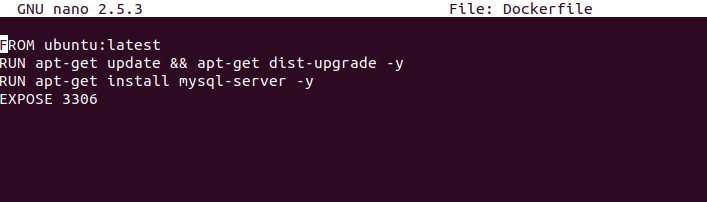
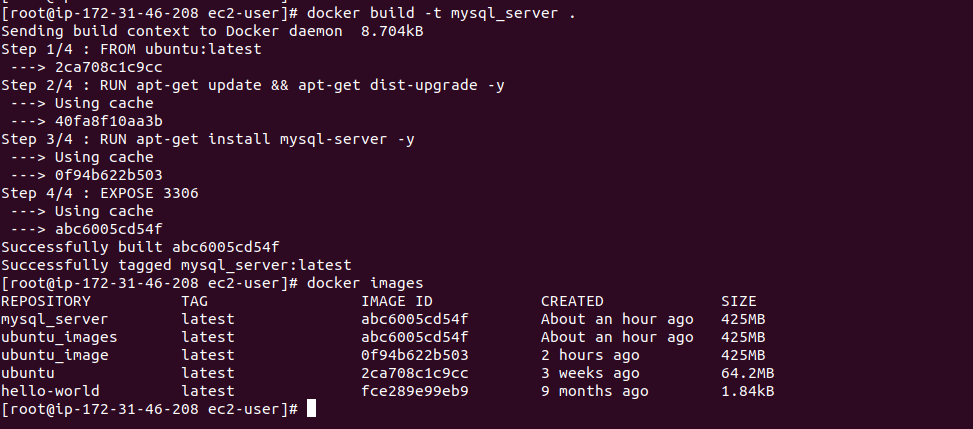
### **Docker:**

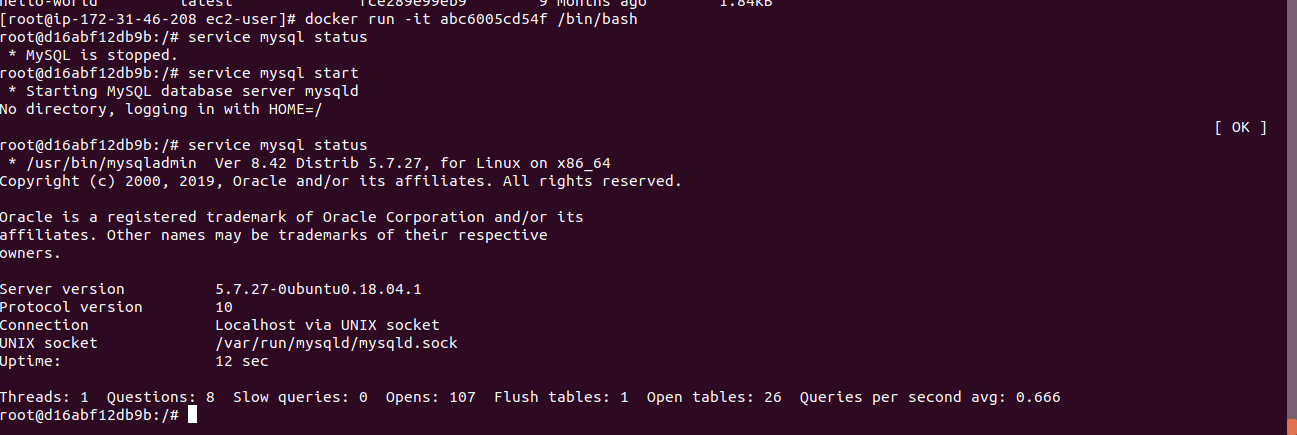
Create a docker file



Build a docker image using the docker file



Build container from that image using image id



#### **Docker Compose:**

Write a docker-compose file to run mongodb and mysql

Open another terminal and see the running containers

docker exec -it <container-id or name> bash

docker exec -it mongodb bash

Type mongo and it will enter into mongo shell

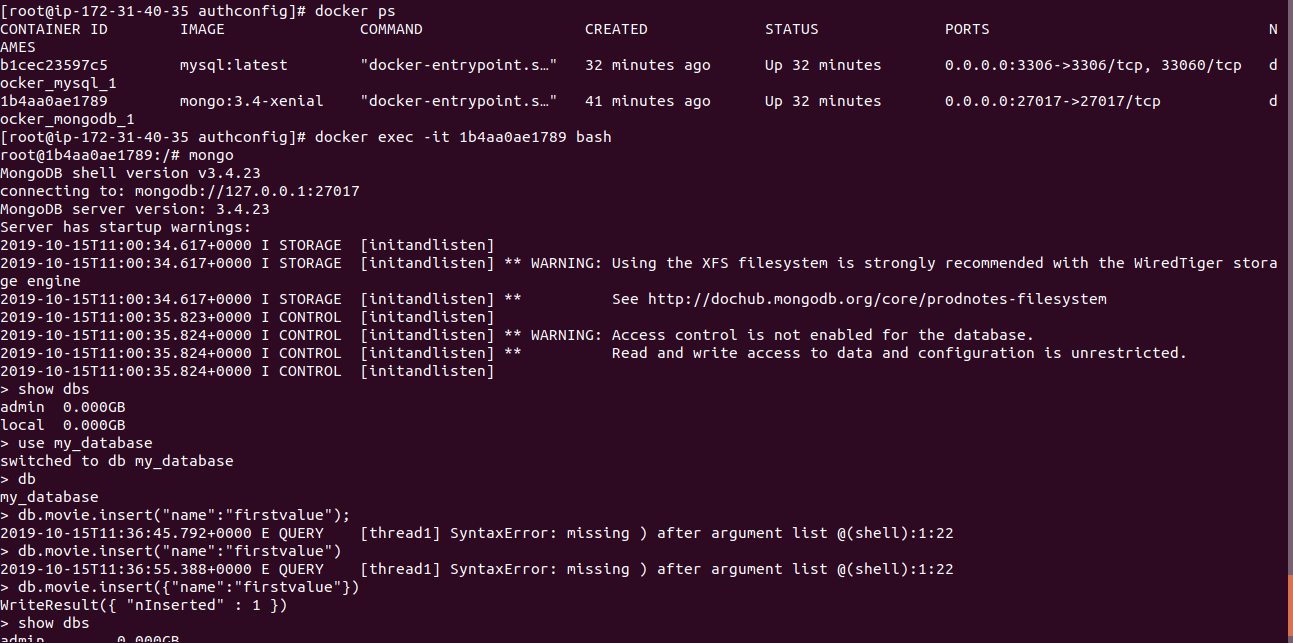
To create a database *use DATA\_BASE* command

*use my\_database*

Database is empty and will not show in dbs

First insert some values into the database and see the database

*show dbs*



version: '3'

services:

mysql:

image: mysql:latest

ports:

- 3306:3306

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=dev\_root

- MYSQL\_DATABASE:devdb

- MYSQL\_USER:devuser

- MYSQL\_PASSWORD:devpwd

mongodb:

image: mongo:3.4-xenial

ports:

- 27017:27017

restart: always

environment:

- MONGO\_INITDB\_ROOT\_USERNAME:mongoadmin

- MONGO\_INITDB\_ROOT\_PASSWORD:ahchahphai8miSai

volumes:

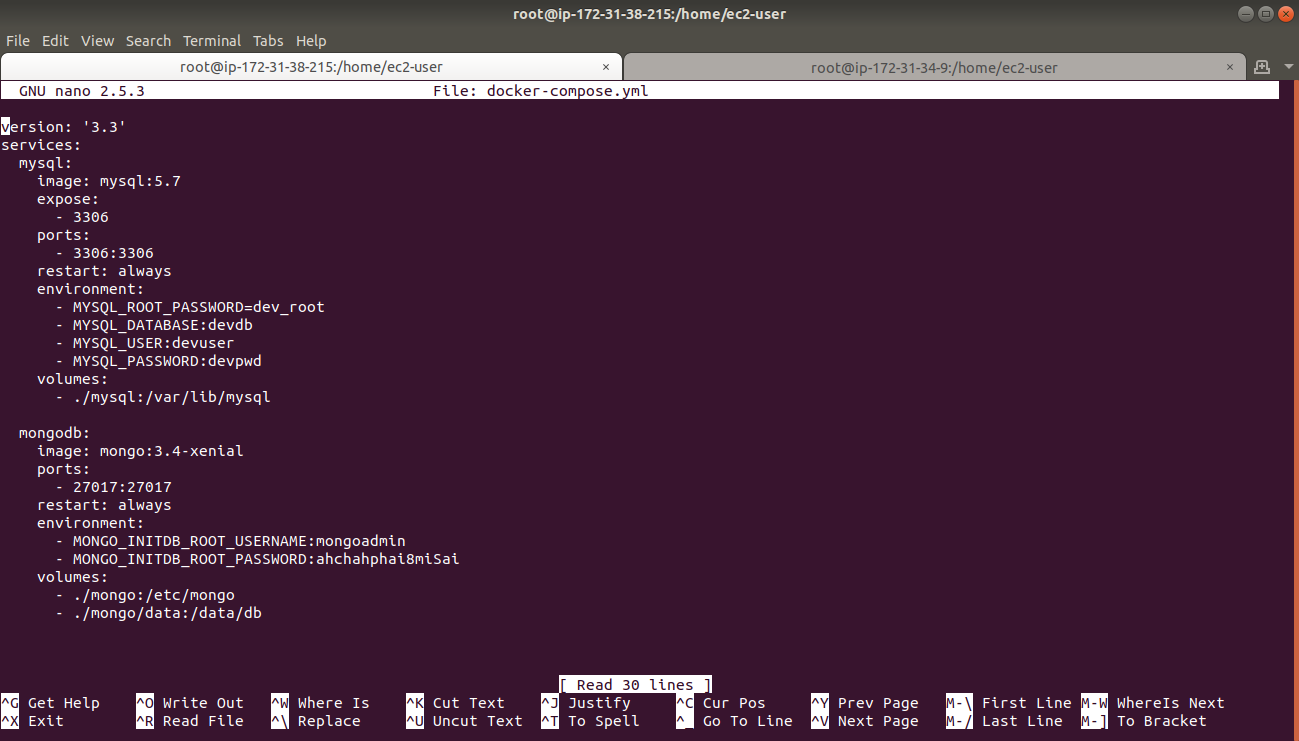
- ./mongo:/etc/mongod.conf

- ./mongo/data:/data/db

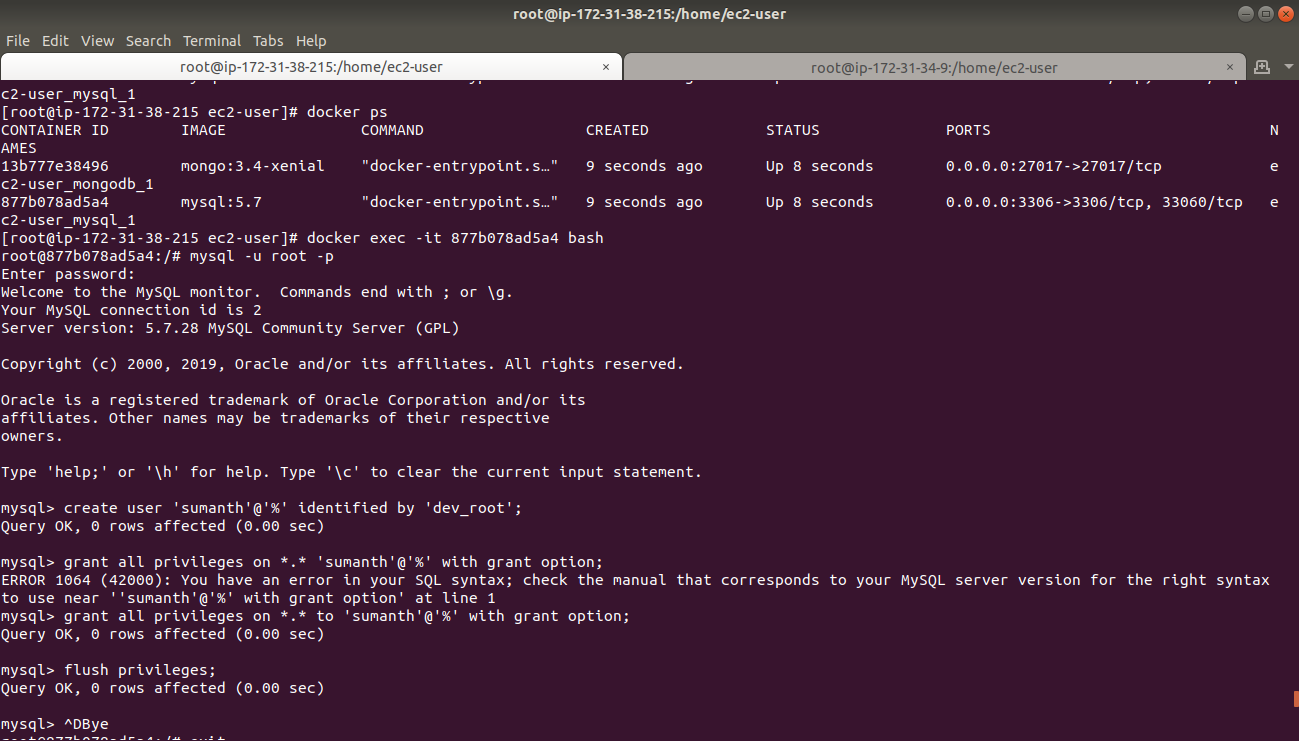
command: mongod --config /etc/mongod.conf

#### **MYSQL:**

Create a docker compose file



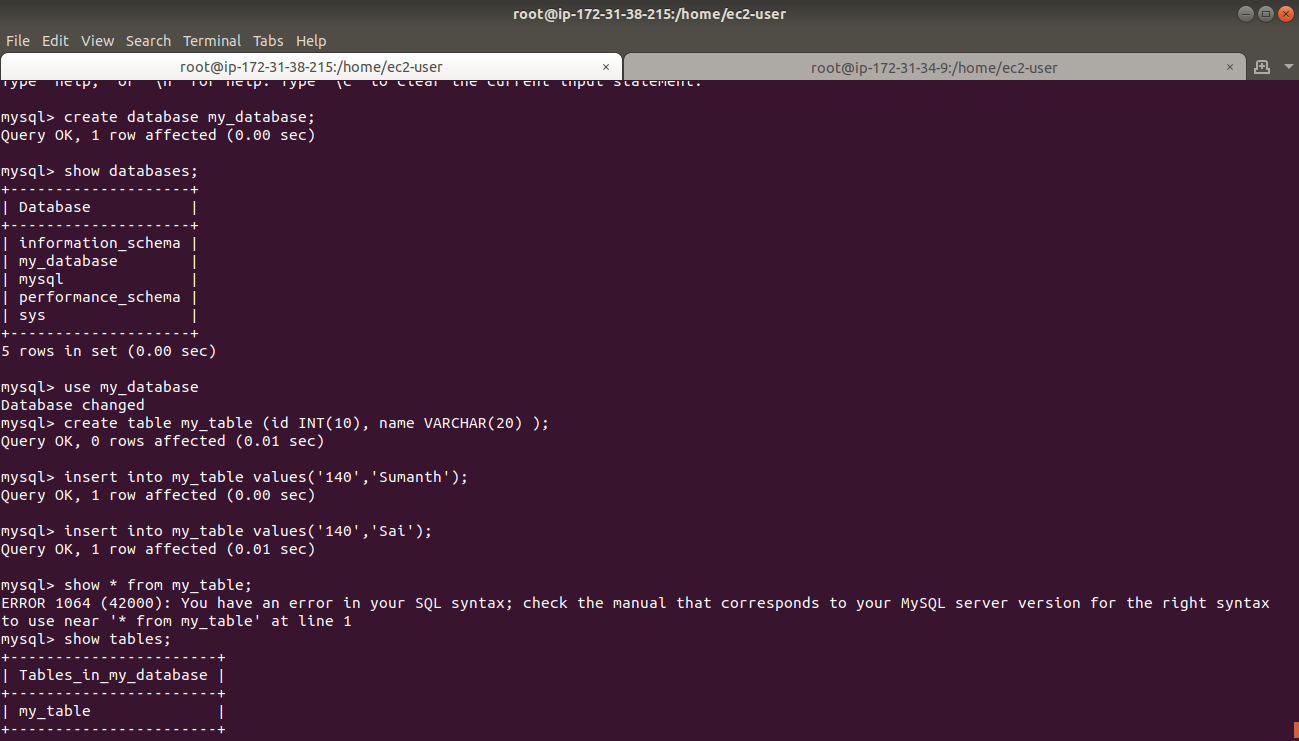
Check the docker containers whether the containers are running or not.



Execute the mysql container and login into mysql as a root user.

*docker exec -it <docker-container-id> bash*

*mysql -u root -p*



Create a database by using the command

*create database <database\_name>;*

And add table into the database

*create table <table\_name>;*

And add values into the table

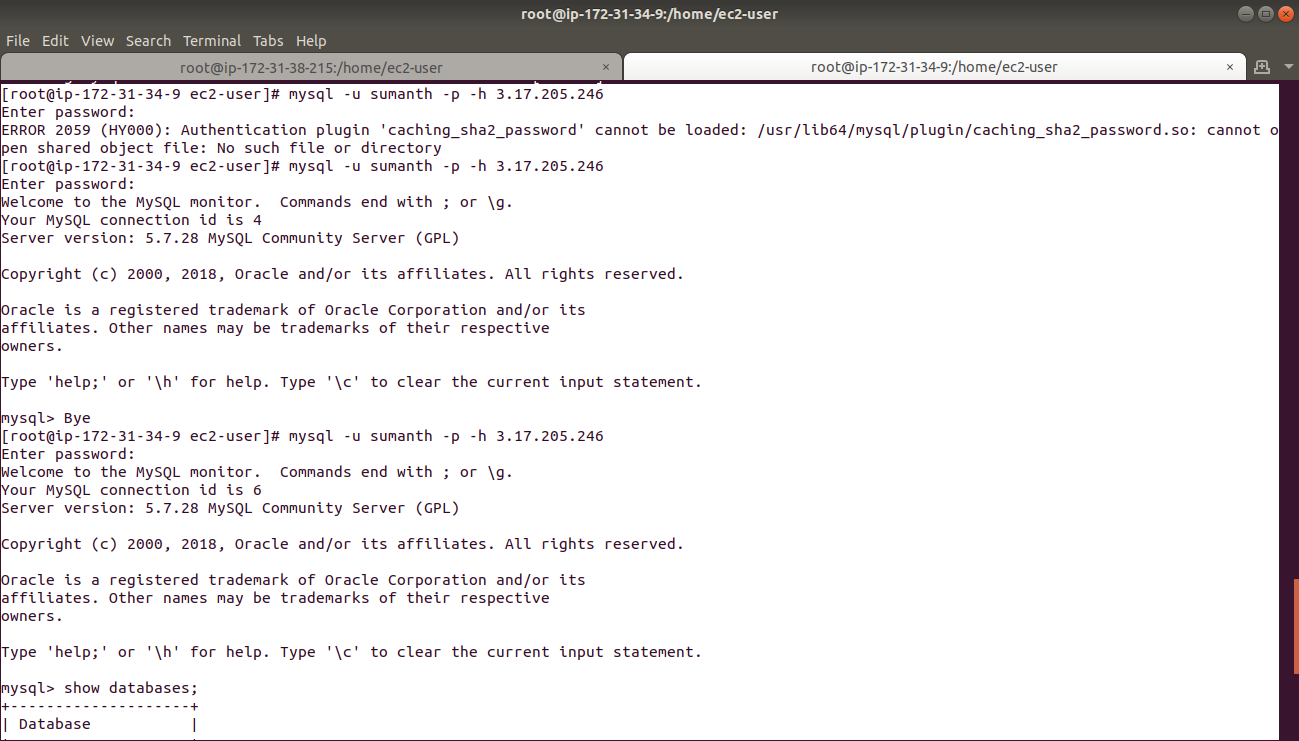
Then open remote server and install sql in that server

*sudo yum install mysql-server;*

And access the mysql which is installed in another server

*mysql -u root -p -h <ip-address-of-instance-in-which-mysql-is-installed>;*

You can access the mysql from any remote server

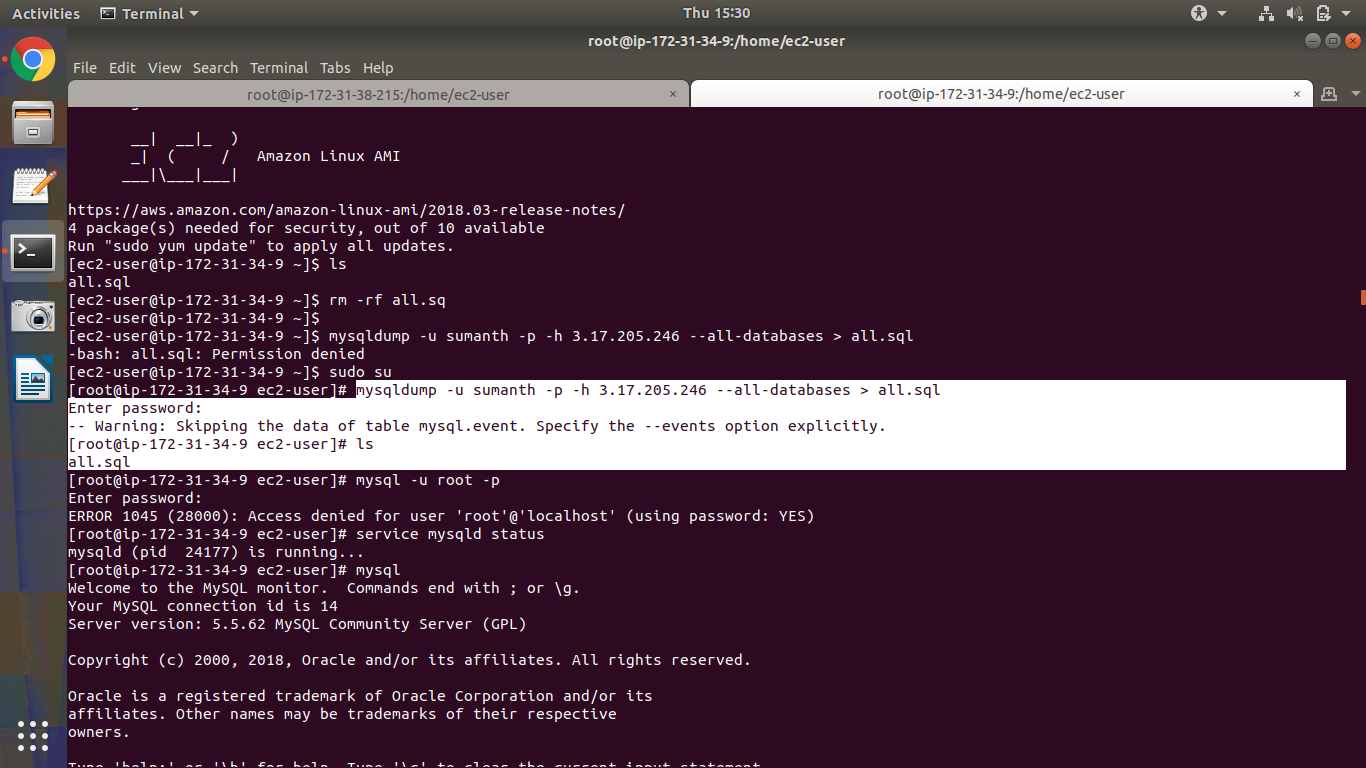


Now dump all the databases into remote server

Ssh the server in which you want to dump the databases;

*mysqldump -u <user\_name> -p -h <ip-address-mysql-server> --all-databases > all.sql*

All databases will be dumped into all.sql

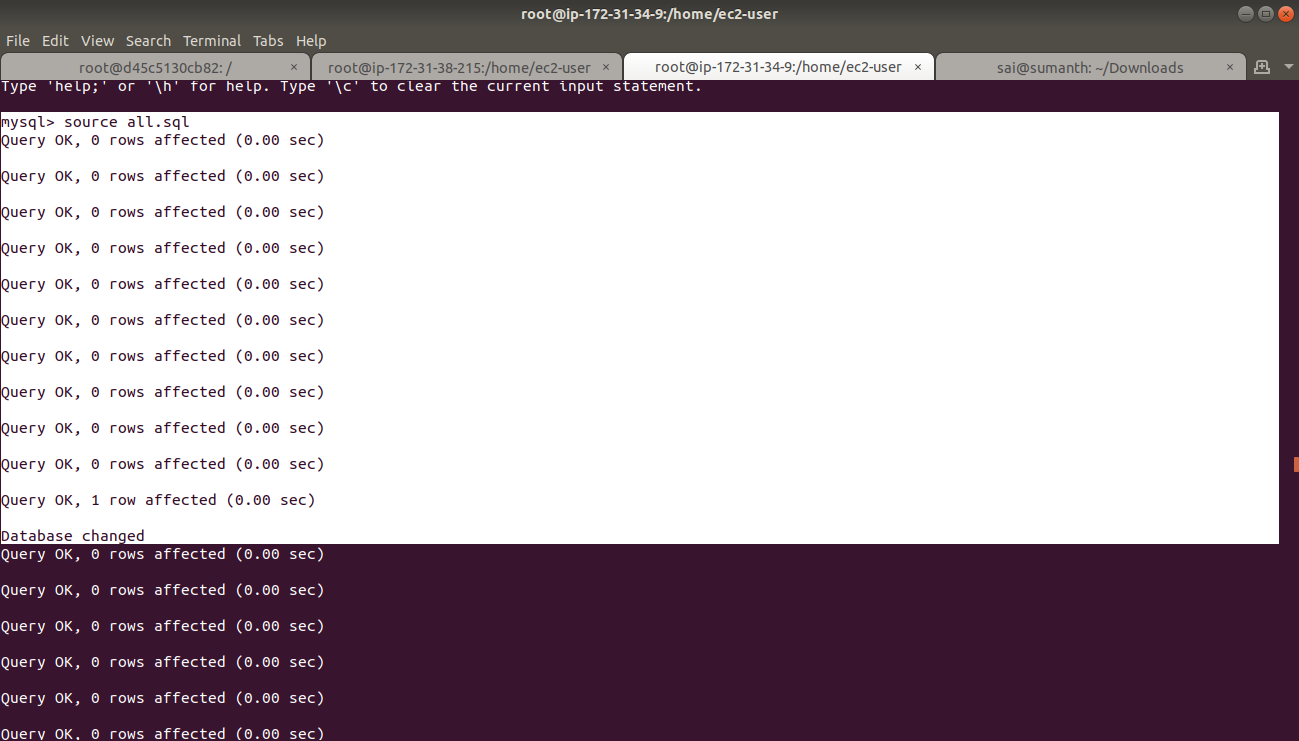


And to restore the databases use all.sql file in which we recently we dumped all the databases

For that go to the folder in which all.sql is present and login into mysql

And use command

*source all.sql*



#### **Mongo:**

Execute the container in which mongo is running

*docker exec -it <container\_id> bash;*

Open mongo

show dbs;

use <data-base-name-to-be-created>;

show dbs;

It will not show the newly created database because no data exist in that database.

Insert values into new database and see the database exist or not

show dbs;

You should see the newly created database.

In mongo user will be created for database.

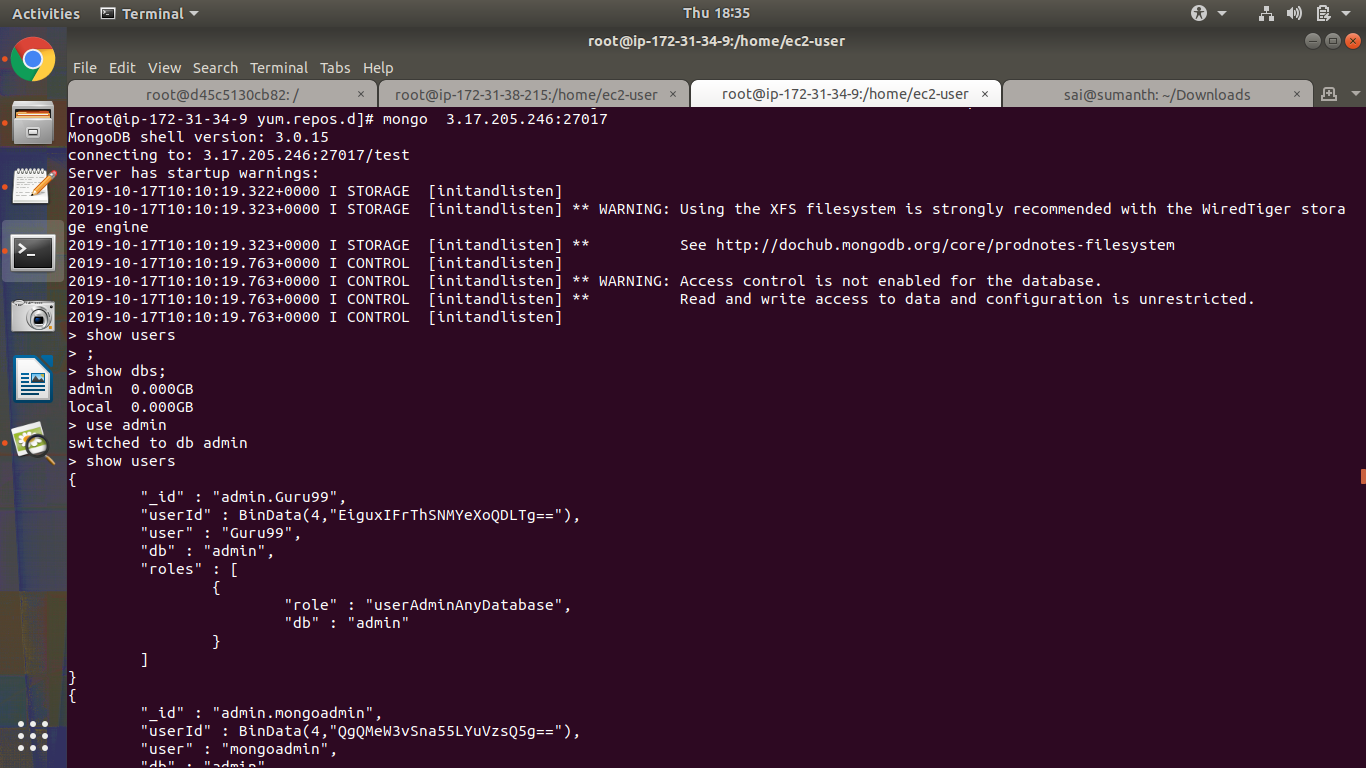
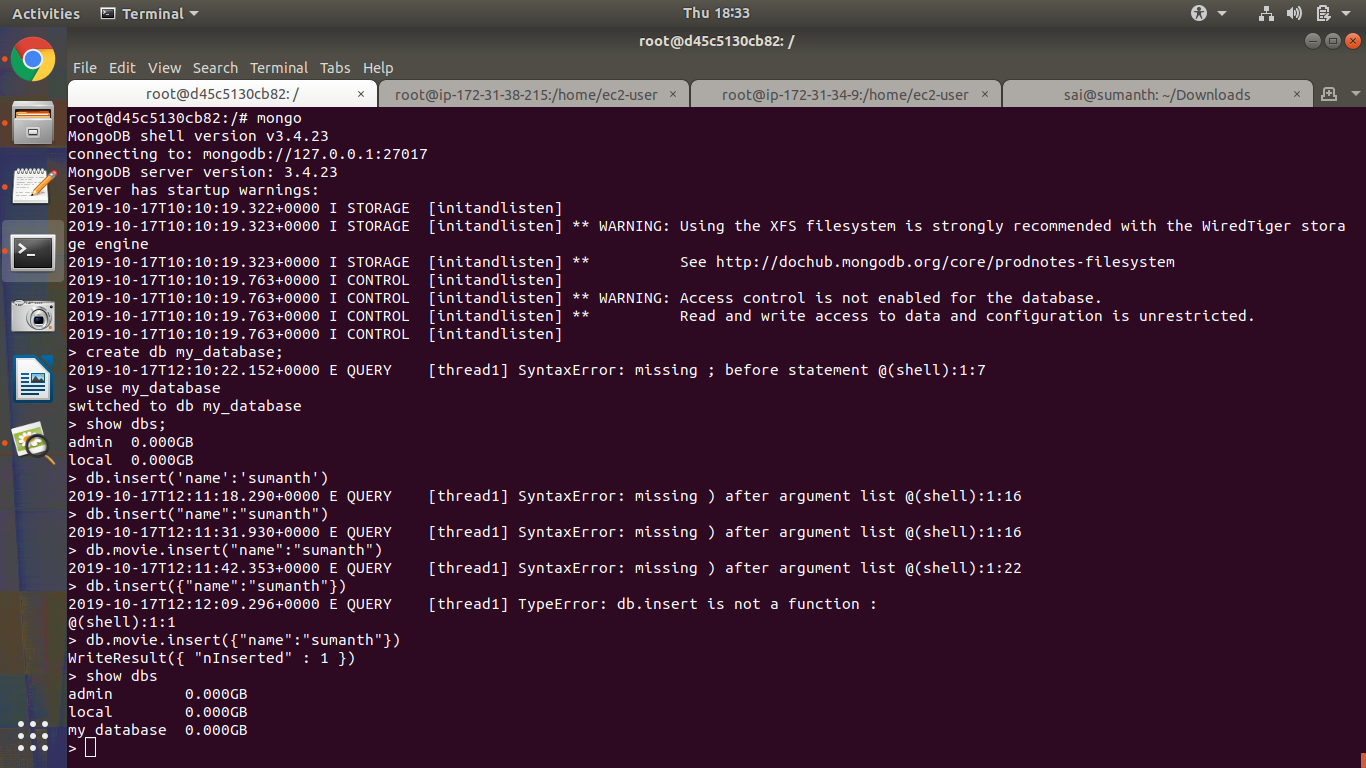
So use a database

use admin;

Create a user in that database.

*db.createUser({user : 'mongoadmin', pwd : '$adminpass', roles : ['root']});*

*db.auth(“mongoadmin”:”$adminpass”);*

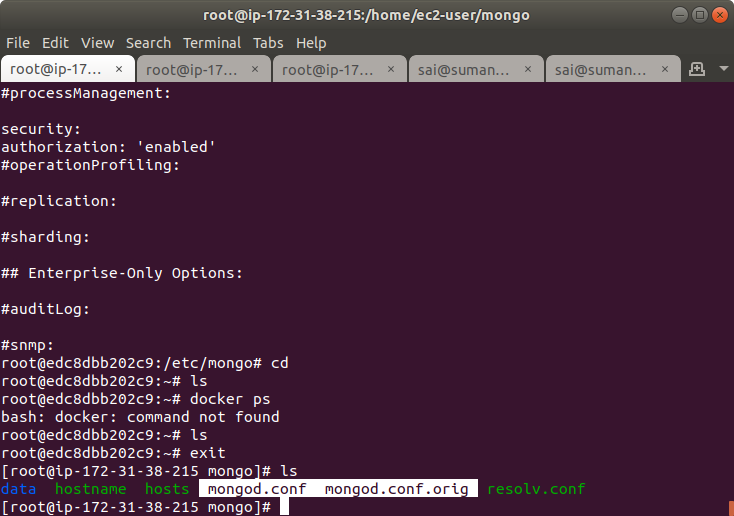


Create directory /etc/mongo in local system and in /mongo create a conf file and the file will transfer to docker container /etc/mongo

Because we mounted volume in yml file

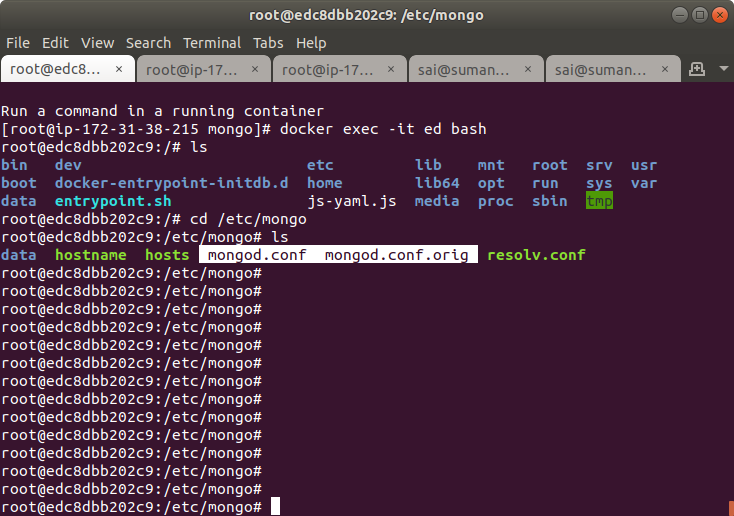
In ec2 instance docker container default conf file will be created as mongod.conf.orig in ~/etc

Here I have created a conf file in /etc/mongo in ec2 instance



And the data folder in ec2 instance is copied from docker container

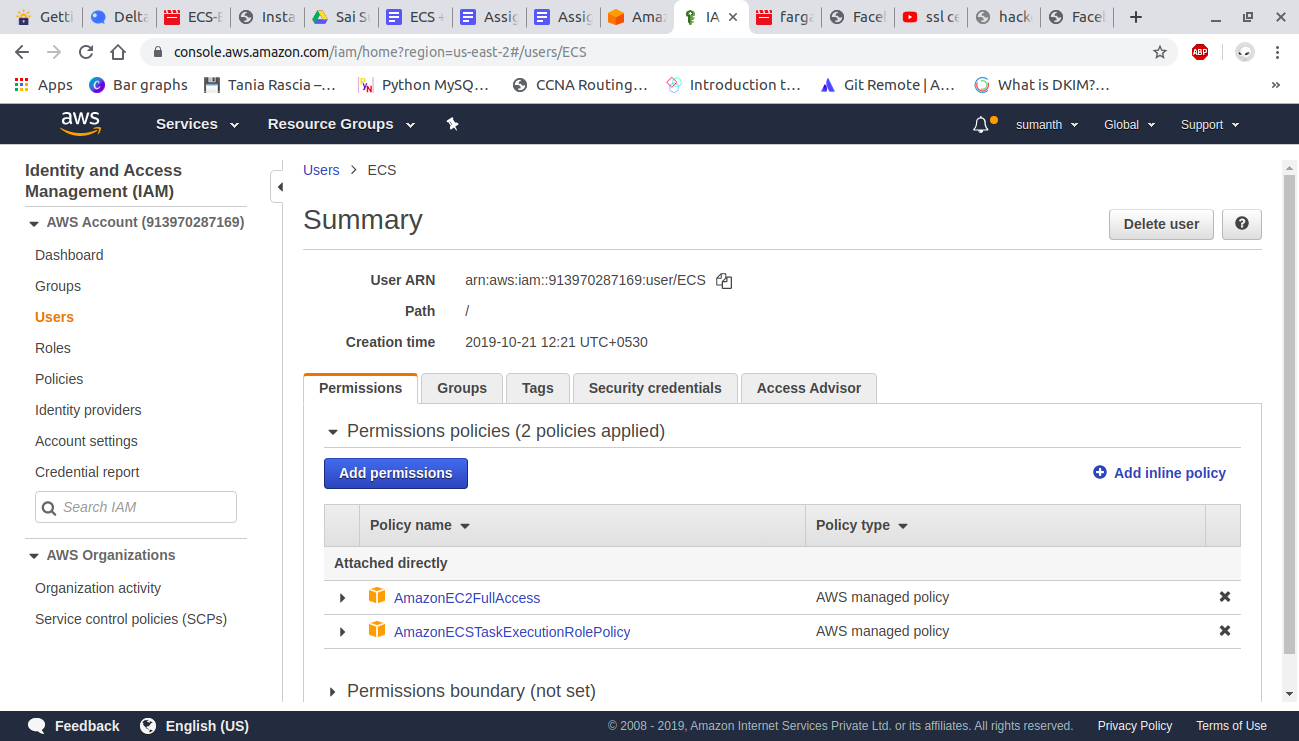
The conf file has been copied to docker container location because volume is mounted



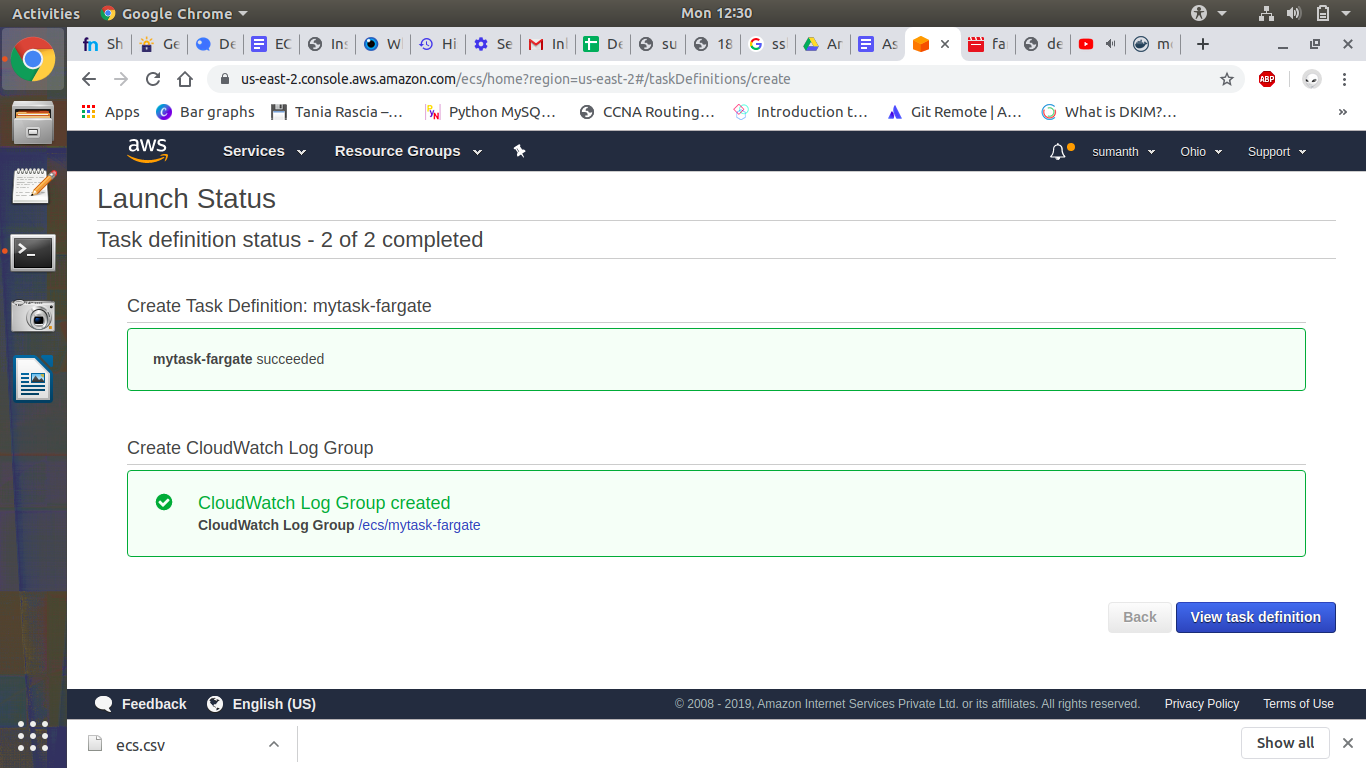
### **ECS:**

#### **ECS with fargate:**

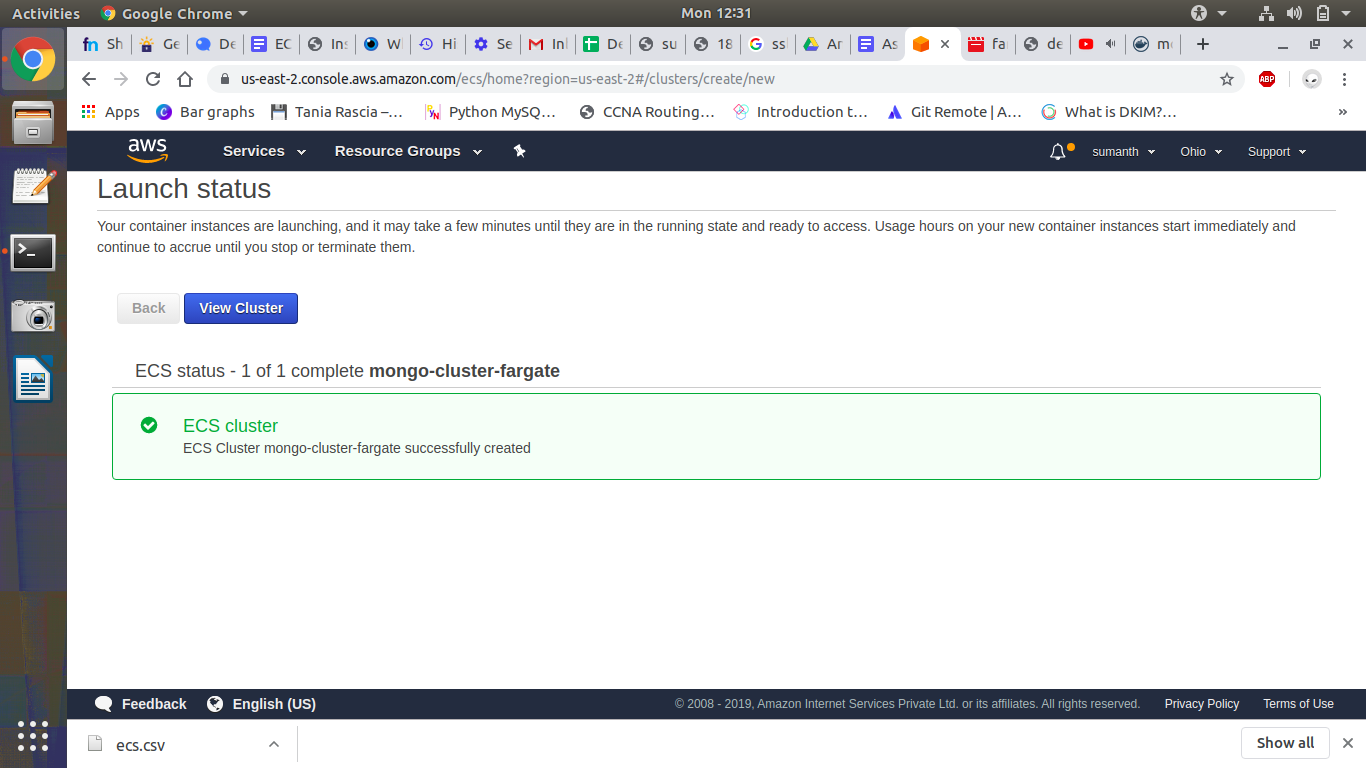
Create IAM user with ECS permissions



Create a task definition.

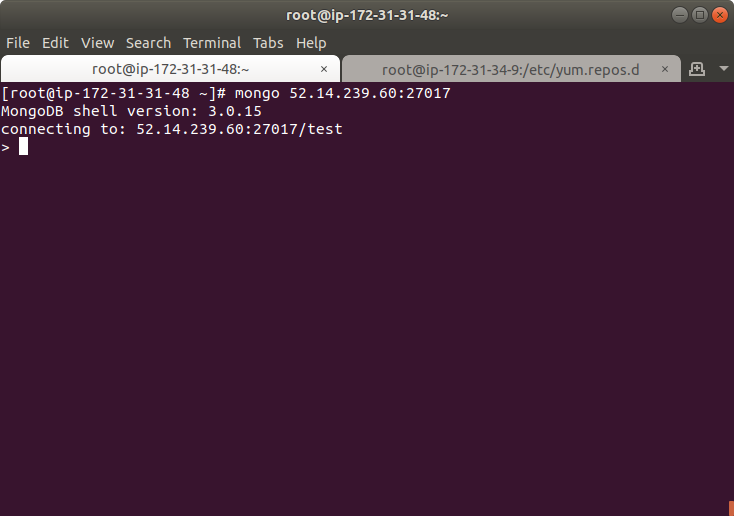


Create a cluster



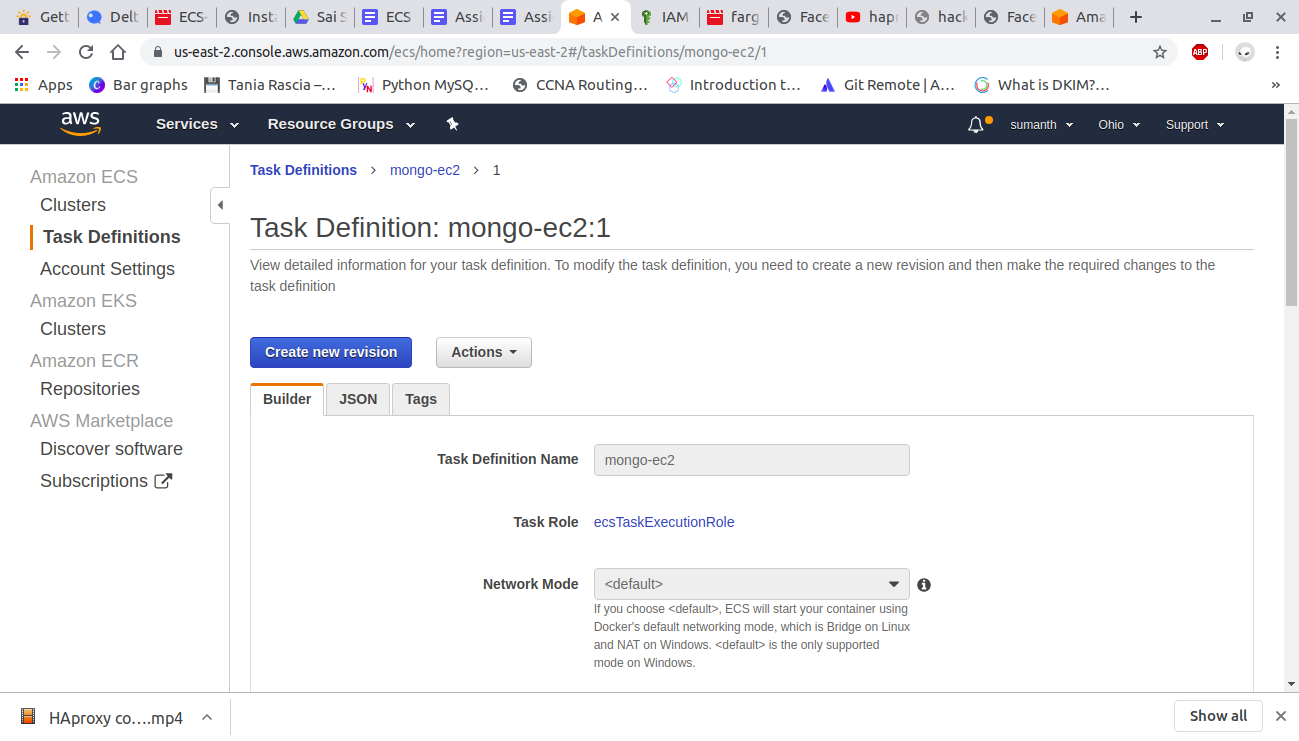
Create service and tasks

And in tasks open running tasks and copy the public ip address and open an ec2 instance and open mongo using -h and ip address.

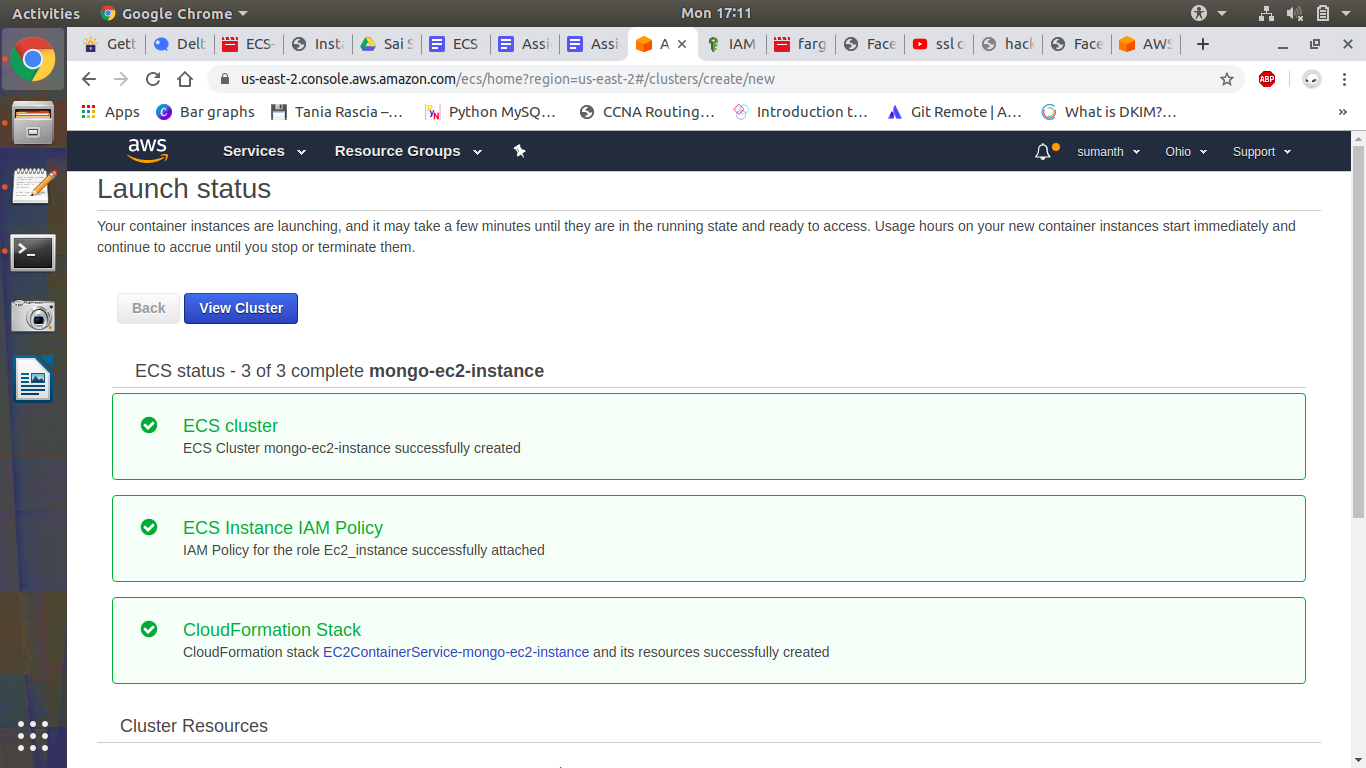


#### **ECS with EC2:**

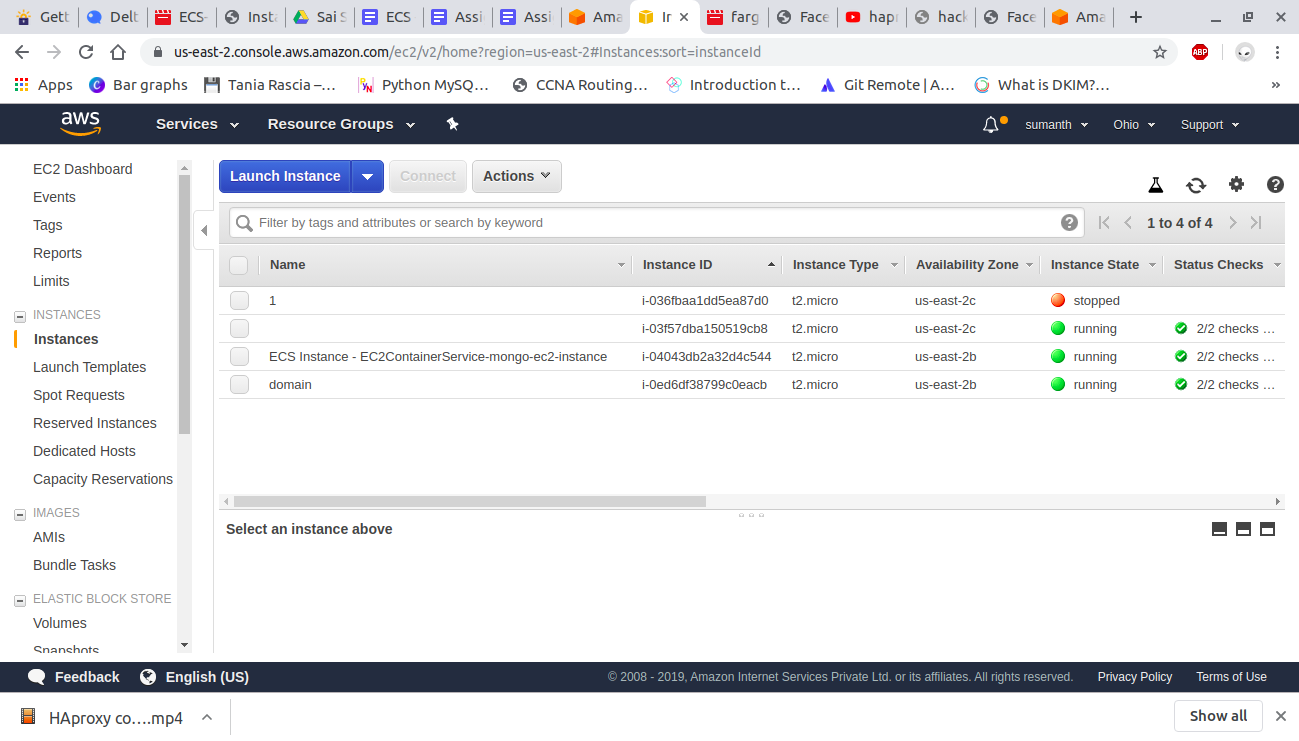
Create task definition:



Create a cluster

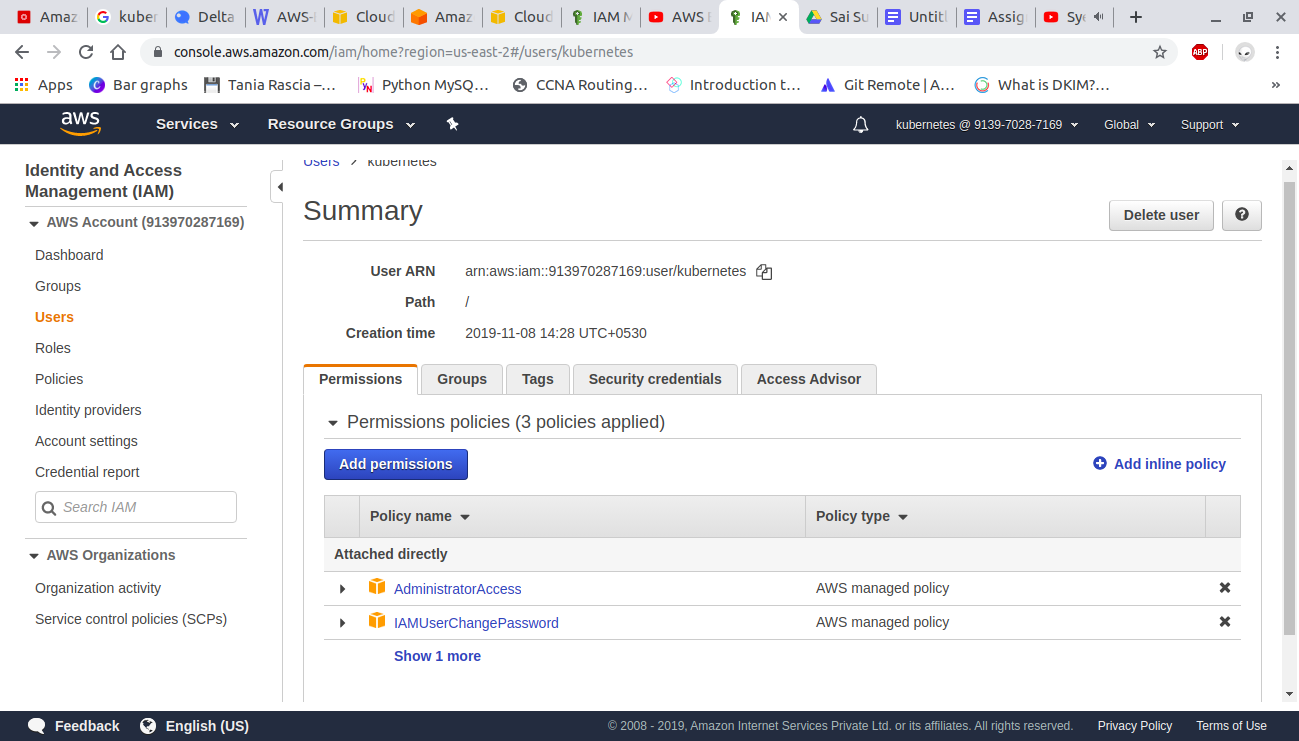


Ec2 instance is created.

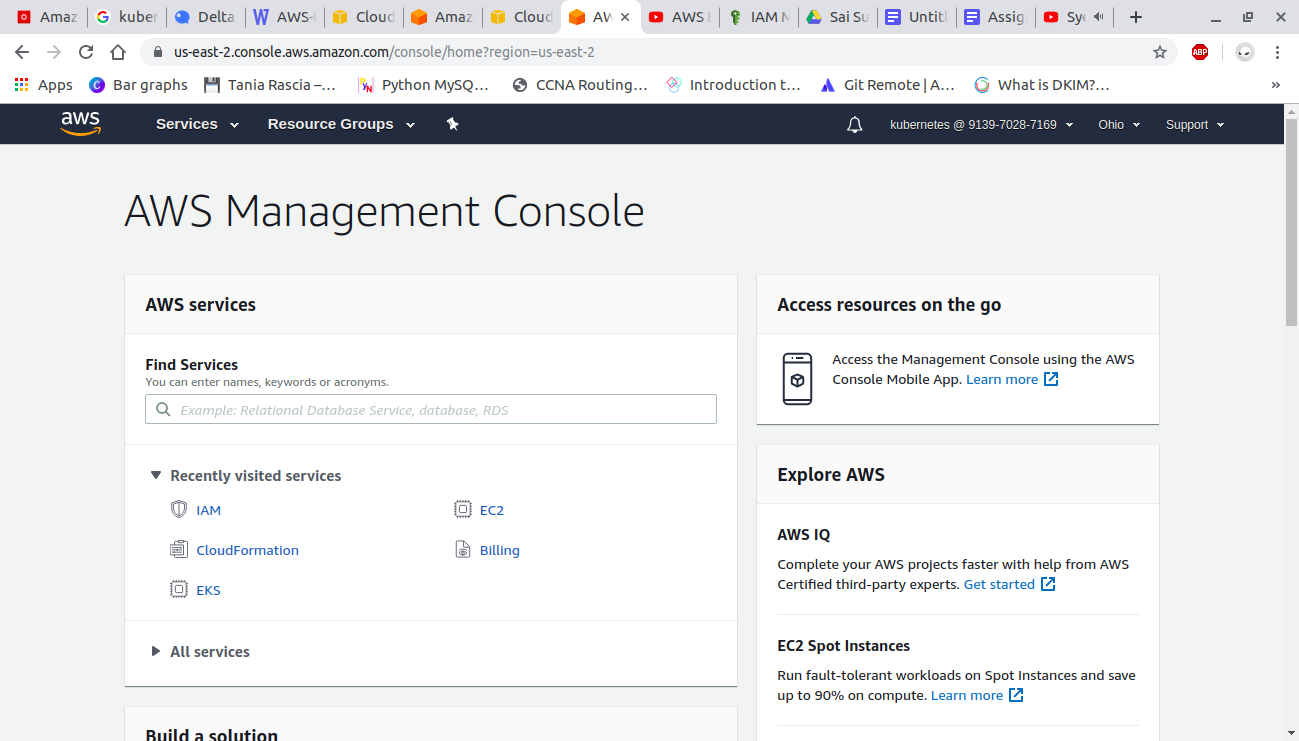


### **EKS:**

Create an IAM user



And login to aws console using that user



##### **Create a VPC :**

We can directly create it using vpc service or we can use cloudformation to create a vpc

To create vpc using cloudformation you have to write a yml file

Open AWS cloudformation console

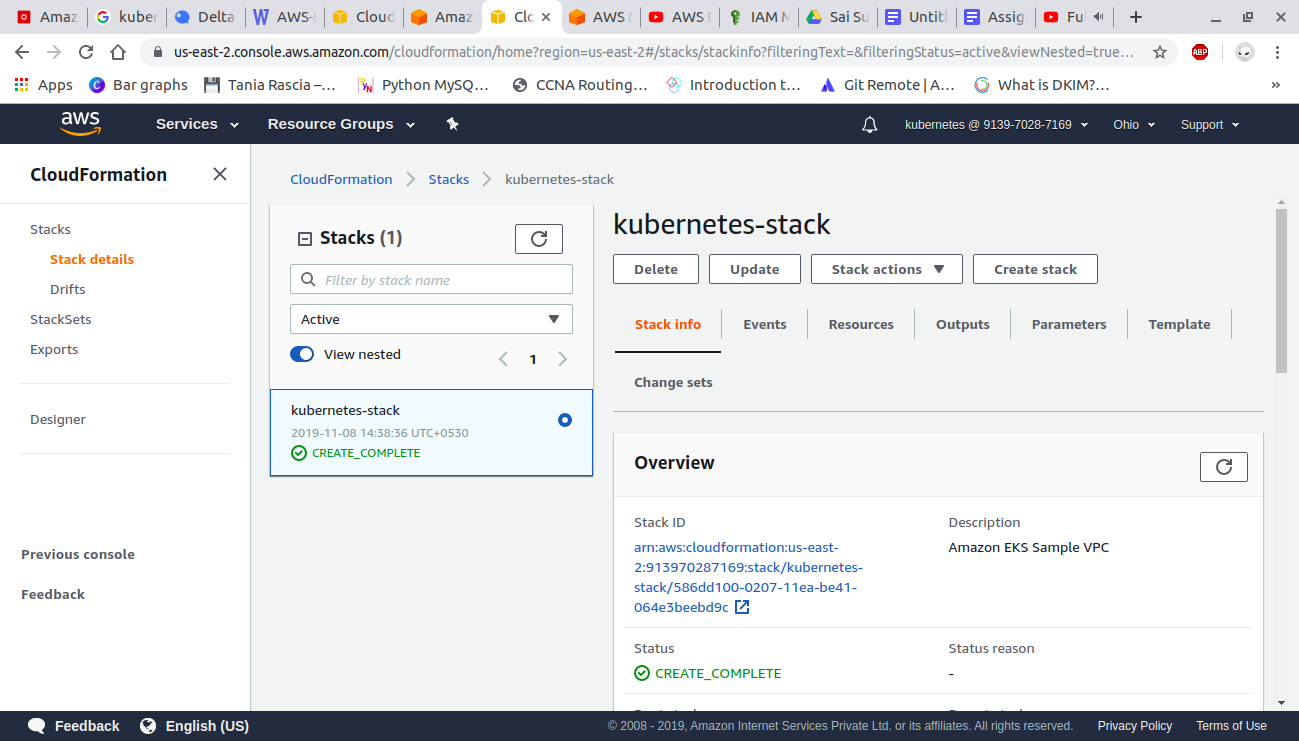
Choose create a stack

And in specify template paste the following url in amazon s3 url

*https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-06-05/amazon-eks-vpc-*

*sample.yaml*

Fill out the parameters accordingly and create a stack



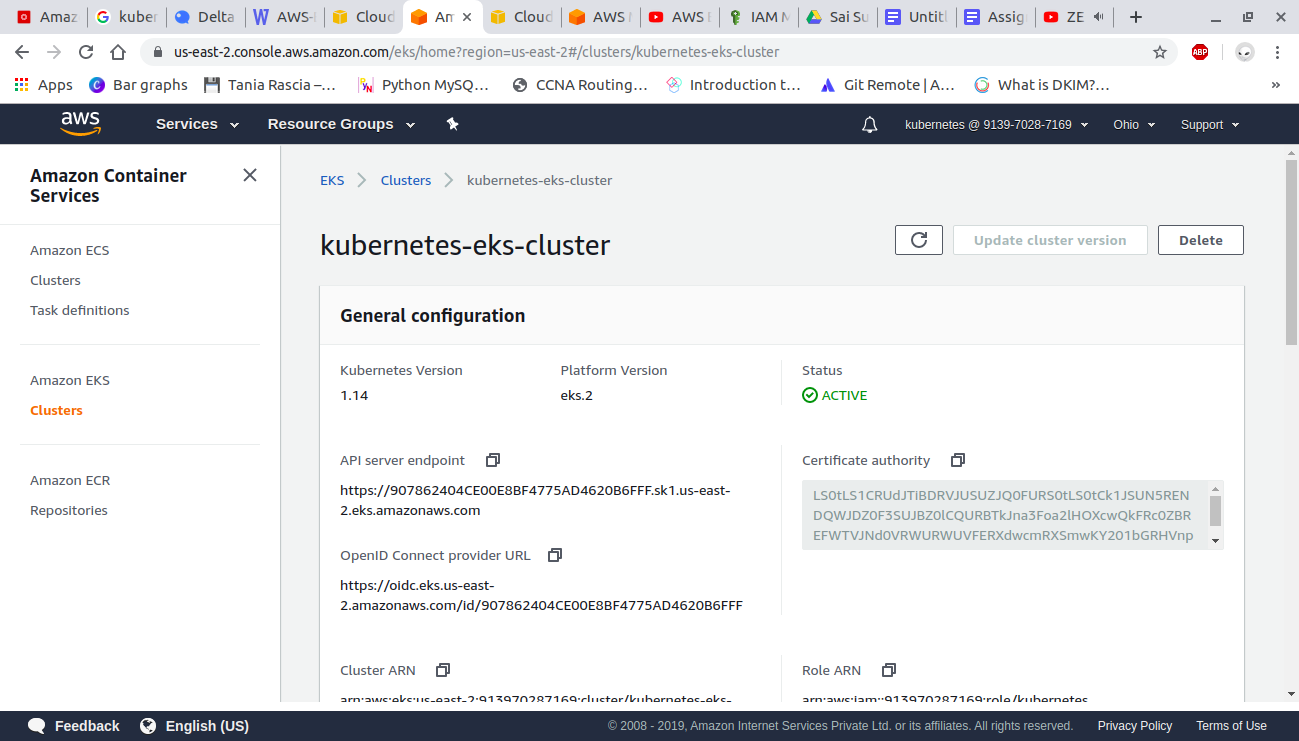
##### **Create EKS cluster:**

Open the amazon EKS console

Choose create cluster

On the Create cluster page, fill in the following fields and then choose Create:

* Cluster Name: A unique name for cluster
* Role name: Select the IAM role that you created
* VPC: Select VPC that you created
* Subnets: Select subnetId values which are recorded from vpc you created
* Security Groups: Select SG value recored from vpc

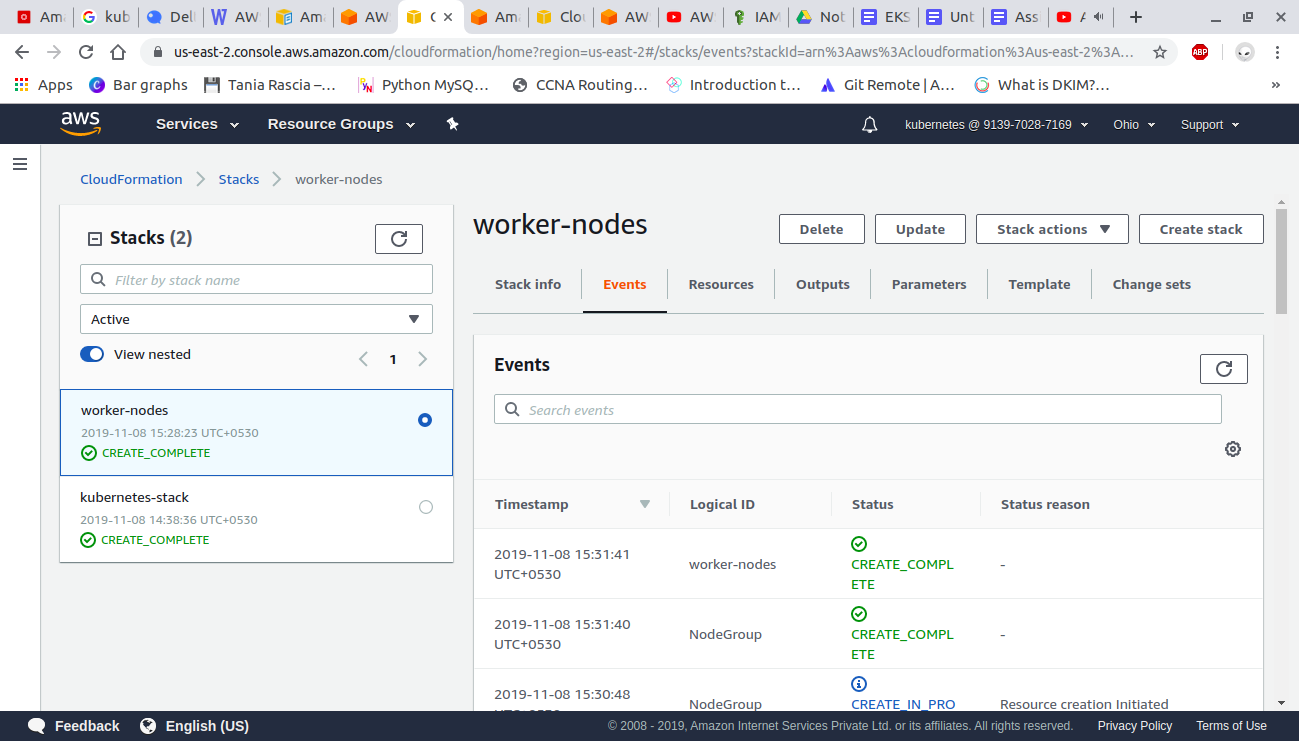


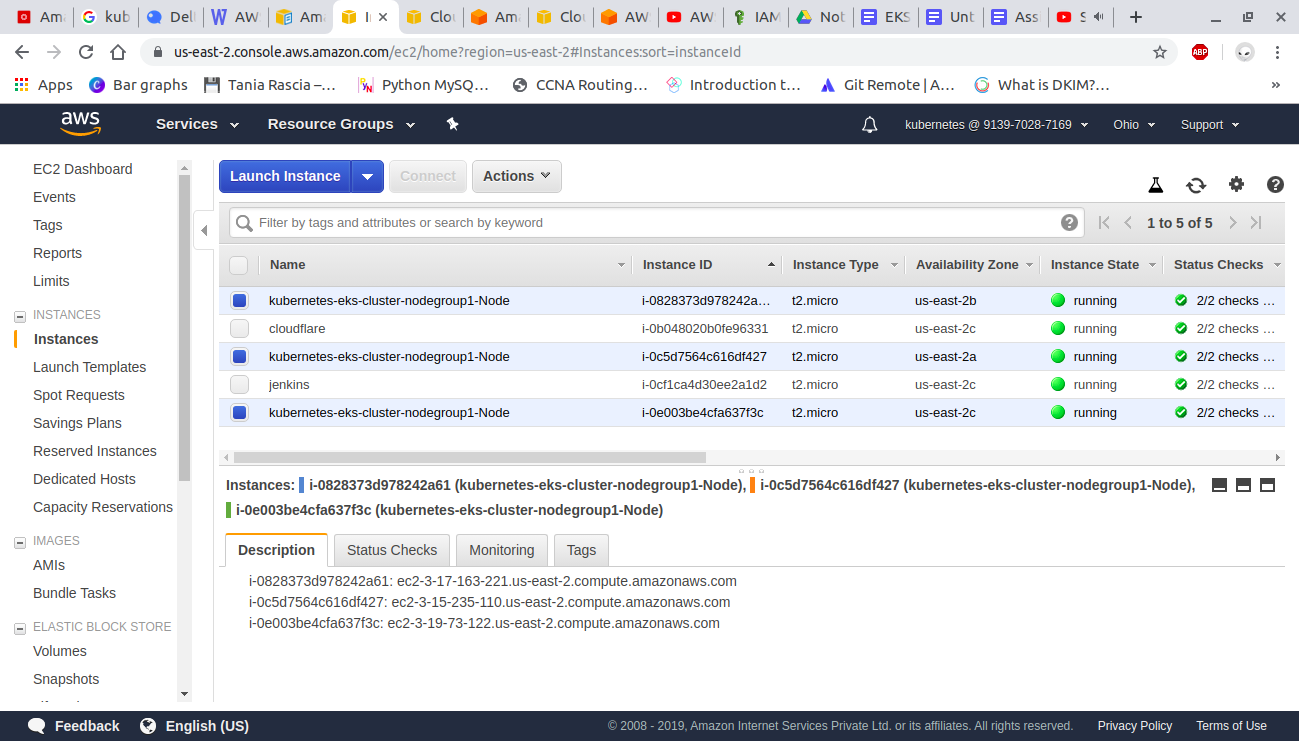
##### **Launch and configure EKS worker nodes:**

Open AWS cloudformation console and create a stack

yml file for creating worker nodes

Link - <https://docs.google.com/document/d/1KmQwMzfgyhJLzFLVesc8oxWPsw4-sMsBXBvZafKRX_0/edit?usp=sharing>





To install aws-iam-authenticator

Download the Amazon EKS-vended aws-iam-authenticator binary from Amazon S3:

*curl -o aws-iam-authenticator* [*https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/amd64/aws-iam-authenticator*](https://amazon-eks.s3-us-west-2.amazonaws.com/1.14.6/2019-08-22/bin/linux/amd64/aws-iam-authenticator)

Apply execute permissions to the binary.

*chmod +x ./aws-iam-authenticator*

Copy the binary to a folder in your $PATH. We recommend creating a $HOME/bin/aws-iam-authenticator and ensuring that $HOME/bin comes first in your $PATH.

*mkdir -p $HOME/bin && cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:$PATH*

Add $HOME/bin to your PATH environment variable.

*echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc*

Test that the aws-iam-authenticator binary works.

*aws-iam-authenticator help*

##### **Create, Launch and configure EKS VPC, cluster and worker nodes VIA EKSCTL utility:**

Refer documentation also : eksctl.io

Download and extract the latest release of eksctl with the following command.

*curl --silent --location "https://github.com/weaveworks/eksctl/releases/download/latest\_release/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp*

Move the extracted binary to /usr/local/bin.

*sudo mv /tmp/eksctl /usr/local/bin*

*eksctl version*

Step 2: Create Cluster

In your terminal,

*eksctl create cluster --name=demo-eks-test --nodes=2 --region=us-west-2*

It will create the cluster on aws within 10-15 minutes, please give the name as per your choice and

choose location accordingly.

Step 3: Delete Cluster

*eksctl delete cluster --region=us-west-2 --name=demo-eks-test*

Kubetcl Common Commands for Connectivity via Terminal:

*kubectl get nodes* #get-nodes

*kubectl create -f nginx.yaml*. #configuration-apply

*kubectl get services -o wide* #getall-services

*kubectl desribe svc nginx-deployment* #describe-service

*kubectl get deployment* #get-deployment-details

*kubectl get rs* #get-replica-details

kubectl describe deployment | grep Image #get-deployment-image

kubectl scale deployment.v1.apps/nginx-deployment --replicas=9 #scale-pods