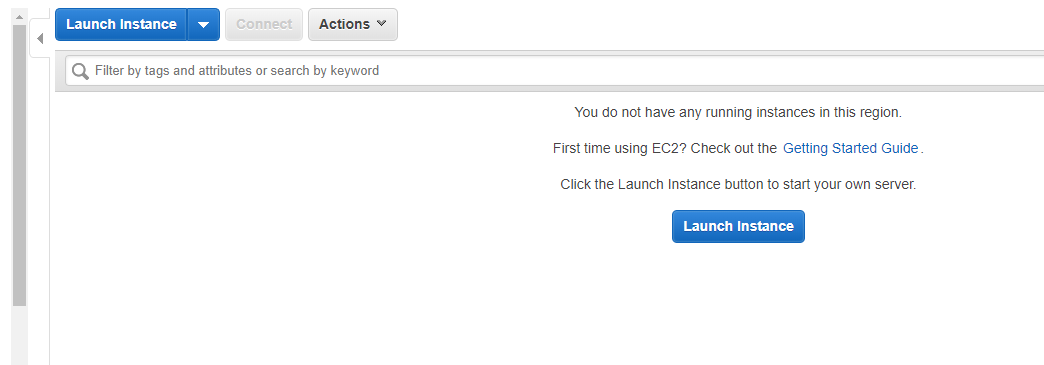
Now our “VPC with two avalability zone has created”

Now we will laumch the EC2 instances.

Select EC2 from Services

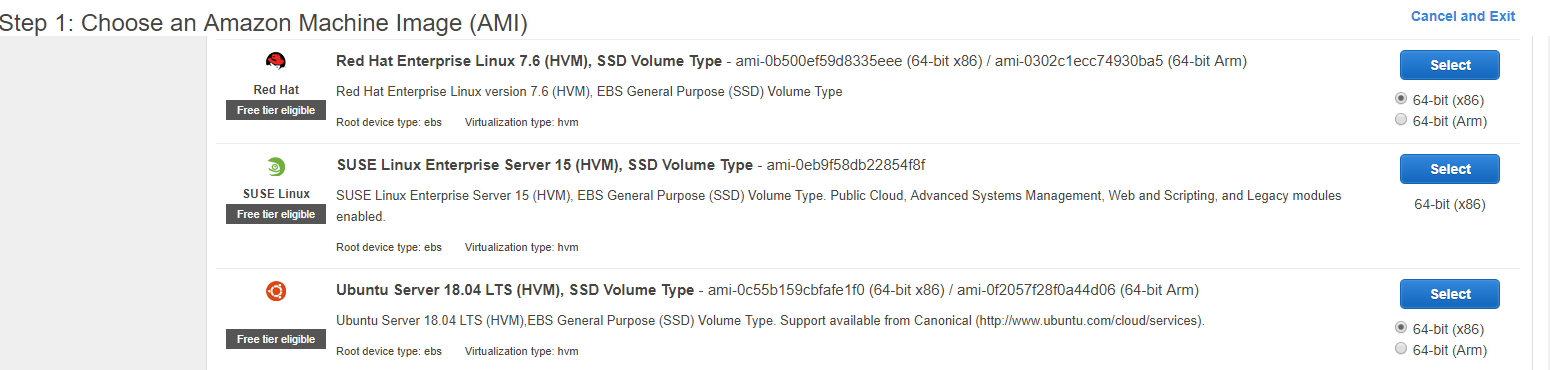


The above diagram shows the “EC2 Dashboard View”



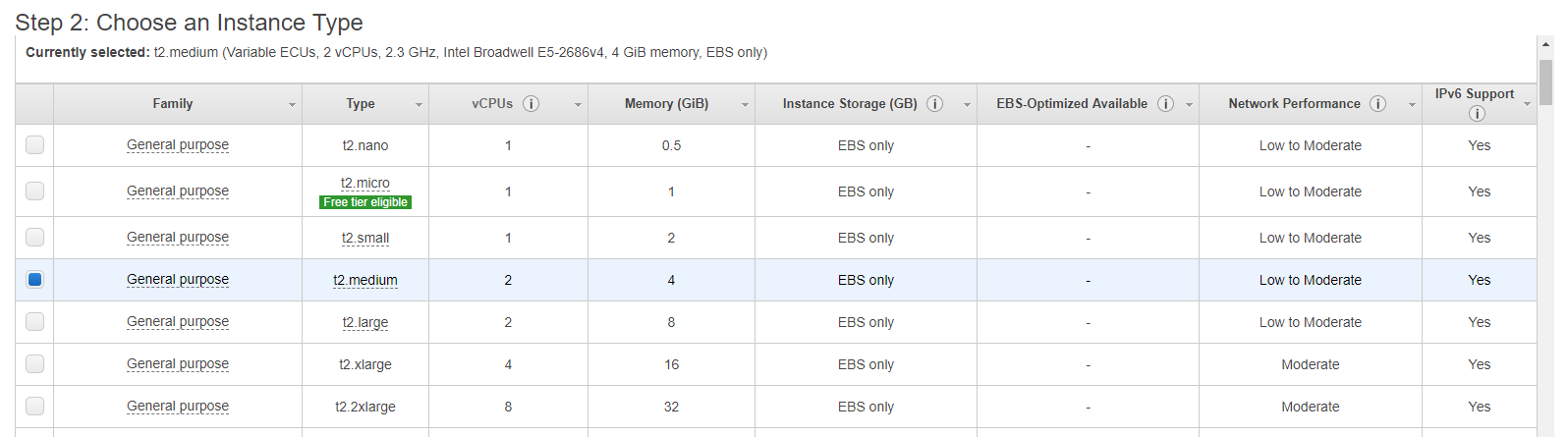
Now click on “Launch instance”

Step 1:

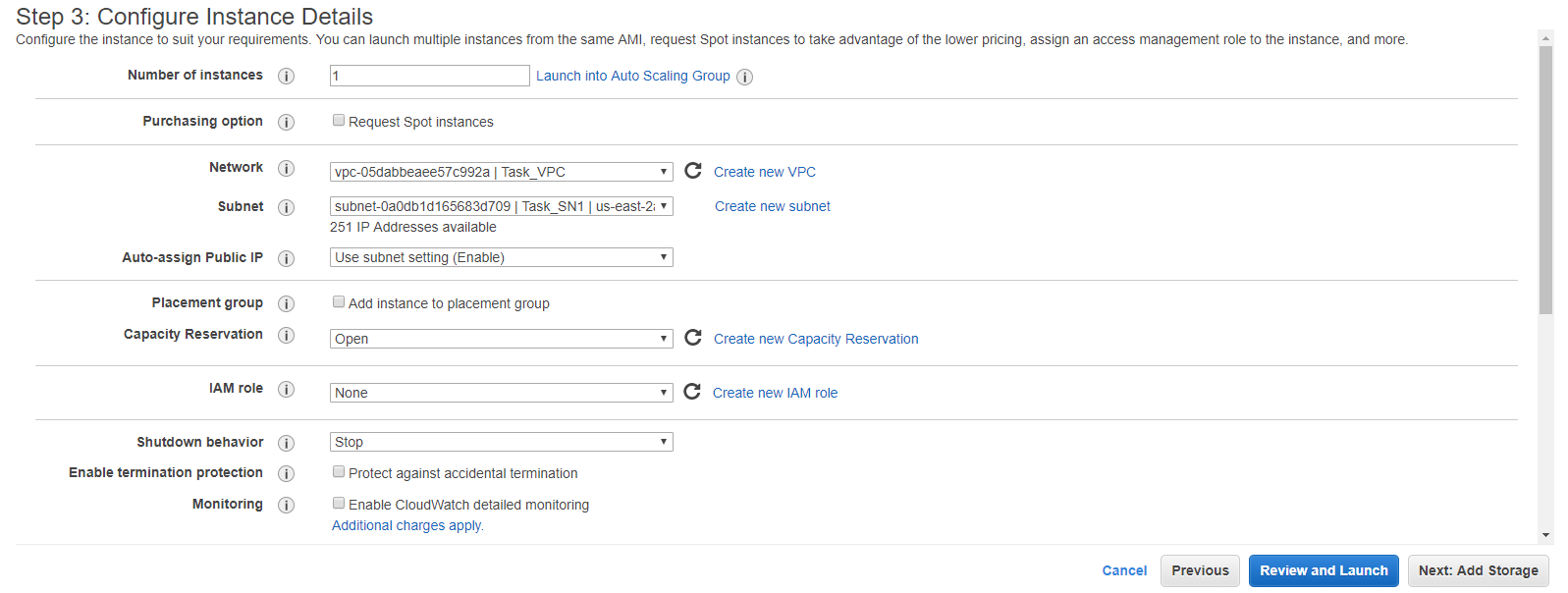


Select the Ubuntu Server 18.04 LTS (HVM)

Step 2: Select the “t2.medium” from the instance type

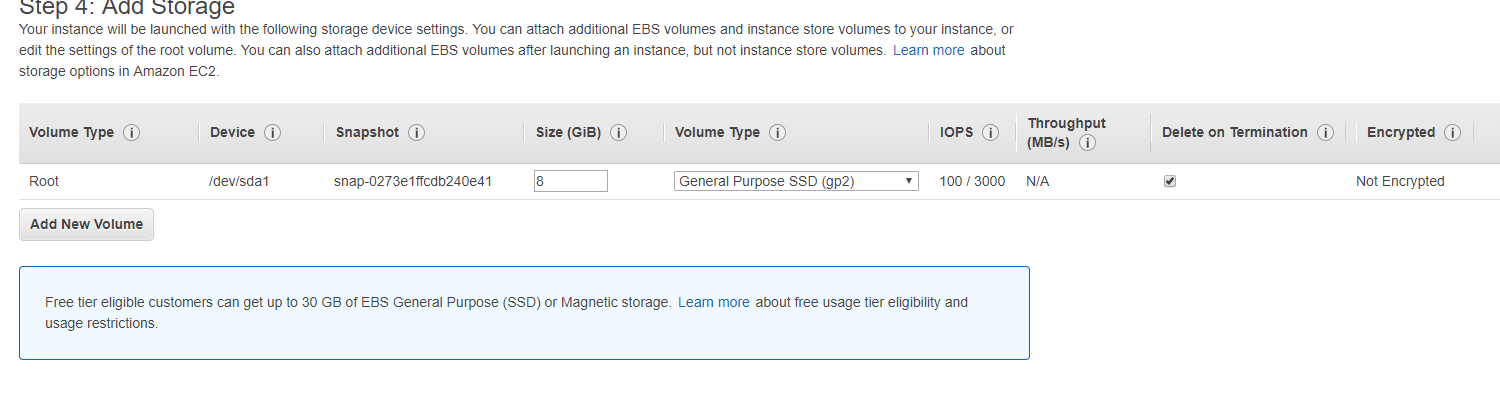


Step 3:

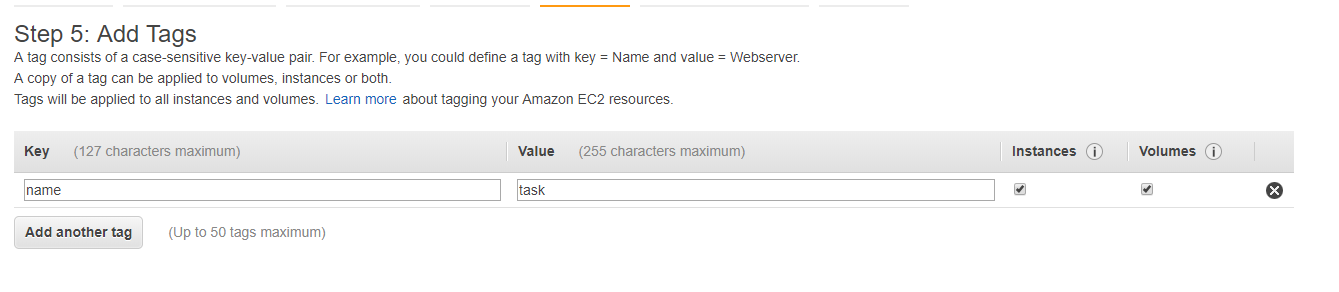


Selct the VPC which we have created “Task\_VPC”

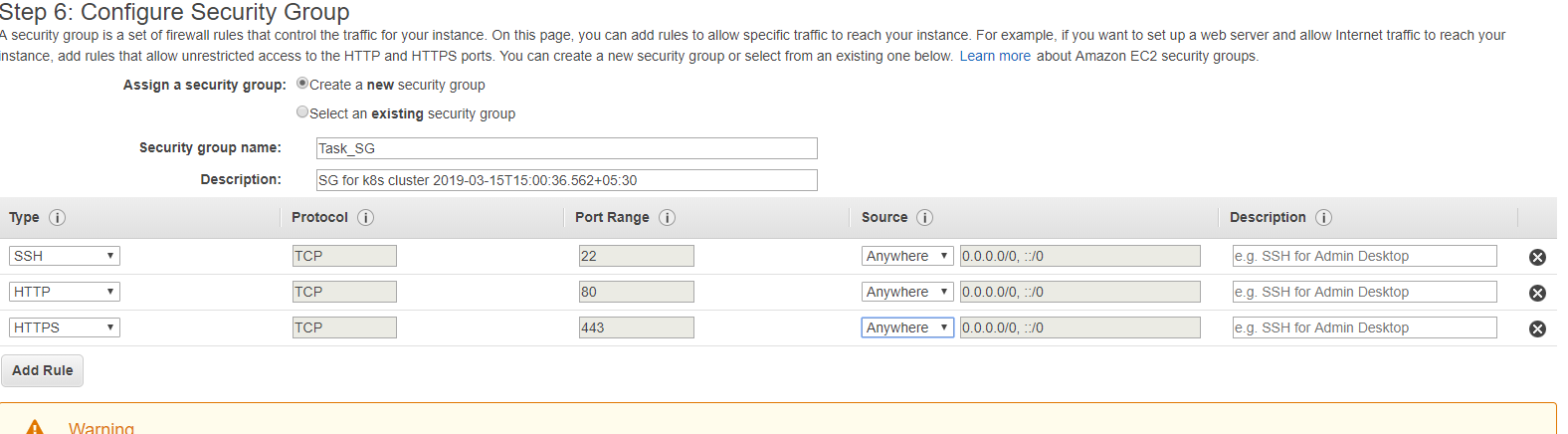
Step 4: Add storage



Step 5: Add tag



Step 6: Configure the Security Group



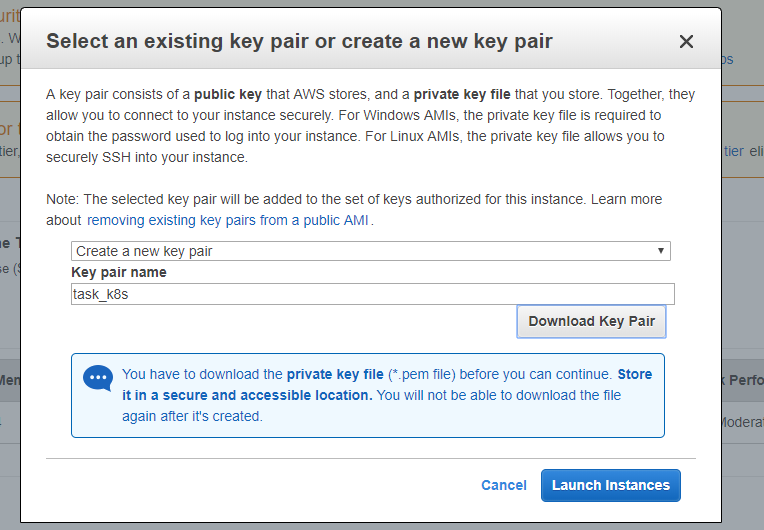
Add SSH 22 Anywhere

Add HTTP 80 Anywhere

Add HTTPS 443 Anywhere

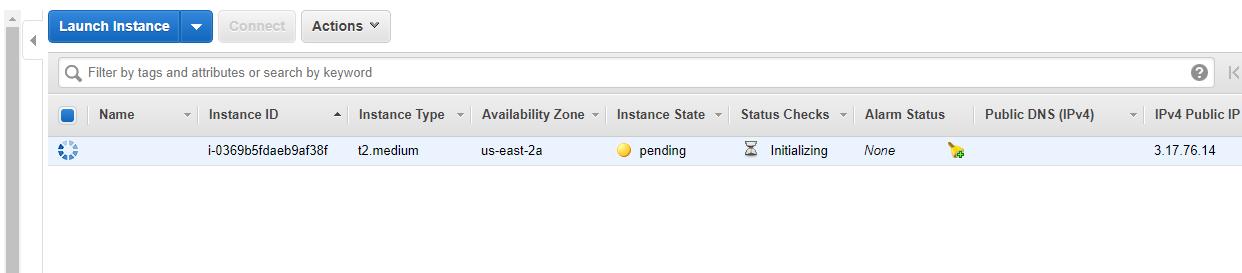
Click on review and launch

Create a new key pair “task\_k8s”



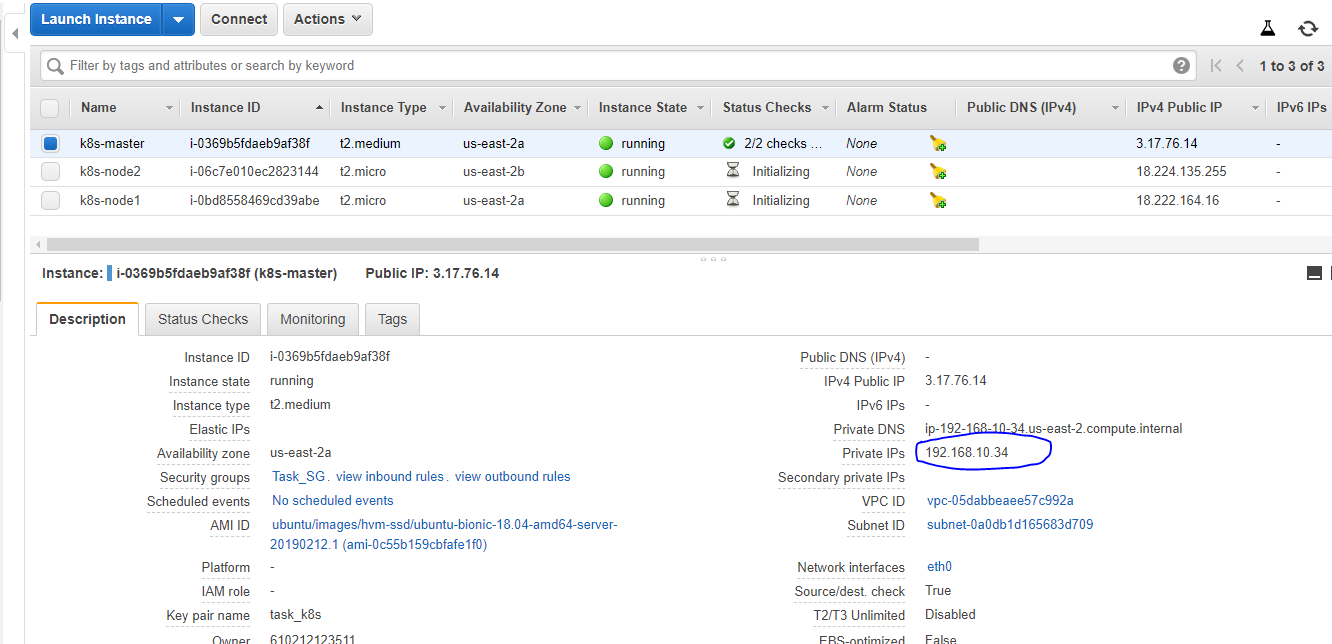
Now click on “Launch instances”

Now look at the dashboard instance is getting created



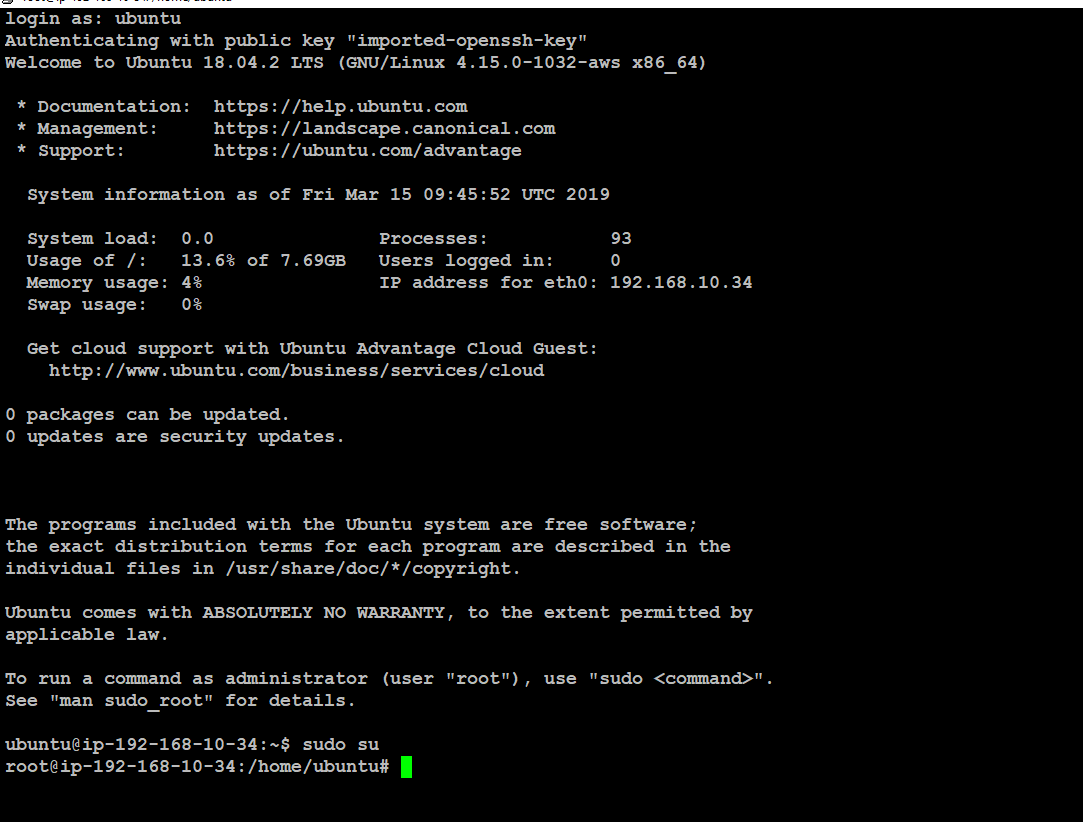
Rename the instance name as k8s-master

Now create the other 2 instances for slaves following the same steps



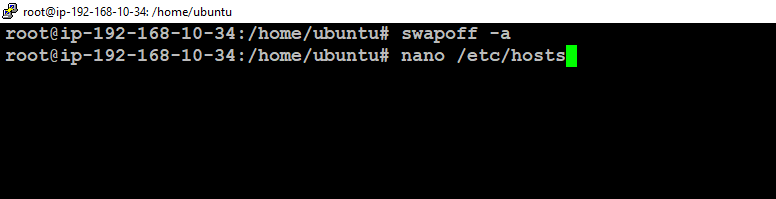
To login to the instances download the putty in your system from putty official site.

Login to the k8s-master

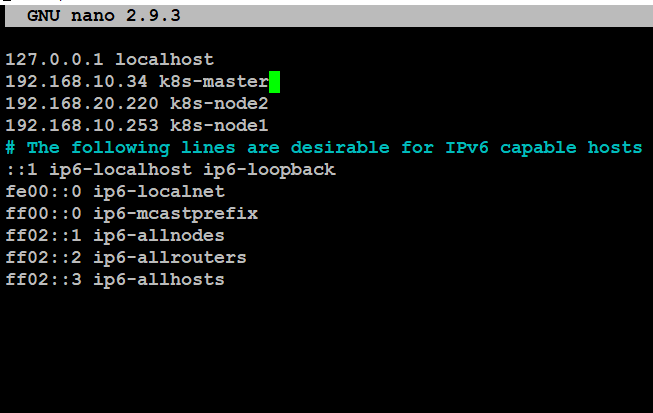


Like wise login to the two other systems as well

List of commands should be entered to install all the dependencies

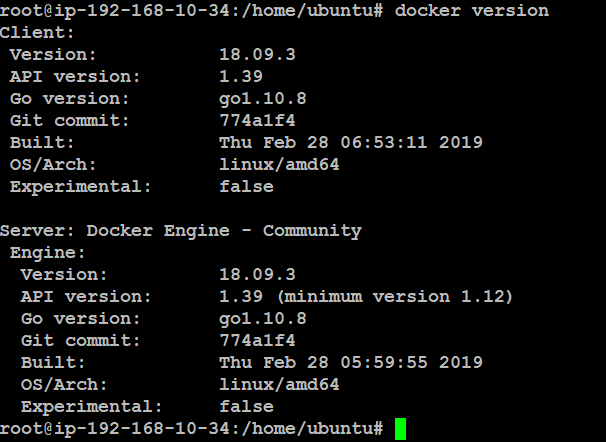


Edit with all instances private ip address



Save and exit

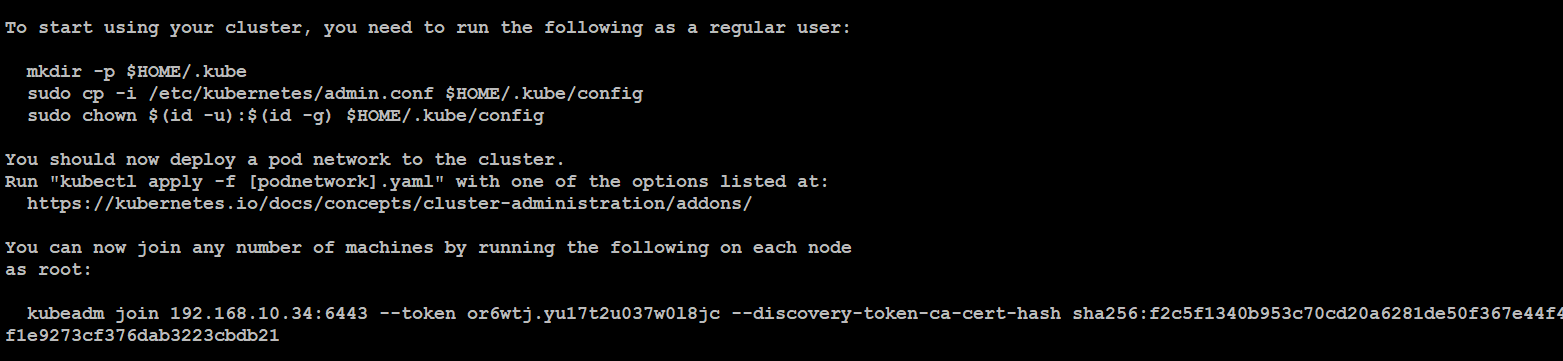
Installing docker



Once after entering all the above commands

We have to initialize the cluster by using $kubeadm init

kubeadm init --apiserver-advertise-address 192.168.10.34 --pod-network-cidr=192.168.0.0/16



Now run the below commands in the master

mkdir -p $HOME/.kube

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

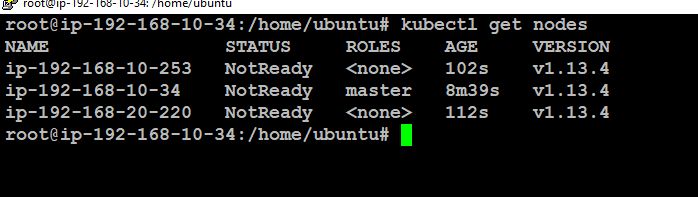
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Now use the “Kubeadm join command” in two nodes

kubeadm join 192.168.10.34:6443 --token or6wtj.yu17t2u037w0l8jc --discovery-token-ca-cert-hash sha256:f2c5f1340b953c70cd20a6281de50f367e44f48f1e9273cf376dab3223cbdb21

Now in the master machine give

#kubectl get nodes

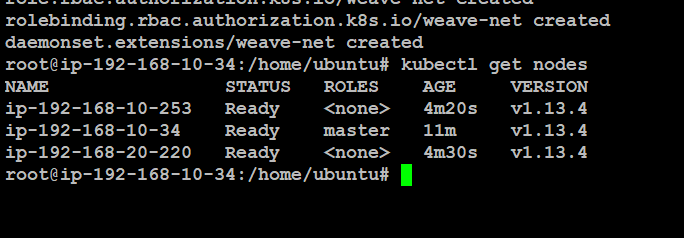


In the output we can see the status is not ready that is because of weave net configuration that can be done by using the following command

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

Now we hit the same command

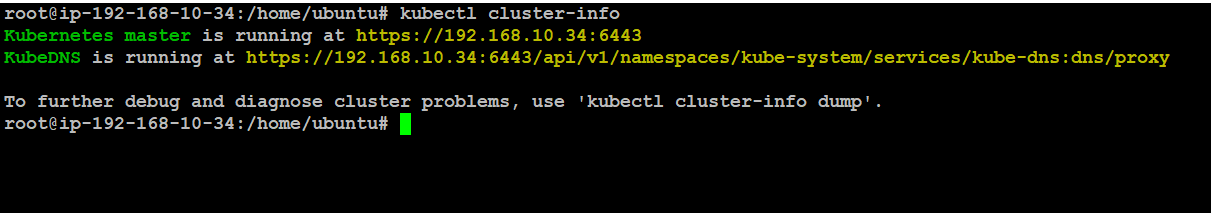
#kubectl get nodes



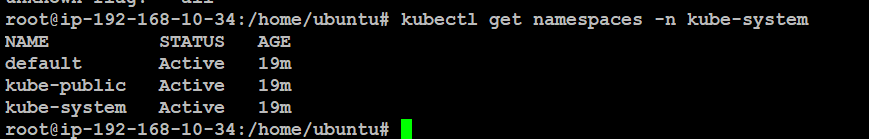
So successfully we have created the K8s cluster

Now we can for the deployment of webapp on this cluster

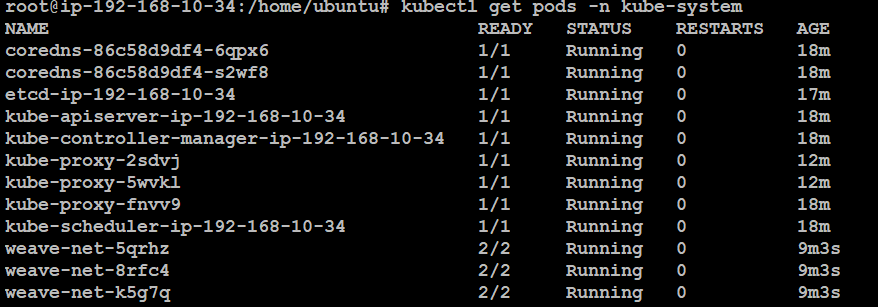
Cluster info



Checking the namespaces

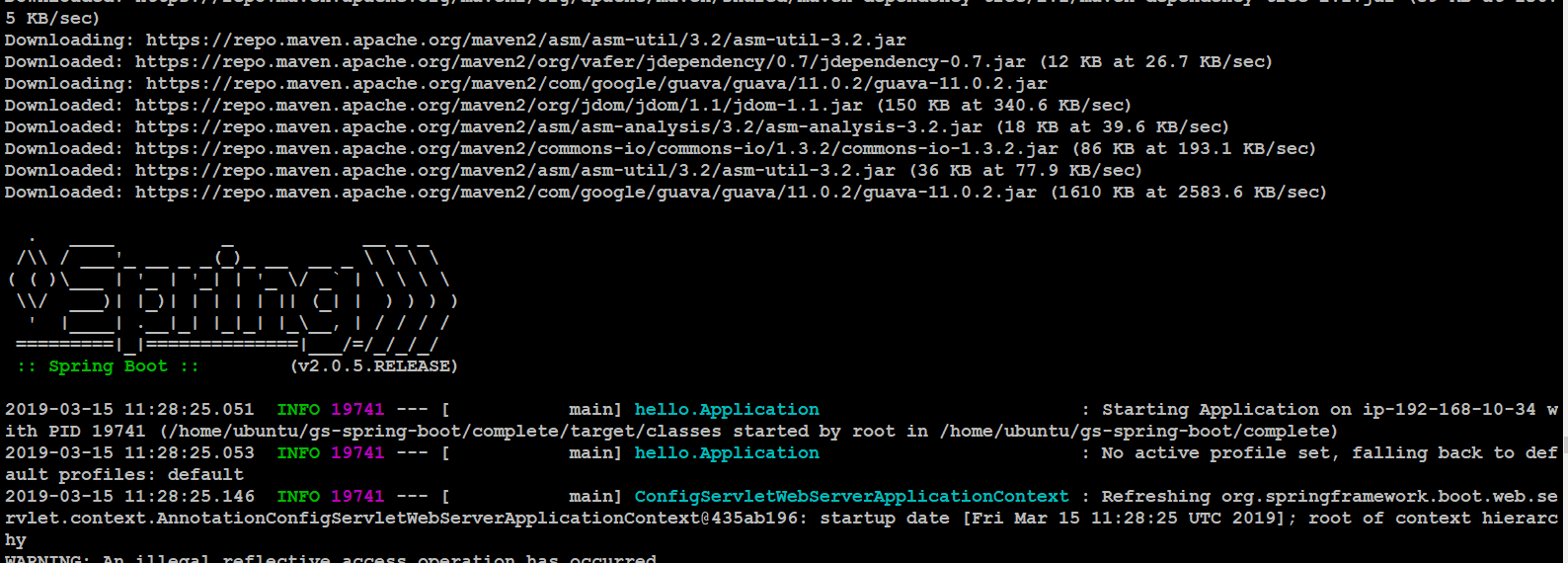


Checking the pods

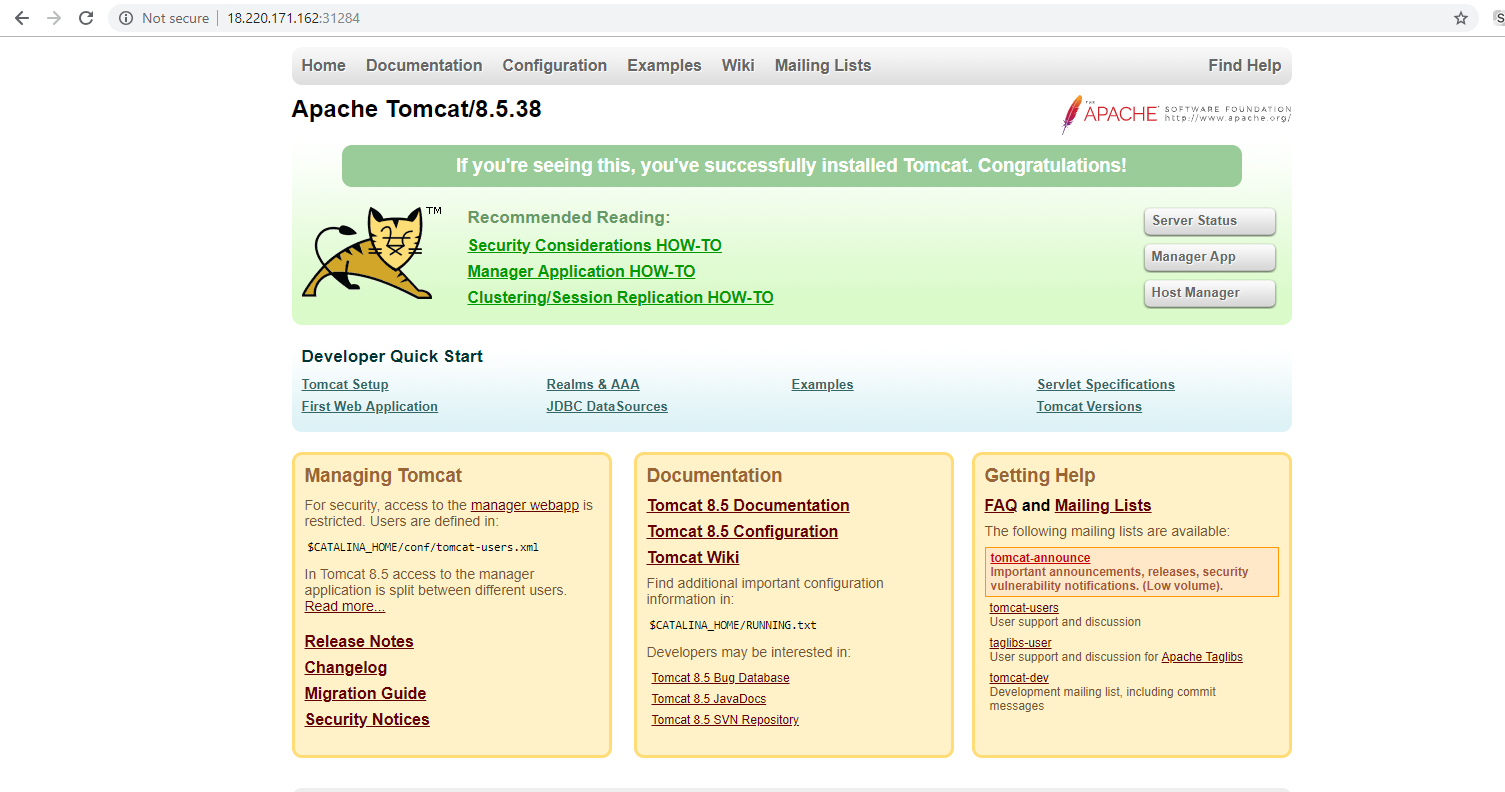


Now apply the above file tomcat.yml

Spring boot configuration



Deploying Tomcat:



I have a knowledge on Helm I have used it for the Stolon application.