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

## 

**A typical pre-DevOps situation**

We describe the common conditions, in which companies develop their software and conduct IT operations. These conditions may be far from infallible, and the *flaws* in them may determine the companies' decision to shift to DevOps as a way to improve their software development process and IT operations.

## Logo, company name Description automatically generated

**Traditionally, there are three main options to organize software development:**

1. Software is developed by a company’s in-house team with a good deal of financial and human resources dedicated to the development process.

2. Software development is outsourced to a third party, in case there’s a lack of IT resources for development.

3. Software development is conducted with in-house efforts but, for instance, quality assurance (QA) is delegated to a third-party QA services provider due to the gap in QA competencies in a company.

**High probability of post-release errors**

Test engineers conduct functional, performance, integration, security, usability, and other types of testing. However, continuous testing is not implemented at each stage of the software development process. Due to the resulting testing gaps, software users often detect severe post-release bugs. Further, test engineers may not be able to reproduce the identified defects in the testing environment. This may happen because of the differences in:

Configurations of testing and production environments.

Build versions deployed in production and in testing environment.

**Lack of users’ trust in software quality**

Due to severe post-release errors, business users are not sure of software quality and have to conduct thorough manual acceptance testing. And because potential users are busy with their main responsibilities, their feedback comes with considerable delays.

**Weeks for updates and fixes**

As the collaboration between the teams involved in the software development process is not efficient enough, it generally takes 2-4 weeks to detect and fix bugs or implement and release minor changes in the software. Such a long waiting period is especially harmful when software under development is supposed to support or transform critical business operations, for instance, if it is a supply chain management application or a CRM software.

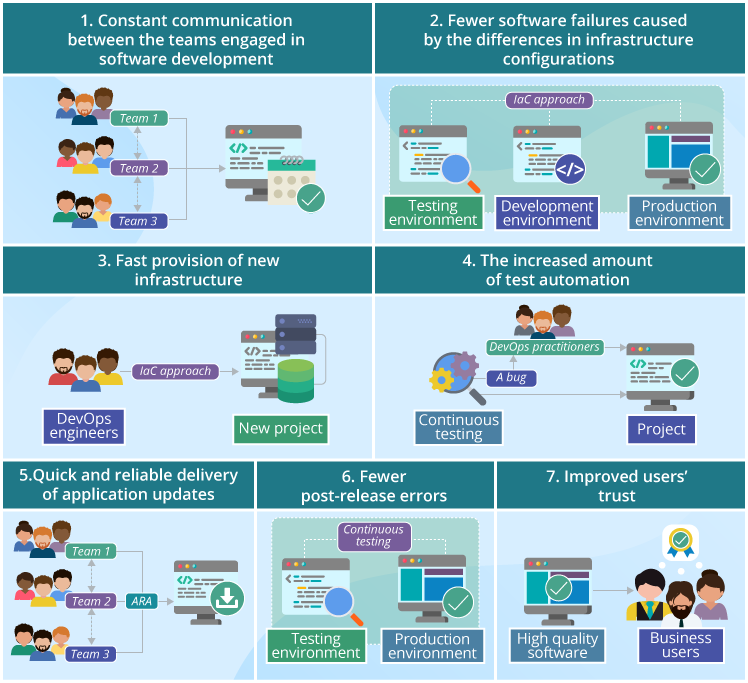
**Time-consuming deployment of the infrastructure**

System administrators spend days or even weeks to prepare a new development, testing and production infrastructure for a new project. Misconfigurations often occur due to the manual nature of infrastructure preparation. Modifications and tuning take even more time.

**What to expect from DevOps:**

To overcome the drawbacks of a traditional way to develop software and carry out IT operations, we suggest considering the DevOps approach.

Due to a range of new practices and technologies we describe further in the article, businesses implementing DevOps can develop and deliver stable applications with thoroughly tested functionality faster than organizations sticking to the traditional approach to software development and operations. We accumulated our DevOps-related experience to provide you with a list of the key advantages DevOps brings.



**Constant communication between the teams engaged in software development**

The implementation of the DevOps approach presupposes that developers, QA engineers and system administrators now work in the alignment with each other. The ongoing communication between the DevOps practitioners allows preparing and delivering new software pieces to production faster and with a decreased number of bugs missed.

**Fewer software failures caused by the differences in infrastructure configurations**

With DevOps, it’s possible to create identical working environments for development, testing and IT operations teams. It becomes achievable when [infrastructure as code](https://dzone.com/articles/bliki-infrastructure-as-code) (IaC) is applied. A DevOps engineer can create a new development or testing environment aligned with the production environment. After that, developers and test engineers can work with new builds being sure that the created development and testing environments are completely identical to the production.

As a result, there will be no situations when software works properly in the testing environment but fails on production.

**Fast provision of new infrastructure**

To quickly prepare and deliver new infrastructure for development, testing and production, DevOps engineers apply the IaC approach. As the infrastructure exists as ready-made code, any time developers need a new infrastructure for a new project, they don’t have to wait before system administrators provide them with it.

**The increased amount of test automation**

Continuous testing is one of the key components of the DevOps approach. To achieve it, testing is automated with the use of specialized tools, e.g., Selenium, Zephyr, and Tricentis Tosca that are designed to automatically perform various types of tests (such as unit, functional, integration testing) and promptly notify the DevOps practitioners on detected bugs.

**Quick and reliable delivery of application updates**

Thanks to close collaboration between the DevOps-related teams and implementing application release automation (ARA), software is updated faster than in the traditional process of software development. ARA allows accelerating the process of deploying new builds with the minimal downtime and fewer configuration errors usually occurring when the deployment of new build is carried out manually.

**Fewer post-release errors**

With the implementation of continuous testing and the alignment of testing and production environments, the QA team spends much less time on QA and testing activities and misses fewer bugs.

**Improved users’ trust**

Within the DevOps approach, the rate of test automation increases while the number of post-release errors declines. To convince business users of high software quality, it’s important to establish strong communication with them. Business users can be involved in defining the tests to be automated (including acceptance tests as well) to ensure sufficient testing coverage for the software functions that are the most critical for them. Since business users know that software functionality is tested thoroughly, their confidence in the application quality rises. Thus, they lose the need to carry out their own manual acceptance tests afterwards, which reduces the overall lead time.

DevOps stands for Development and Operations. It is a new form of software development that has revolutionized the way software products are developed and distributed. The DevOps methodology focuses on providing frequent smaller upgrades rather than rare big feature sets.

IT operations benefit from DevOps. Before the advent of DevOps, several concerns remained intrinsic to the IT team. This resulted in the IT team facing some level of undesired suspicion.

You can summarize by saying Agile software development methodology focuses on the development of software but DevOps on the other hand is responsible for development as well as deployment of the software in the safest and most reliable way possible.

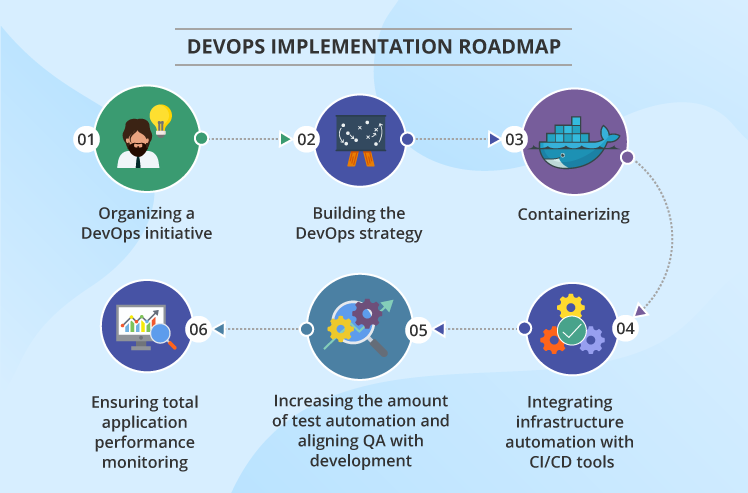
However, that all changed with the entry of DevOps, which allows the IT operations to share these concerns with the rest of the organization resulting in enhanced transparency and better coordination between the IT operations and the other teams.

Nowadays, DevOps are in great demand in the current industry, and many businesses are eagerly wanting to invest in DevOps talent. Some of the huge multi-national companies such as Facebook and Netflix are investing their money and time in DevOps for automation and pacing up application deployment as every large industry wants to see some automation in the coming years. It helps the organizations to grow and expand their businesses to generate large revenues. Its popularity continues to grow in demand as tech competition increases as most companies start adopting DevOps practices; then, it becomes even more important for the competitors to invest in similar or better development practices, increasing demand.

DevOps implementation has given provable results in businesses which contend higher efficiency, with its new technology standards; tech workers can implement codes faster than ever before, and with lesser errors. As now, more consumers and businesses rely on cloud software as it requires fast deployments to meet the consumer needs without interrupting services, this increases user adoption of cloud software like DevOps over the years.

**DevOps implementation roadmap**

Once you have considered all the factors and decided to implement DevOps within your company, you need a step-by-step plan to smoothly shift from traditional software development to the DevOps approach. Here, we outline the key steps.



**1. Organizing a DevOps initiative**

A company’s CIO organizes a DevOps initiative as a part of the IT department program. Thus, the IT department gets an opportunity to make changes in the development and operations activities the least painfully for the entire company. CIO, in his turn, is able to arrange financial investments, human resources in the most optimal way. A program manager becomes responsible for designing the DevOps strategy and monitoring its implementation.

**2. Building the DevOps strategy**

To draw up an effective DevOps strategy, a program manager should utilize best practices that will improve the interdepartmental collaboration and enable the new ways of infrastructure provisioning, software development and testing. Among the most important practices are the following:

Put the company’s development, testing, design, operations and other teams in a shared working DevOps environment, making all the members focus on the outcomes of the software development cycle and understand each other’s motives and duties. Set the common goal – to accelerate a software development cycle and ensure the high quality of software – for everyone involved in software development and operations.

Implement IaC to ensure the prompt provision of the IT infrastructure upon developers’ or test engineers’ requests, whenever they need it for creating a new build or checking its quality. This will allow DevOps practitioners to get new infrastructure for development or testing in one click, as well as avoid human errors that often result from the manual configuration of IT infrastructures.

Automate software building, unit testing, application testing via UI, software integrating, deploying, and releasing processes to speed up the software development-testing-releasing cycle.

**3. Containerizing**

Containerization implemented with such tool as Docker solves the problem with the reliability of software, for example, when it travels from the development to the testing environment and then to production. Containers include everything required to run an application, i.e. all the dependencies, libraries, configuration files. The isolation of the containerized parts of the software from the overall IT infrastructure allows for their stable running regardless of the differences in the environments they are put in.

Moreover, since the pieces of application (its database, front end, etc.) are put into several containers, it is easier for an operations team to manage the application, since they have no need to rebuild the entire software when the changes are required in one of its microservices.

**4. Integrating infrastructure automation with CI/CD tools**

When software is put into containers, the containerized application needs to be managed properly. Such infrastructure automation tools as Kubernetes, Ansible, Chef, or Puppet are integrated with CI/CD tools like Jenkins, Bamboo, or GoCD for more efficient configuration management and software deployment.

For example, Kubernetes – for large infrastructures – or Ansible – for smaller ones – allows managing containers for fault tolerance, monitoring their health and rolling software updates, and Jenkins is used to create, test and deploy new builds into Kubernetes.

**5. Increasing the amount of test automation and aligning QA with development**

To achieve faster delivery with DevOps, sufficient automated testing must be ensured. However, not each testing type must be automated. For instance, exploratory, usability, and security testing should still be performed manually. Depending on the efforts needed to write automated tests, functional testing may partially remain manual.

The development and testing activities are carried out in tandem to avoid post-release bugs. While the application is still in development, the best practice is to conduct automated tests 1-2 times a day. In case defects are found, developers work on stabilizing software before releasing the next build.

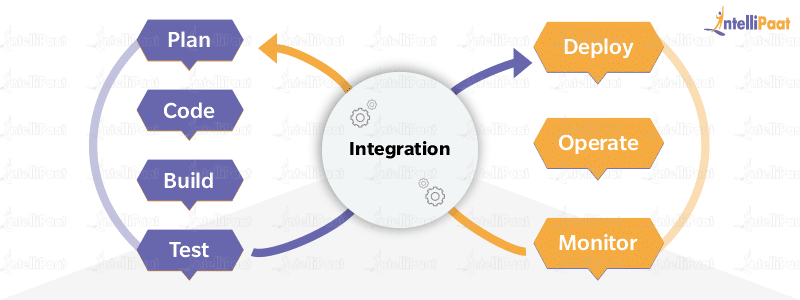
**6. Ensuring total application performance monitoring**

Application performance monitoring provides the DevOps-related teams with transparency over all the performance issues, e.g., slow response, memory leaks, runtime errors. The issues may be revealed during application server monitoring, user experience monitoring, etc.

Application performance monitoring allows detecting, prioritizing and isolating application defects before end users find them, as well as finding the root causes of the errors quickly with the use of special application monitoring software, such as Zabbix, Nagios, or Prometheus customized for monitoring a particular application.

 What is CI? What is its purpose?

CI or Continuous Integration is the process of compiling the entire code base, every time a member of the software development team checks the code, into the shared source code repository.

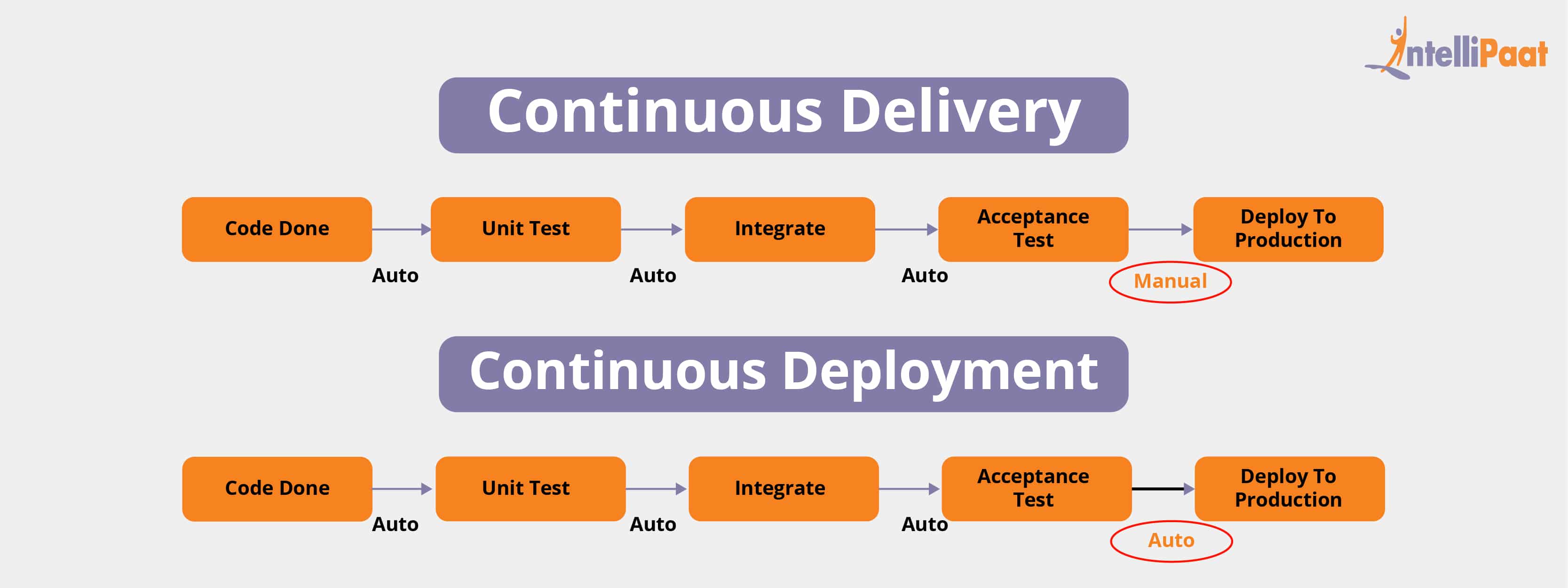


If a team member checks into the code file with a bug, then the build gets broken. In this sort of scenario, other developers can’t synchronize the shared source code repository without introducing compilation errors into their own local workspaces. Thus, collaborative and shared software development cannot go forward.

When a CI build breaks, it is crucial that the problem is corrected immediately. A CI process often includes a suite of unit, and integration and regression tests that run every time the compilation succeeds. If any of these tests fail, the build will be considered unstable, not broken.

What is the difference between continuous deployment and continuous delivery?

Continuous deployment is fully automated, and the deployment to production needs no manual intervention in continuous deployment; whereas, in continuous delivery, the deployment to production requires some manual intervention for change management in the organization, and it needs to be approved by the manager or higher authorities to be deployed in production. According to your organization’s application risk factor, continuous deployment/delivery approach will be chosen.



## 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

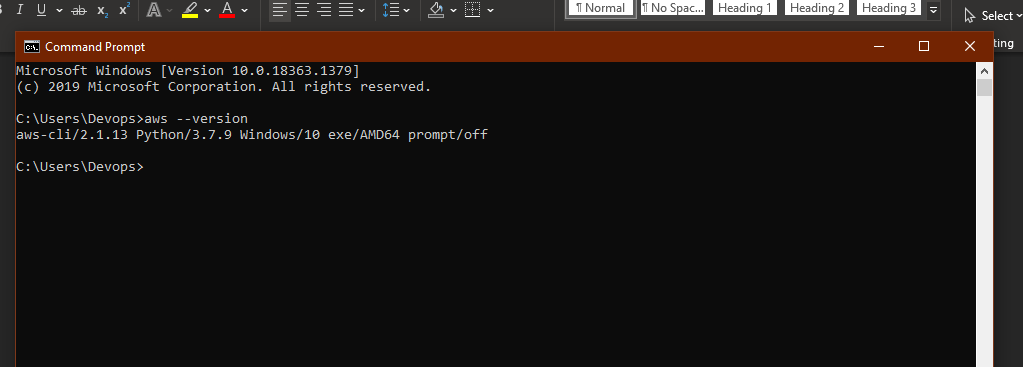
**Ekstl – setup and operation of EKS cluster**

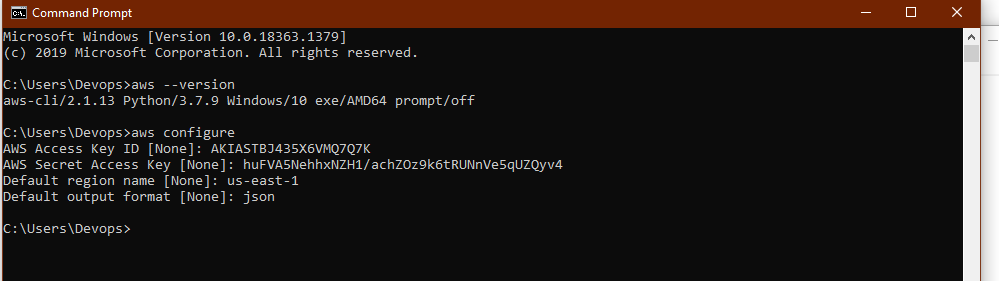
**Setup of eksctl :**

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

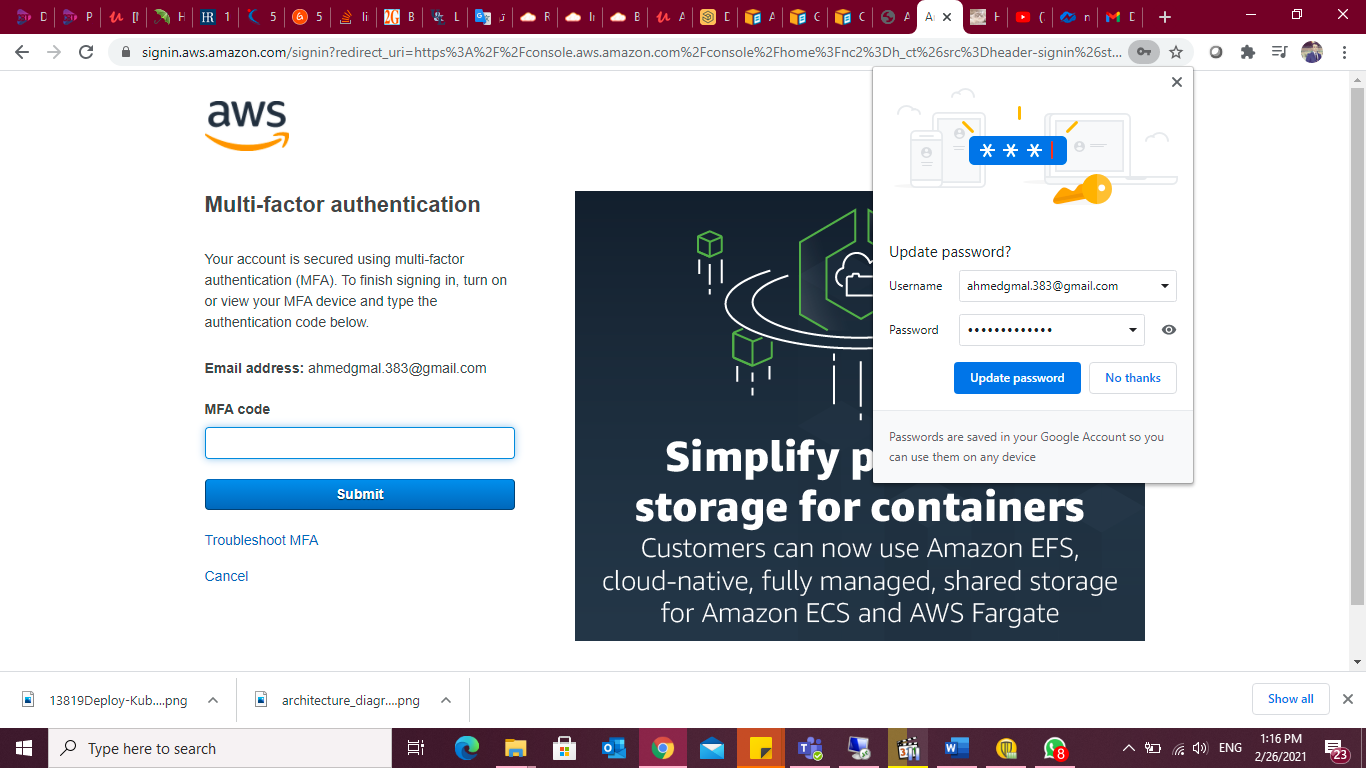
#sudo mv /tmp/eksctl /usr/local/bin

**Setup AWS CLI**



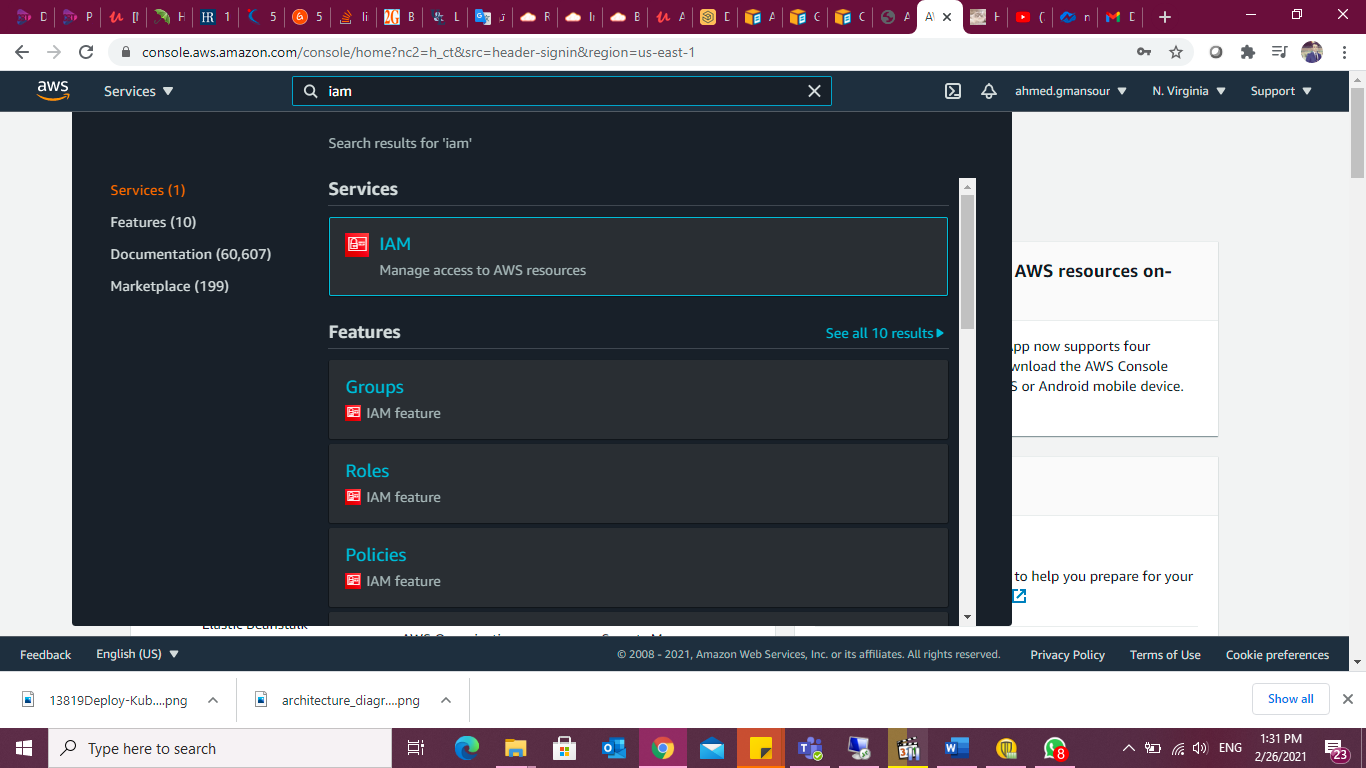


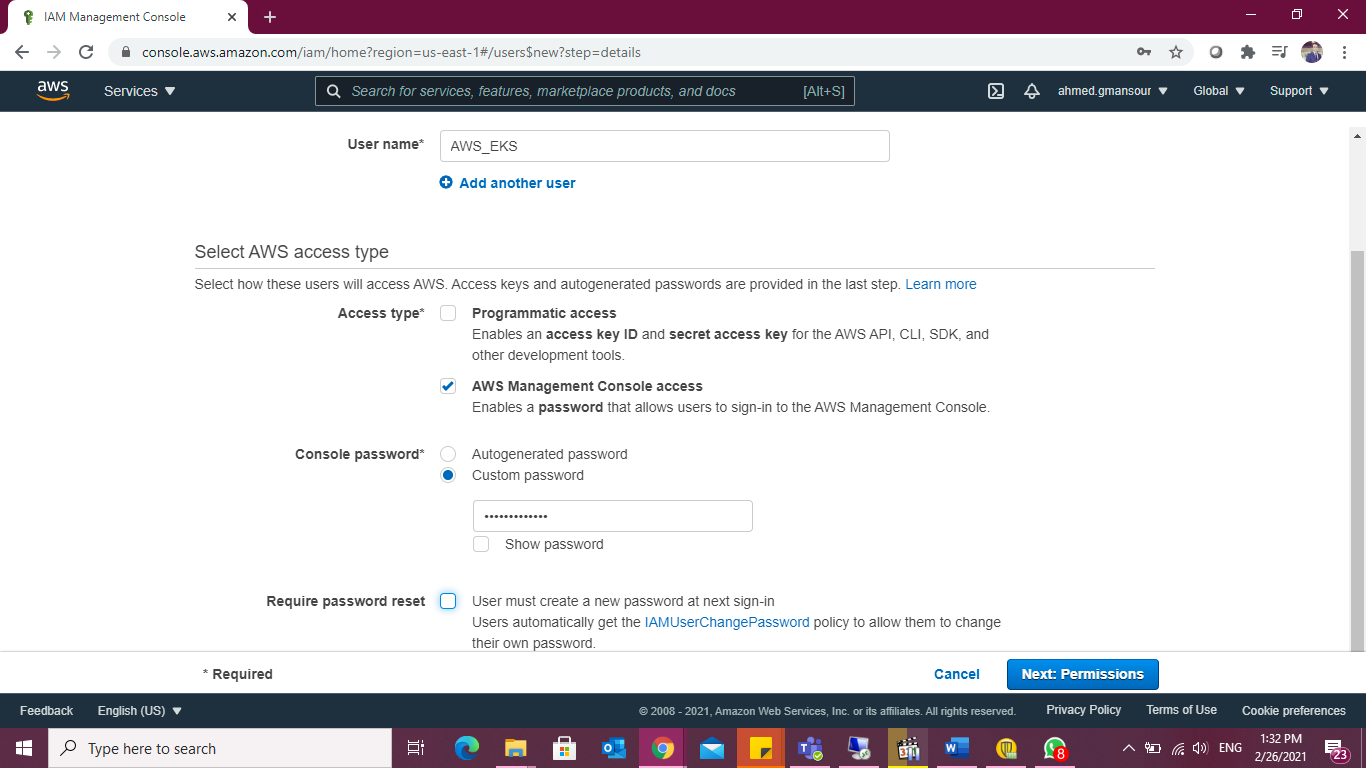
**Sign in to your AWS account at**[**https://aws.amazon.com**](https://aws.amazon.com/)**with an IAM user role that has the necessary**

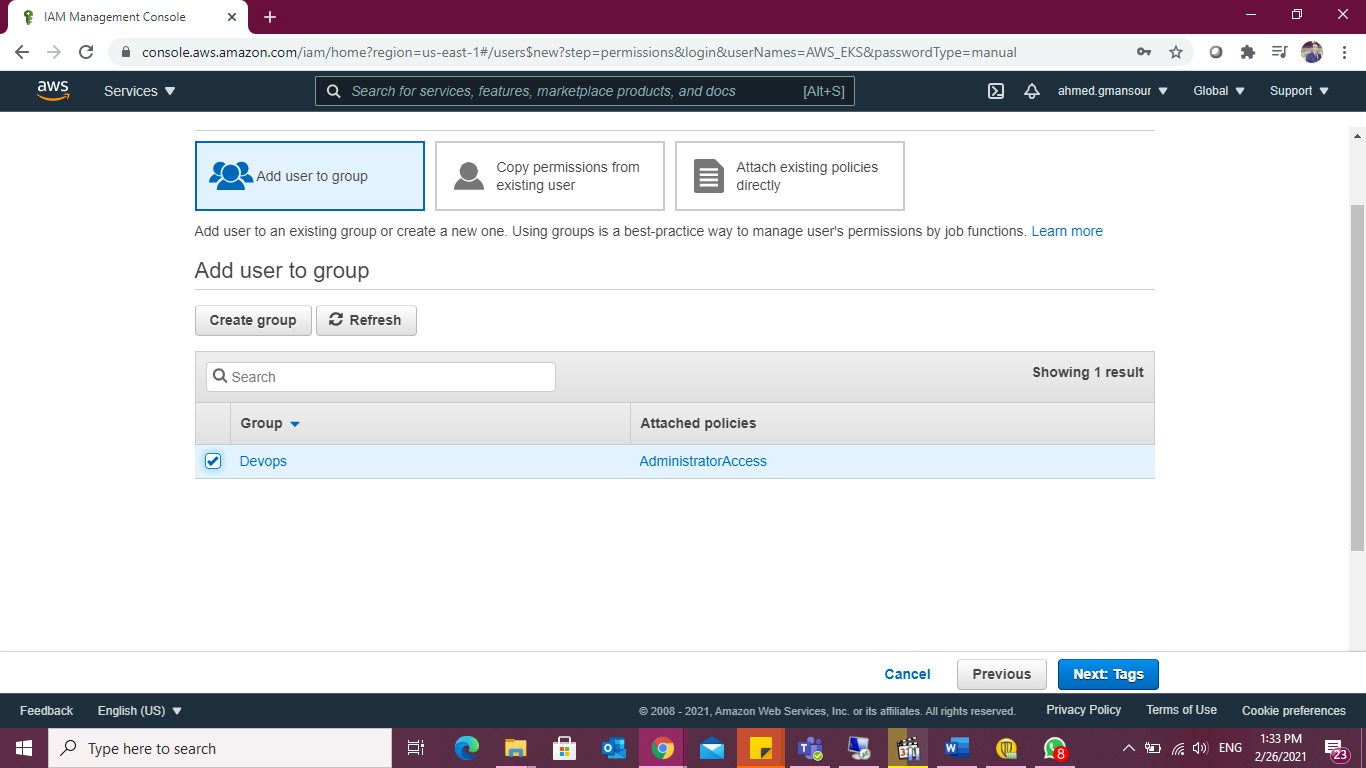


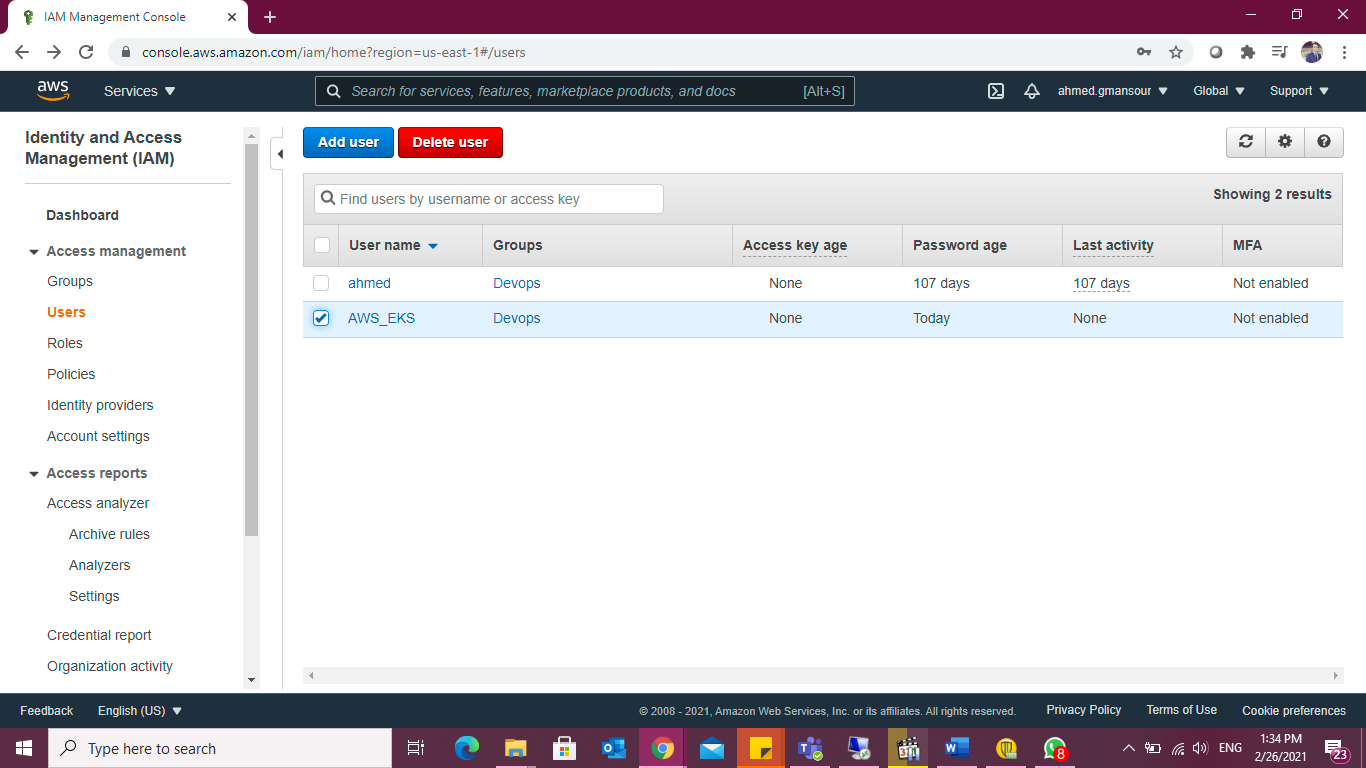
## Step 1: Creating an EKS role

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

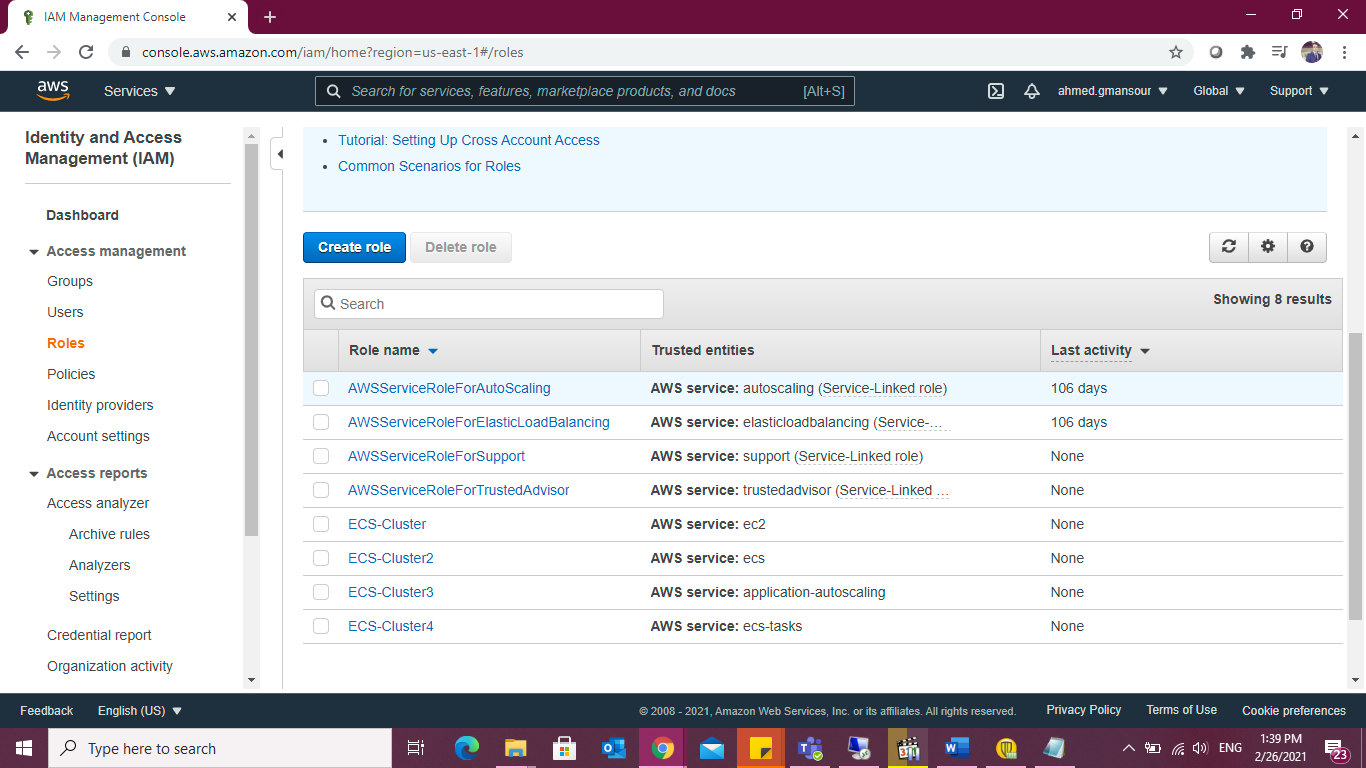




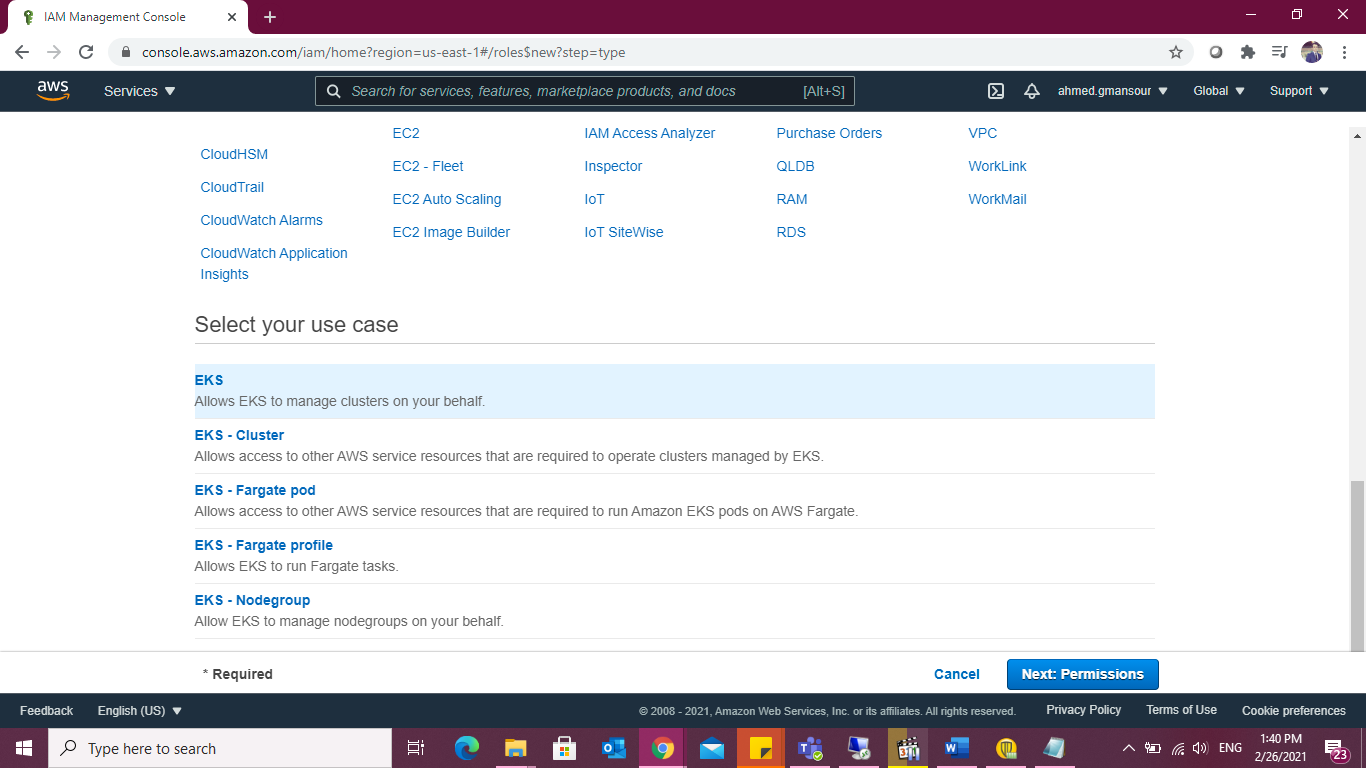


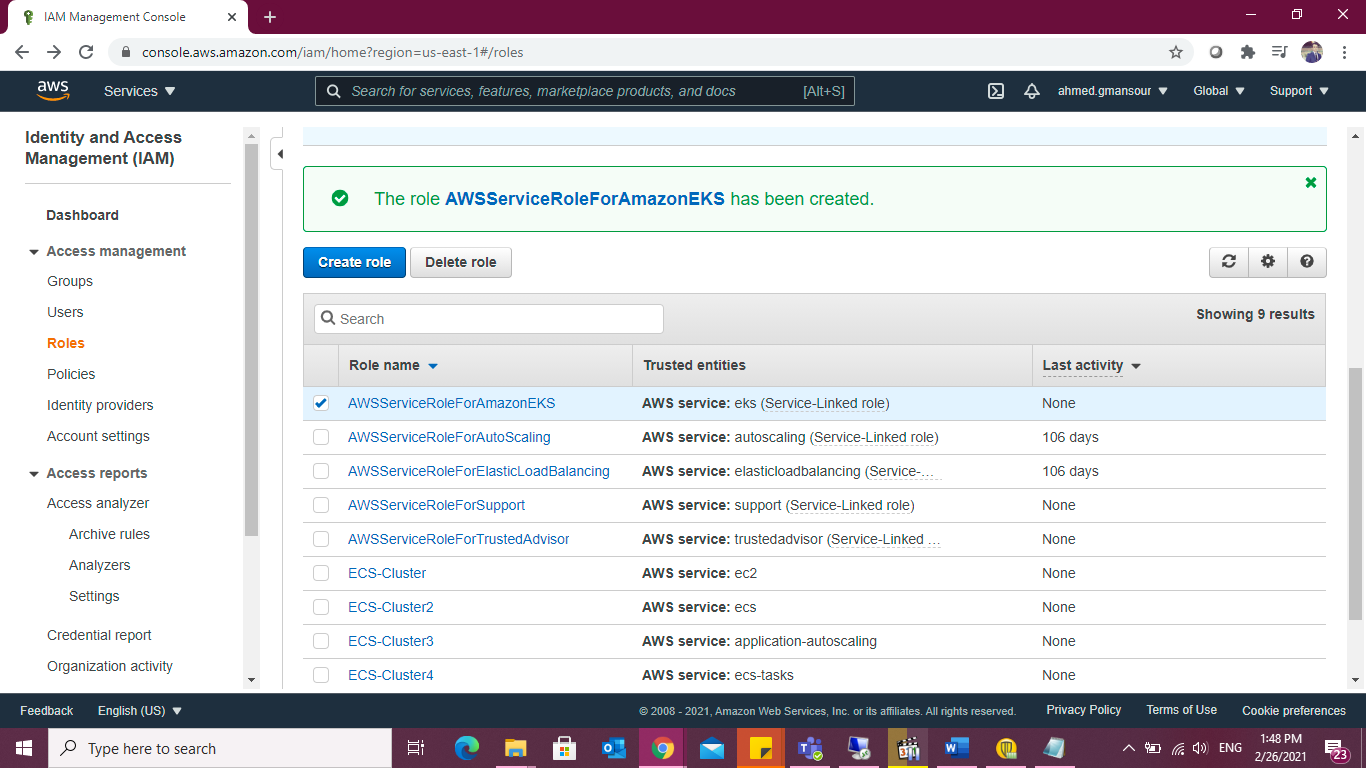


Open the [IAM console](https://console.aws.amazon.com/iam/), select Roles on the left and then click the Create Role button at the top of the page.

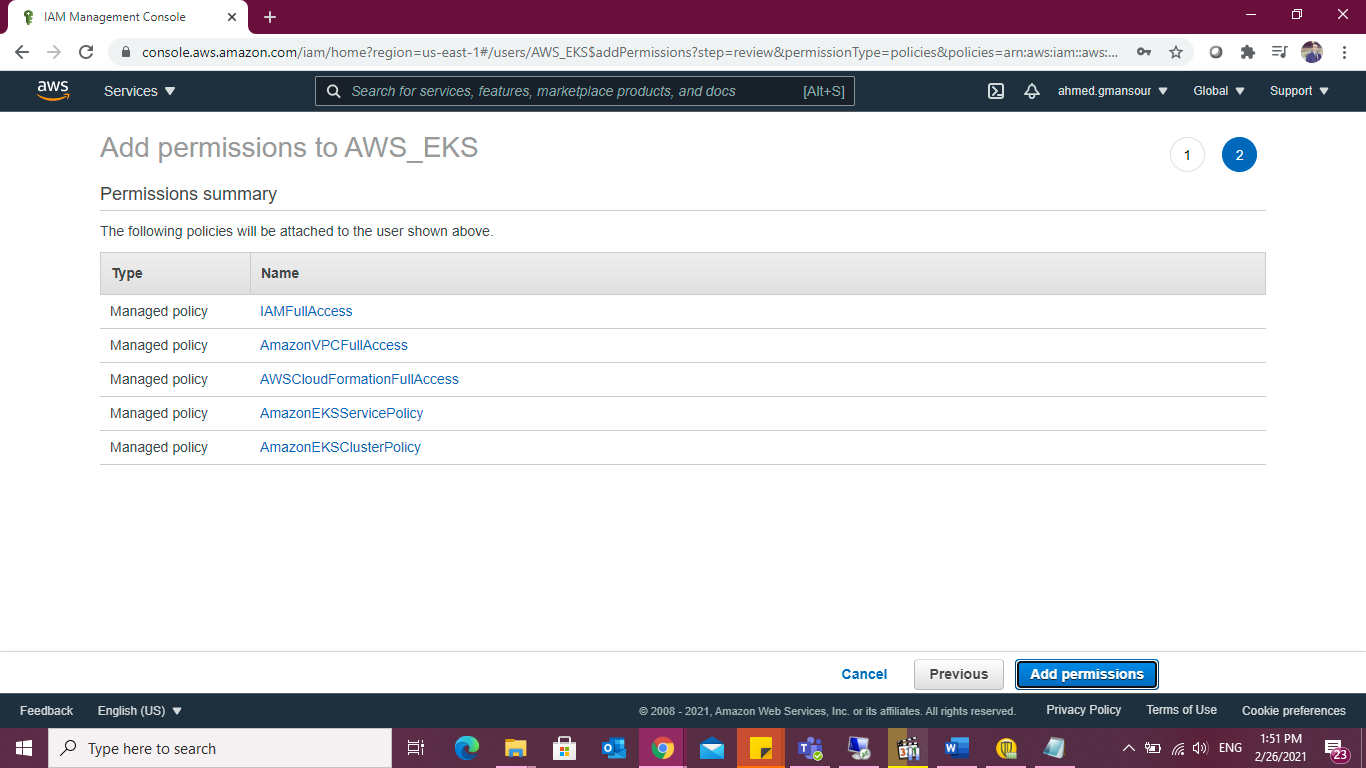


From the list of AWS services, select EKS and then Next: Permissions at the bottom of the page.





Give Permission to the user:



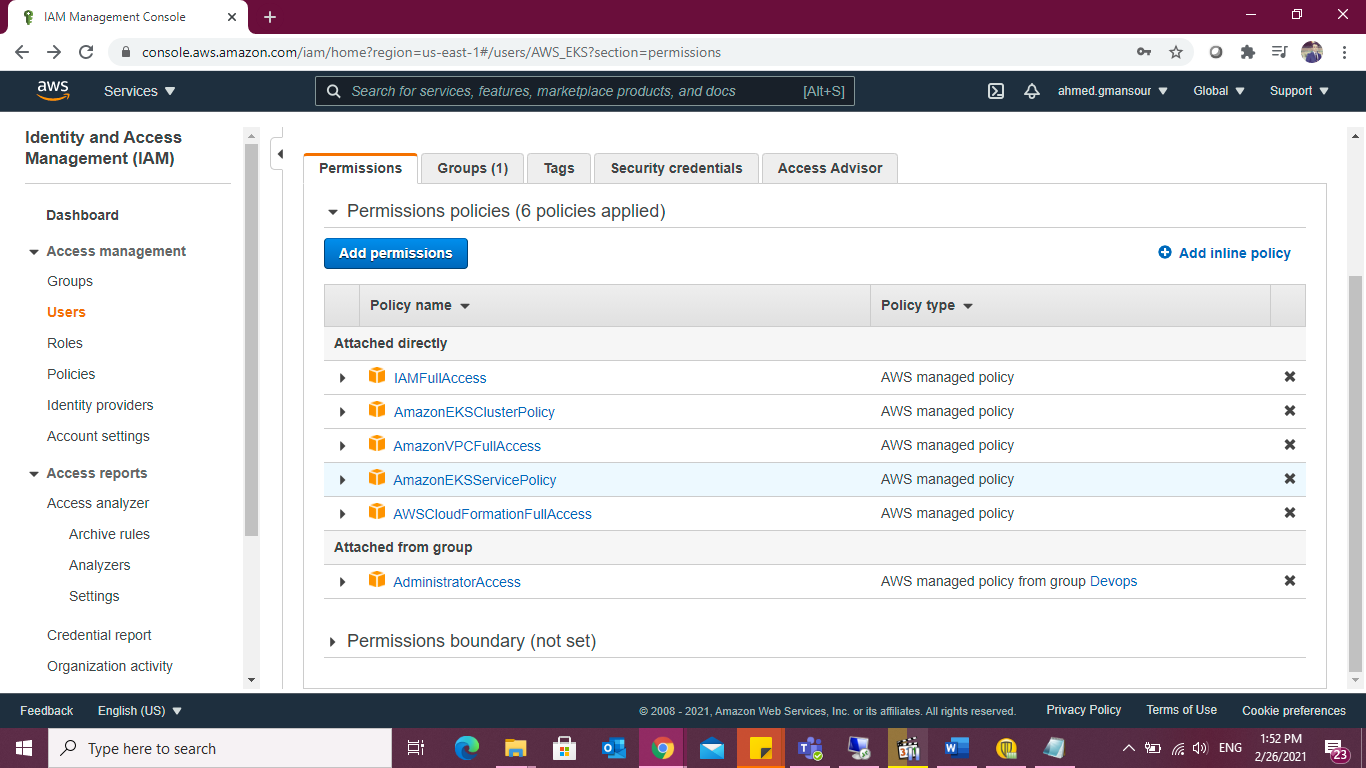
- AmazonEC2FullAccess

- IAMFullAccess

- AmazonVPCFullAccess

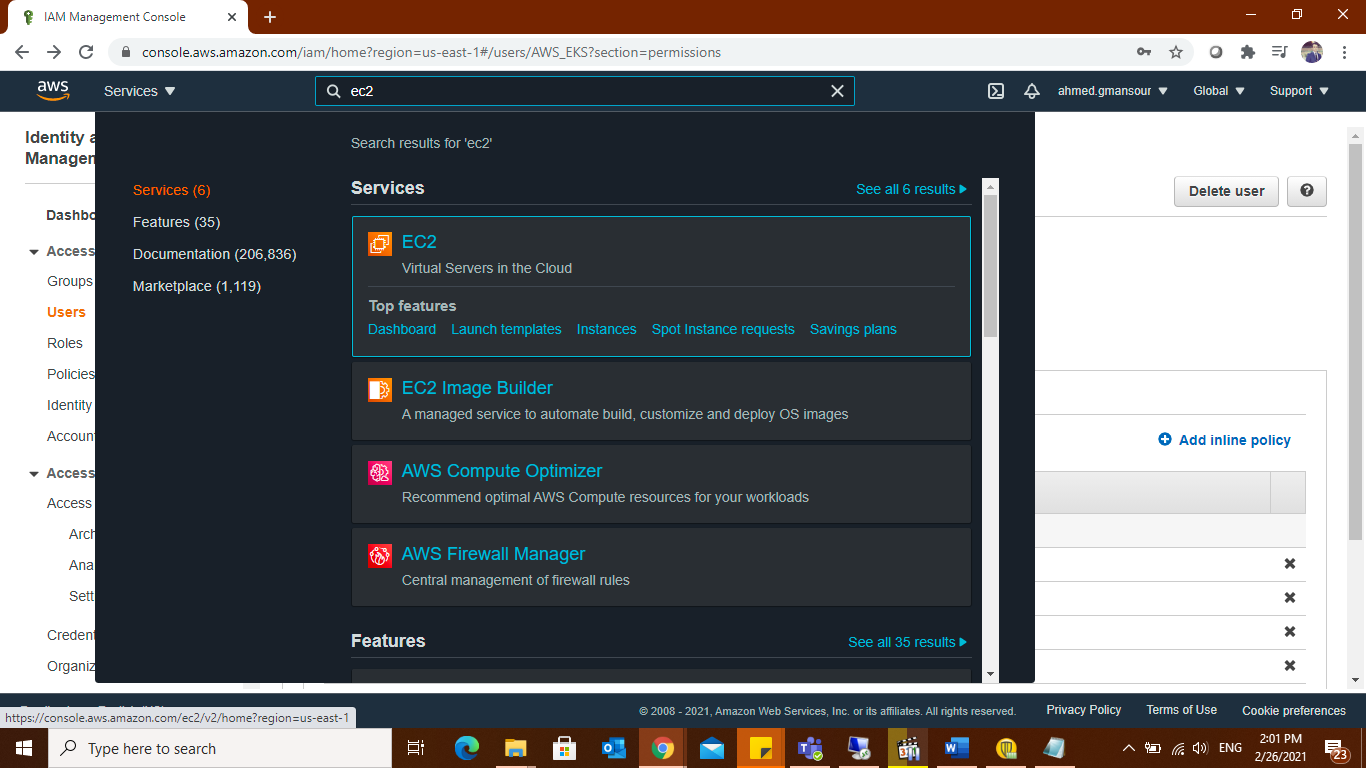
- CloudFormation-Admin-policy

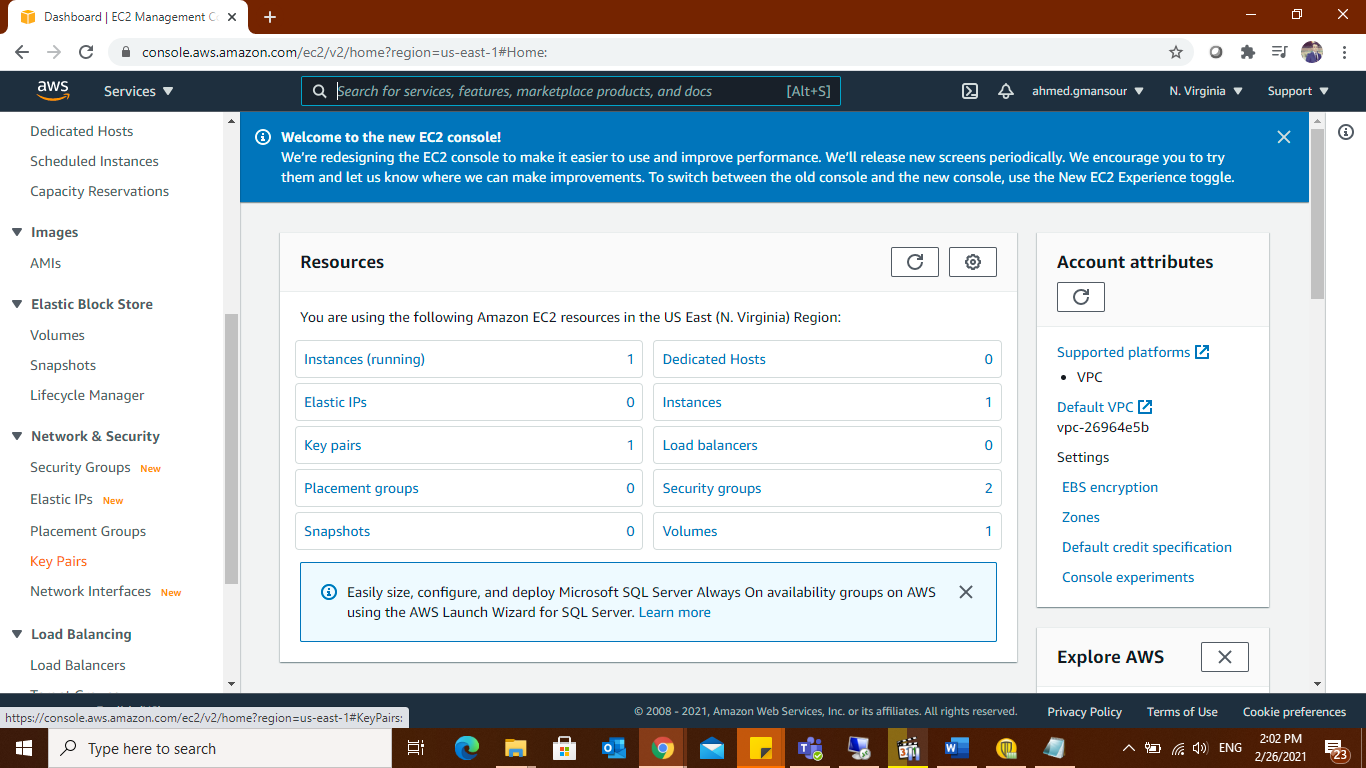
- EKS-Admin-policy

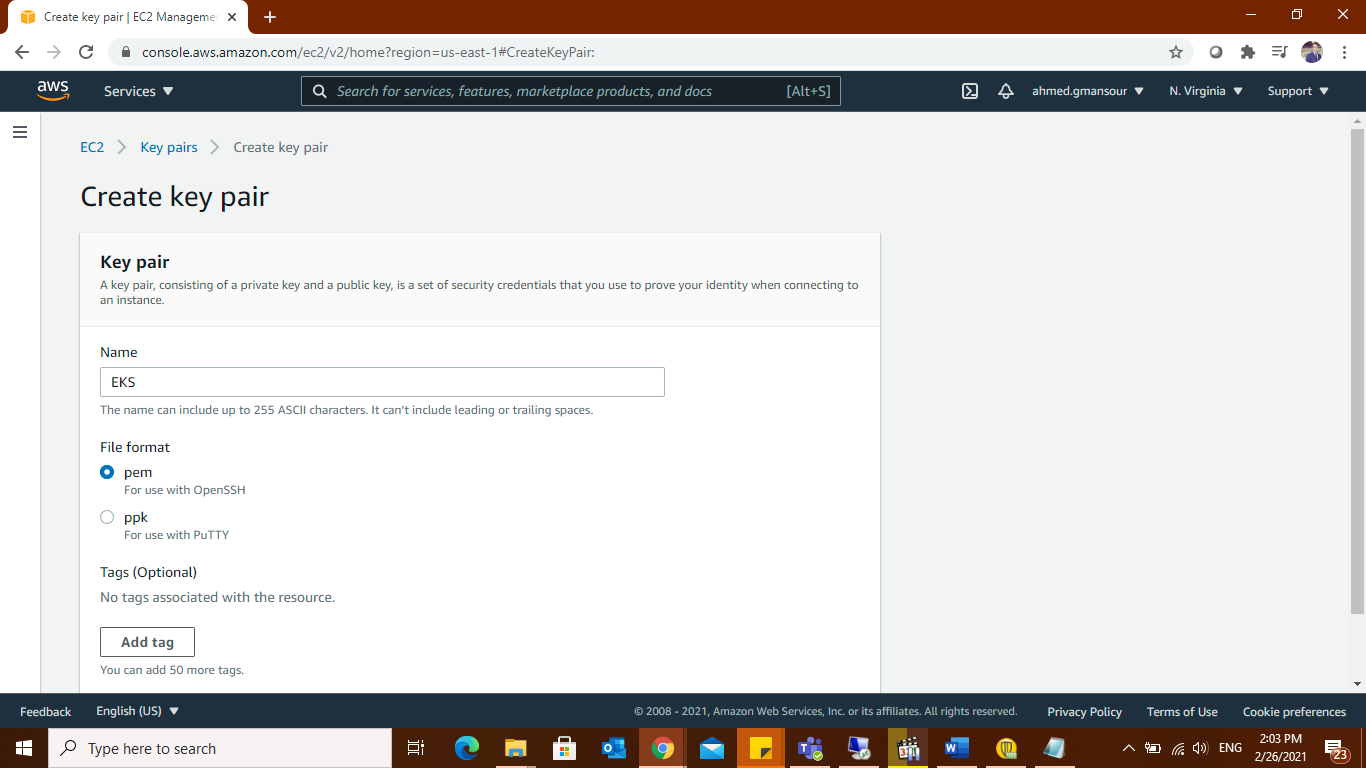


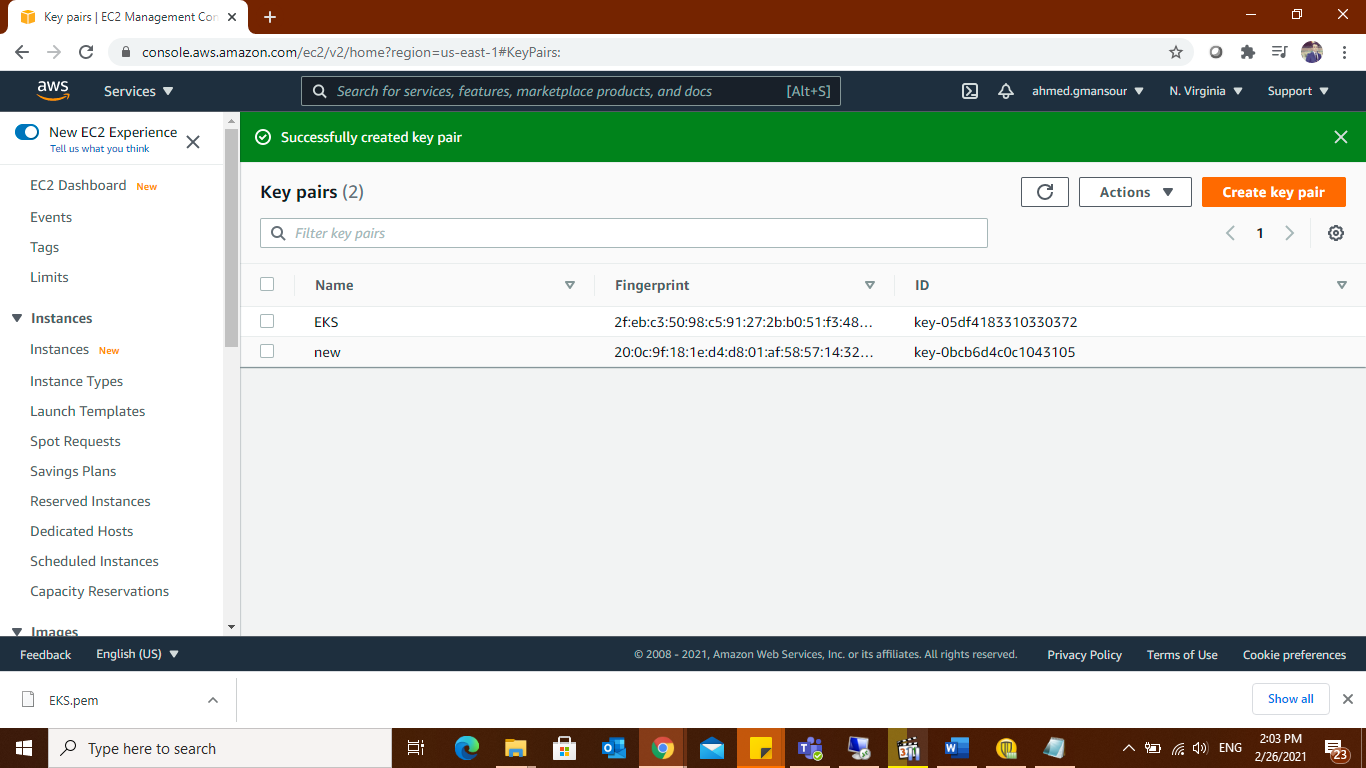
Be sure to note the Role ARN, you will need it when creating the Kubernetes cluster in the steps below.

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



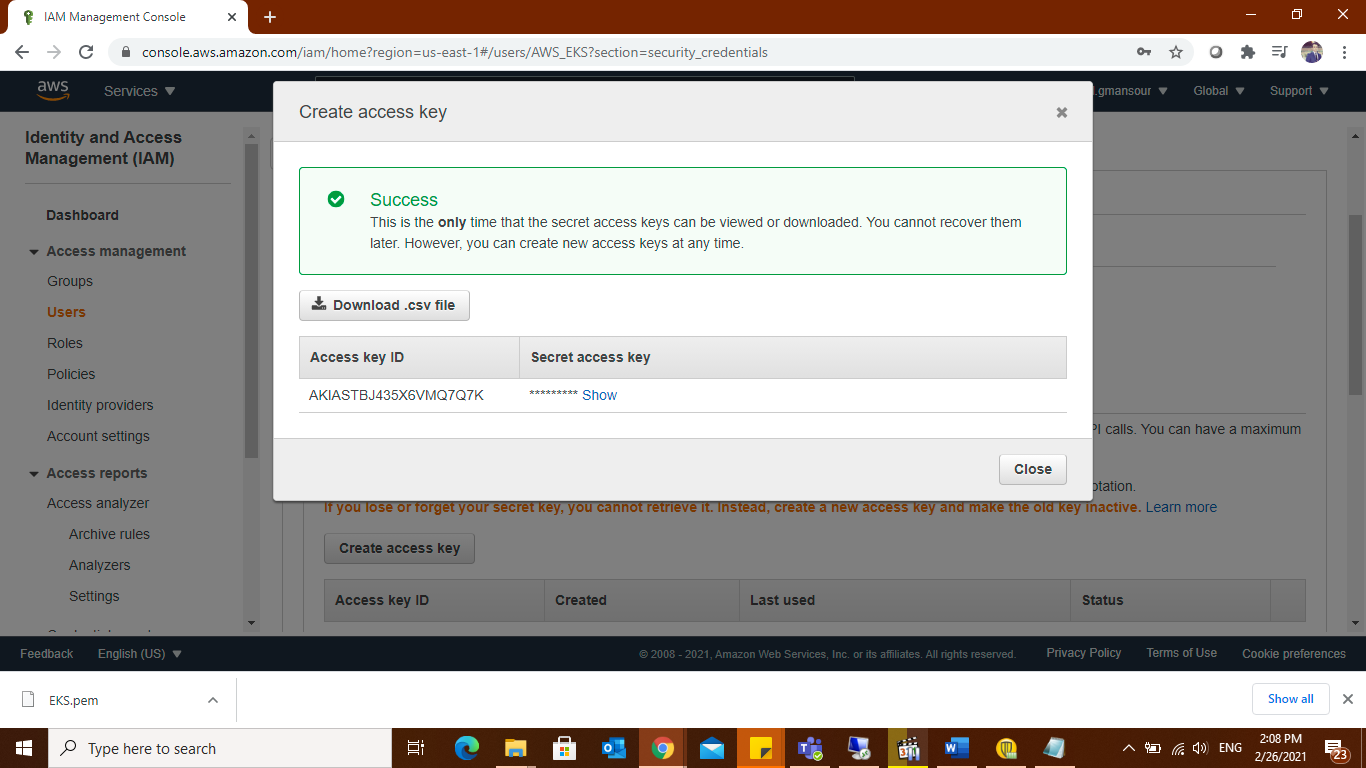






As you may have seen according to the PEM file including the key please ensure that you copy it to proper and safe folder so that no one except you has access to this folder.

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

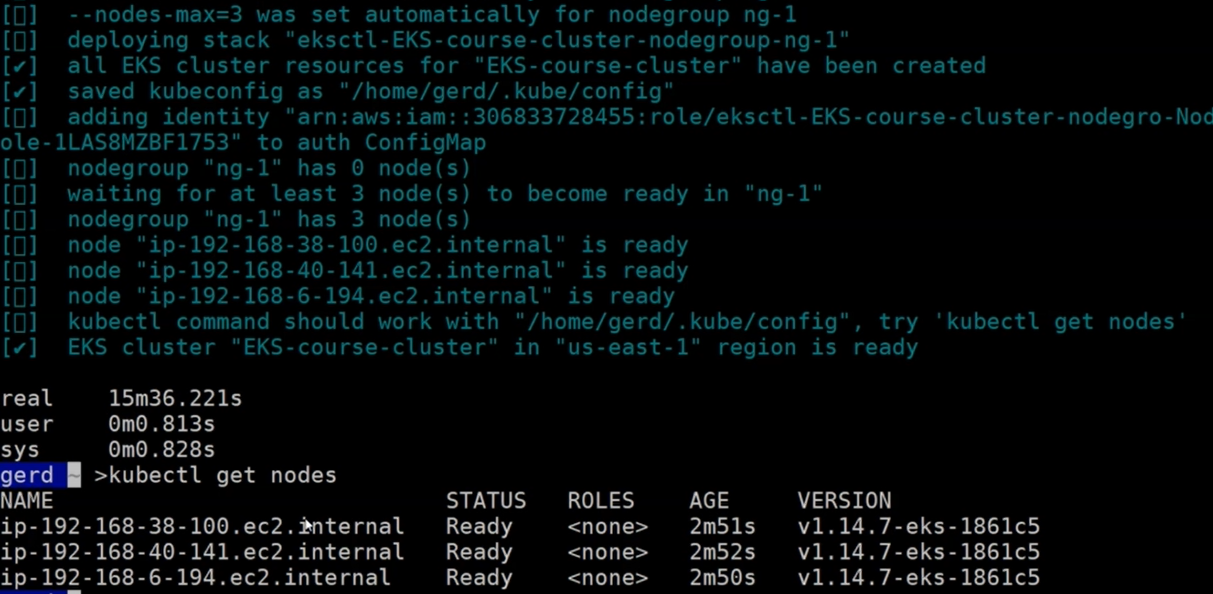
**-eksctl will automatically creating 2 subnets and two different availability zone**

then run the below command:

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

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

#kubectl get nodes



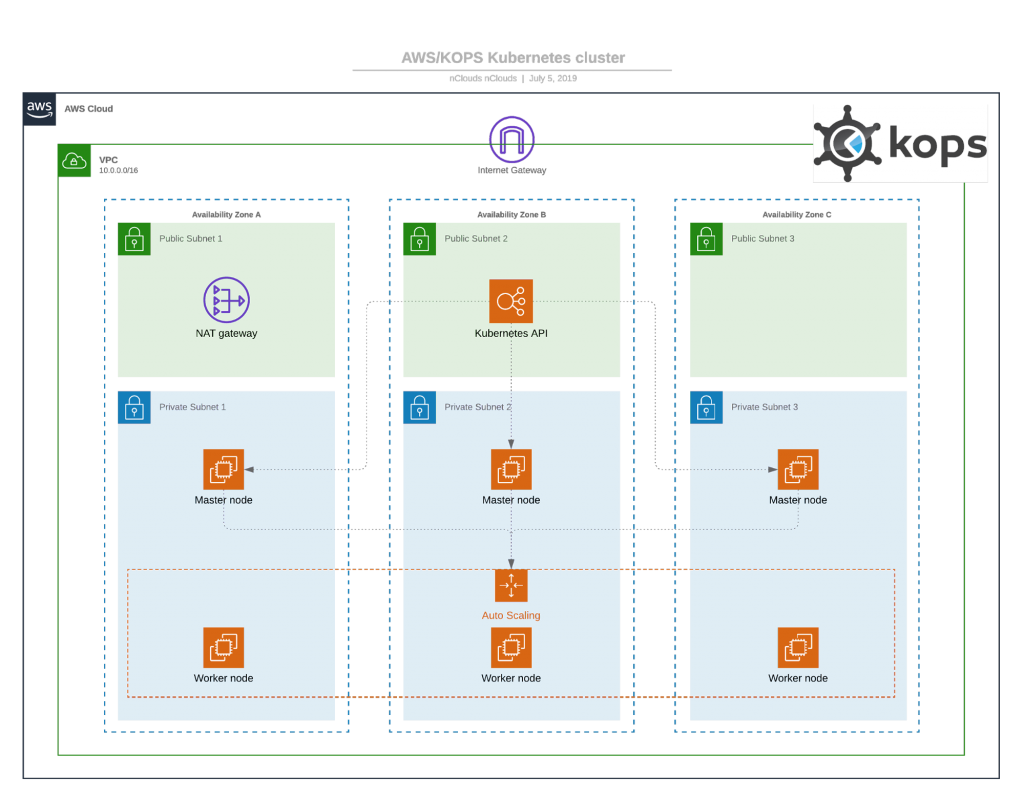
**The cluster up and running.**

OR

## Creating the EKS cluster (Hard Way)

Architecture

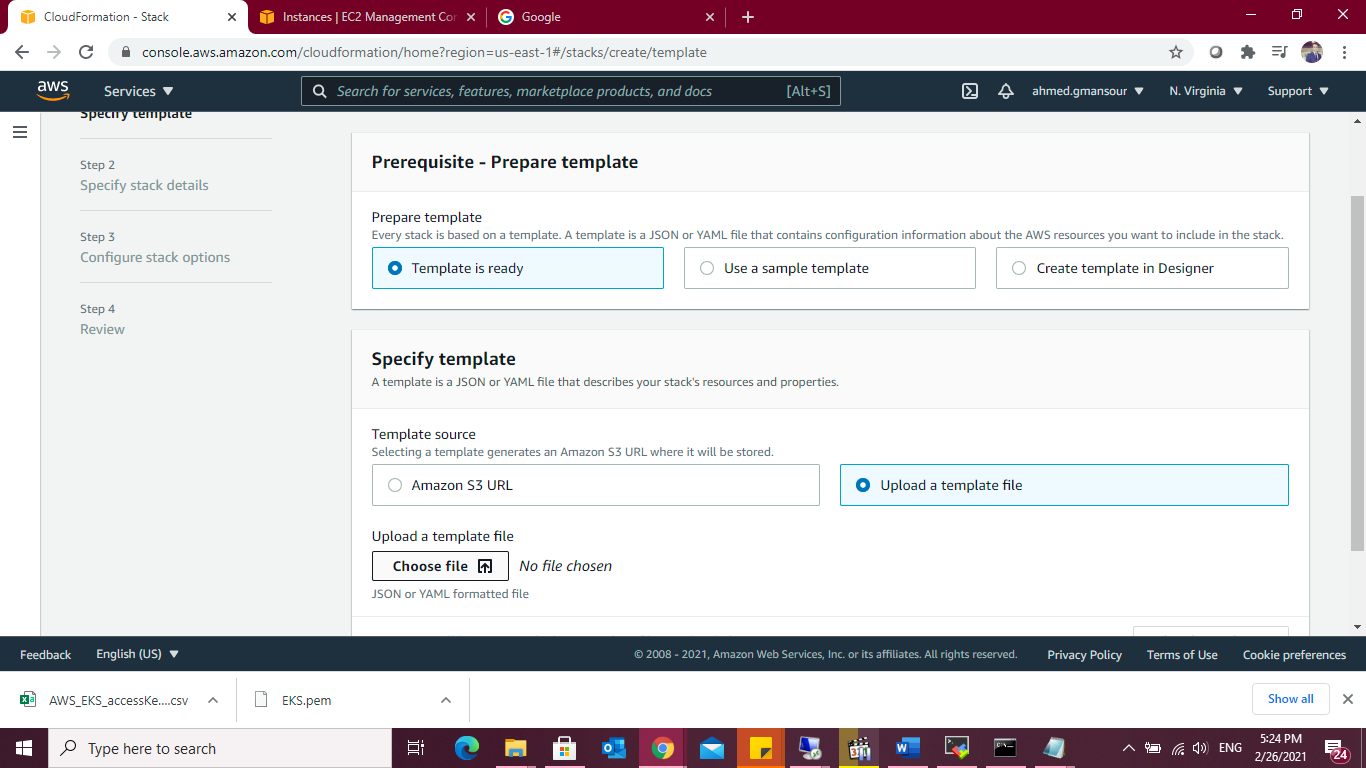
Deploying this Quick Start for a new virtual private cloud (VPC) with default parameters builds the following Amazon EKS environment in the AWS Cloud.

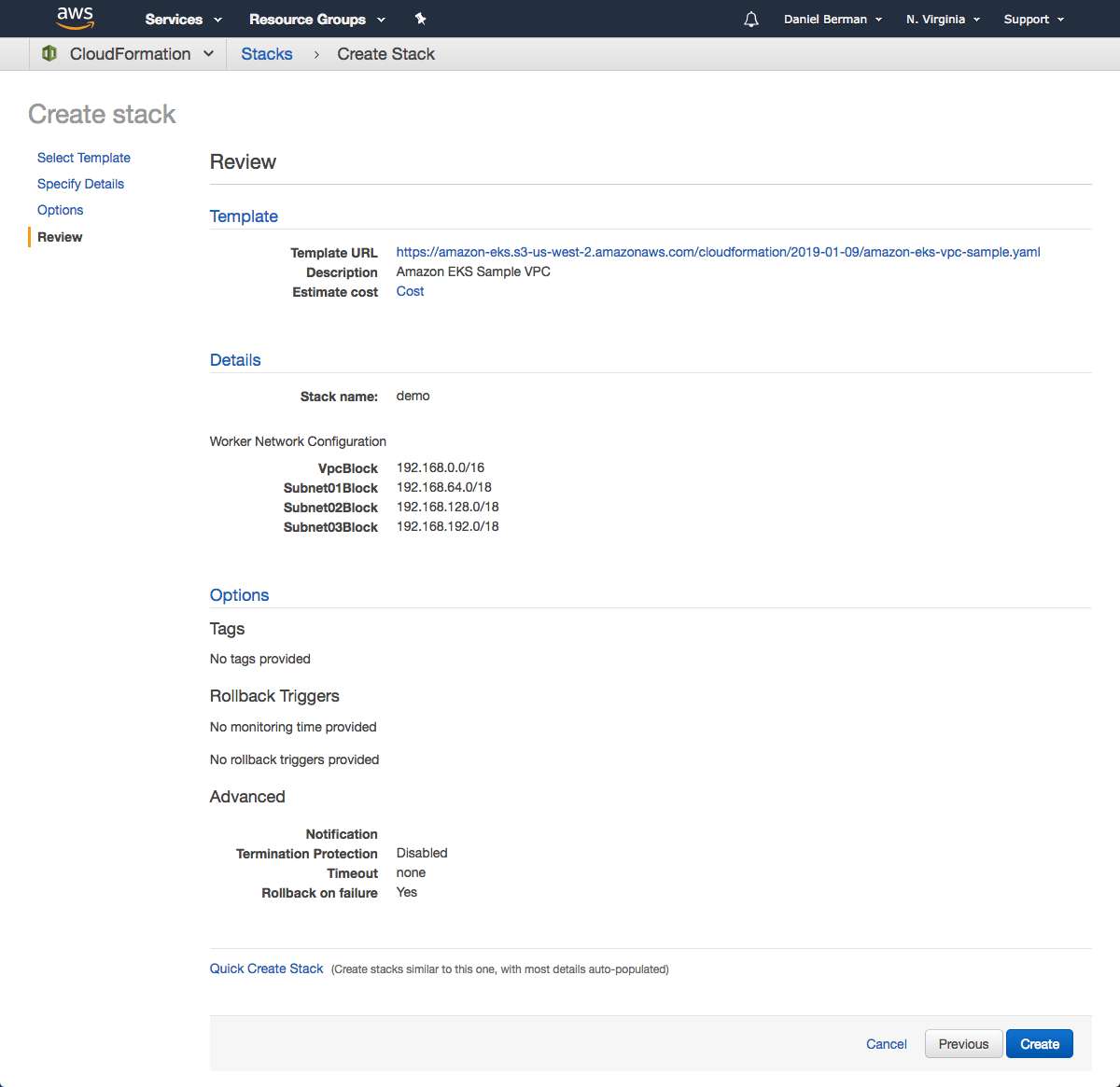


The Quick Start sets up the following:

* 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.

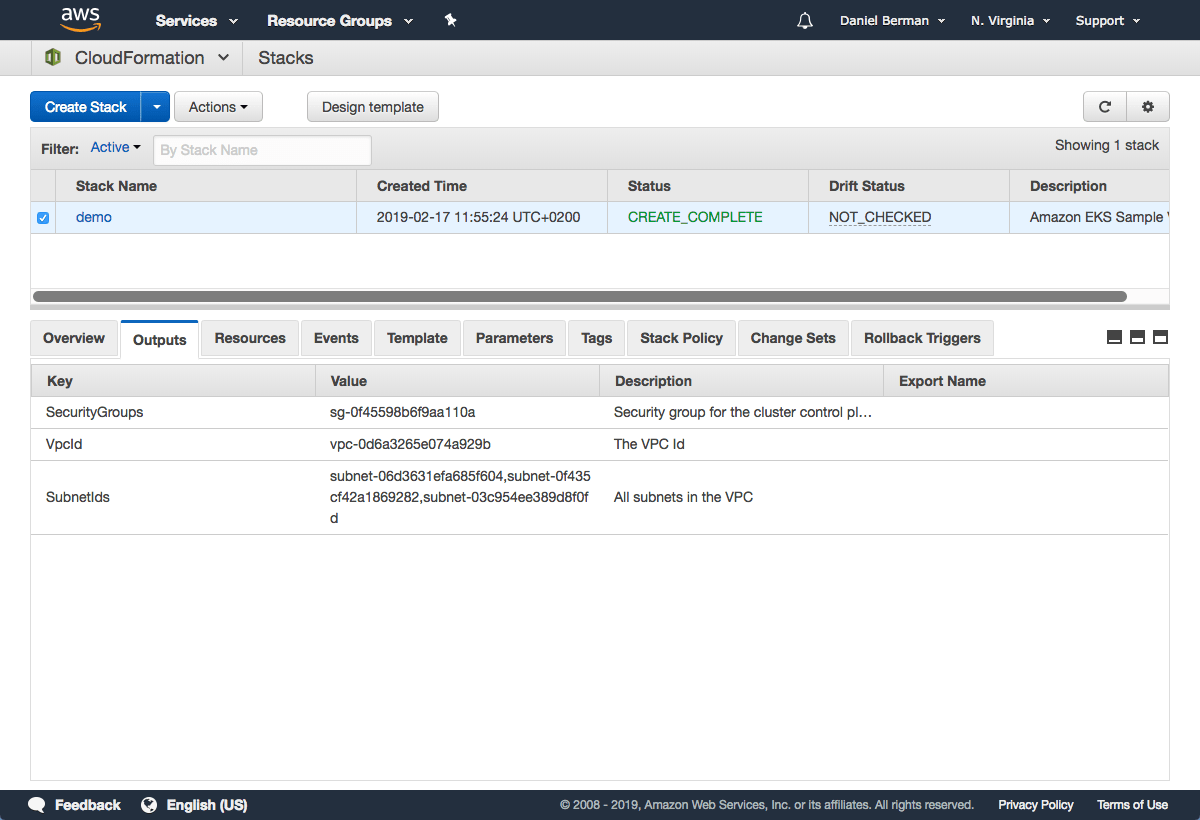
we are going to create a separate VPC—a Virtual Private Cloud that enable communication between worker nodes and the AWS Kubernetes API server— for our EKS cluster. To do this, we are going to use a CloudFormation.( amazon-eks-vpc-sample.yaml) attached.





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.

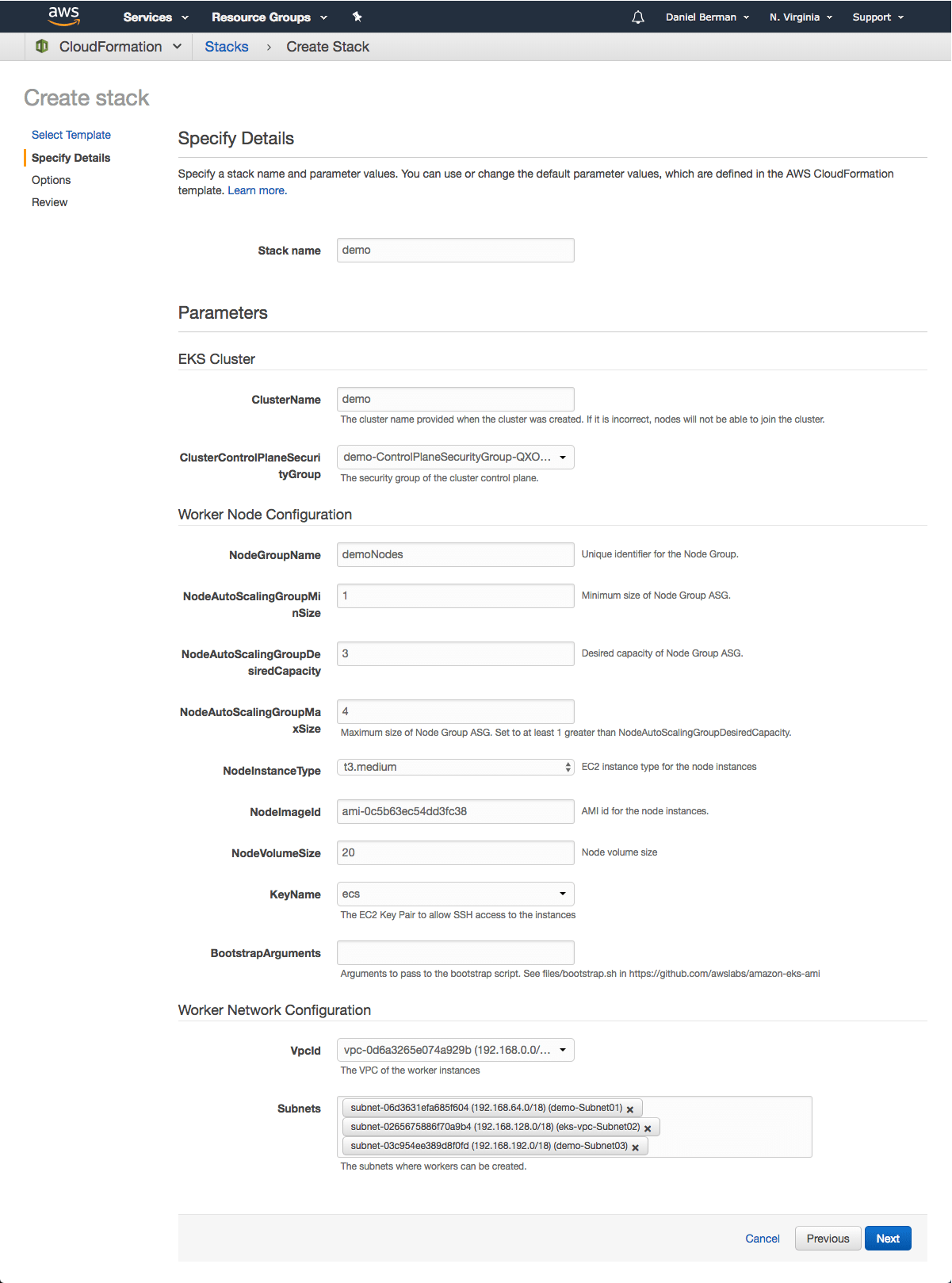
To do this, we will use the AWS CLI update-kubeconfig command:

#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.

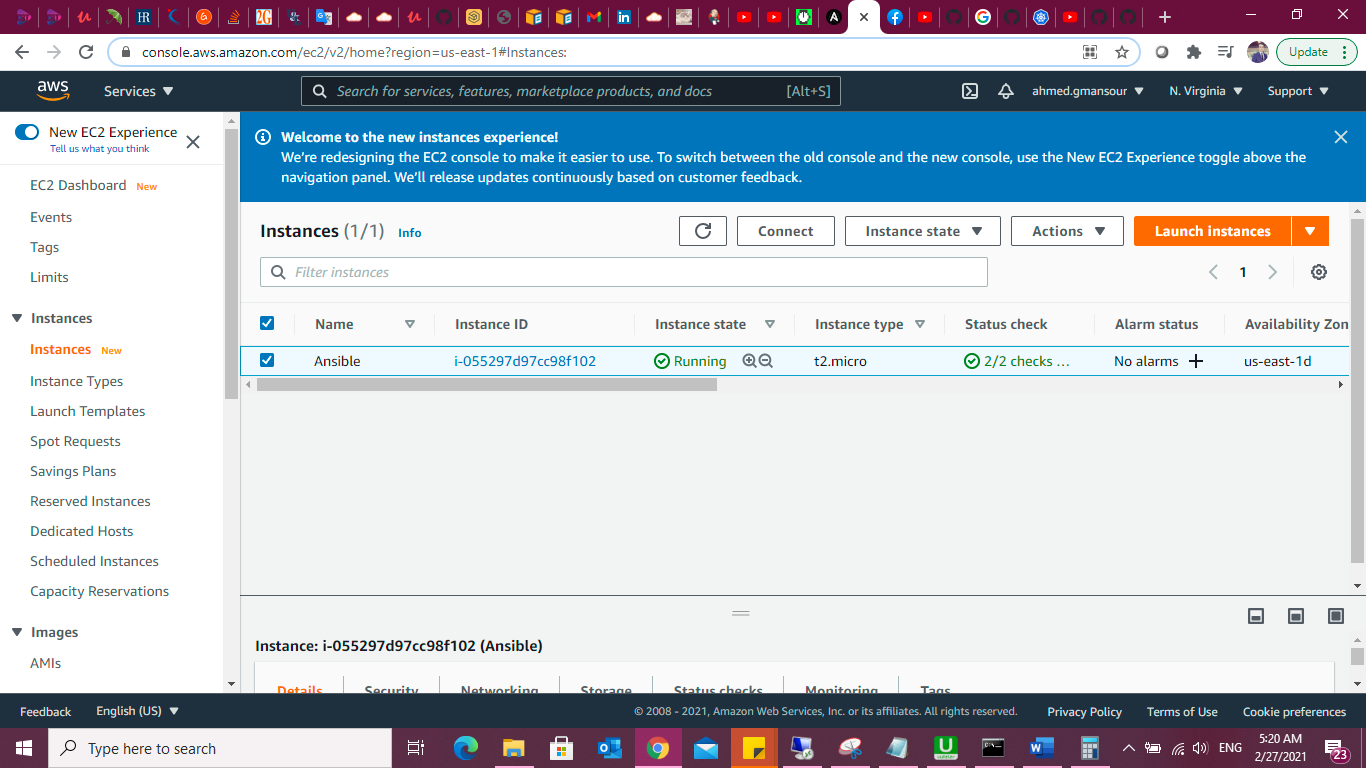
**Connect to the server (Ansible) using SSH:**

#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:**

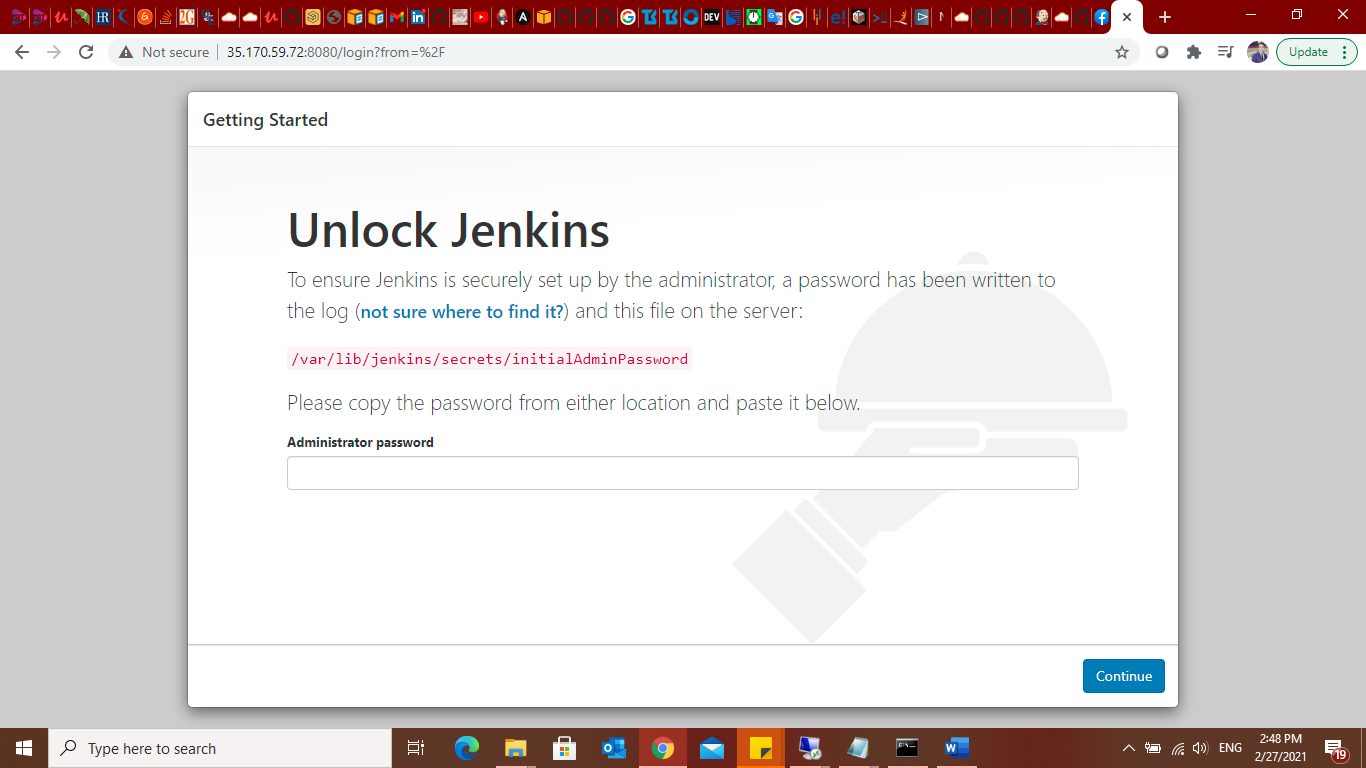
$ kubectl get pods -n Jenkins

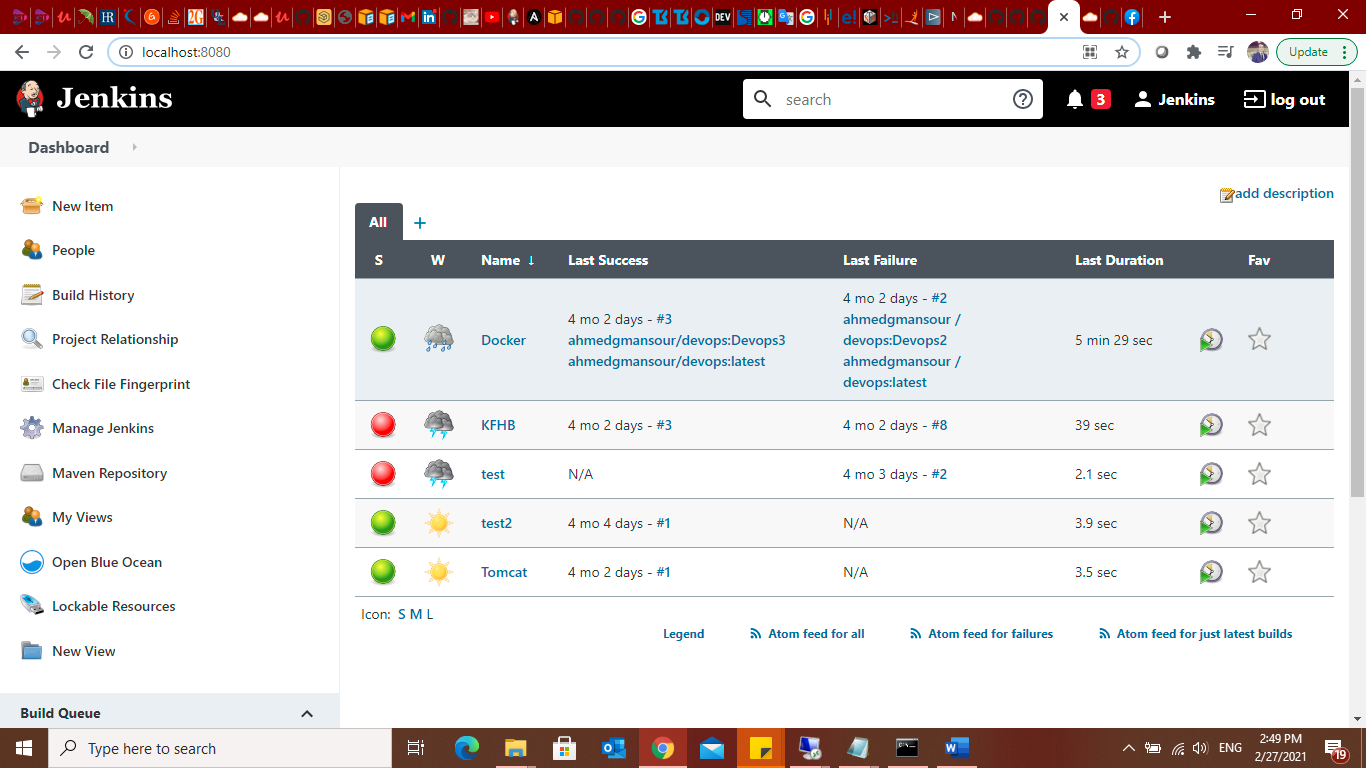
**Connect to Jenkins (actual Kubernetes cluster)**

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

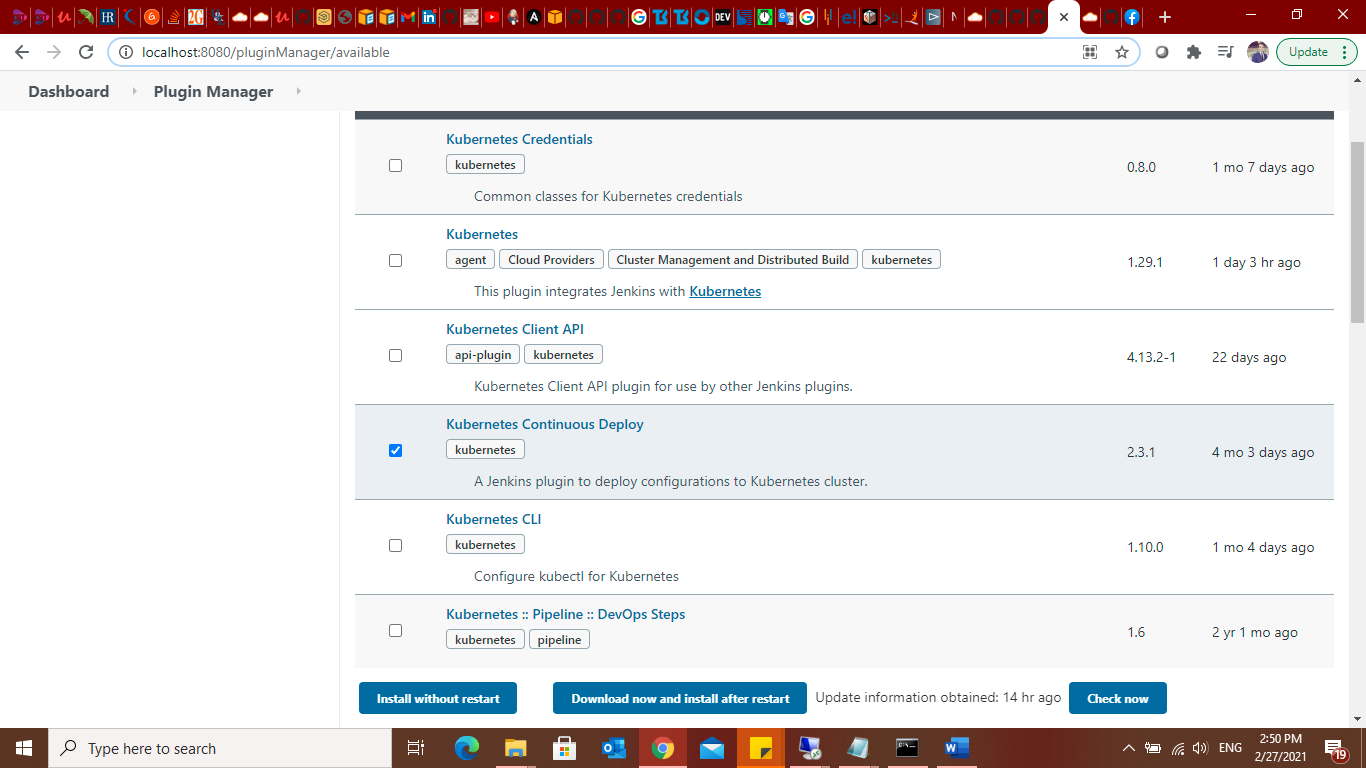
**CI/CD pipeline:**

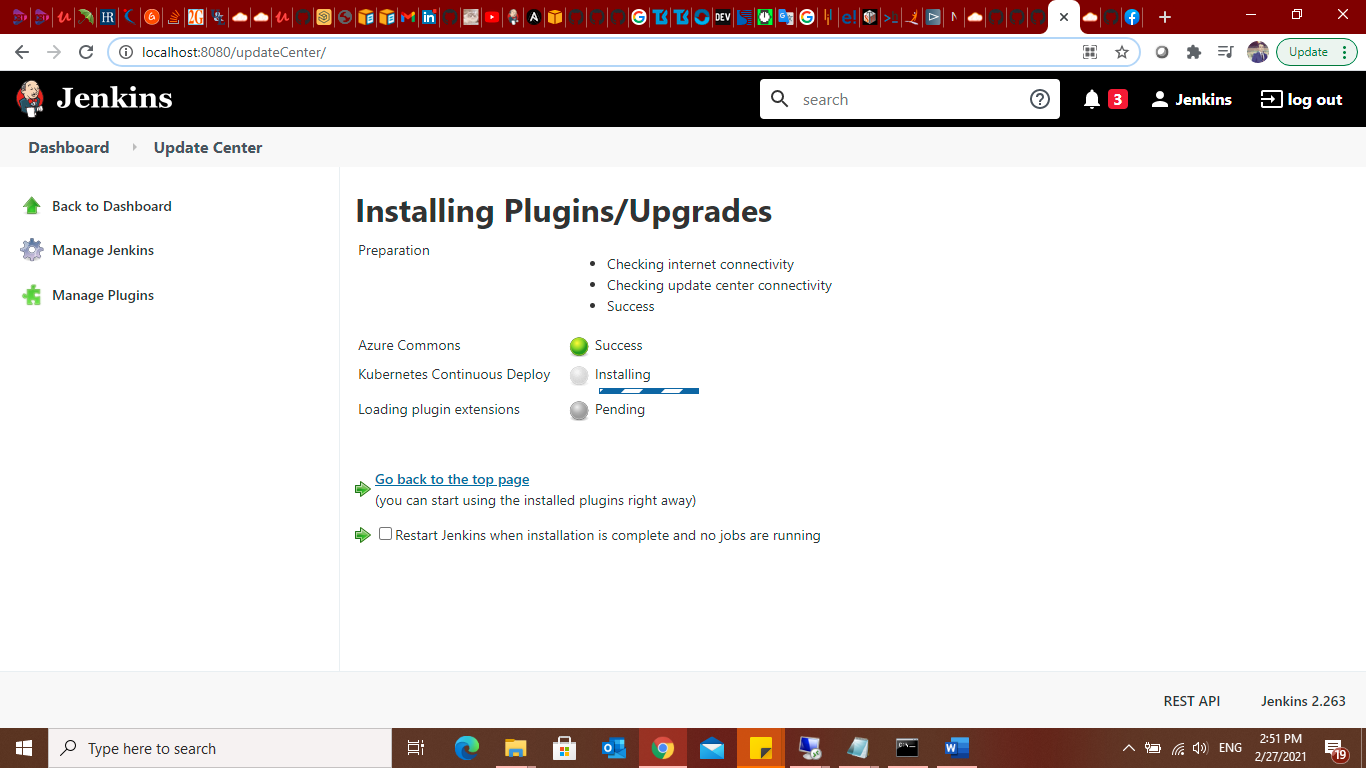
**Log in to Jenkins:**



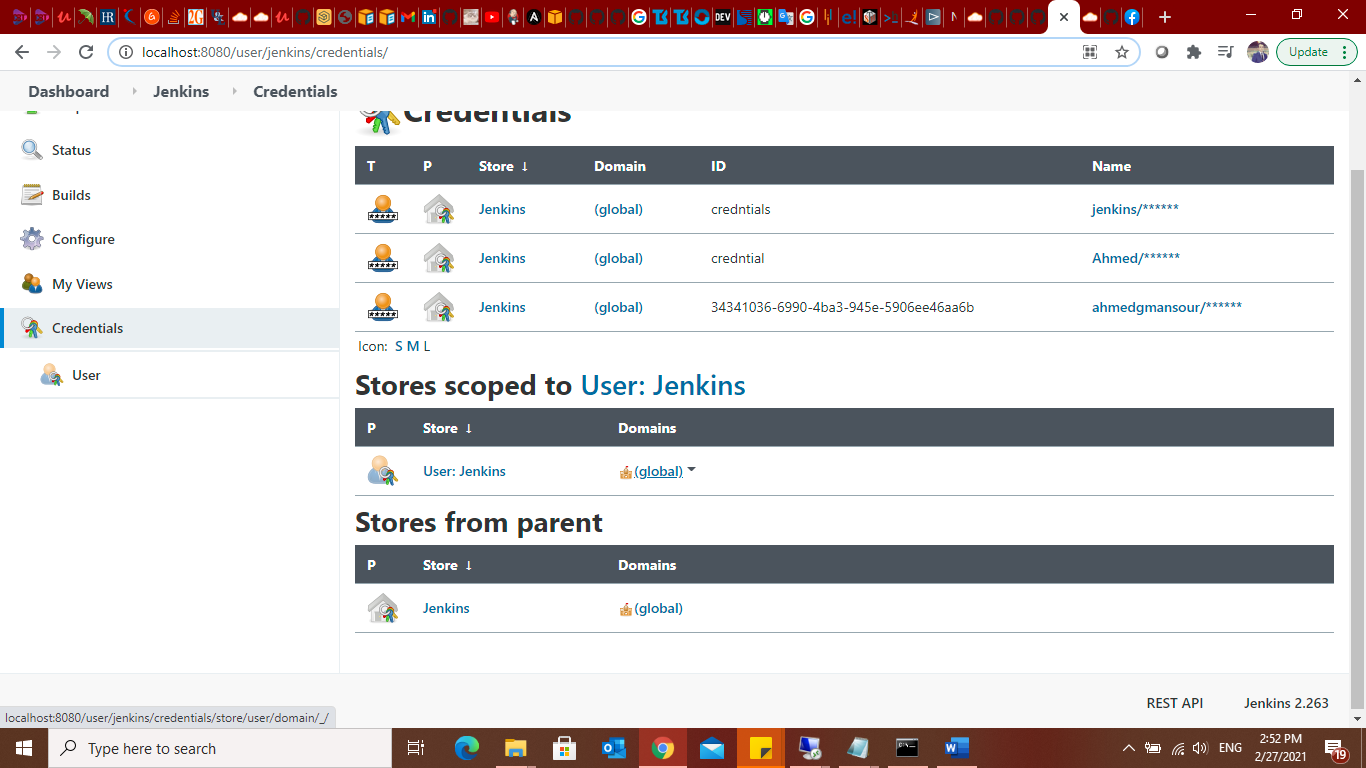


**1-Install plugins of Kubernetes.**

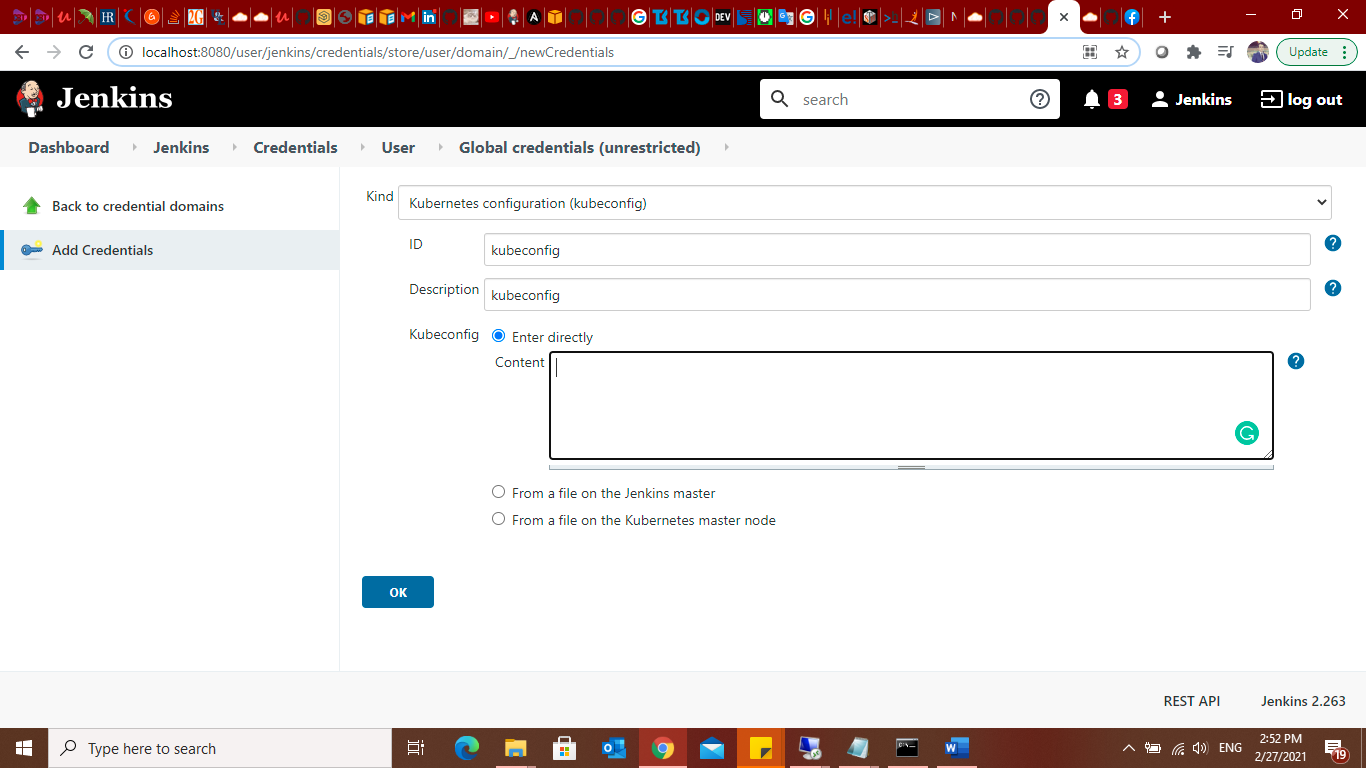




**2- add Kubernetes credentials.**



**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**

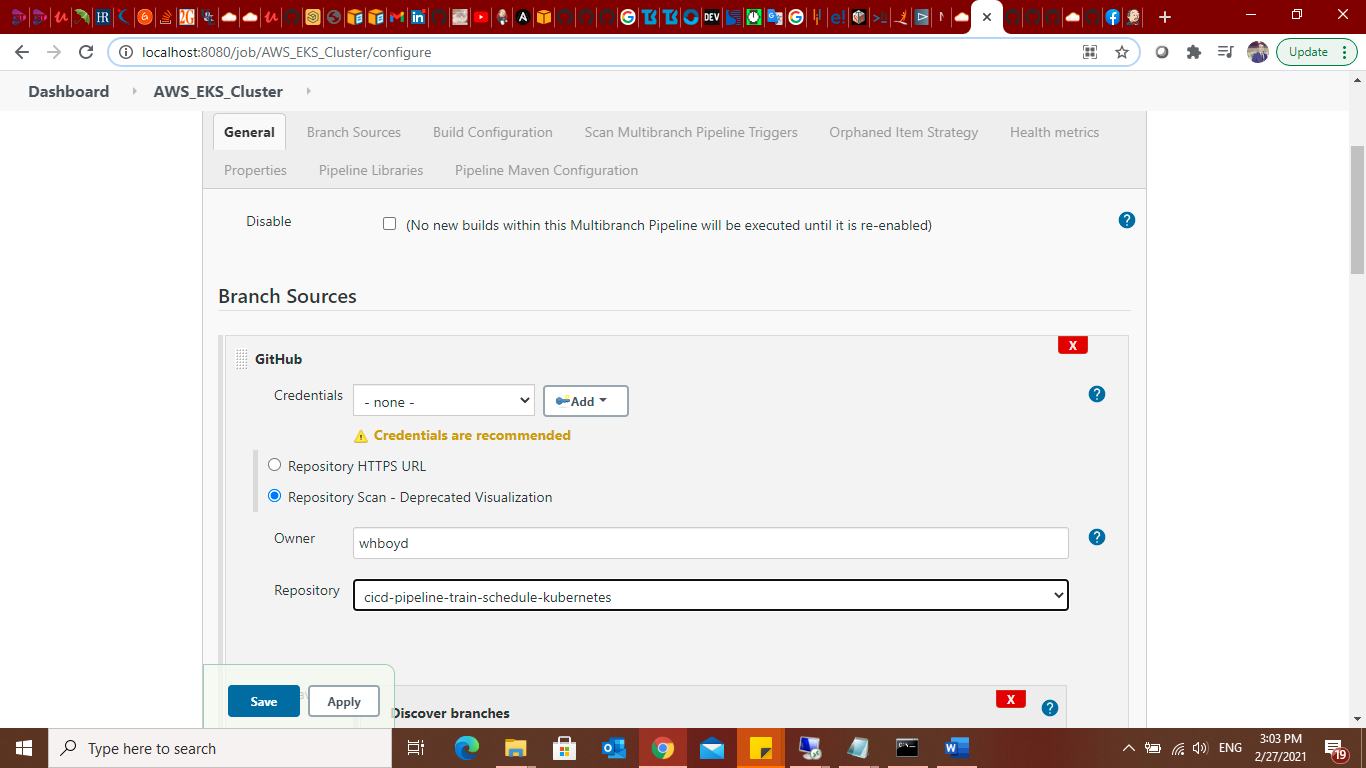
**The copy the entire context to kubconfig in jenkins configuration.**

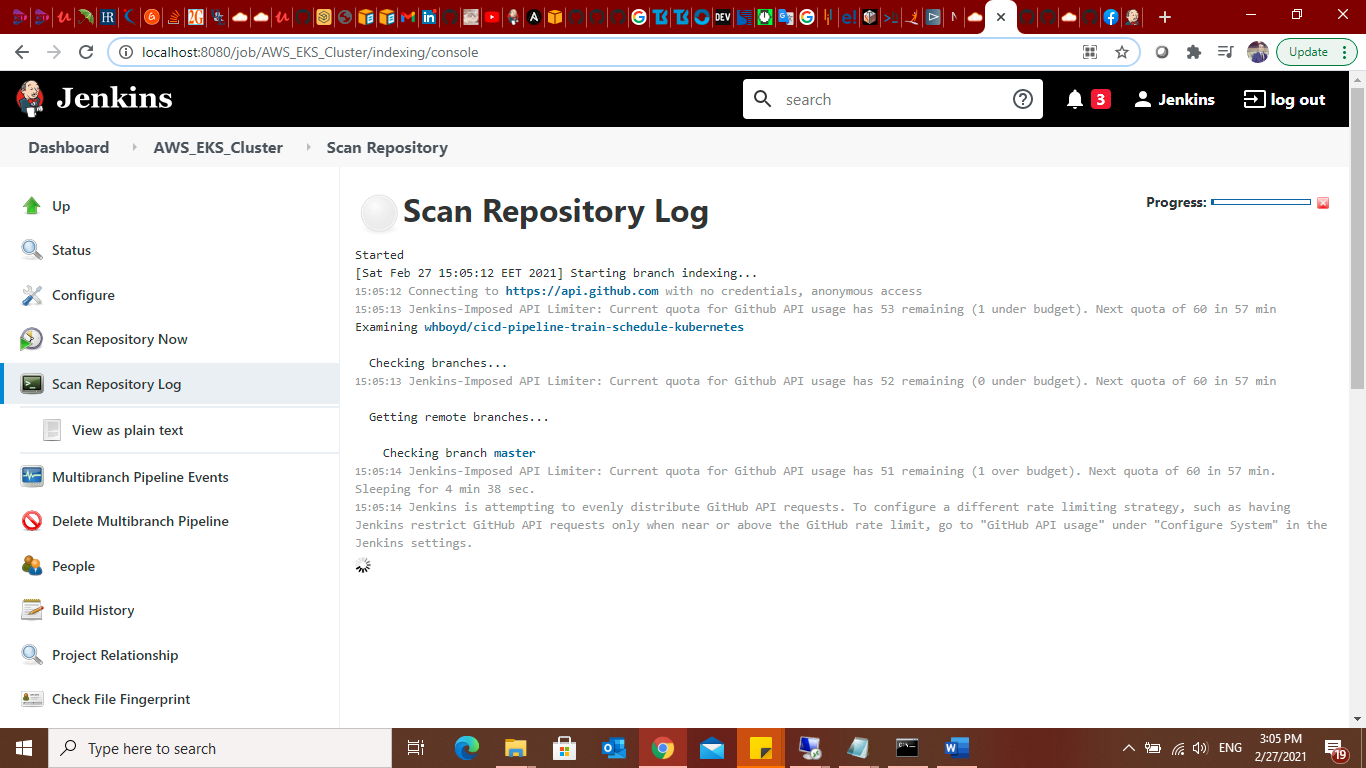
**Create Pipeline:**



**Then choose multibranch pipleline.**



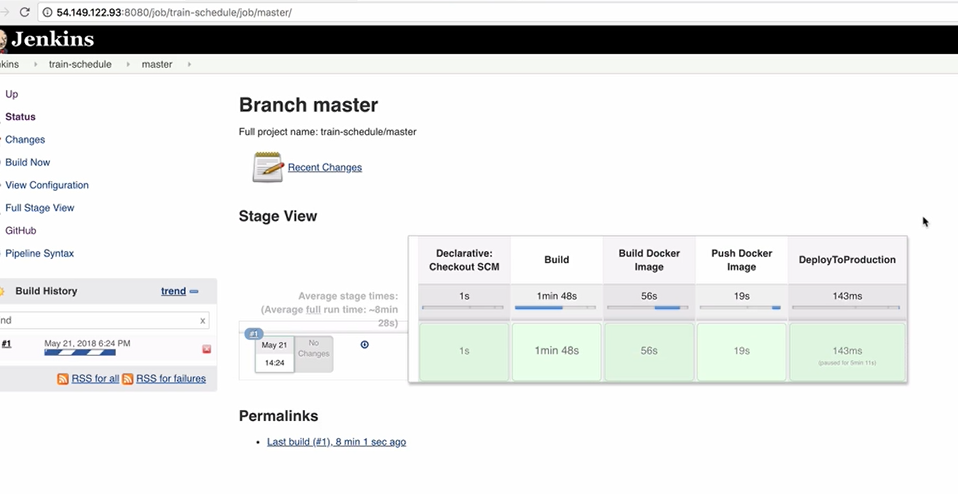




**Deployiong on kuberentes :**

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

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

****

**Kubernetes spinning up my application.**

