**What is Amazon EKS?**

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

Amazon EKS runs Kubernetes control plane instances across multiple Availability Zones to ensure high availability. Amazon EKS automatically detects and replaces unhealthy control plane instances, and it provides automated version upgrades and patching for them.



Amazon EKS runs Kubernetes control plane instances across multiple Availability Zones to ensure high availability. Amazon EKS automatically detects and replaces unhealthy control plane instances, and it provides automated version upgrades and patching for them.

Amazon EKS is also integrated with many AWS services to provide scalability and security for your applications, including the following:

* Amazon ECR for container images
* Elastic Load Balancing for load distribution
* IAM for authentication
* Amazon VPC for isolation

Amazon EKS runs up-to-date versions of the open-source Kubernetes software, so you can use all the existing plugins and tooling from the Kubernetes community. Applications running on Amazon EKS are fully compatible with applications running on any standard Kubernetes environment, whether running in on-premises data centers or public clouds. This means that you can easily migrate any standard Kubernetes application to Amazon EKS without any code modification required.

Amazon EKS uses Amazon VPC network policies to restrict traffic between control plane components to within a single cluster. Control plane components for a cluster cannot view or receive communication from other clusters or other AWS accounts, except as authorized with Kubernetes RBAC policies.

This secure and highly-available configuration makes Amazon EKS reliable and recommended for production workloads.

## **How does Amazon EKS work?**


    How Amazon EKS works
   

Getting started with Amazon EKS is easy:

1. First, create an Amazon EKS cluster in the AWS Management Console or with the AWS CLI or one of the AWS SDKs.
2. Then, launch worker nodes that register with the Amazon EKS cluster. We provide you with an AWS CloudFormation template that automatically configures your nodes.
3. When your cluster is ready, you can configure your favorite Kubernetes tools (such as **kubectl**) to communicate with your cluster.
4. Deploy and manage applications on your Amazon EKS cluster the same way that you would with any other Kubernetes environment.

## **Benefits:-**

### High Availability

### Serverless option

### Secure

### Built with the Community

## **Use cases:-**

### Hybrid Deployment

### Machine Learning

### Web Applications

**Task:-**

For this task, we are going to deploy our pods of Wordpress and a Mysql database on the top of amazon EKS cluster.

Basically, what we are trying to do is:

* Create a user with admin access
* Create a cluster using eksctl.
* Integrate EKS with EC2,ELB,EBS,EFS.
* Create an EFS that has the ability to attach more than one container/pods.
* Create deployment of the application WordPress and mysql
* Create a secret that contains wordpress root and mysql db password.

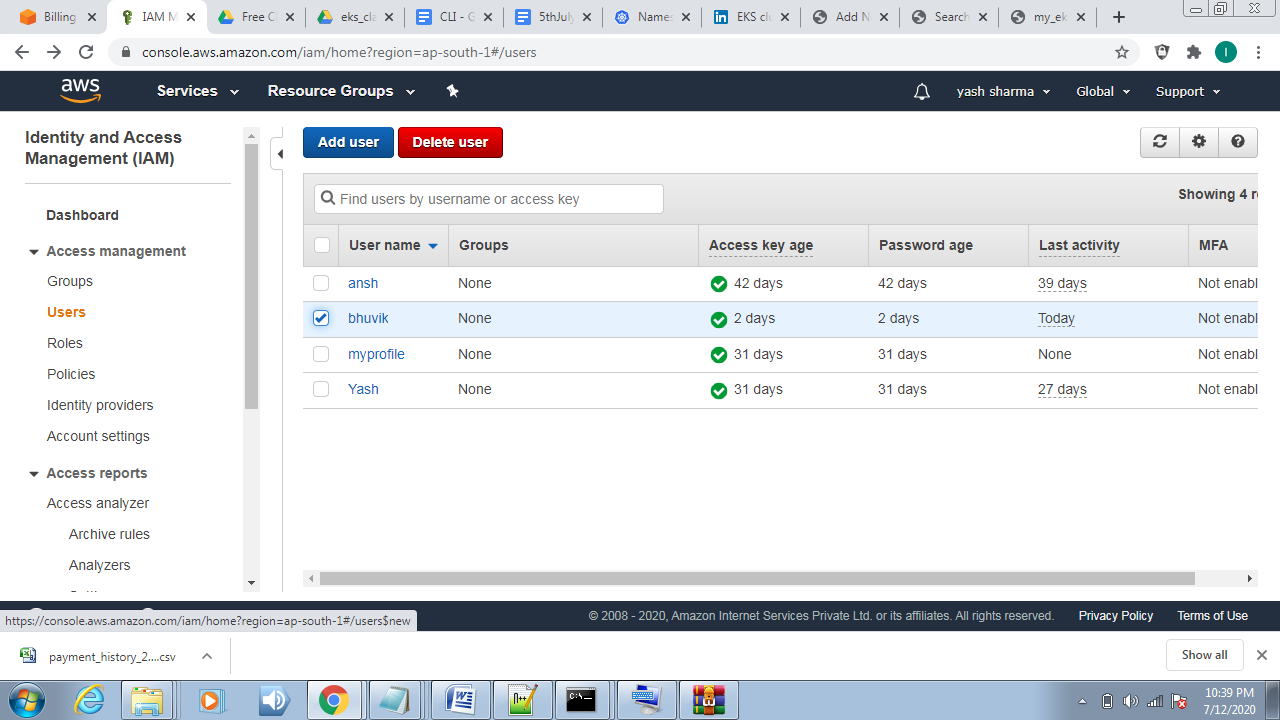
**Prerequisite:-**

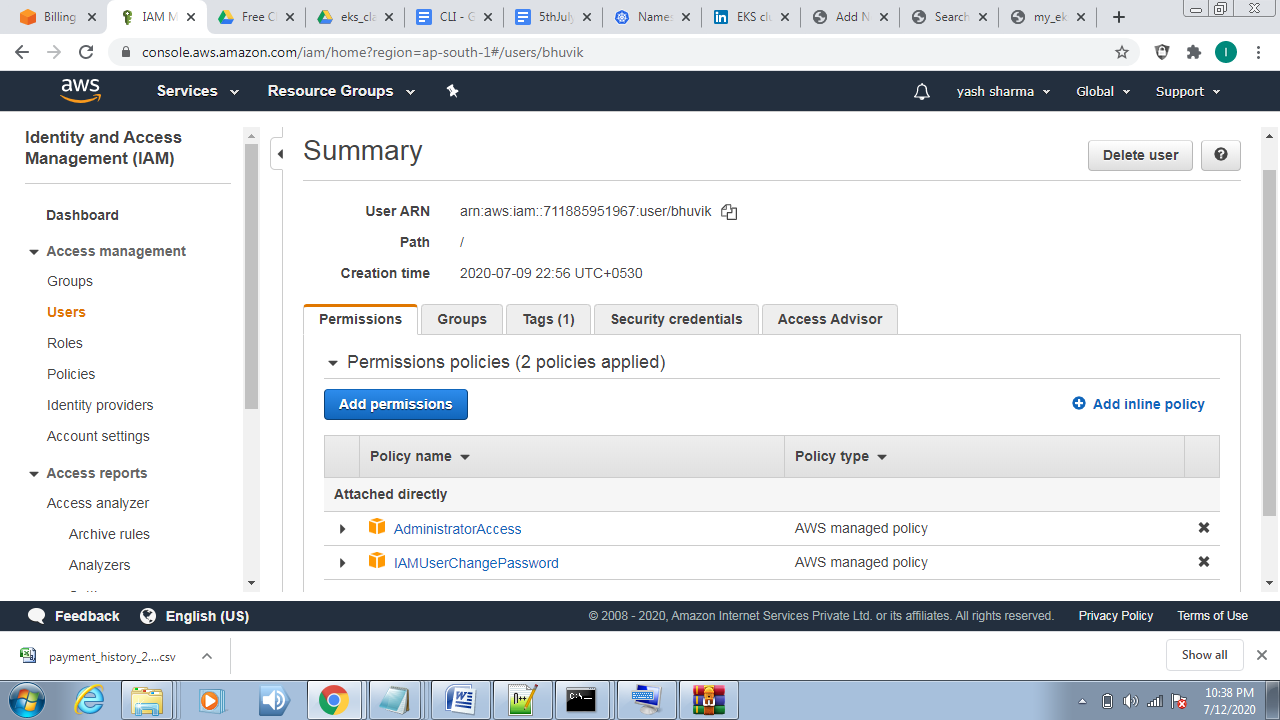
* AWS account
* AWS CLI
* Eksctl
* Basic of k8s
* Kubectl

**Lets start:**

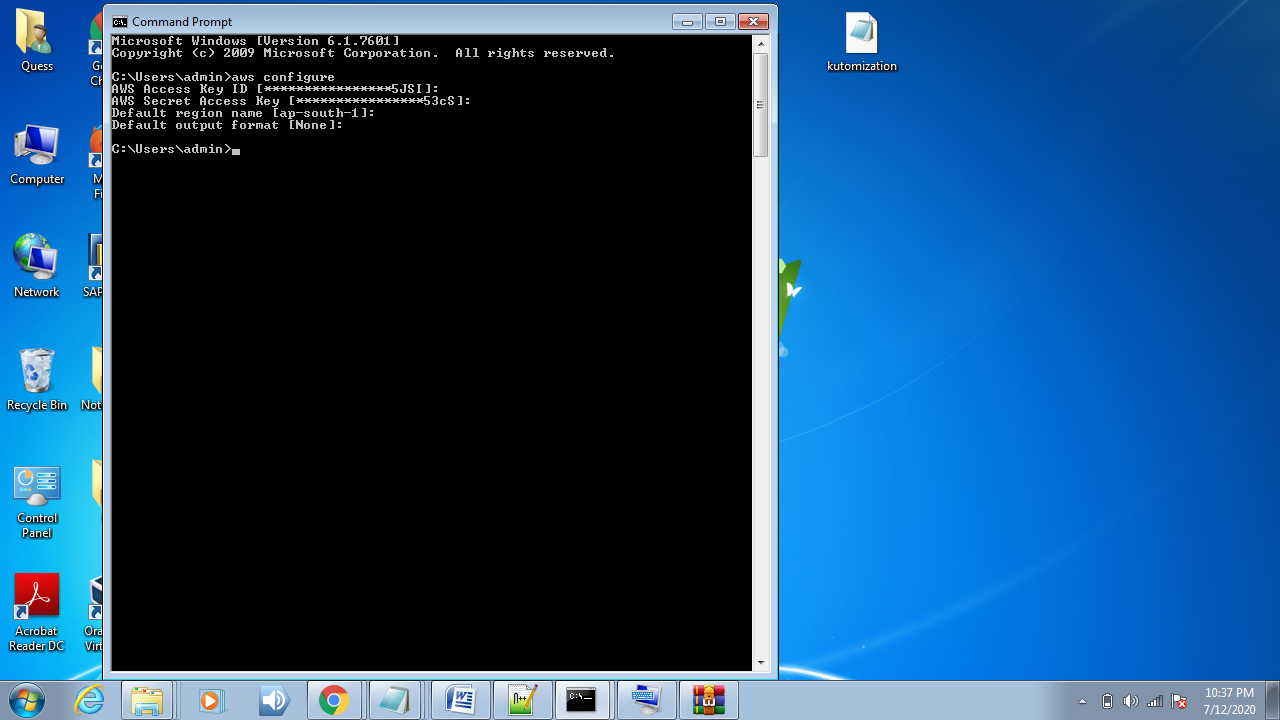
**Step 1.-** **Create IAM User having administration power.**

[ aws login 🡪 services🡪 IAM🡪 user 🡪Add User🡪add power]

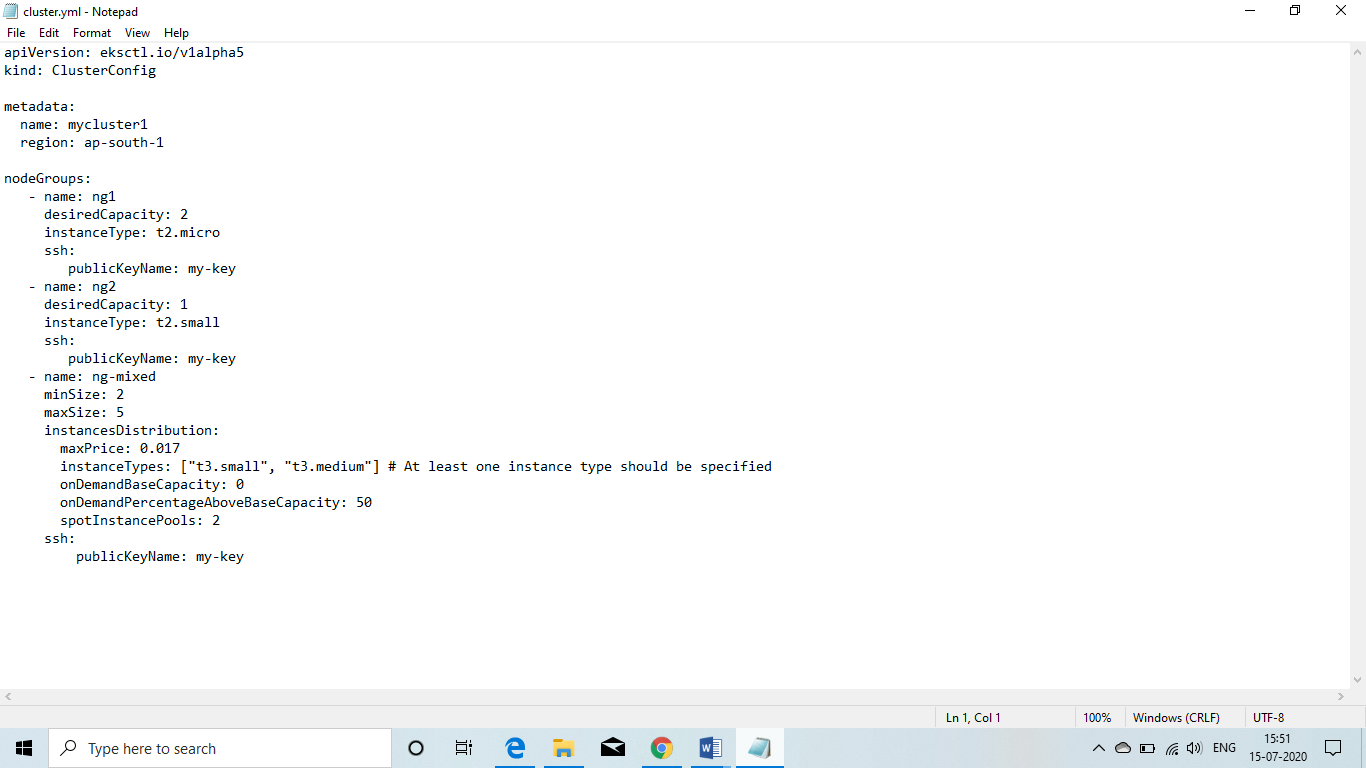




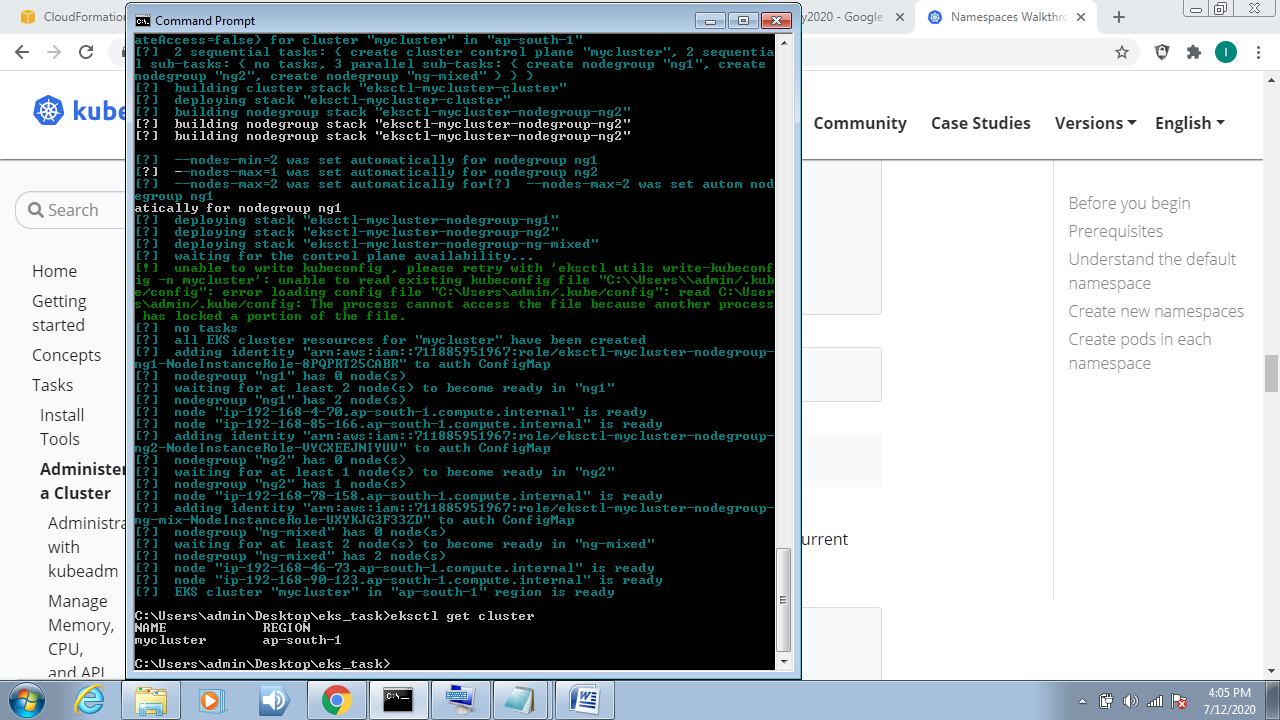
**Step 2.** **Configure the aws** :- Type Command:- aws configure



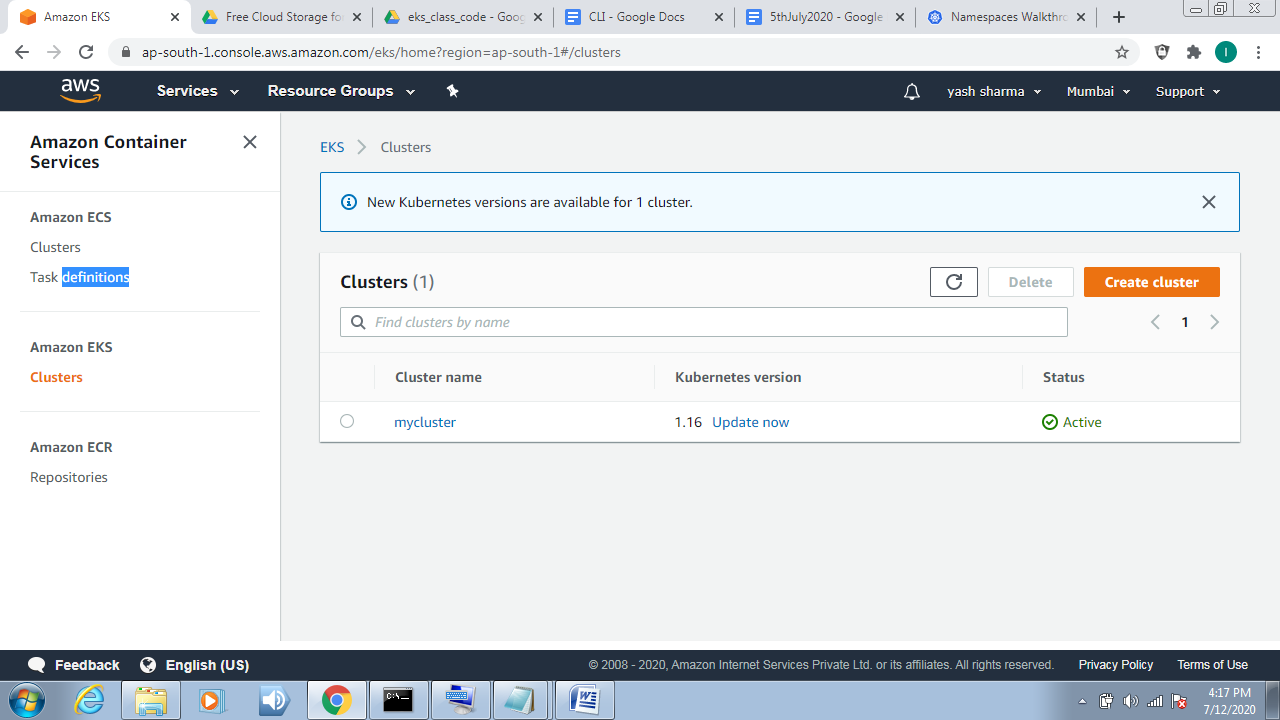
**Step 3. Create Cluster:-** Create cluster.yml (This file create cluster)



* Then type command:- eksctl create cluster -f cluster.yml



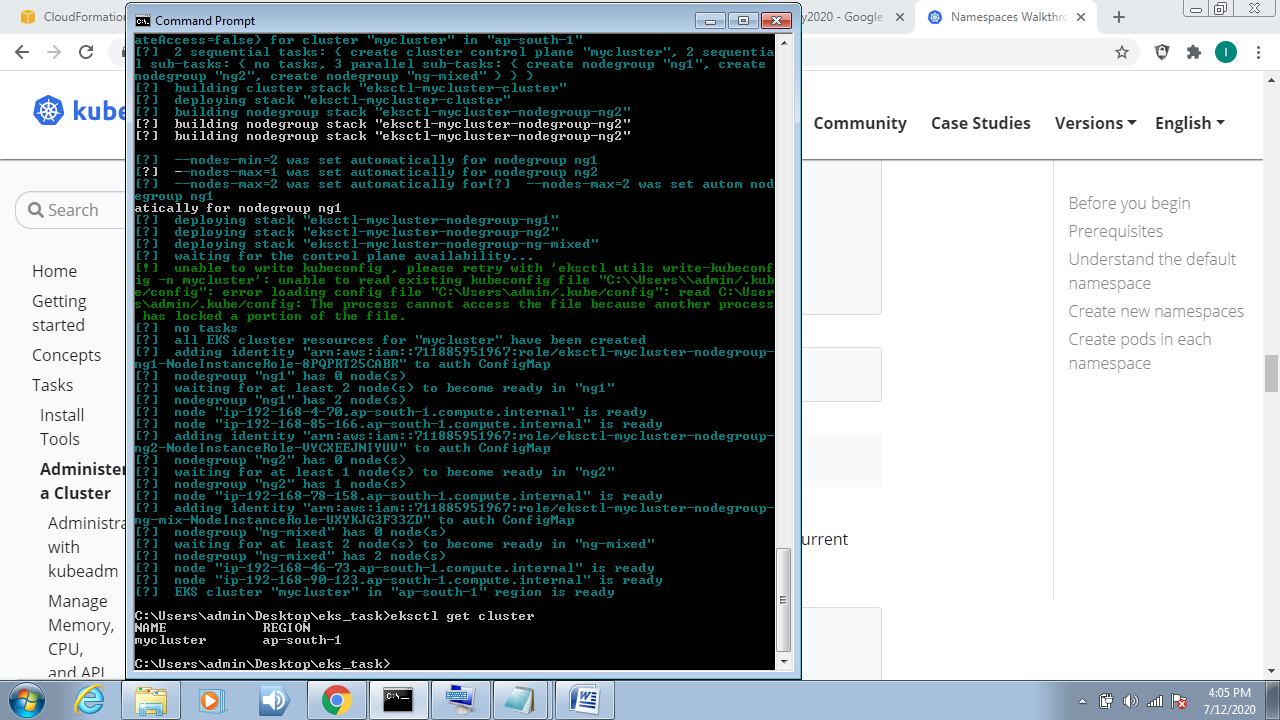
\*Cluster has been created



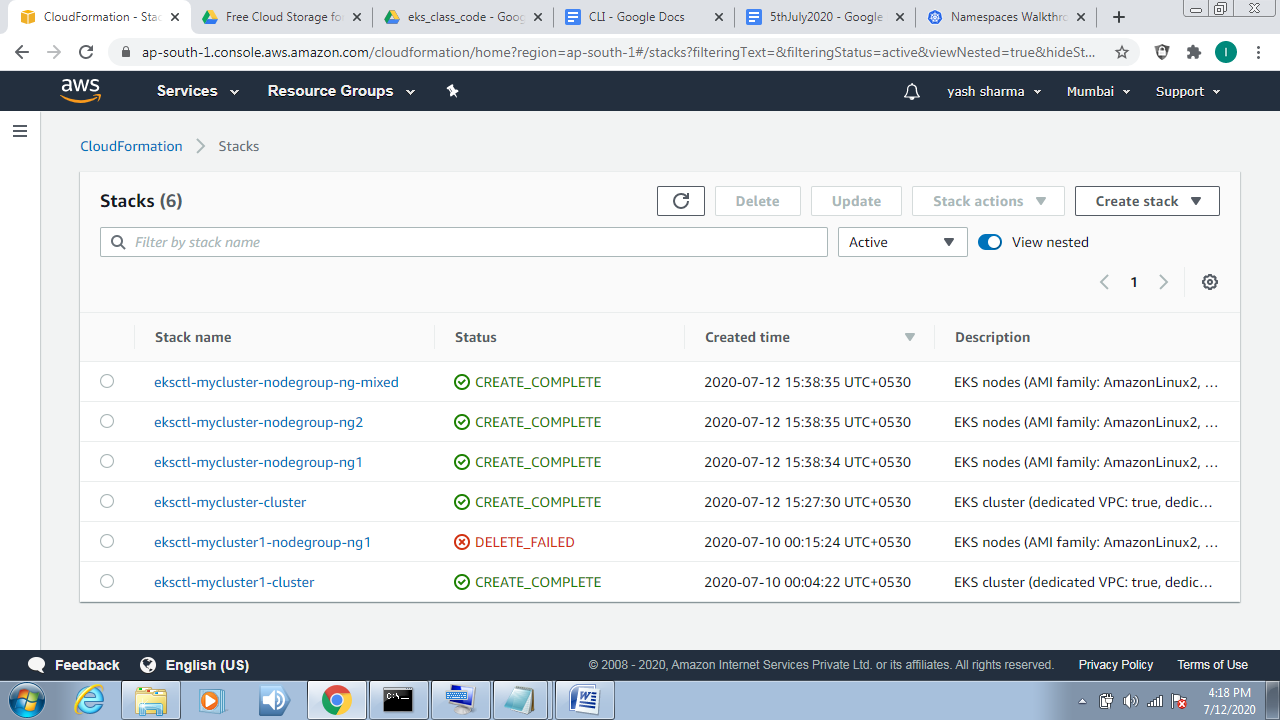
**Step4 Update config file :-** aws eks update-kubeconfig –name cluster\_name



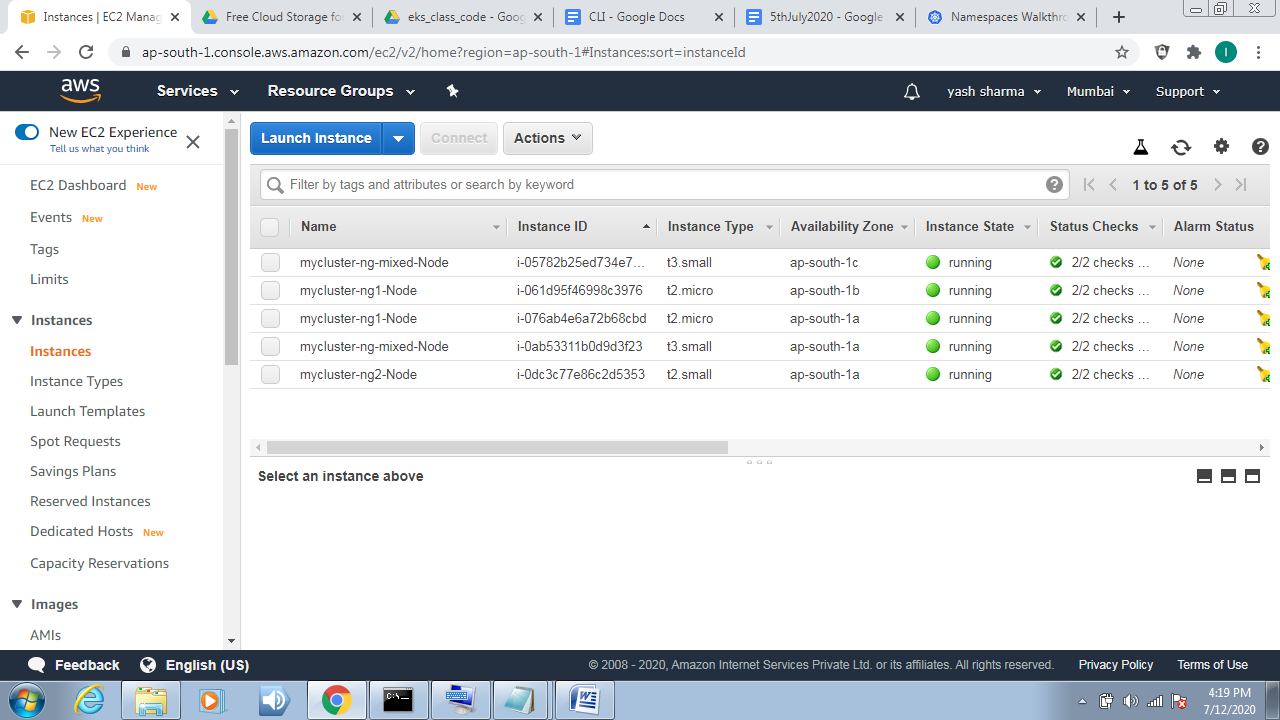
**Step5. Access Cluster:-**



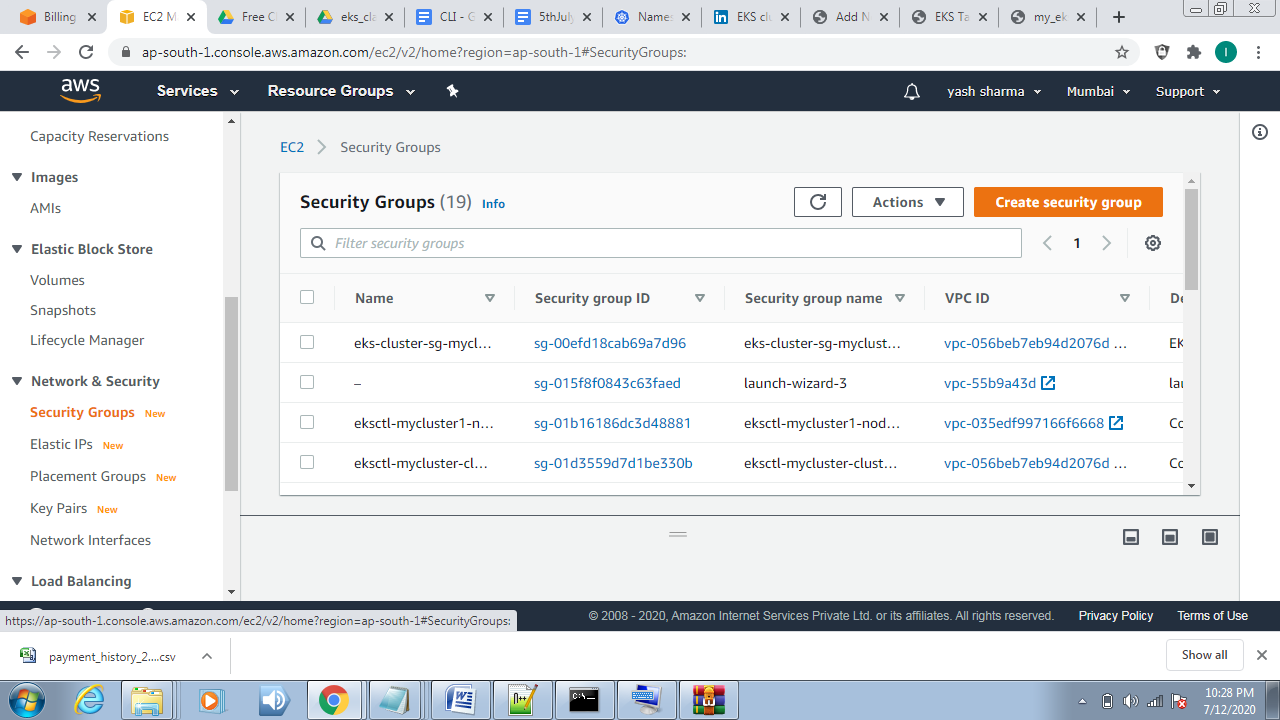
* **CloudFormation>Stacks:-**

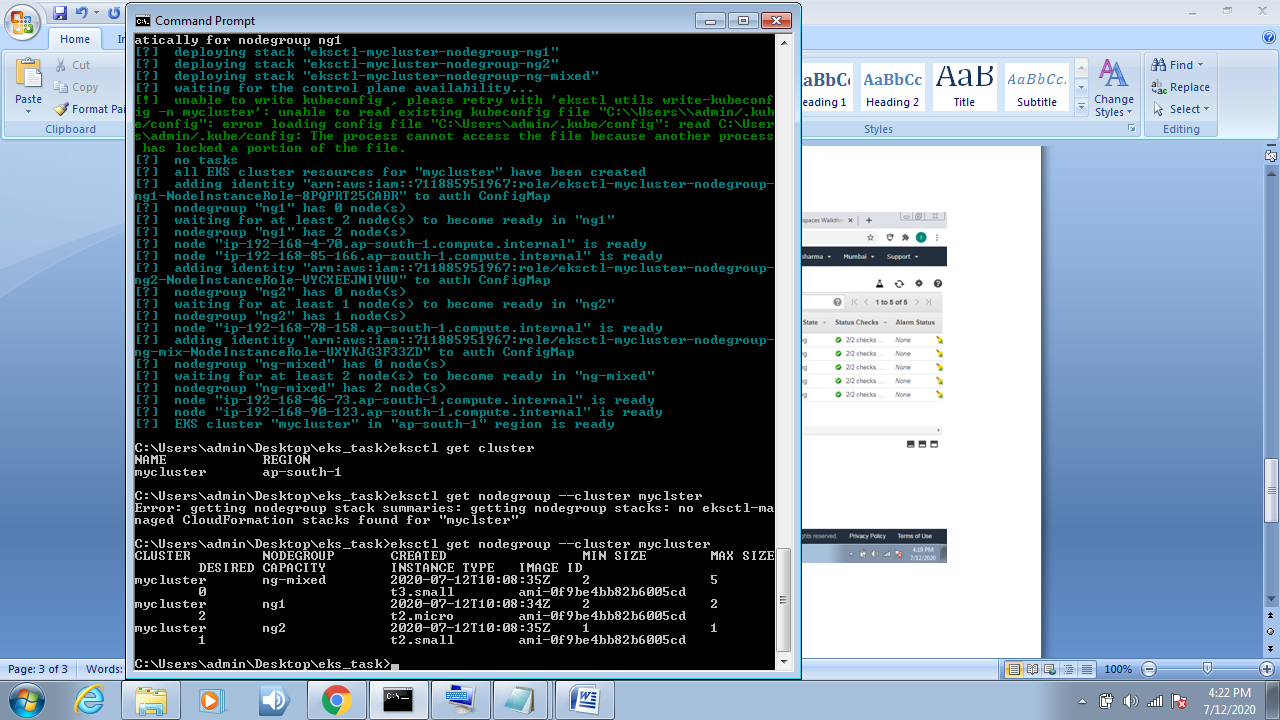


* **Instances Created**

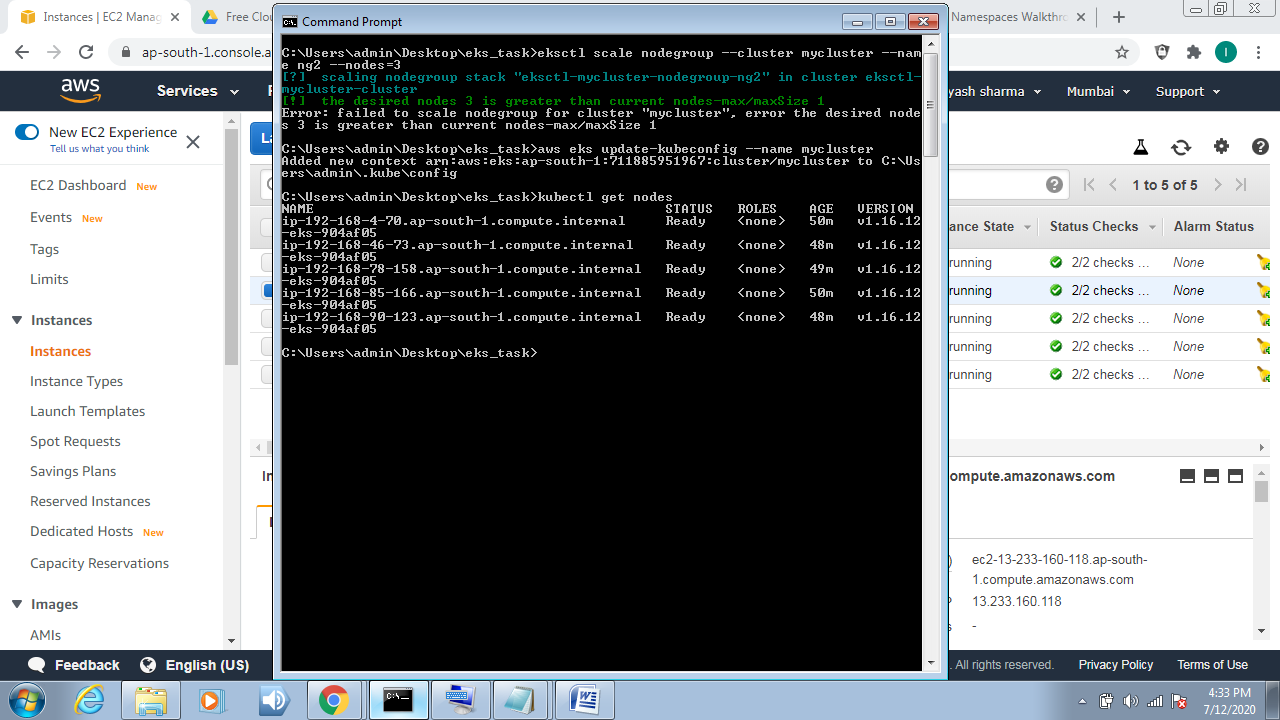


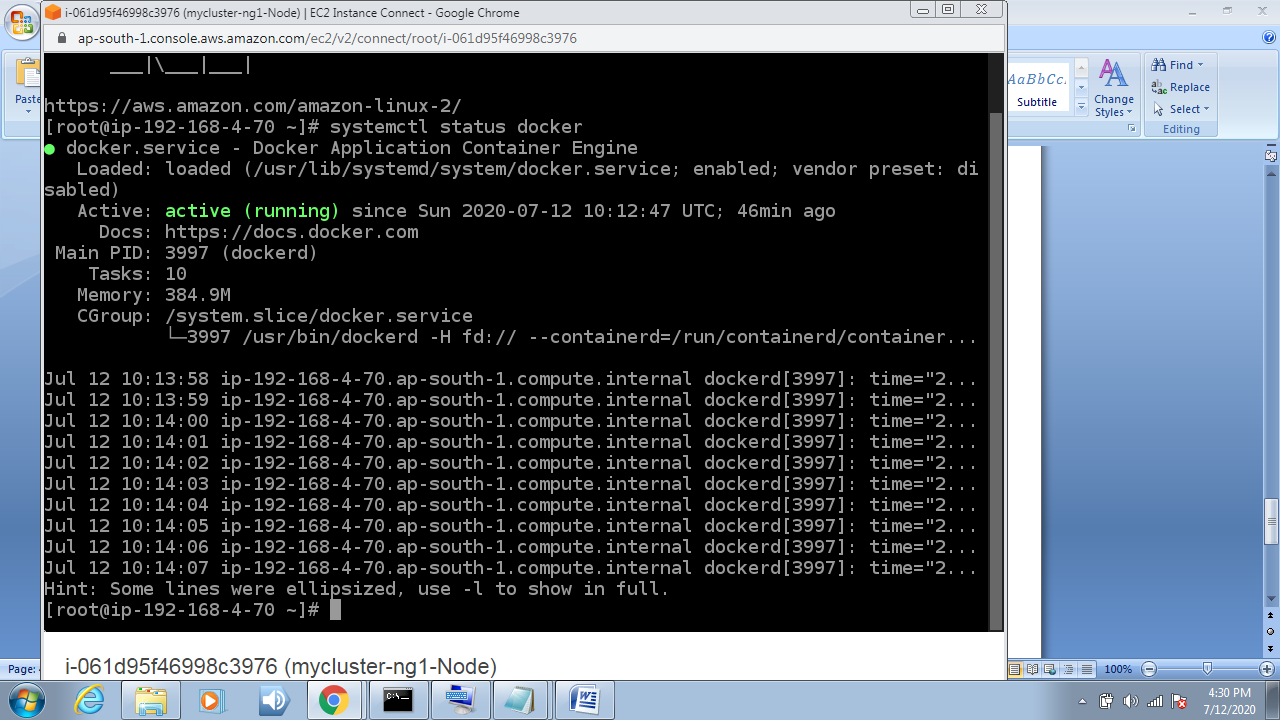
* **Security group**

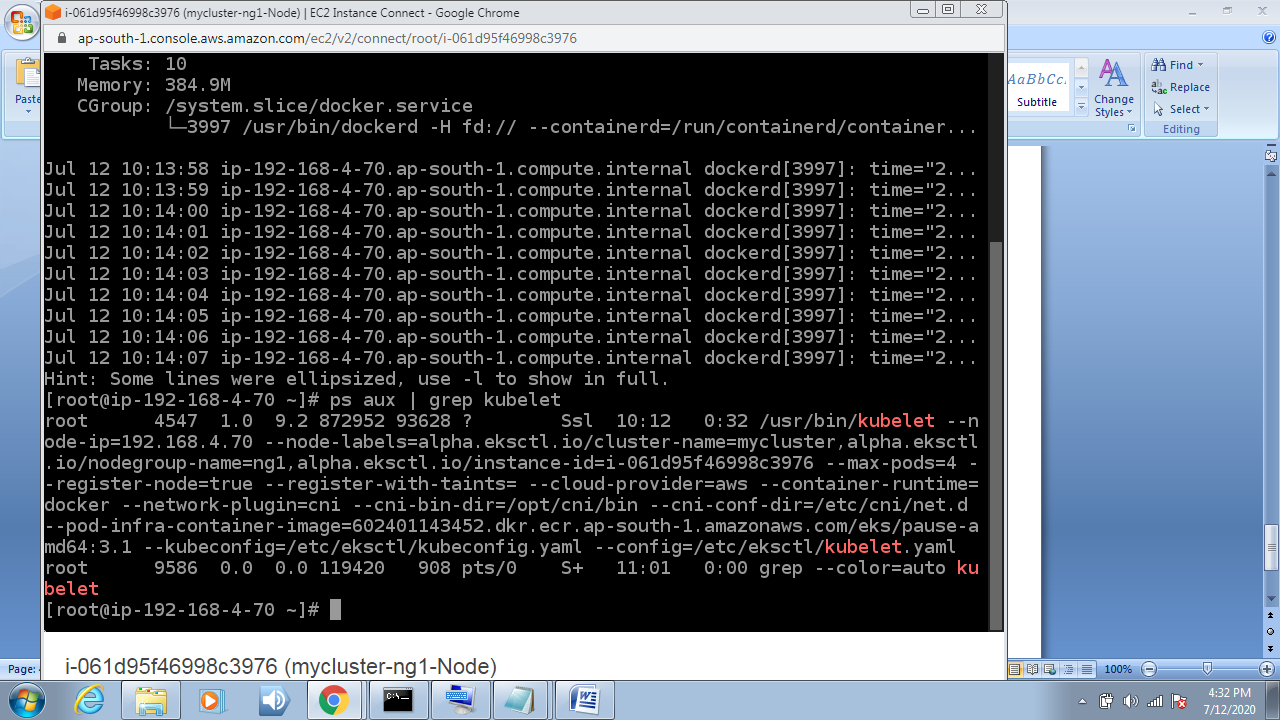


* **View your Kubernetes config file using:-** kubectl config view
* **Command:**-eksctl get nodegroup --cluster mycluster

**To get nodes🡪 command🡪** kubectl get nodes







**Step6.** In this we create secret file for store our user name and password

**Kubectl create secret generic <your\_secret\_name>**

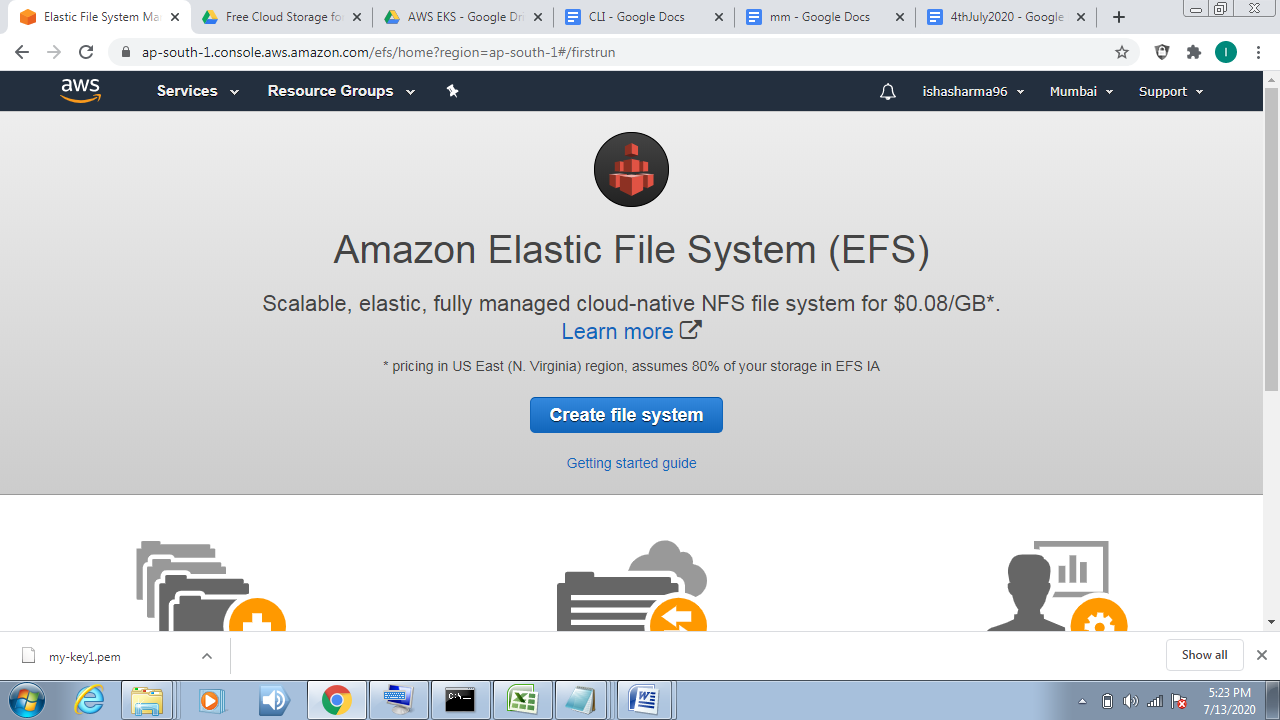
**--from –literal=password=<your\_password>**

**Step7. Going to create EFS .**

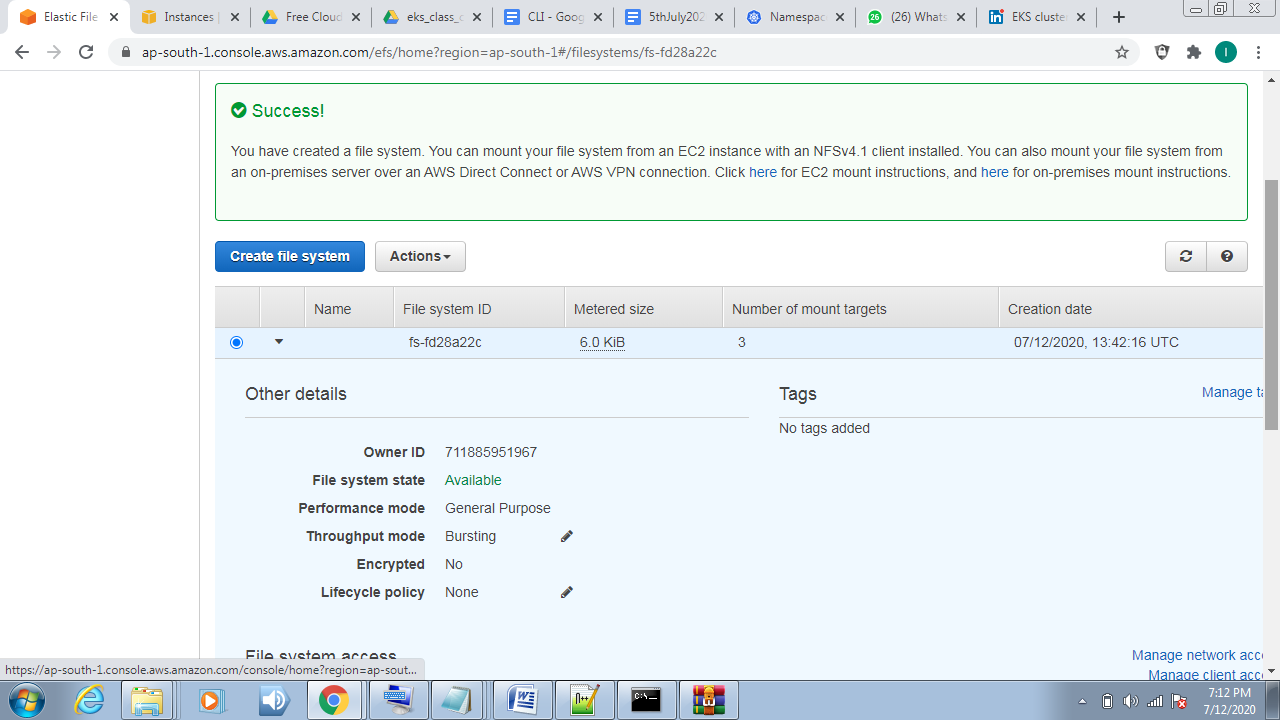
**“Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth.”**

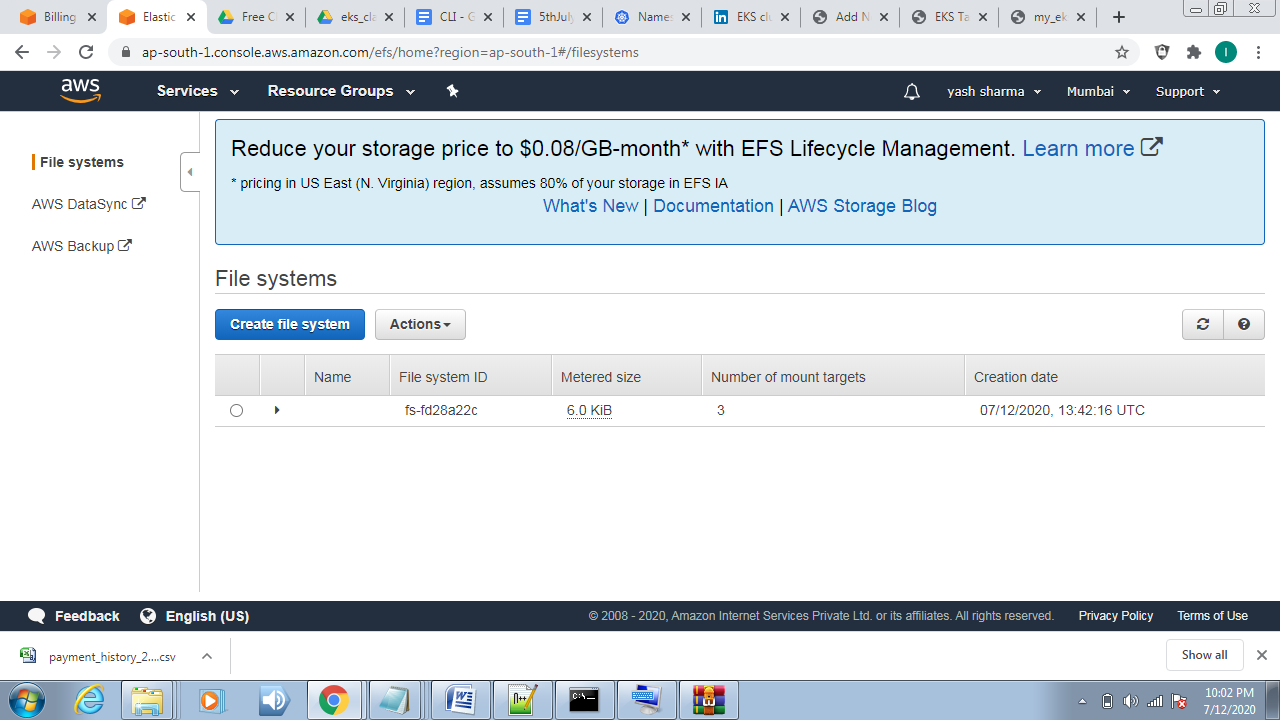
aws console -> Services -> EFS -> Create EFS

In this you have to config security group very carefully ,first choose the VPC of your cluster then choose the security group accordingly. Do not go for default VPC.









Now we gonna launch the MySQL and Wordpress for this we have to again write the yaml code.But with this we want that our data must be permanent in case if our pods goes down we don't want to loose our data . So for this we have to create the PVC.

**Step 8. Create mysql pod (**for this we create mysql-deployment.yaml file)

**CODE:-**

apiVersion: v1

kind: Service

metadata:

name: wordpress-mysql

labels:

app: wordpress

spec:

ports:

- port: 3306

selector:

app: wordpress

tier: mysql

clusterIP: None

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pv-claim

labels:

app: wordpress

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 10Gi

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: wordpress-mysql

labels:

app: wordpress

spec:

selector:

matchLabels:

app: wordpress

tier: mysql

strategy:

type: Recreate

template:

metadata:

labels:

app: wordpress

tier: mysql

spec:

containers:

- image: mysql:5.6

name: mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pass

key: password

ports:

- containerPort: 3306

name: mysql

volumeMounts:

- name: mysql-persistent-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-persistent-storage

persistentVolumeClaim:

claimName: mysql-pv-claim

**Step 9. Create WordPress pod (**for this we create wordpress-deployment.yaml**)**

**CODE:-**

apiVersion: v1

kind: Service

metadata:

name: wordpress

labels:

app: wordpress

spec:

ports:

- port: 80

selector:

app: wordpress

tier: frontend

type: LoadBalancer

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: wp-pv-claim

labels:

app: wordpress

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 5Gi

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: wordpress

labels:

app: wordpress

spec:

selector:

matchLabels:

app: wordpress

tier: frontend

strategy:

type: Recreate

template:

metadata:

labels:

app: wordpress

tier: frontend

spec:

containers:

- image: wordpress:4.8-apache

name: wordpress

env:

- name: WORDPRESS\_DB\_HOST

value: wordpress-mysql

- name: WORDPRESS\_DB\_PASSWORD

valueFrom:

secretKeyRef:

name: mysql-pass

key: password

ports:

- containerPort: 80

name: wordpress

volumeMounts:

- name: wordpress-persistent-storage

mountPath: /var/www/html

volumes:

- name: wordpress-persistent-storage

persistentVolumeClaim:

claimName: wp-pv-claim

**Step 10. Now both wordpress\_deployment and mysql-deployment has been created . Now we shall create kustomization.yml file as it let us deploy whole setup with just one command and few other functionalities.**

**Kustomization.yml**

apiVersion: kustomize.config.k8s.io/v1beta1

kind: Kustomization

secretGenerator:

- name: mysql-pass

literals:

- password=redhat

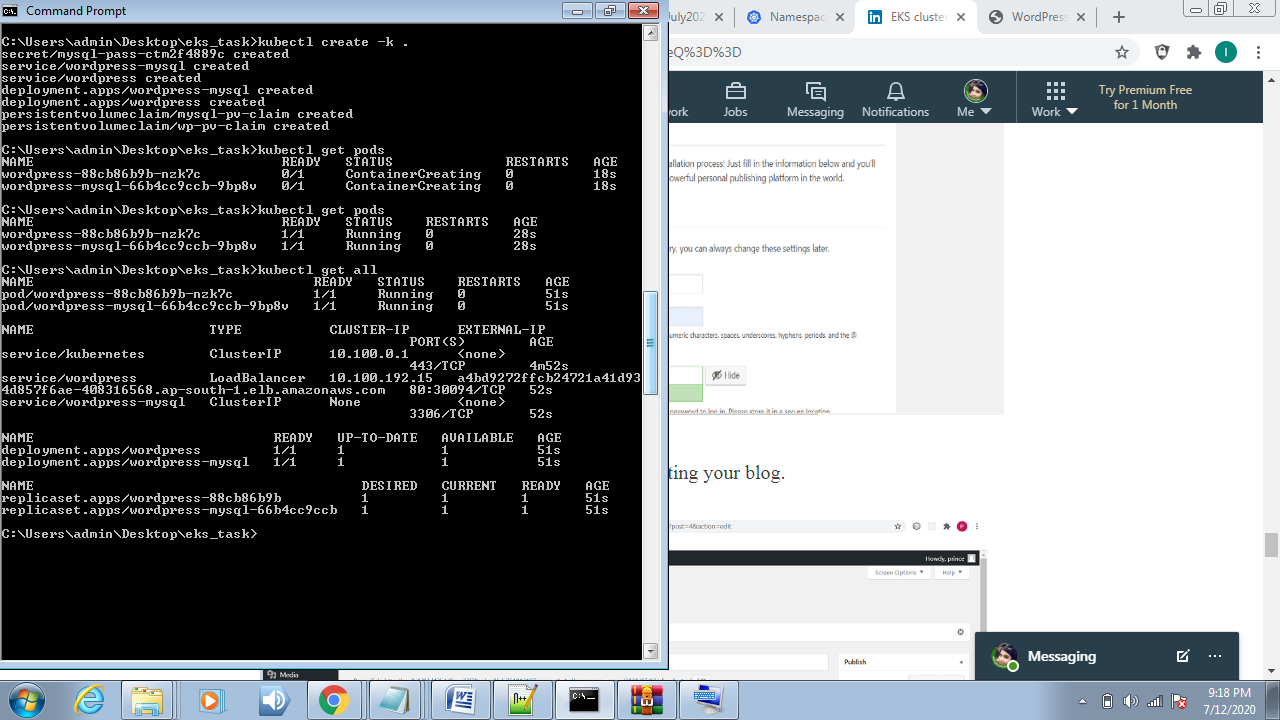
resources:

- mysql-deployment.yaml

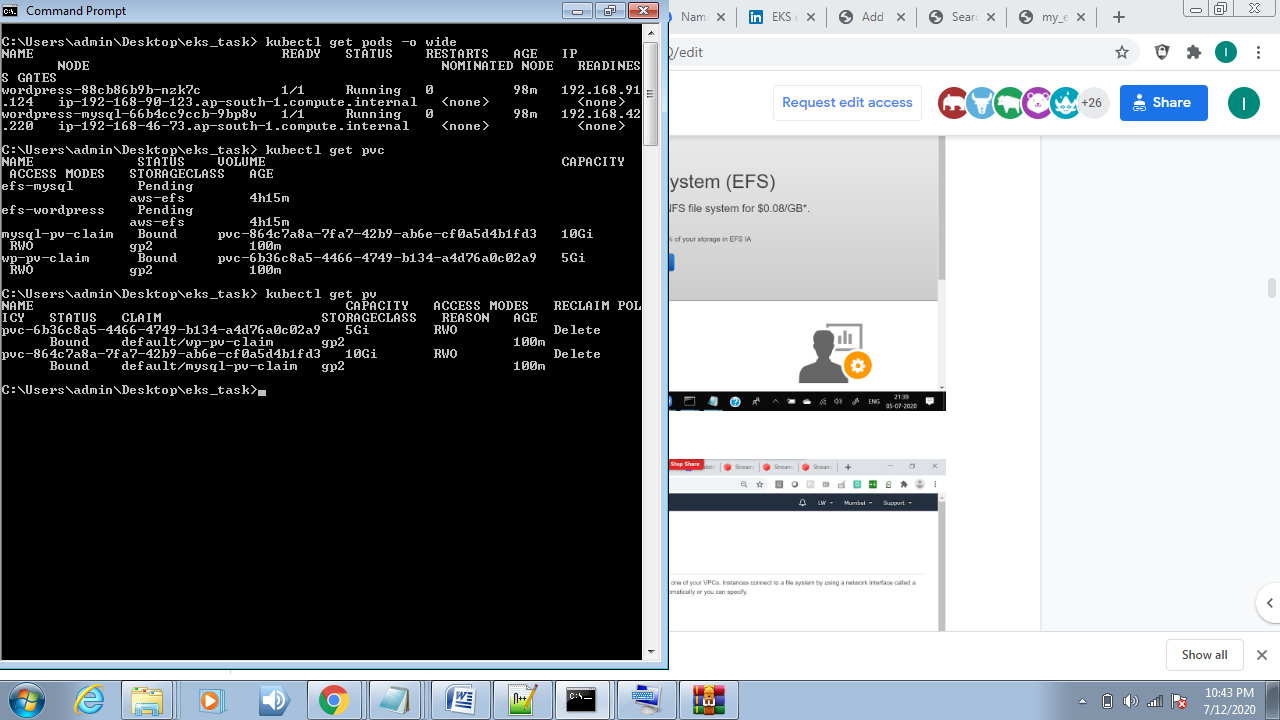
- wordpress-deployment.yaml

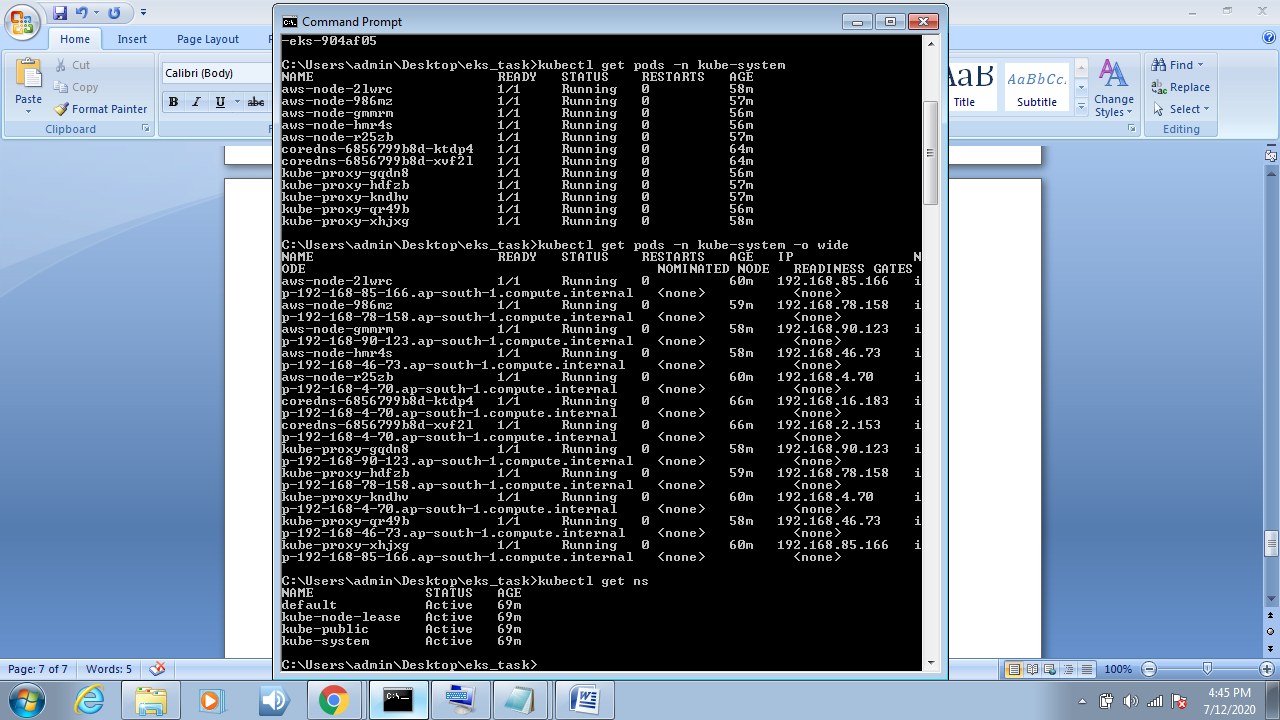
**Step 11. Deploy the whole setup and run these commands:-**

* **kubectl create –k .**
* **kubectl get pods //to show all pods**
* **kubectl get all //provide all details**

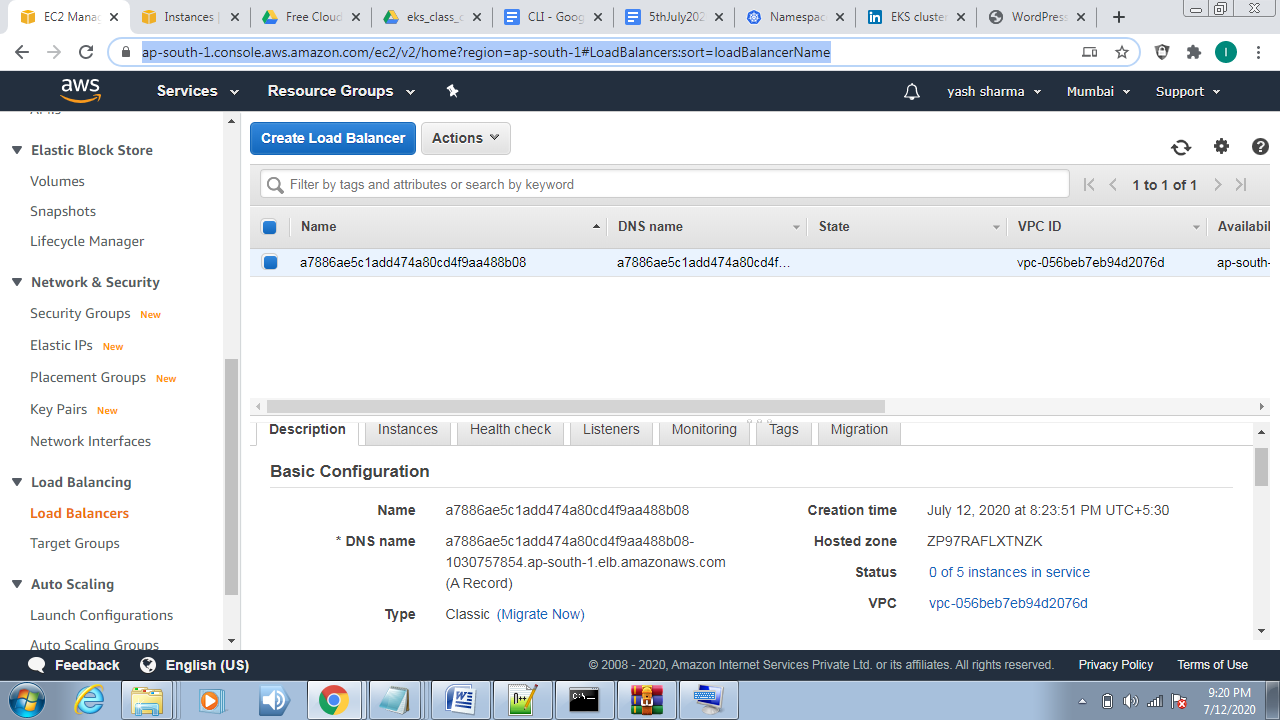


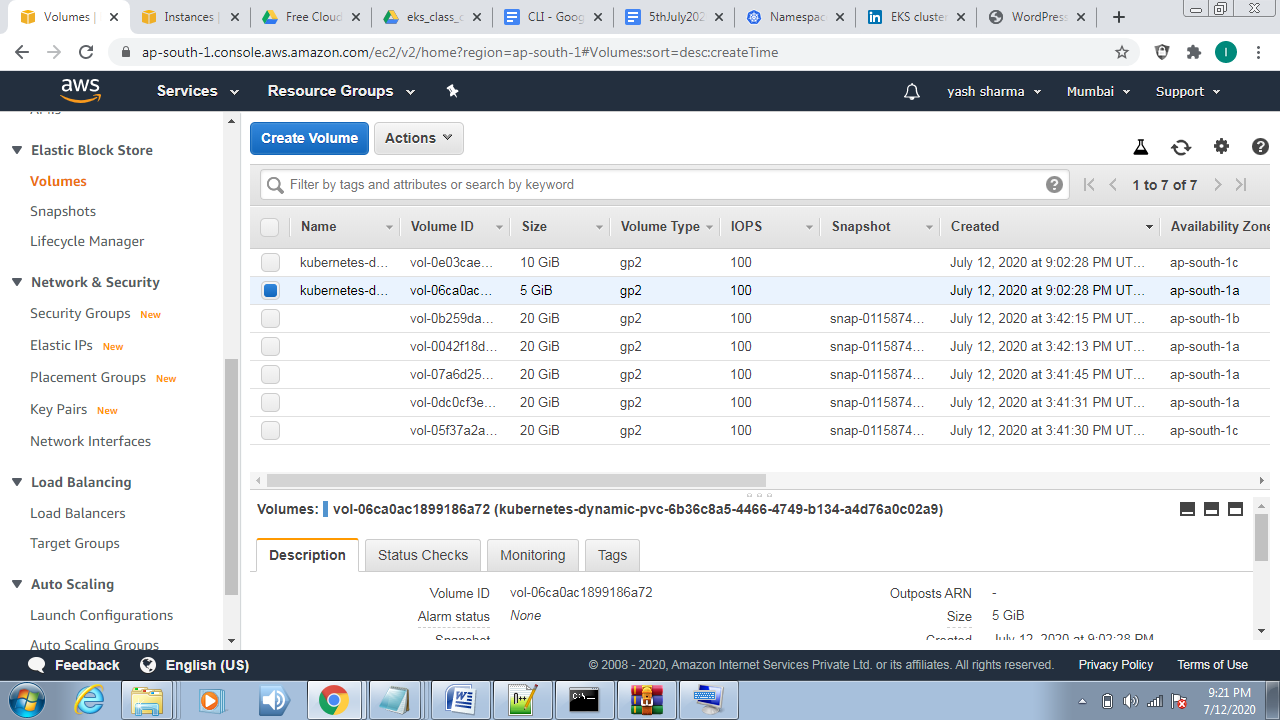
* **To get VPC and pv by using CLI :- Command**🡪kubectl get pvc and kubectl get pv



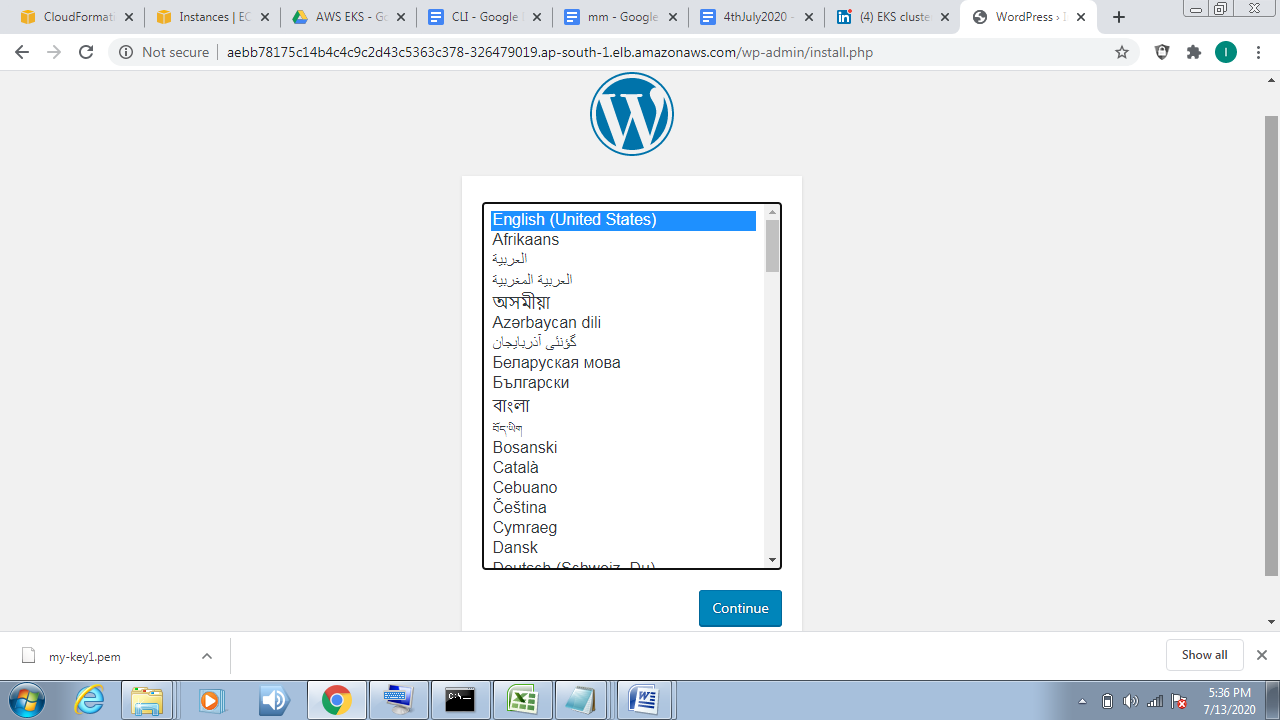
****

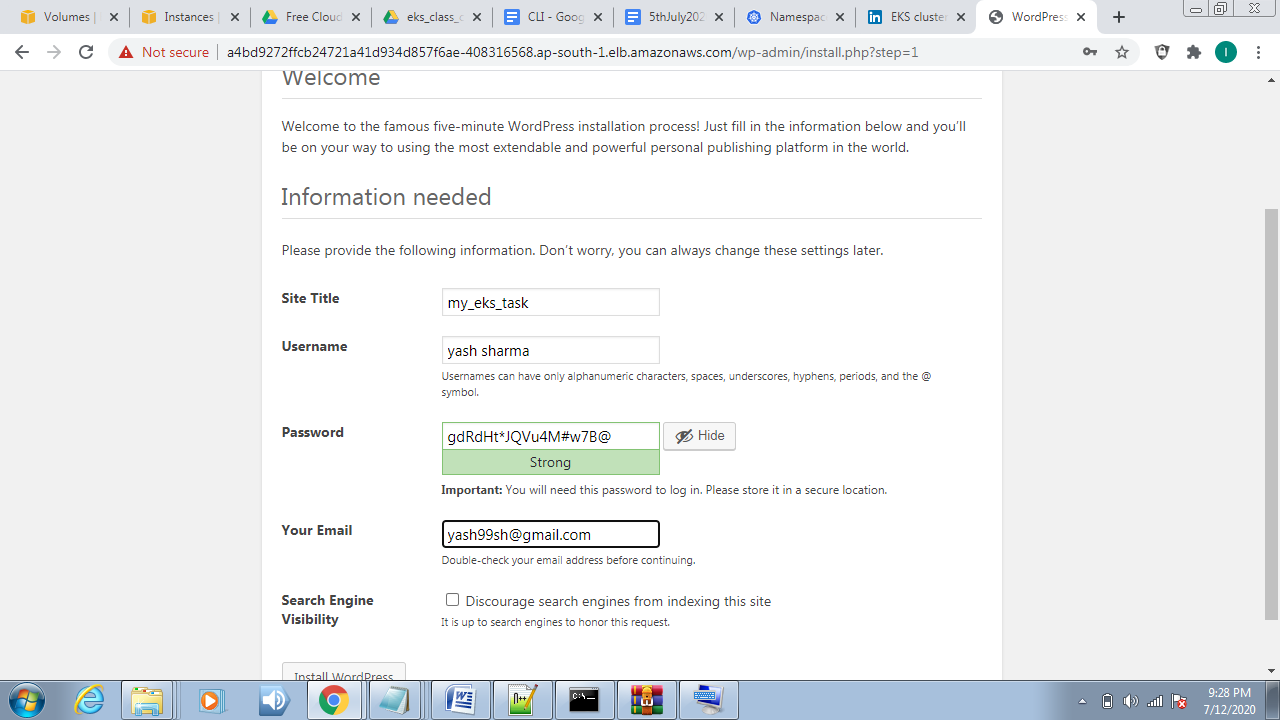
**Load Balancer:-**

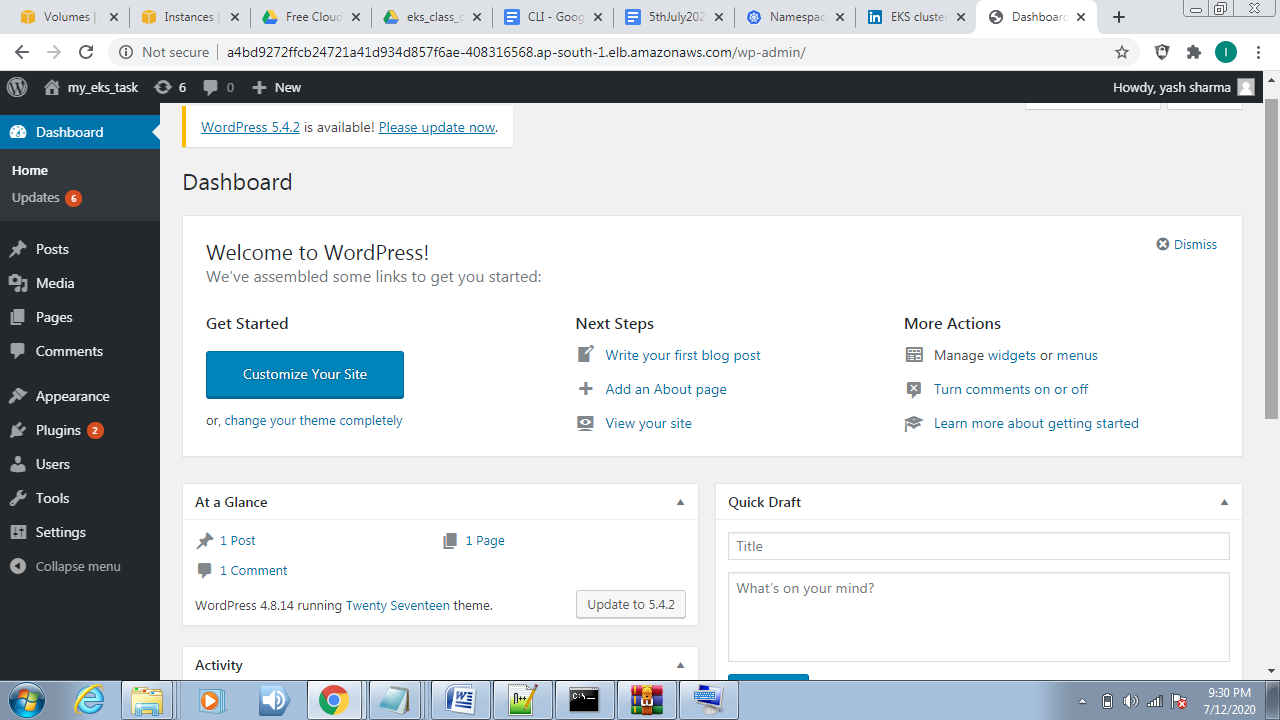


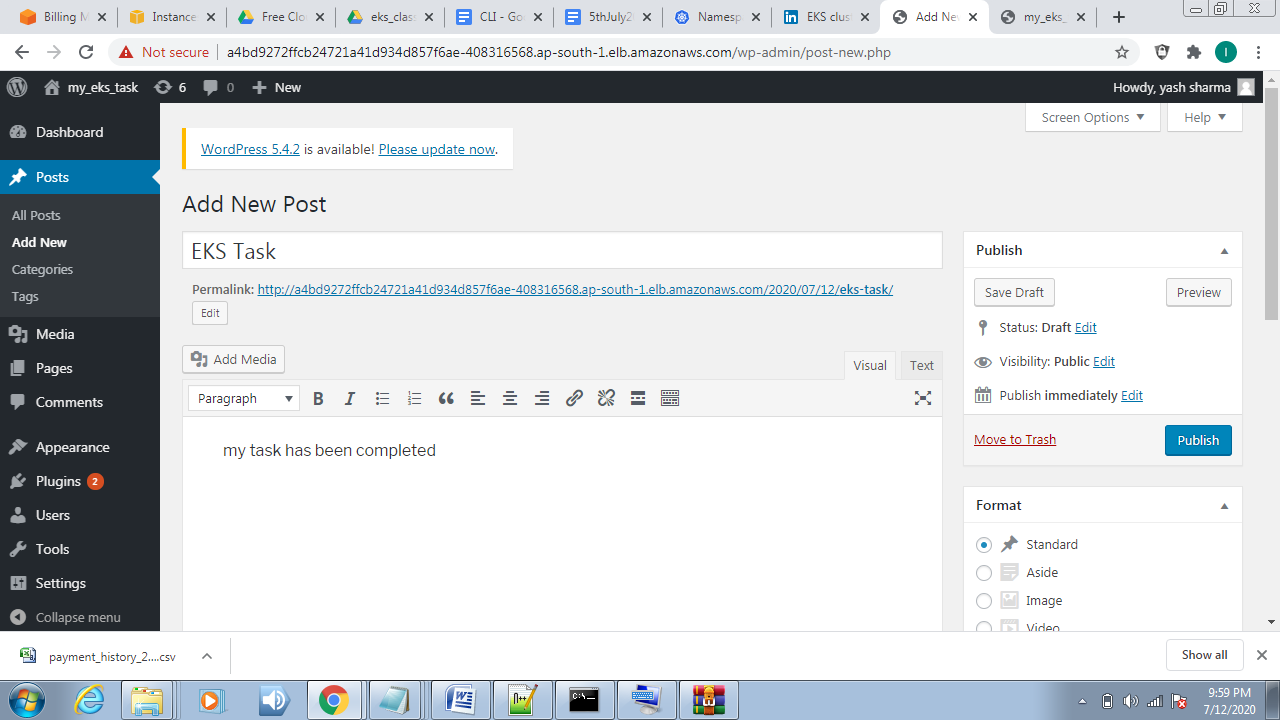
* **Volume: - (** in this we see persistence volume of wordpress-deployment and mysql-deployment added)
* 

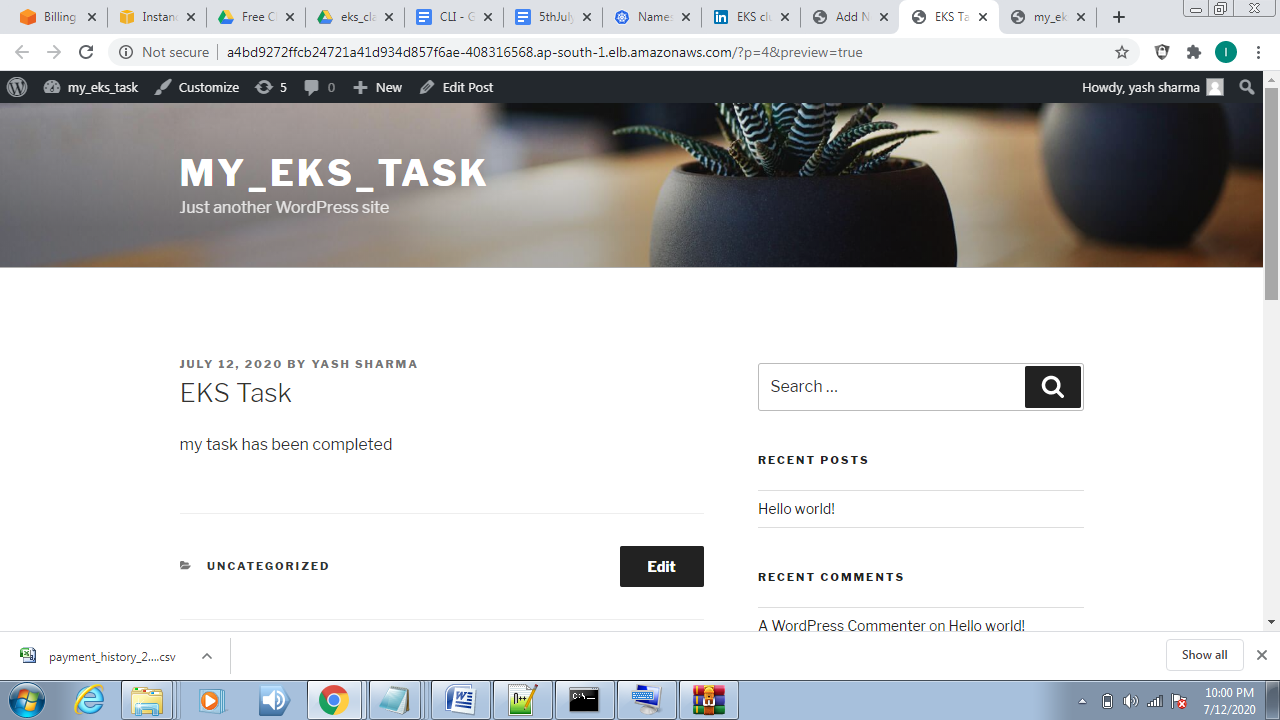
**Step 12 .Now copy the external ip (** get it from command kubectl get all**) and paste it into your browser.**











**In last check your bill and delete your cluster by CLI using command -> “**eksctl delete cluster –f cluster.yml**” . Otherwise your bill increases per hour bases .**

**“My task has been completed. I thanks to vimal daga sir and linuxWorld\_India for guidance**

**Thanks for reading…**