## What are the steps you are going to take to lead this transformation?

## 

## What is Amazon EKS?

Amazon EKS (Elastic Container Service for Kubernetes) is a managed Kubernetes service that allows you to run Kubernetes on AWS without the hassle of managing the Kubernetes control plane.

The [Kubernetes control plane](https://kubernetes.io/docs/concepts/#kubernetes-control-plane) (API)plays a crucial role in a Kubernetes deployment as it is responsible for how Kubernetes communicates with your cluster — starting and stopping new containers, scheduling containers, performing health checks, and many more management tasks.

EKS will provision, scale and manage the Kubernetes control plane for you to ensure high availability, security and scalability.

**You will need to make sure you have the following components installed and set up before you start with Amazon EKS:**

AWS CLI – while you can use the AWS Console to create a cluster in EKS, the AWS CLI is easier.

Kubectl – used for communicating with the cluster API server.

eksctl : is a simple CLI tool for creating clusters on EKS - Amazon's new managed Kubernetes service for EC2. It is written in Go, and uses CloudFormation.

**install \_kubectl**

sudo apt-get update && sudo apt-get install -y apt-transport-https

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

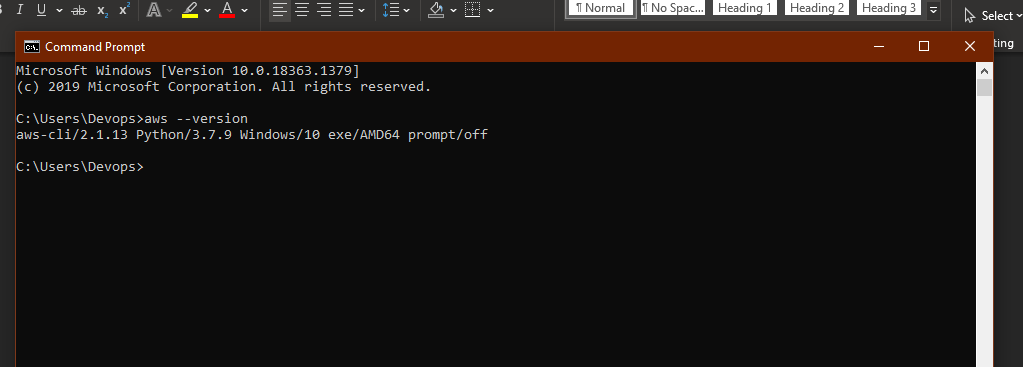
**Ekstl – setup and operation of EKS cluster**

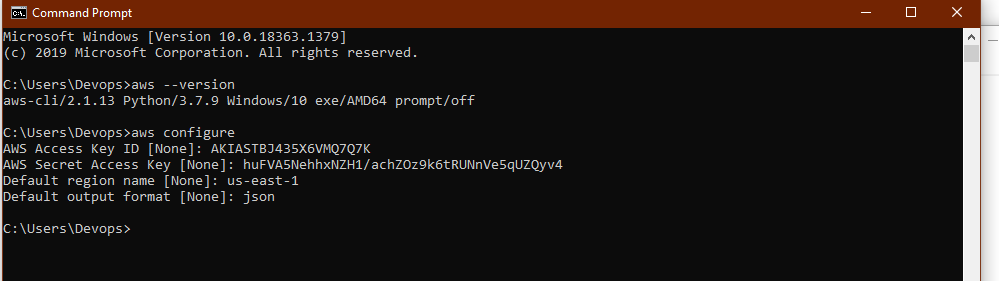
**Setup of eksctl :**

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

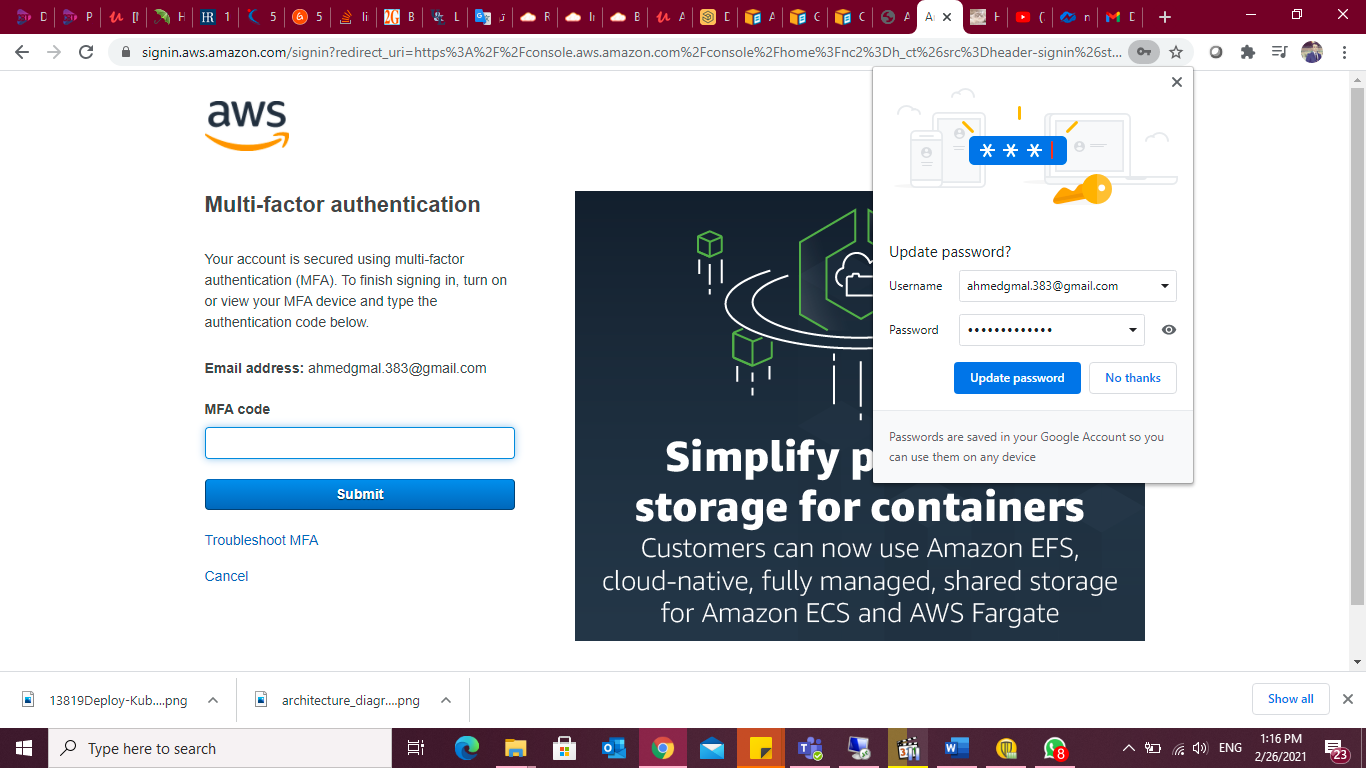
#sudo mv /tmp/eksctl /usr/local/bin

**Setup AWS CLI**



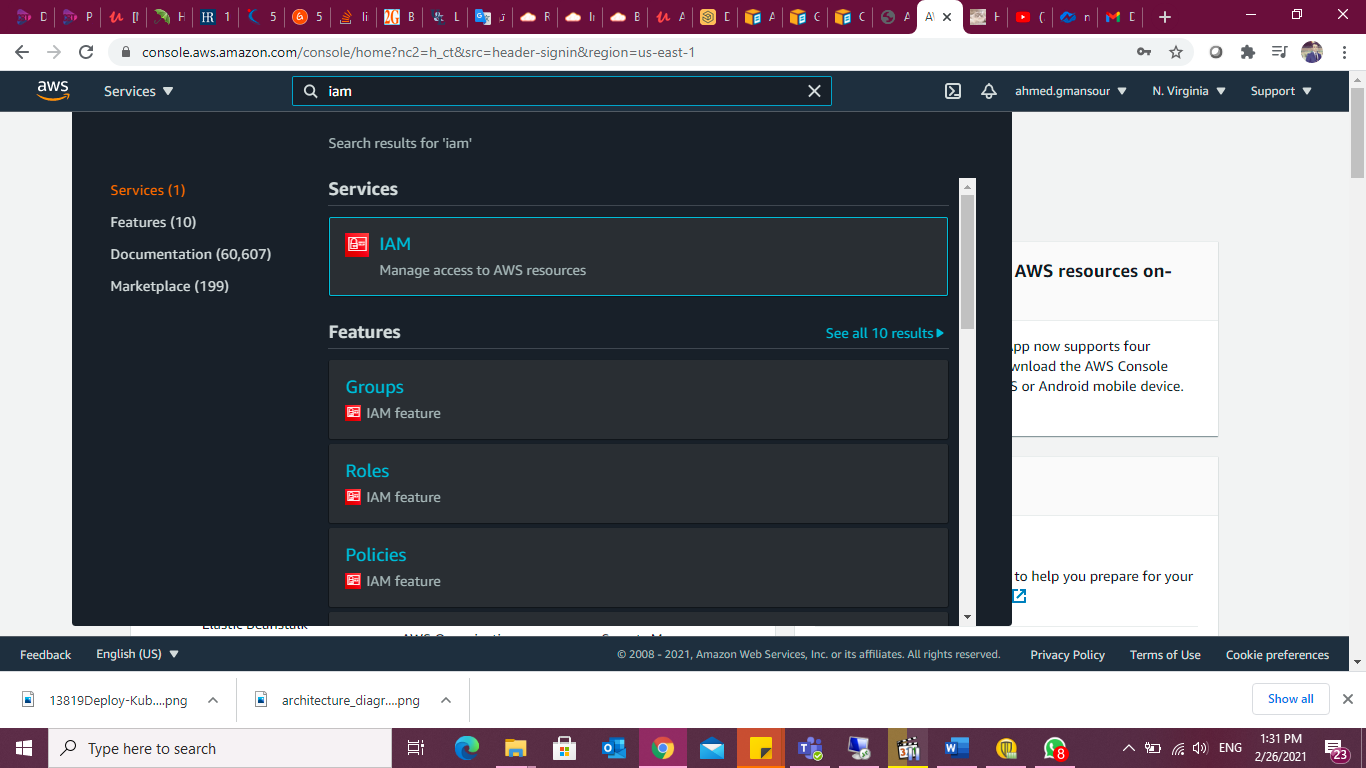


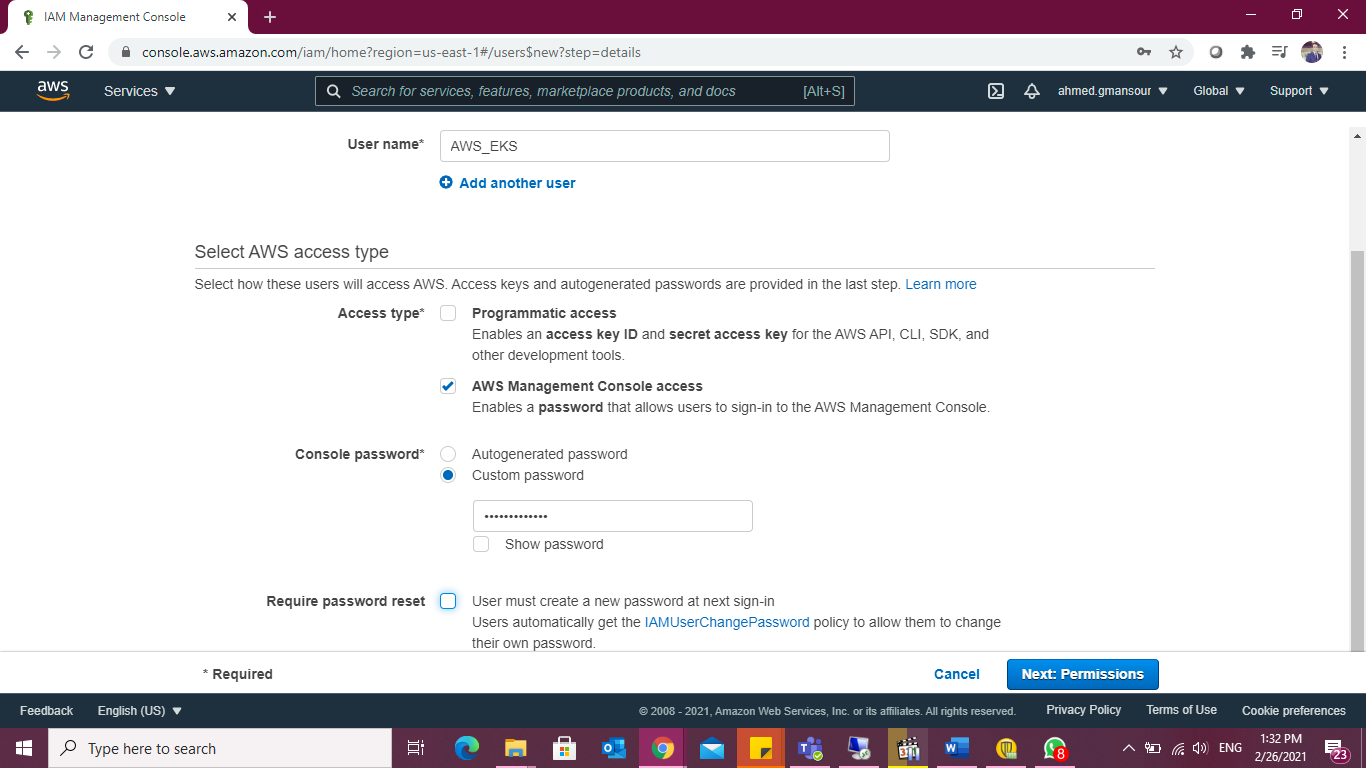
**Sign in to your AWS account at**[**https://aws.amazon.com**](https://aws.amazon.com/)**with an IAM user role that has the necessary**

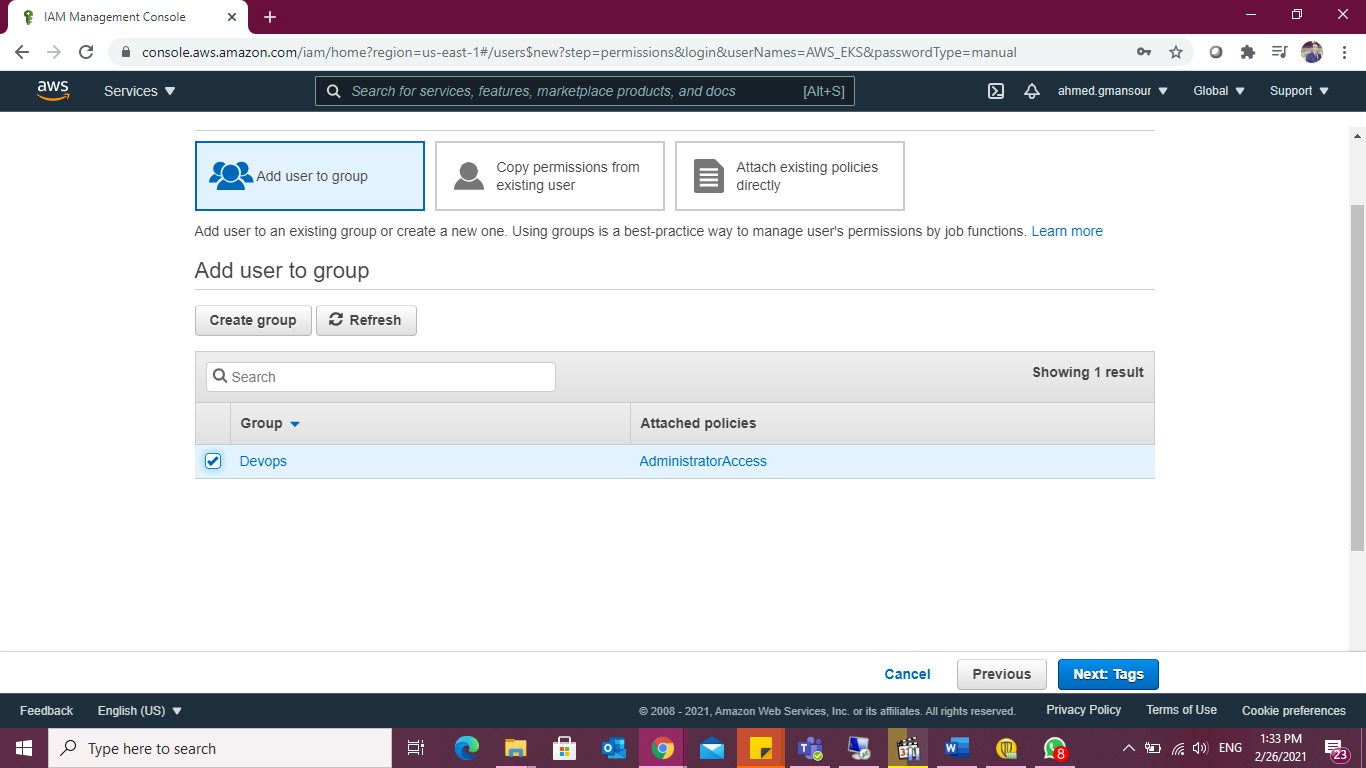


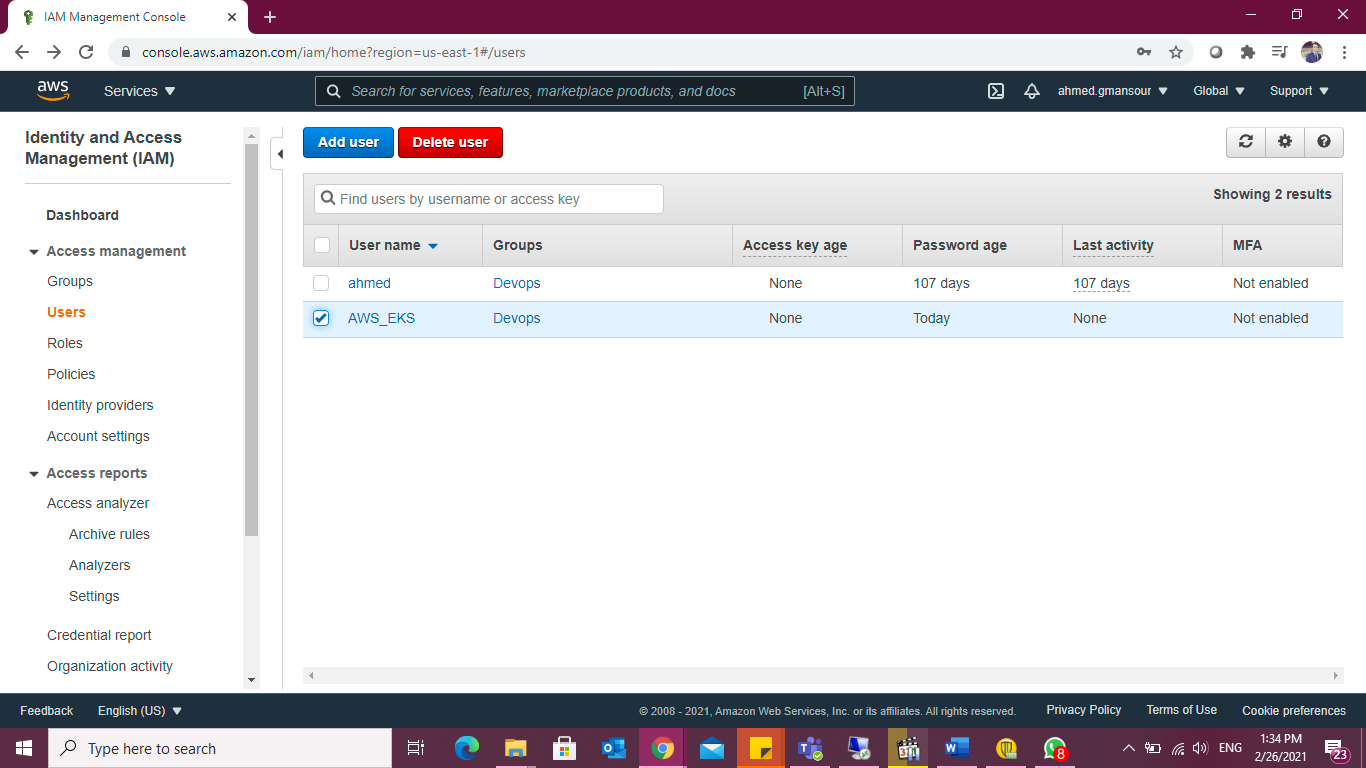
## Step 1: Creating an EKS role

Our first step is to set up a new IAM role with EKS permissions.

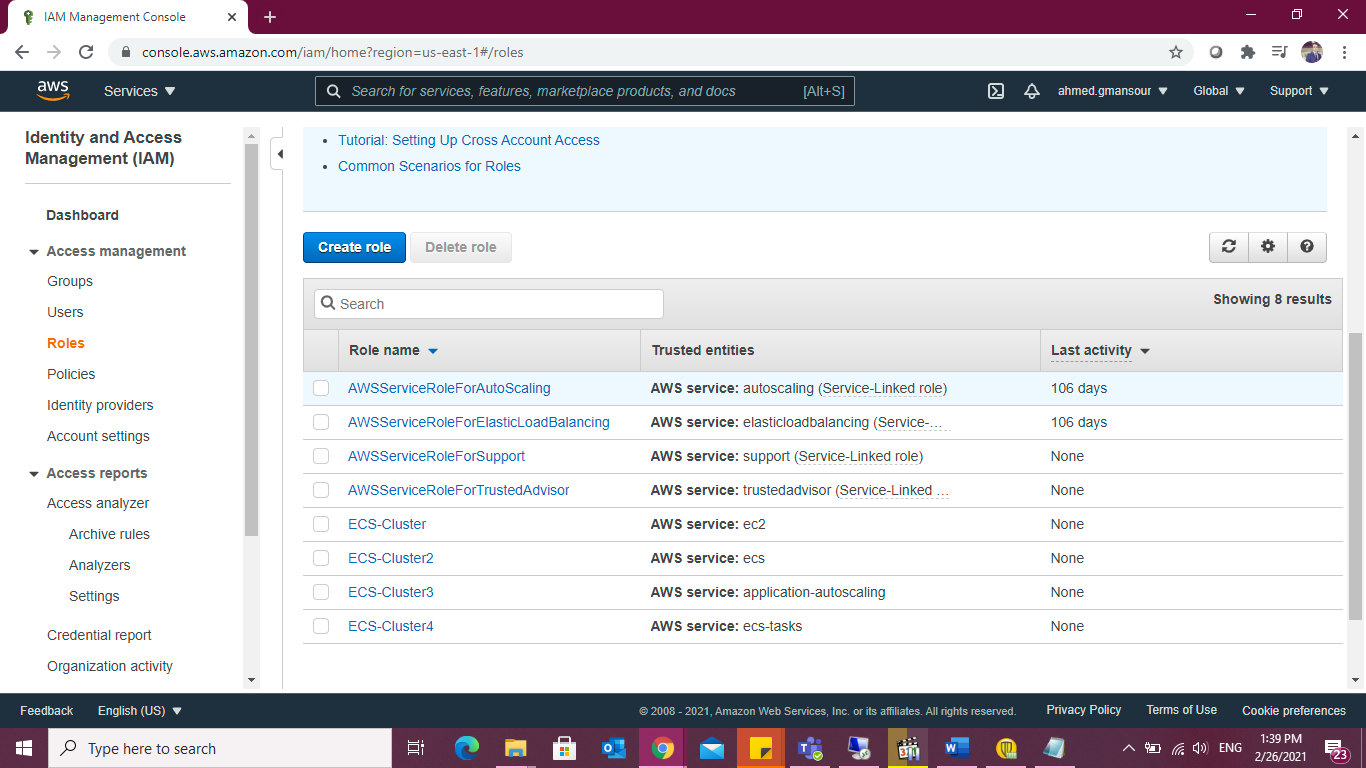




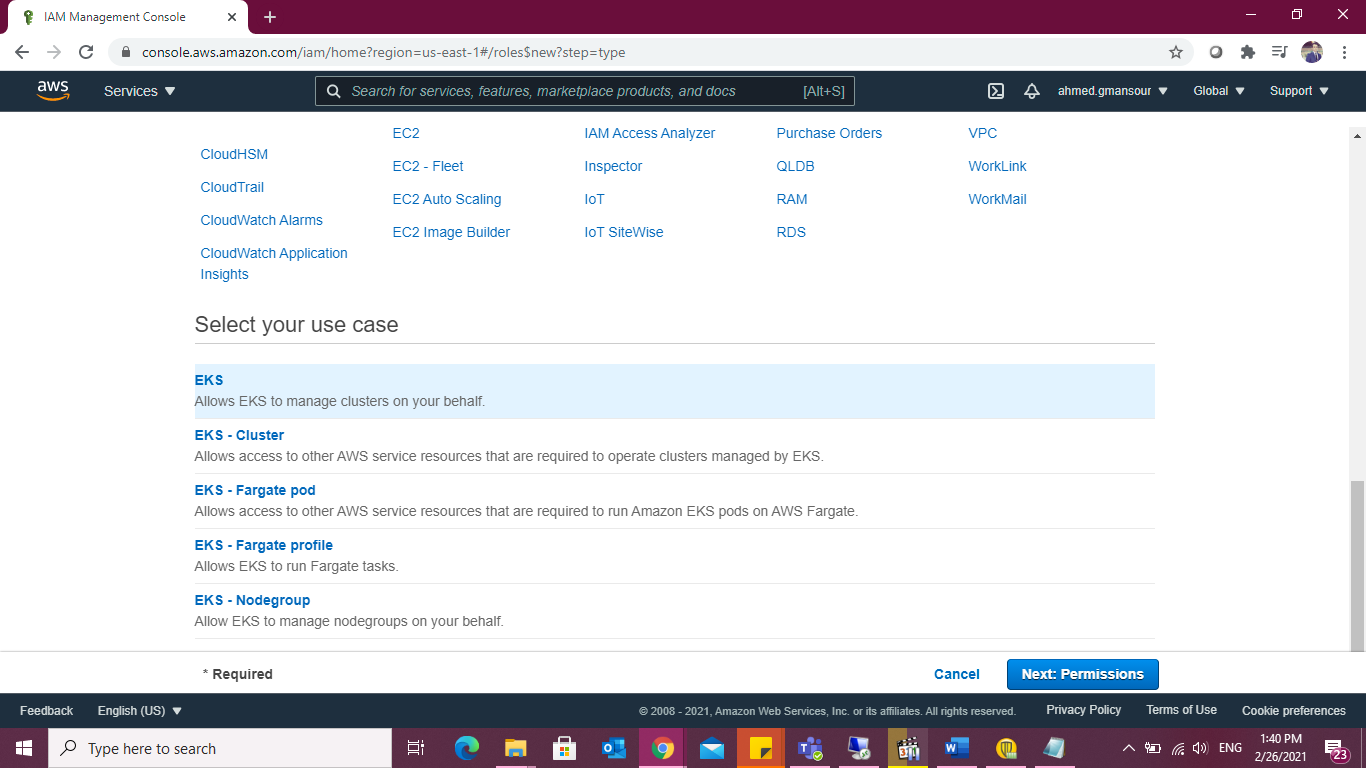


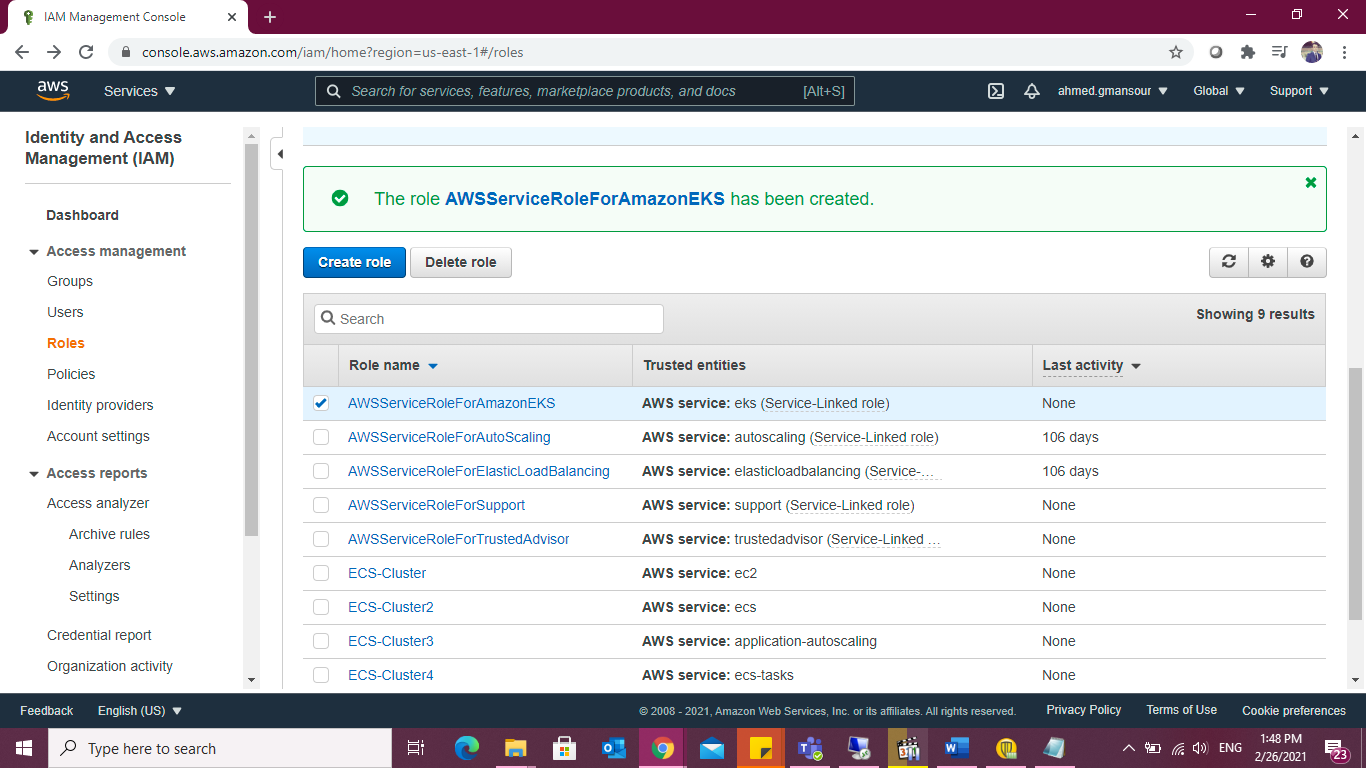


Open the [IAM console](https://console.aws.amazon.com/iam/), select Roles on the left and then click the Create Role button at the top of the page.

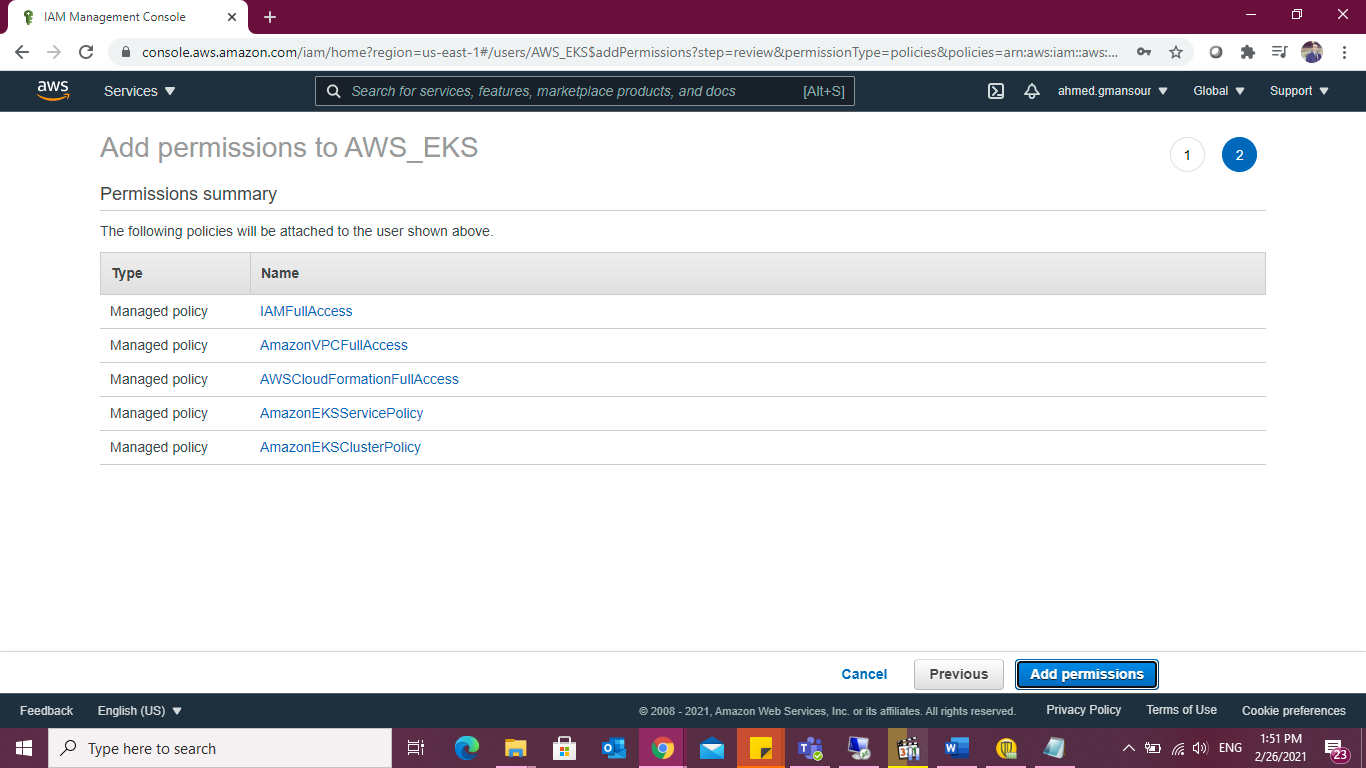


From the list of AWS services, select EKS and then Next: Permissions at the bottom of the page.





Give Permission to the user:



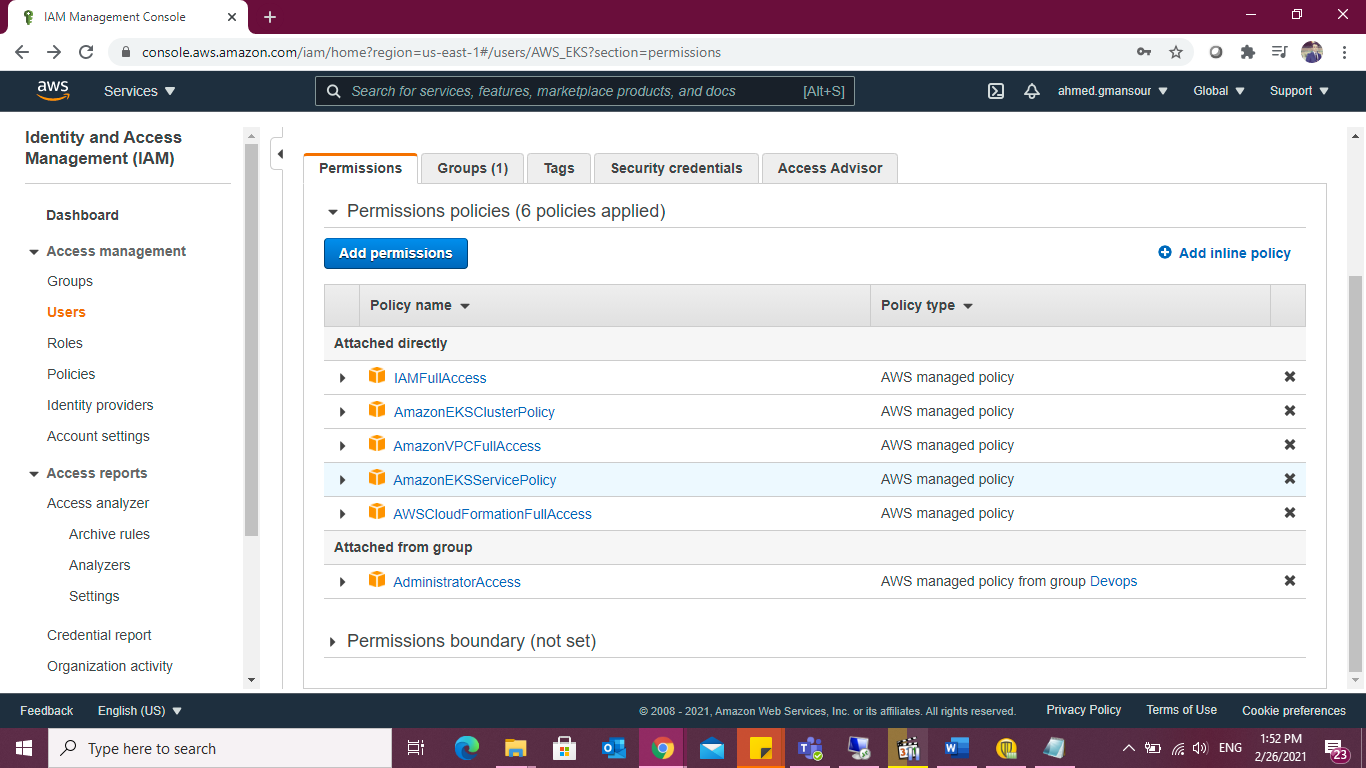
- AmazonEC2FullAccess

- IAMFullAccess

- AmazonVPCFullAccess

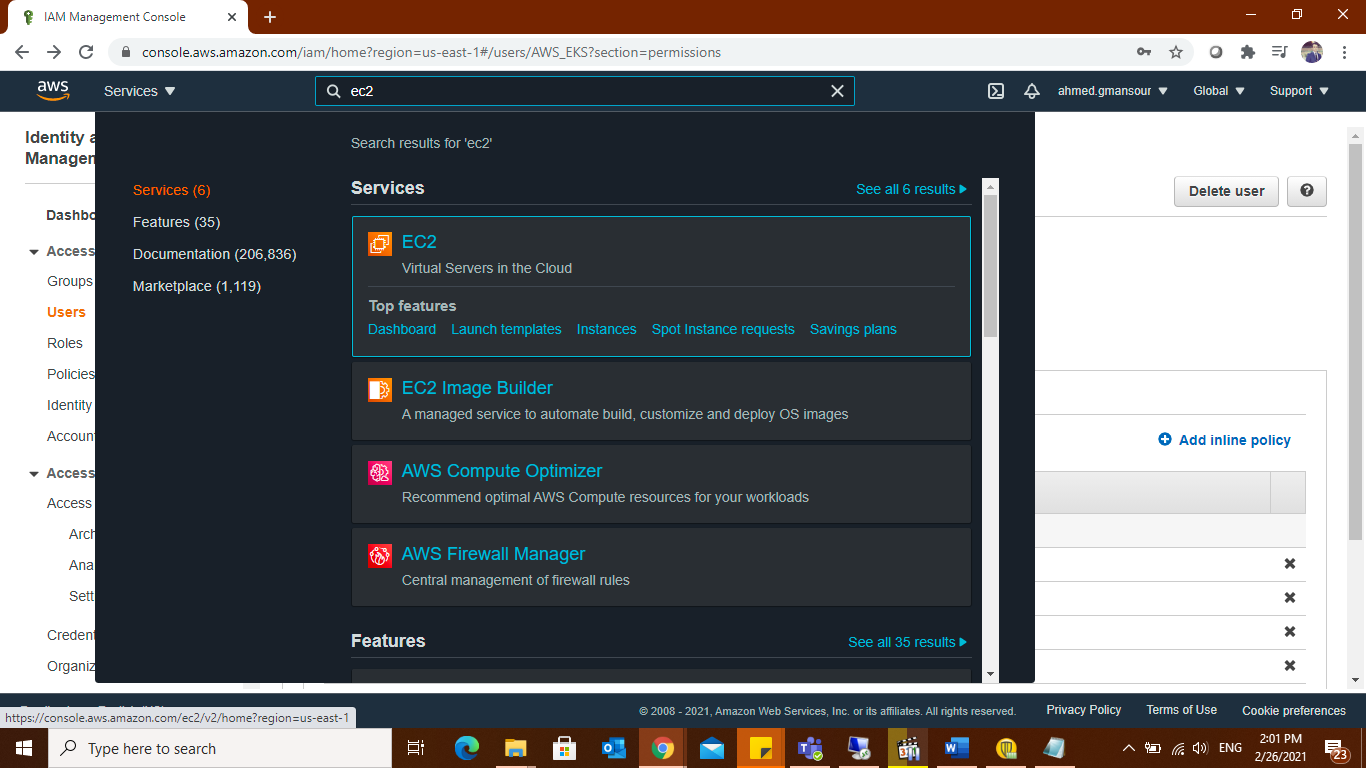
- CloudFormation-Admin-policy

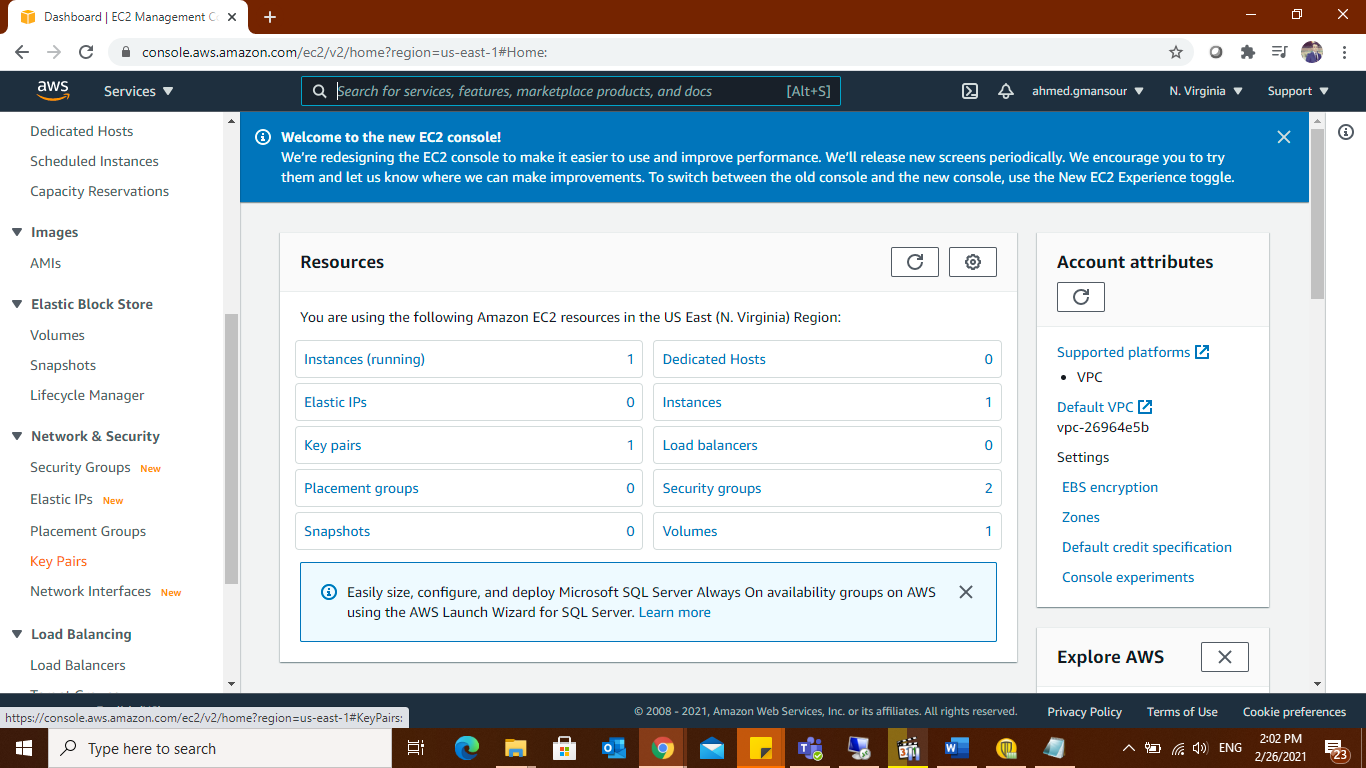
- EKS-Admin-policy

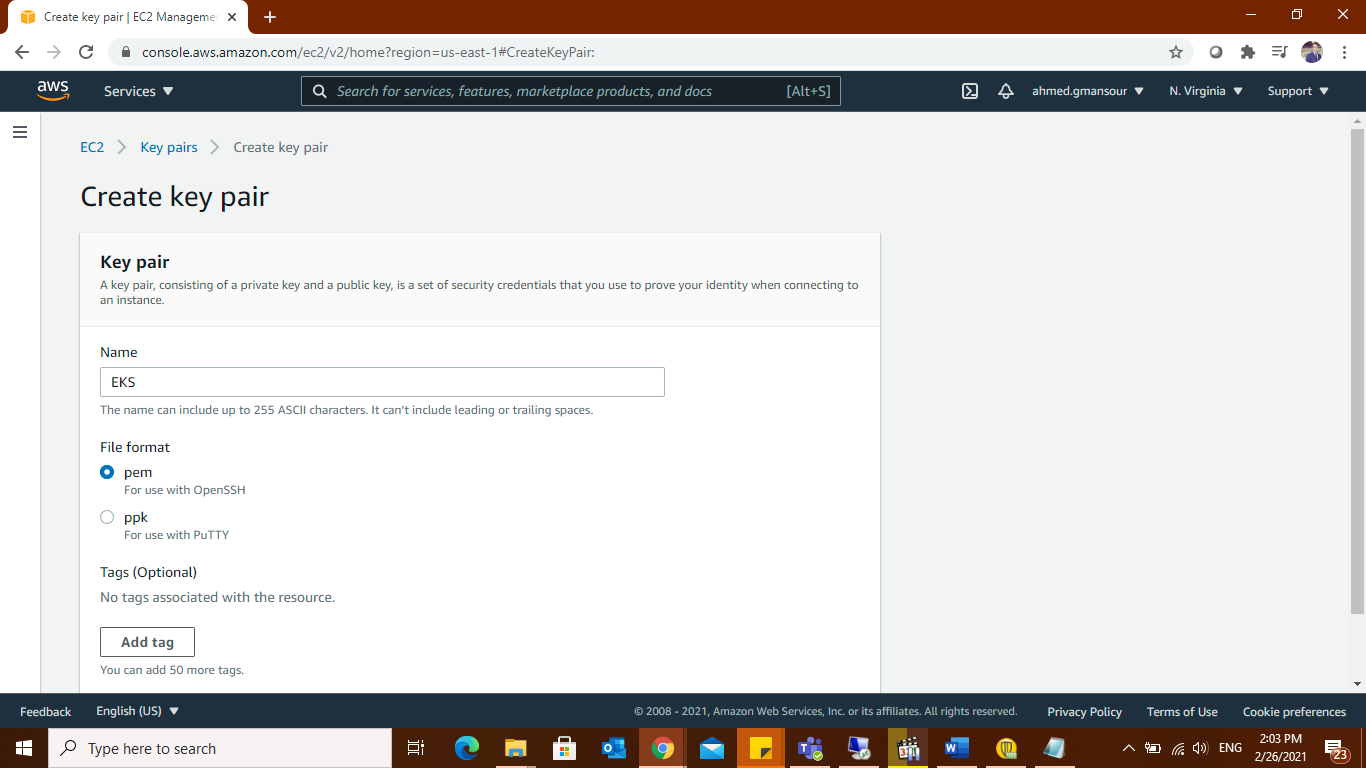


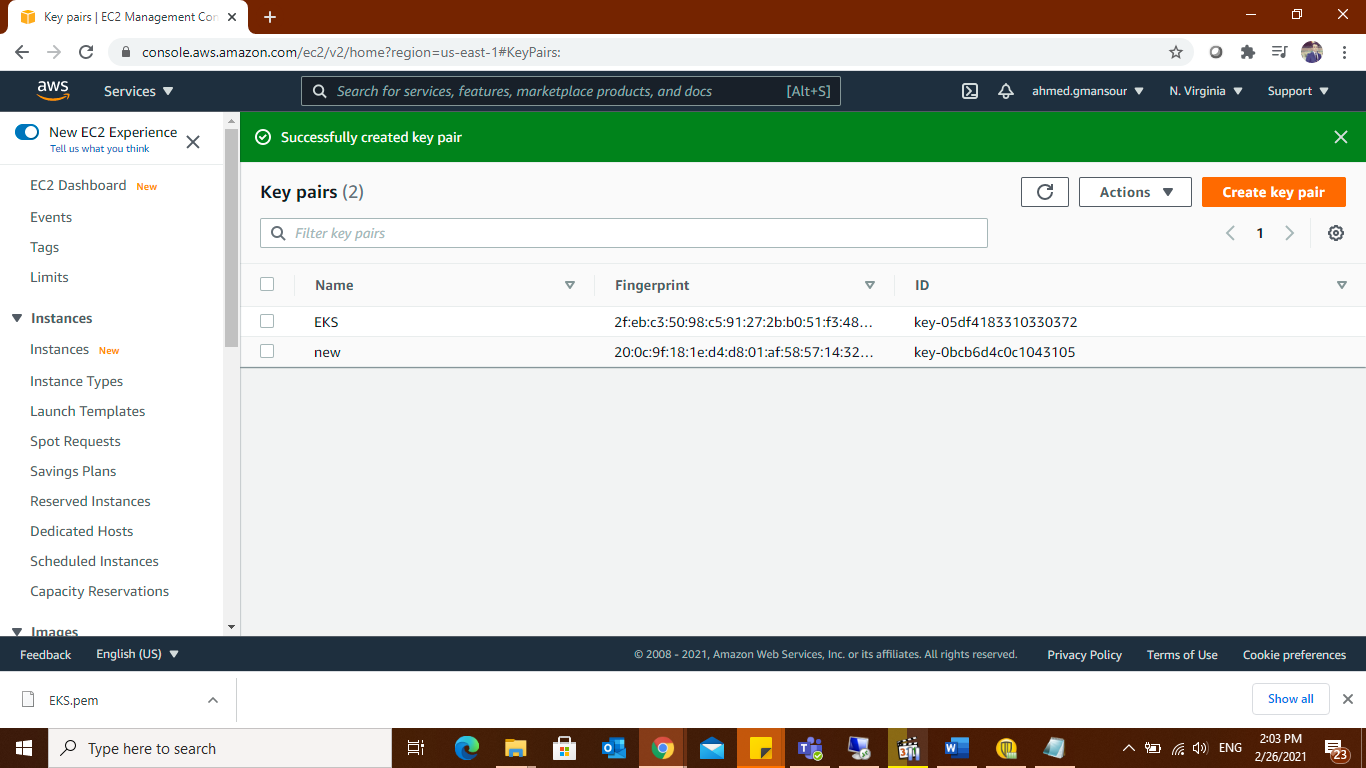
Be sure to note the Role ARN, you will need it when creating the Kubernetes cluster in the steps below.

**Create SSH key For IAM User to be able to ssh to my EC2 instances.**



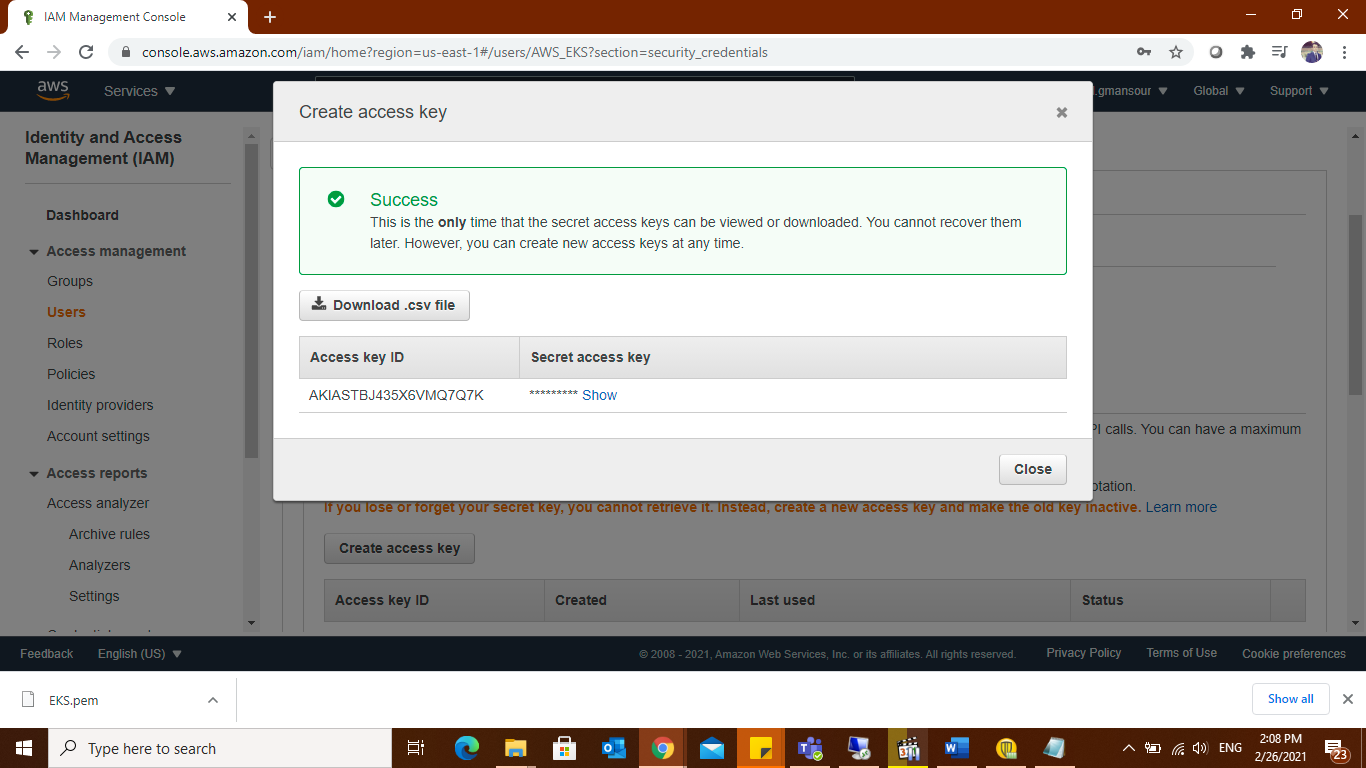






As you may have seen according to the PEM file including the key please ensure that you copy it to proper and safe folder so that no one except you has access to this folder.

Then back to dashboard and search for IAM to user then choose secuirty credetials



This is the only time that the secret access keys can be viewed or downloaded. You cannot recover them later. However, you can create new access keys at any time. (Use access keys to make programmatic calls to AWS from the AWS CLI, )

## Creating the EKS cluster with the easy way

**On local machine:**

#mkdir eksctl

#vi eksctl/eks-aws.yaml

apiVersion: eksctl.io/v1alpha5

kind: ClusterConfig

metadata:

name: EKS-AWS

region: us-east-1

nodeGroups:

- name: ng-1

instanceType: t2.small

desiredCapacity: 3

ssh: # use existing EC2 key

publicKeyName: eks-AWS

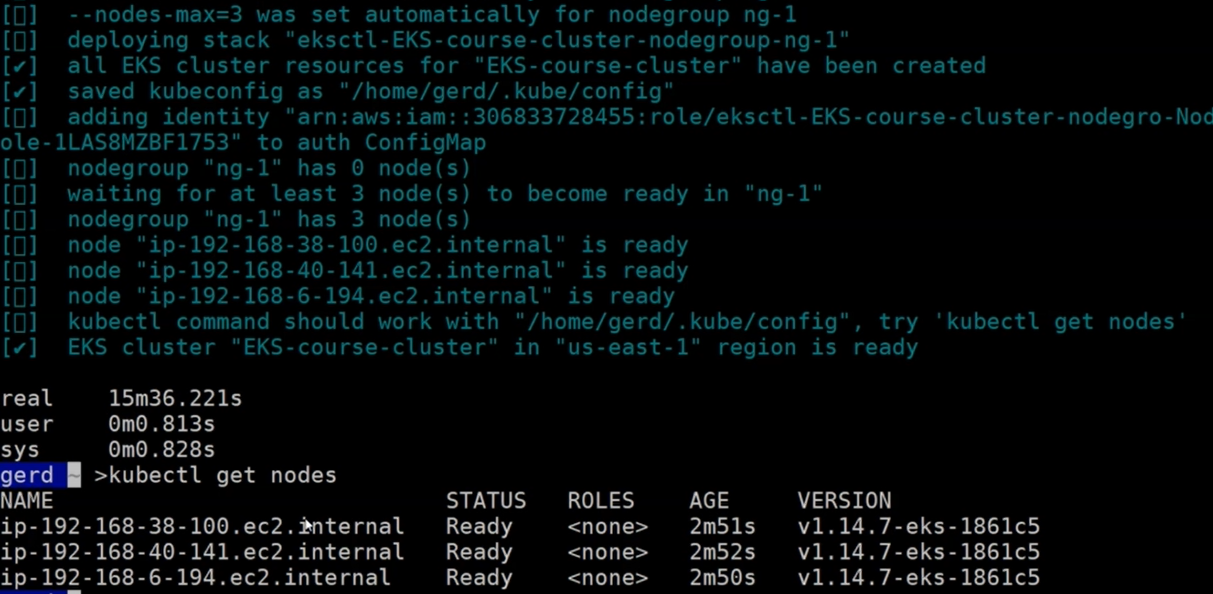
**-eksctl will automatically creating 2 subnets and two different availability zone**

then run the below command:

#eksctl create cluster -f eksctl/eks-aws.yaml

# aws eks update-kubeconfig --name AWS\_EKS

#kubectl get nodes



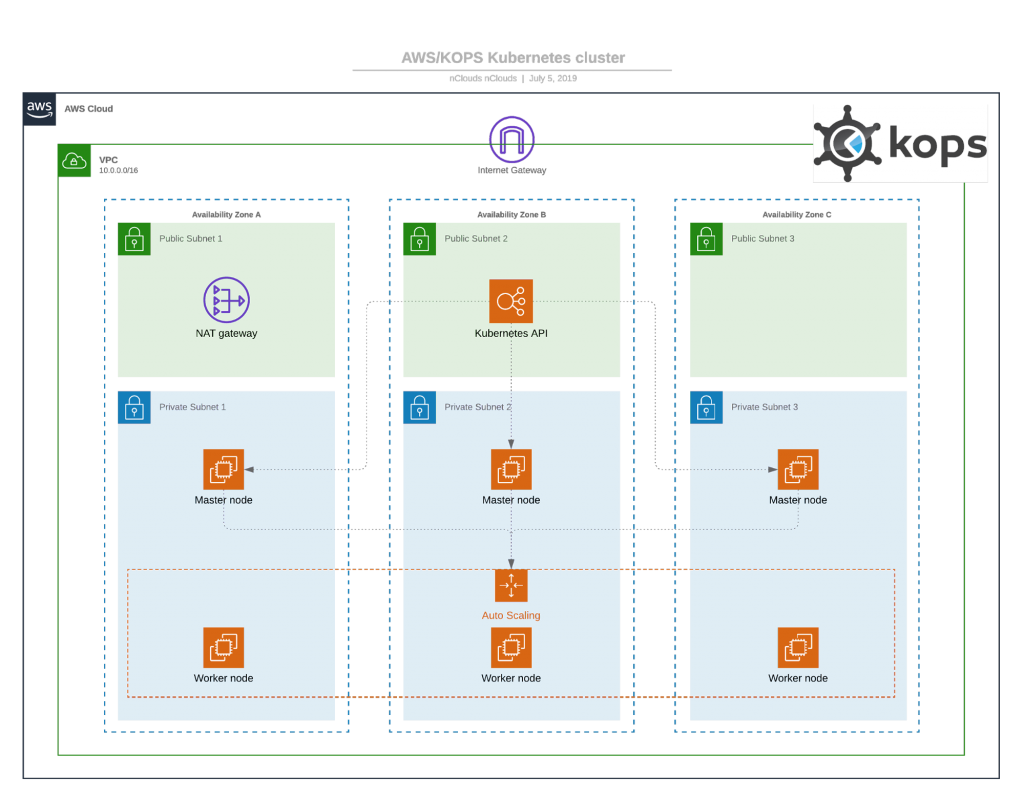
**The cluster up and running.**

OR

## Creating the EKS cluster (Hard Way)

Architecture

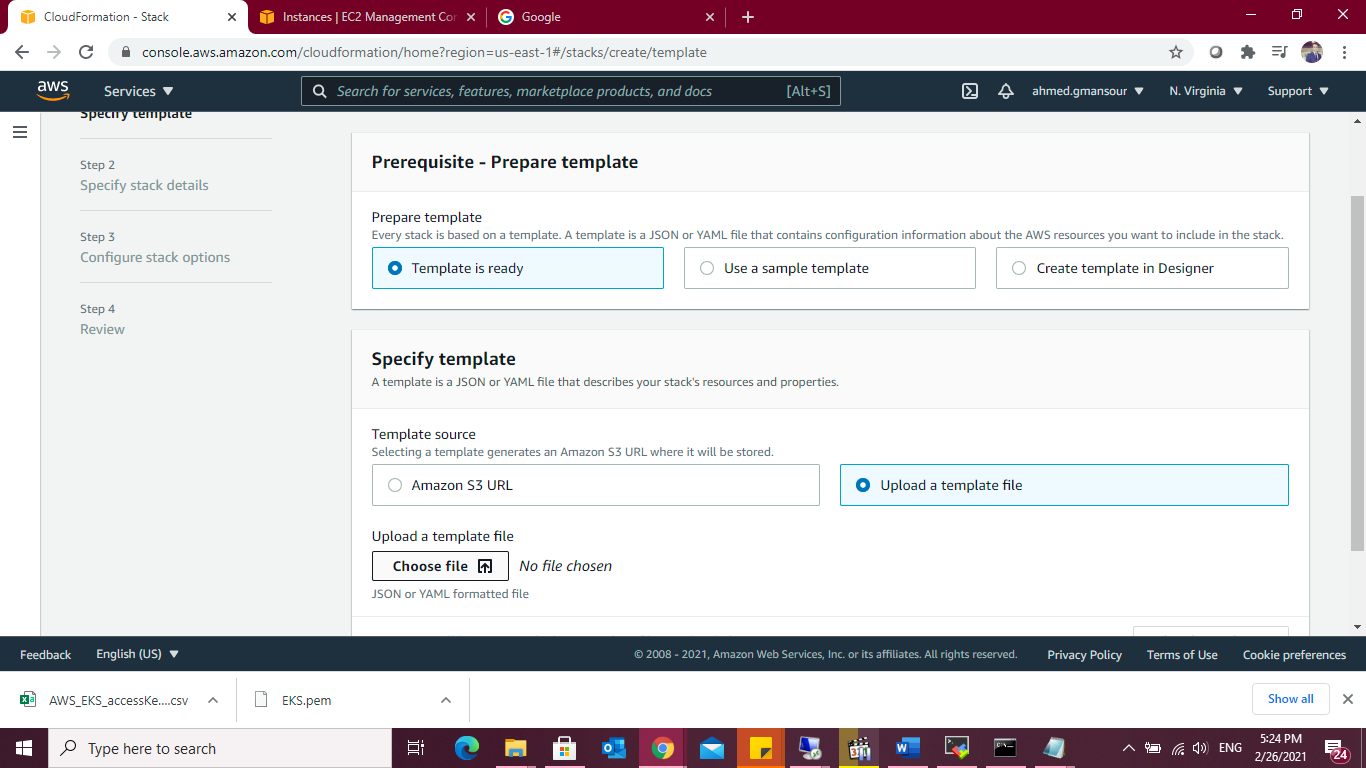
Deploying this Quick Start for a new virtual private cloud (VPC) with default parameters builds the following Amazon EKS environment in the AWS Cloud.

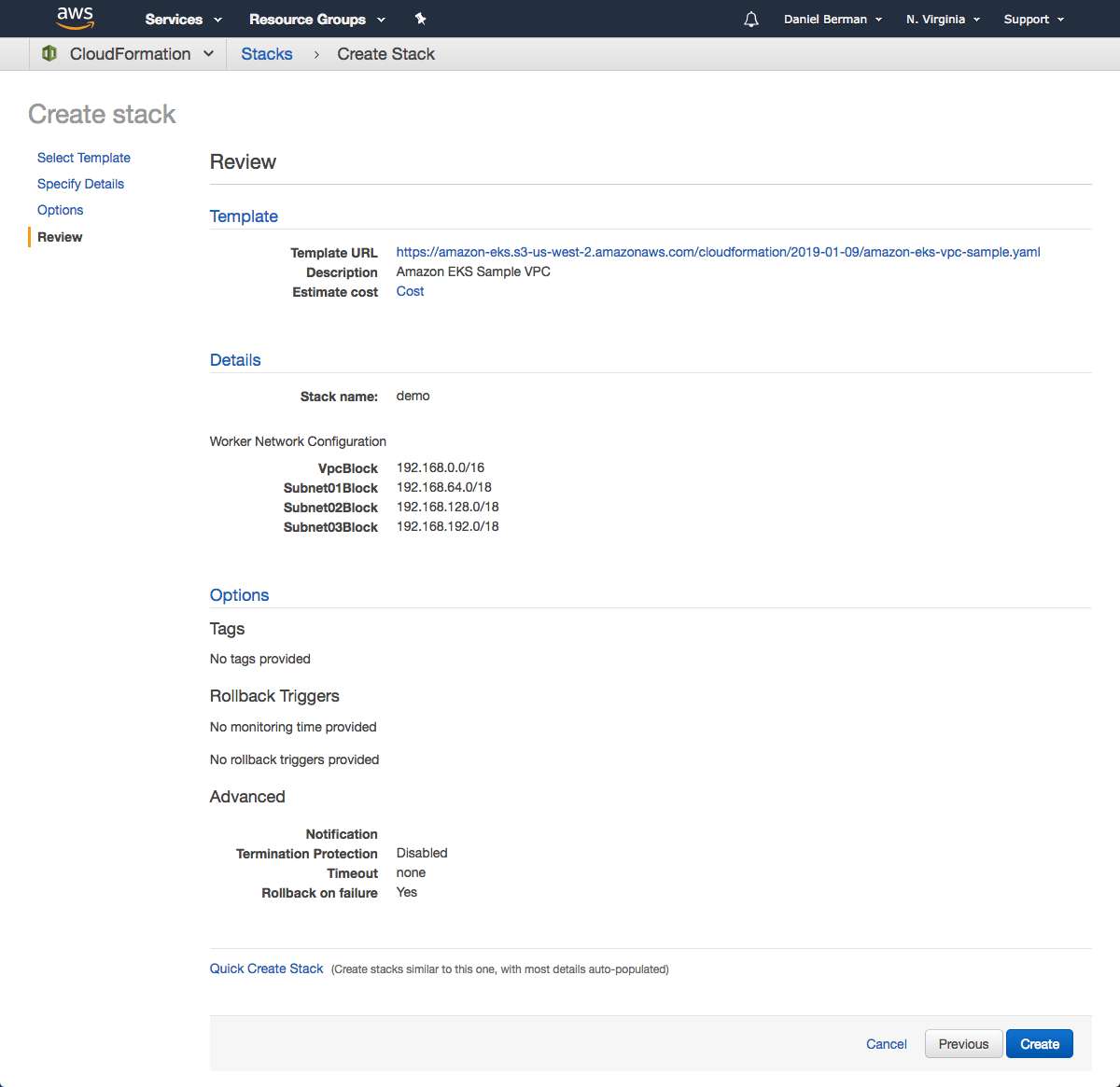


The Quick Start sets up the following:

* A highly available architecture that spans three Availability Zones. \*
* A VPC configured with public and private subnets.
* In the public subnets, managed NAT gateways to allow outbound internet access for resources in the private subnets.
* In the private subnets, a group of Kubernetes nodes.

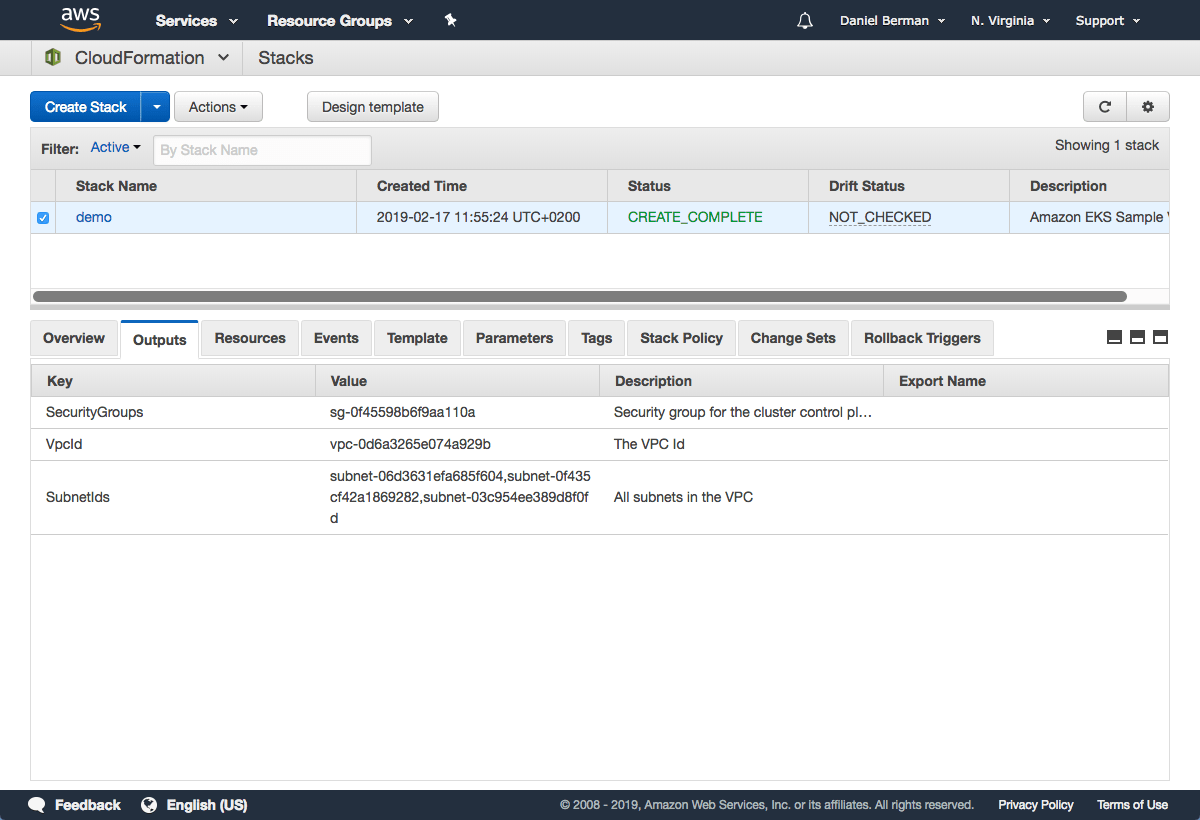
we are going to create a separate VPC—a Virtual Private Cloud that enable communication between worker nodes and the AWS Kubernetes API server— for our EKS cluster. To do this, we are going to use a CloudFormation.( amazon-eks-vpc-sample.yaml) attached.





As mentioned above, we will use the AWS CLI to create the Kubernetes cluster. To do this, use the following command:

#aws eks --region us-east-1 create-cluster --name AWS\_EKS --role-arn:aws:iam::178329018223:user/eksServiceRole --resources-vpc-config subnetIds=subnet-06d3631efa685f604,subnet-0f435cf42a1869282,subnet-03c954ee389d8f0fd,securityGroupIds=sg-0f45598b6f9aa110a



You can ping the status of the command using this CLI command:

#aws eks --region us-east-1 describe-cluster --name AWS\_EKS--query cluster.status

Once the status changes to “ACTIVE”, we can proceed with updating our kubeconfig file with the information on the new cluster so kubectl can communicate with it.

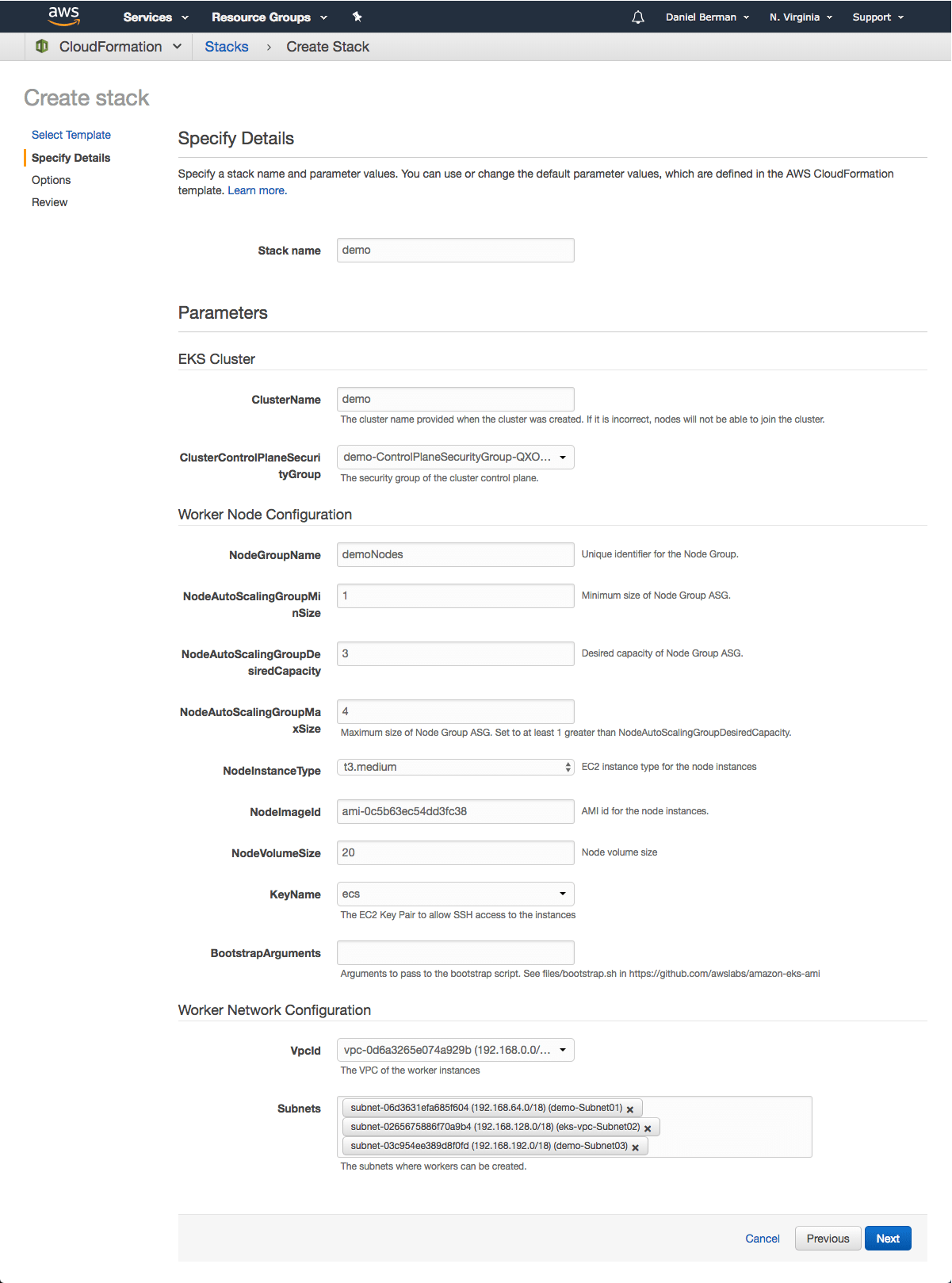
To do this, we will use the AWS CLI update-kubeconfig command:

#aws eks --region us-east-1 update-kubeconfig --name AWS\_EKS

## Launching Kubernetes worker nodes

Now that we’ve set up our cluster and VPC networking, we can now launch Kubernetes worker nodes. To do this, we will again use a CloudFormation template.

Open CloudFormation, click Create Stack.(amazon-eks-nodegroup.yaml)



CloudFormation creates the worker nodes with the VPC settings we entered — three new EC2 instances are created.

Note the value for NodeInstanceRole as you will need it for the next step — allowing the worker nodes to join our Kubernetes cluster.

To do this, first download the AWS authenticator configuration map:

#curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2019-01-09

/aws-auth-cm.yaml

Open the file and replace the rolearn with the ARN of the NodeInstanceRole created above:

apiVersion: v1

kind: ConfigMap

metadata:

name: aws-auth

namespace: kube-system

data:

mapRoles: |

- rolearn: <ARN of instance role>

username: system:node:{{EC2PrivateDNSName}}

groups:

- system:bootstrappers

- system:nodes

Save the file and apply the configuration:

#kubectl apply -f aws-auth-cm.yaml

#kubectl get nodes

My Kubernetes cluster is created and set up…

**Installing Ansible:**

For installing ansible We will create EC2 instance on the same VPC and subnet of cluster to assign IP in the same range.

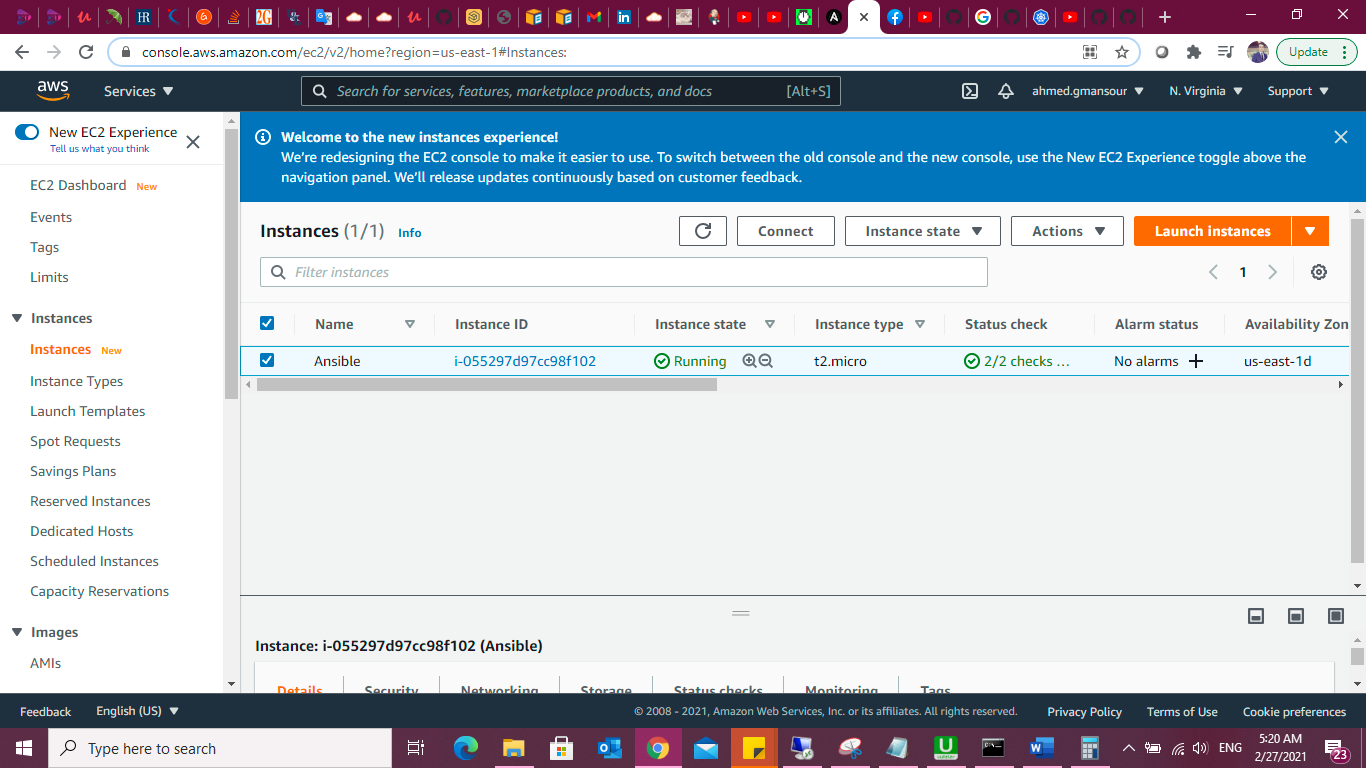
**Connect to the server (Ansible) using SSH:**

#ssh -i C:\Users\Devops\Downloads\EKS.pem [ec2-user@34.224.27.254](mailto:ec2-user@34.224.27.254)

**Then install the below packages:**

#sudo amazon-linux-extras install ansible2

#sudo yum install -y python3 && sudo yum install -y python3-pip



we are using Ansible as our deployment tool. There are many other ways to deploy Kubernetes resources, but I thought Ansible is a much easier option. Ansible uses playbooks to organize its instructions.

Ansible already includes the [k8s module](https://docs.ansible.com/ansible/latest/modules/k8s_module.html?ref=hackernoon.com) for handling communication with the Kubernetes API server.

**The**[**Jenkins Operator**](https://jenkinsci.github.io/kubernetes-operator/docs/) is a Kubernetes native Operator which manages operations for Jenkins on Kubernetes. It was built with immutability and declarative configuration as code in mind. The Jenkins Operator is easy to install with just a few manifest and allows users to

configure and manage Jenkins on Kubernetes.

**Create a manifest e.g. jenkins\_instance.yaml with the following data**

apiVersion: jenkins.io/v1alpha2

kind: Jenkins

metadata:

name: example

spec:

master:

containers:

- name: jenkins-master

image: jenkins/jenkins:lts-jdk11

imagePullPolicy: Always

livenessProbe:

failureThreshold: 12

httpGet:

path: /login

port: http

scheme: HTTP

initialDelaySeconds: 80

periodSeconds: 10

successThreshold: 1

timeoutSeconds: 5

readinessProbe:

failureThreshold: 3

httpGet:

path: /login

port: http

scheme: HTTP

initialDelaySeconds: 30

periodSeconds: 10

successThreshold: 1

timeoutSeconds: 1

resources:

limits:

cpu: 1500m

memory: 3Gi

requests:

cpu: "1"

memory: 500Mi

seedJobs:

- id: jenkins-operator

targets: "cicd/jobs/\*.jenkins"

description: "Jenkins Operator repository"

repositoryBranch: master

repositoryUrl: <https://github.com/jenkinsci/kubernetes-operator.git>

**Ansible uses playbooks to organize its instructions. Our playbook.yml file looks as follows:**

- hosts: NodeGroup

  tasks:

  - name: Deploy the service

    k8s:

      state: present

      definition: "{{ lookup('template', 'jenkins\_instance.yaml') | from\_yaml }}"

      validate\_certs: no

      namespace: default

#ansible-playbook -i Playbook.yaml

Browse to <http://localhost:8080> or Public IP (or whichever port you configured for Jenkins when installing it) .

**Watch the Jenkins instance being created:**

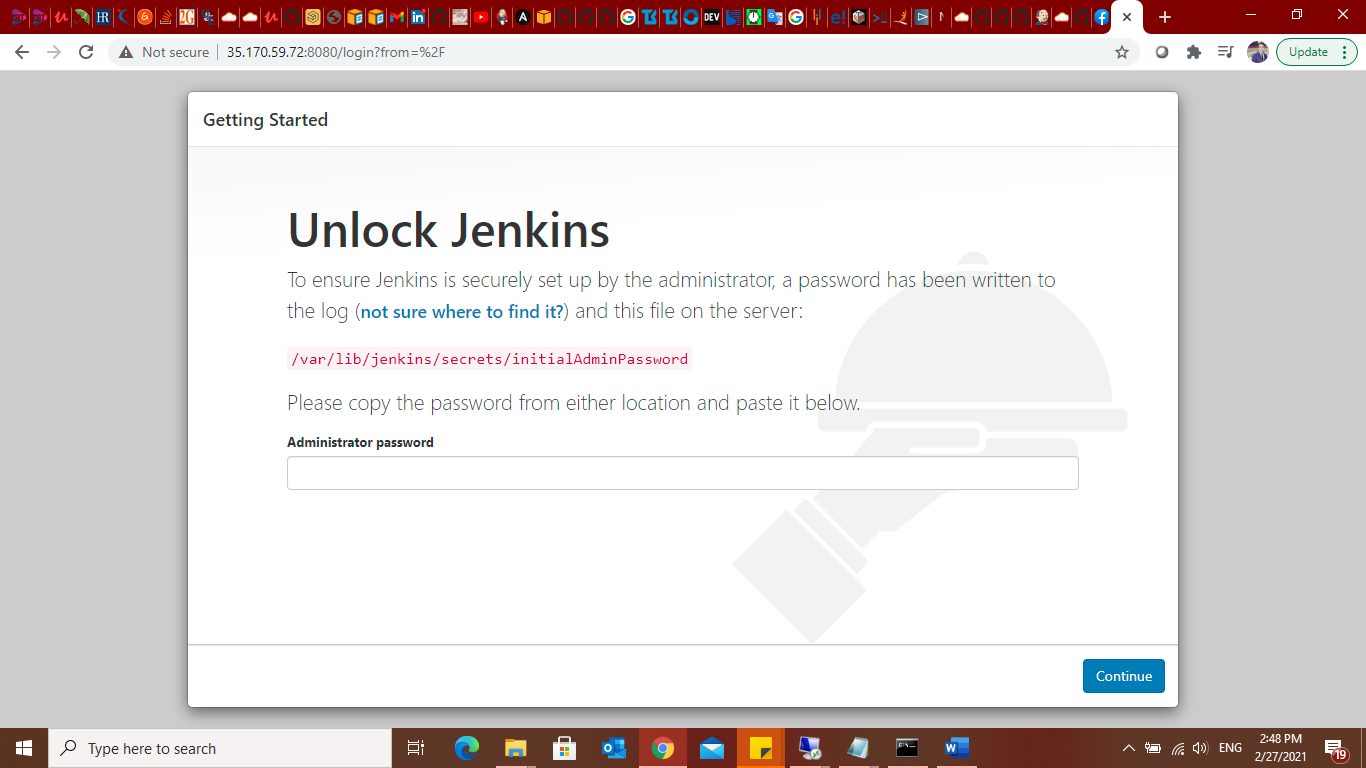
$ kubectl get pods -n Jenkins

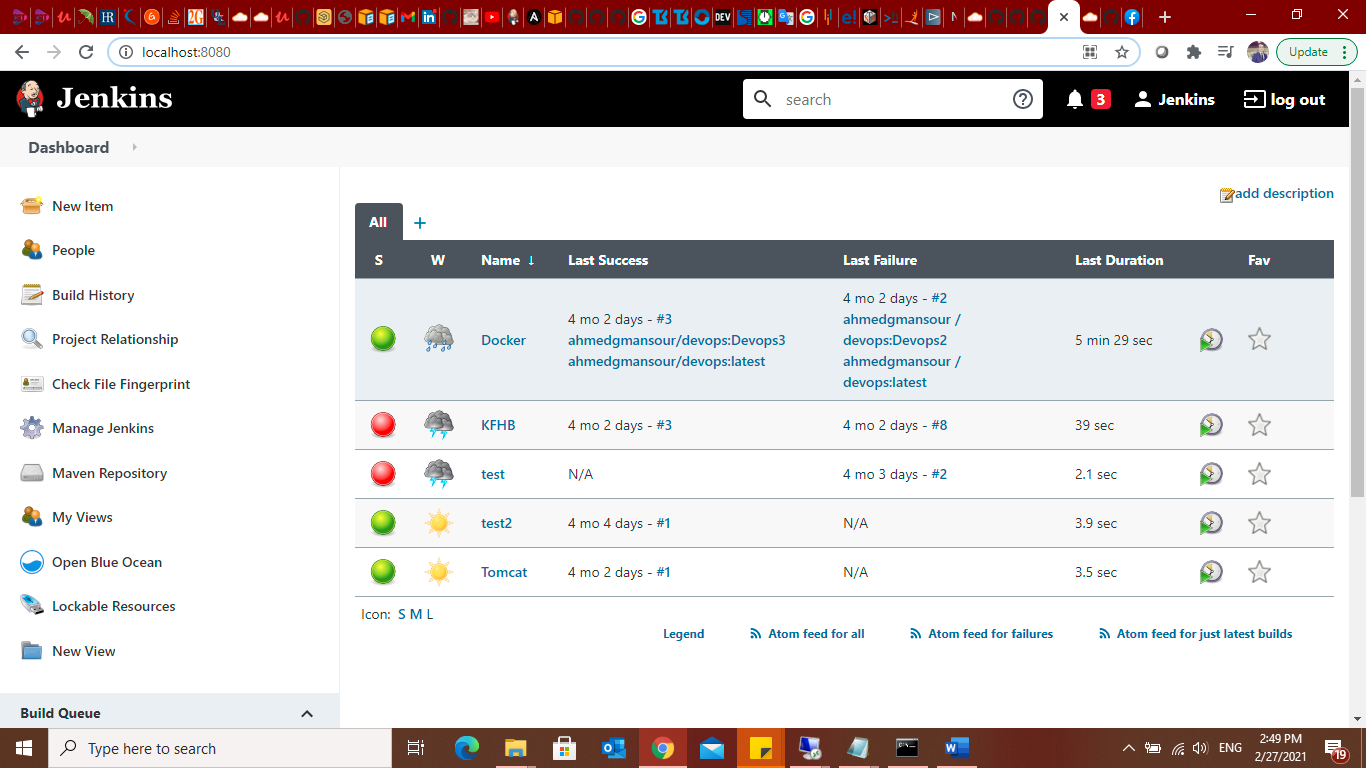
**Connect to Jenkins (actual Kubernetes cluster)**

$ kubectl port-forward Jenkins-AWS\_EKS 8080:8080

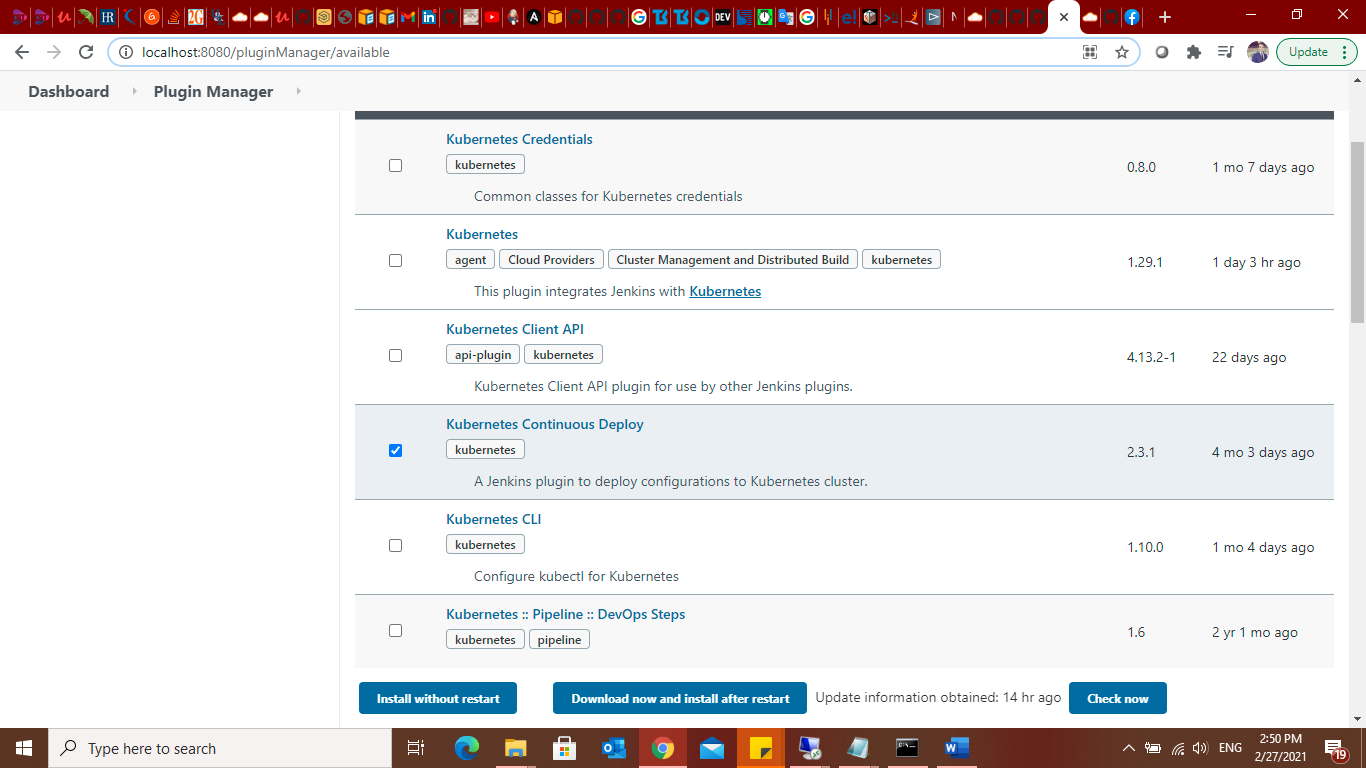
**CI/CD pipeline:**

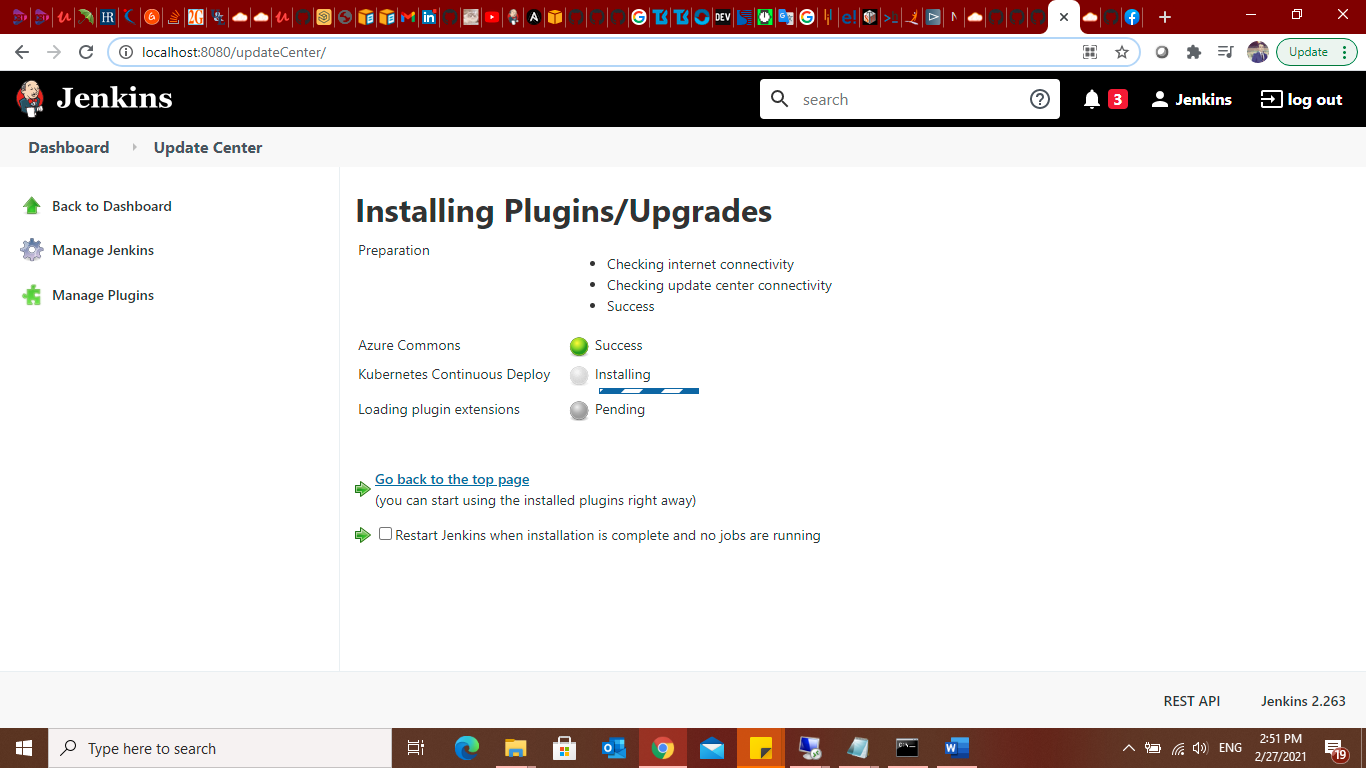
**Log in to Jenkins:**



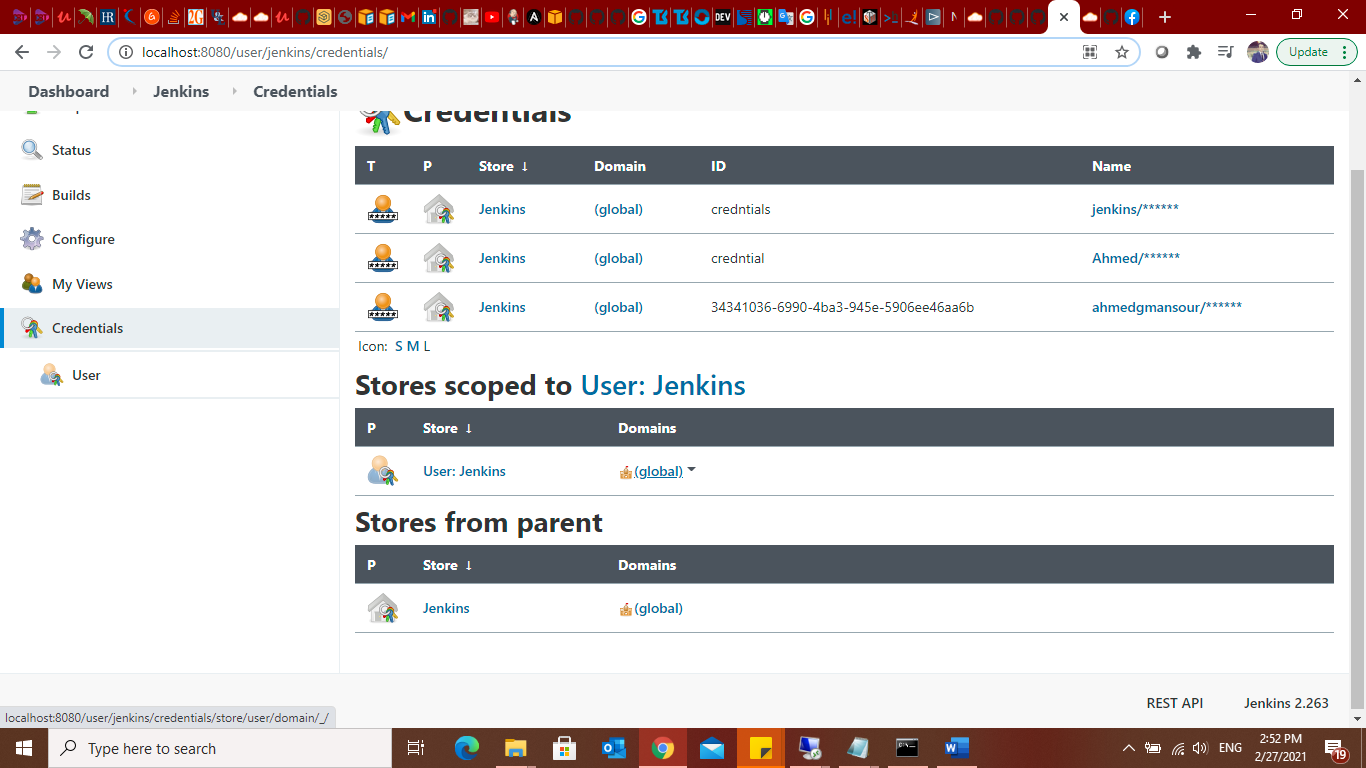


**1-Install plugins of Kubernetes.**

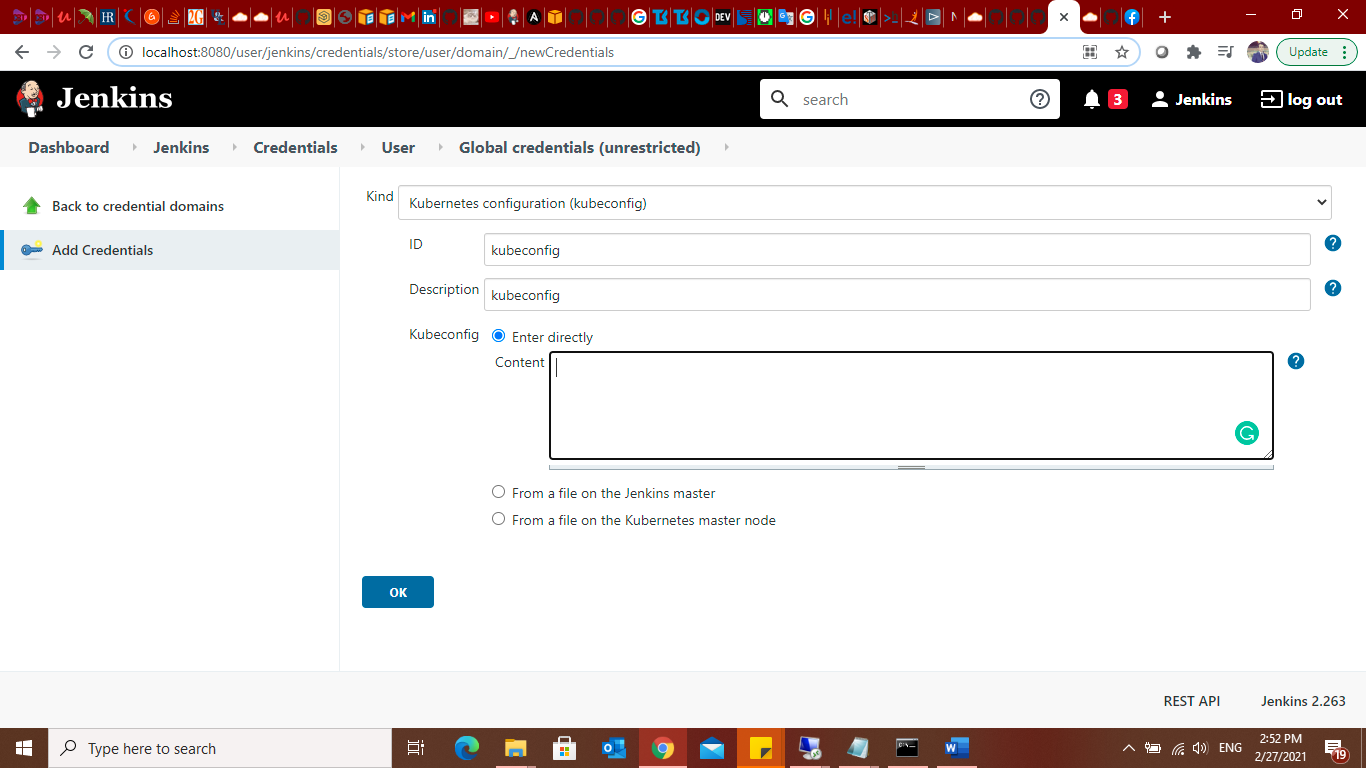




**2- add Kubernetes credentials.**



**3- will add my kubernetes credentials so that jenkins access and authenticate with k8s master to carry out deployment**



**To add kubeconfig key I have to login to the Master Node to get conext:**

**#cat ~/.kube/config**

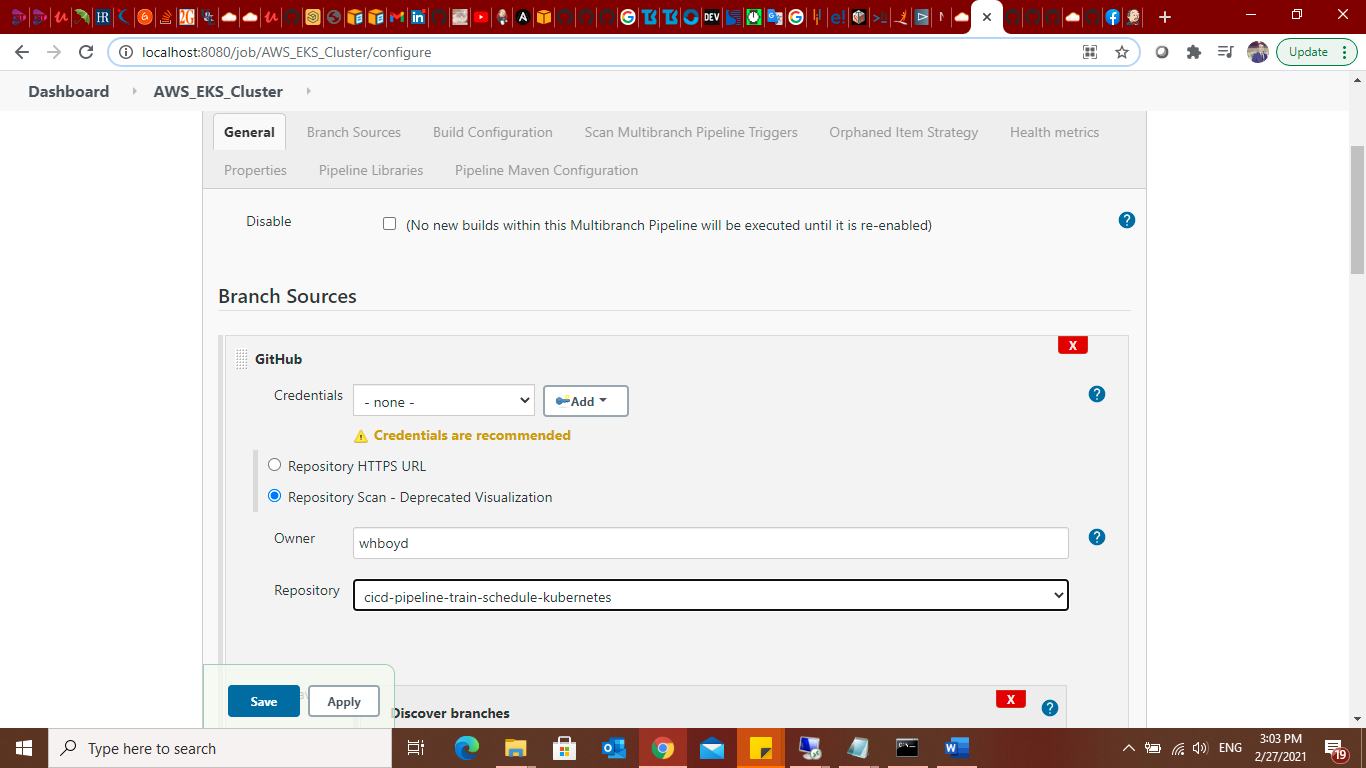
**The copy the entire context to kubconfig in jenkins configuration.**

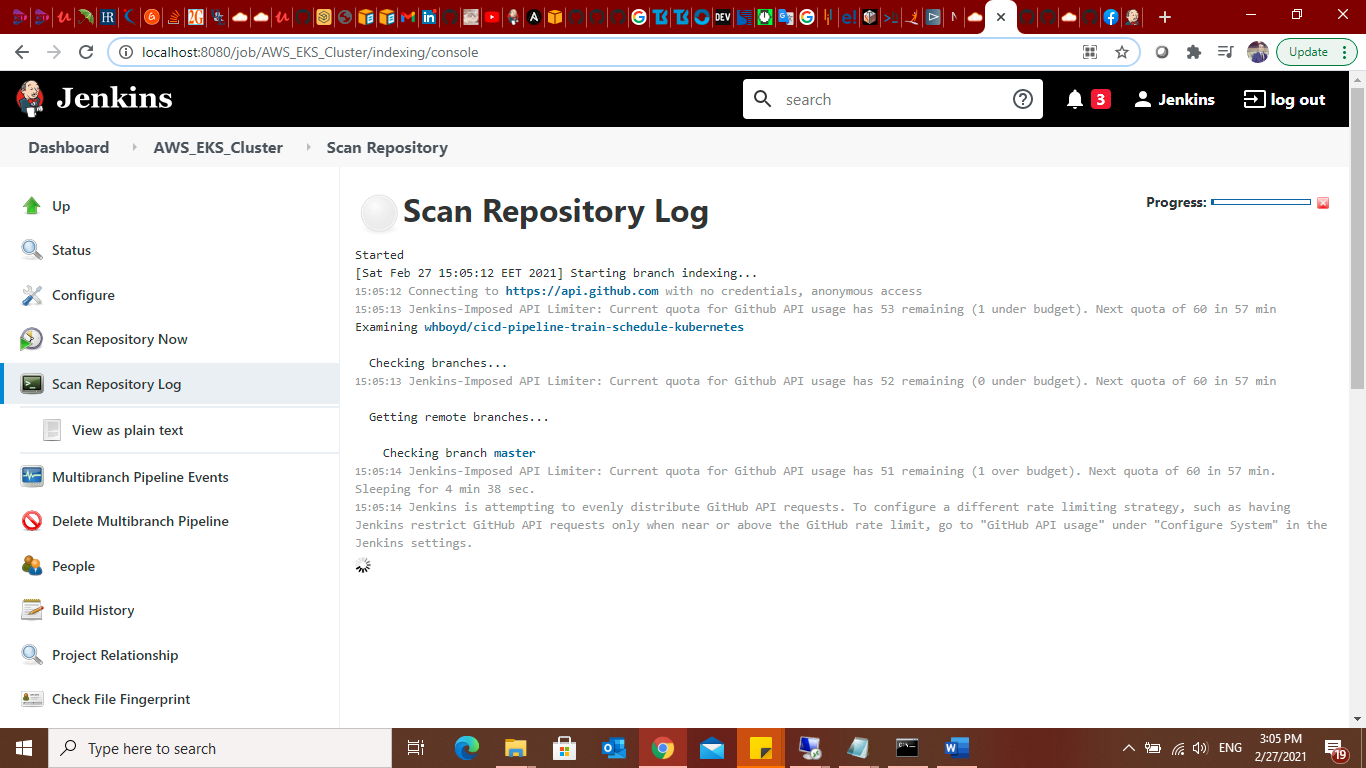
**Create Pipeline:**



**Then choose multibranch pipleline.**



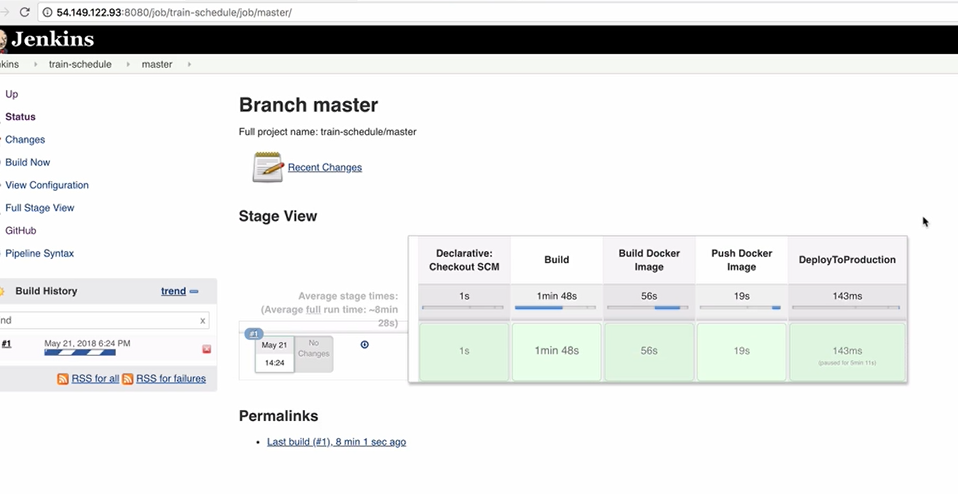




**Deployiong on kuberentes :**

**Create new file (AWS\_EKS.yml)**

**After committing it will automatically build thew new changes.**

****

**Kubernetes spinning up my application.**

