VASANTDADA PATIL PRATISHTHAN'S COLLEGE OF
ENGINEERING AND VISUAL ARTS
Department of Information Technology

LAB MANUAL OF

Advance DevOps Lab
Code: ITL504

Class: TE Information Technology

Semester: V (Rev-2019 'C' Scheme)

Sachin Barahate
Lab Incharge

H.O.D
(Dr. Pradip Mane)

College Vision

- To provide an environment to educate, encourage and explore students by facilitating innovative research, entrepreneurship, opportunities and employability to achieve social and professional goals.

College Mission

- To foster entrepreneurship & strengthen industry institute interaction to enhance career opportunities for the employability of students.
- To encourage collaborations with industries and academic institutes in terms of projects & internships by creating area for Research and Development.
- To build up appropriate moral and ethical skills and to promote holistic development of students through various academic, social and cultural activities

Department Vision

- To impart quality education in the field of Information Technology to meet the challenging needs of the society and industry.

Department Mission

- To provide quality education to students by including Problem Solving, Teamwork and Leadership Skills to achieve their goals in the field of Information Technology.
- To develop skilled IT professionals with moral principles and empower them in lifelong learning.
- To educate students for global development including entrepreneurship, employability and the ability to apply technology to real life problems.

Program Educational Objectives (PEO)

- Graduates will be successful with sound foundation in engineering fundamentals, trending technologies and entrepreneurship.
- Graduates will be able to identify and solve real world problems.
- Graduates will become ingenious and responsible citizens by demonstrating ethics with nurtured professional attitude

Program Specific Outcomes (PSO)

- Develop efficient IT based solutions by applying and integrating various domains like Artificial Intelligence, IoT, Computer Networks and Security to solve real time problems.
- Apply technical knowledge in the field of Information Technology to achieve successful career and to pursue higher studies for future endeavors.

Program Outcomes (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identity, formulate complex engineering problems reaching substantiated conclusions using principles of Computer Engineering.
3. **Design / development of solutions:** Design / develop solutions for complex engineering problems and design system components or processes that meet the specified needs with

appropriate consideration for the society.

4. **Conduct investigations of complex problems:** Use knowledge for the design of experiments, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select and apply appropriate techniques and modern engineering tools, including predictions and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply the knowledge to assess social issues and the responsibilities relevant to engineering practices.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively such as being able to comprehend and write effective reports and design documentation, make effective presentations.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management skills and apply the skills to manage projects effectively.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Lab Objectives

The Lab experiments aims:

1. To understand DevOps practices and cloud native environments to achieve continuous software delivery pipelines and automated operations that address the gap between IT resources and growing cloud complexity.
2. To Use Kubernetes services to structure N-tier applications.
3. To be familiarized with Infrastructure as code for provisioning, compliance, and management of any cloud infrastructure, and service.
4. To understand that security and speed in software development are not inversely-related objectives Internalizing the contribution of tools and automation in DevSecOps
5. To understand various troubleshooting techniques by monitoring your entire infrastructure and business processes
6. To understand how software and software-defined hardware are provisioned dynamically.

Lab Outcomes

On successful completion, of course, learner/student will be able to:

1. To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements
2. To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
3. To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure.
4. To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Techniques.

5. To use Continuous Monitoring Tools to resolve any system errors (low memory, unreachable server etc.) before they have any negative impact on the business productivity
6. To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework

Prerequisite: Operating System, Linux Administration, Java /Web Application Programming, Software Engineering, Cloud Computing and DevOps Ecosystem.

Hardware & Software Requirements:

Hardware Requirements	Software Requirements	Other Requirements
PC With following Configuration 1. Intel i3 core or above 2. 4 GB RAM or above 3. 500 GB HDD 4. Network interface card	1. Linux / Windows Operating system 2. VIRTUAL BOX/ VMWARE	1. Internet Connection for installing additional packages 2. GitHub account 3. AWS free tier account

Lab /syllabus:

Sr. No.	Module	Detailed Content	Hours	LO Mapping
0	Prerequisite	Knowledge of Linux Operating system, installation and configuration of services and command line basics, Basics of Computer Networks, Software Development Life cycle, Cloud Computing and DevOps Ecosystem.	02	--
I	Introduction to Devops on Cloud	Learn about various cloud services and service providers, also get the brief idea of how to implement DevOps over Cloud Platforms. • Introduction to high availability architecture and auto-scaling • Set up the DevOps infrastructure on the cloud • Work and set up IDE on Cloud9 • Deploy projects on AWS using Code Build, CodeDeploy, and CodePipeline **Self-Learning Topics:** AWS Codestar	04	LO1
II	Container Orchestration using Kubernetes	In this module, you will learn how Kubernetes automates many of the manual processes involved in deploying, managing, and scaling containerized applications. • Install and configure Kubernetes • Spin Up a Kubernetes Cluster • Check the Nodes of Your Kubernetes Cluster Installing kubectl to manage cluster	04	LO1, LO2

		and deploy Your First Kubernetes Application **Self-Learning Topics:** • Using Services and Ingresses to Expose Deployments • Perform logging, monitoring, services, and volumes in Kubernetes.		
III	Infrastructure Automation with Terraform	In this module you will learn, Infrastructure as code for provisioning, compliance, and management of any cloud infrastructure, and service. • Introduction to Infrastructure as Code with Terraform • Install, Build, change and Destroy Infrastructure using Terraform. **Self-Learning Topics**: Terraform • Create Resource Dependencies • Provision Infrastructure • Define Input Variables, Query Data with output and store remote state	04	LO1, LO3
IV	DevSecOps: Static Application Security Testing (SAST)	In this module, you will learn to identify and remediate application vulnerabilities earlier and help integrate security in the development process using tools like SonarQube / Gitlab / • Perform static analysis on application source code and binaries. • Spot potential vulnerabilities before deployment • Analysis of java / web-based project • Jenkins SonarQube / Gitlab Integration **Self-Learning Topics**: Snyk, OWASP ZAP, Analysis Core Plugin	04	LO1, LO4
V	DevSecOps: Continuous Monitoring	In this module, you will learn to detect, report, respond to the attacks and issues which occur within the infrastructure. • Introduction to Continuous Monitoring • Introduction to Nagios • Installing Nagios • Nagios Plugins (NRPE) and Objects Nagios Commands and Notification • Monitoring of different servers using Nagios **Self-Learning Topics:** Splunk, Snort, Tenable	04	LO1, LO5
VI	NoOps: Serverless	Computing In this module, you will learn serverless computing platform like AWS Lambda, which allows you to build your	04	LO1, LO6

		code and deploy it without ever needing to configure or manage underlying servers. • AWS Lambda - Overview and Environment Setup . • Building and Configuring the Lambda function (NODEJS/PYTHON/JAVA) • Creating & Deploying using AWS Console/CLI • Creating & Deploying using Serverless Framework **Self-Learning Topics:** AWS Lambda • Create a REST API with the Serverless Framework		

Textbooks:

1. AWS Certified SysOps Administrator Official Study Guide: Associate Exam by Stephen Cole (Author), Gareth Digby (Author), Chris Fitch (Author), Steve Friedberg (Author), Shaun Qual
2. AWS Certified Solutions Architect Official Study Guide: Associate Exam by Joe Baron
3. Terraform: Up & Running - Writing Infrastructure as Code, Second Edition by Yevgeniy Brikman , O'Reilly
4. Kubernetes: Up and Running - Dive into the Future of Infrastructure, Second Editionby Brendan Burns,O'Reilly
5. Going Serverless with AWS Lambda: Leveraging the latest services from the AWS cloud by Ajay Pherwani , Shroff/X-Team;
6. Learning Nagios, Packt Publishing.

References:

1. Learning Aws - Second Edition: Design, build, and deploy responsive applications using AWS by Amit Shah Aurobindo Sarkar
2. Mastering Aws Lambda by Yohan Wadia Udita Gupta

List of Experiments

Sr. No	Experiment Title	Lab Outcome
1	To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.	LO1
2	To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS Code Pipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.	LO1
3	To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.	LO1, LO2
4	To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.	LO1, LO2
5	To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.	LO1, LO3
6	To Build, change, and destroy AWS / GCP /Microsoft Azure/ Digital Ocean infrastructure Using Terraform.	LO1, LO3
7	To understand Static Analysis SAST process and learn to integrate Jenkins SAST to Sonar Qube/Git Lab.	LO1, LO4
8	Create a Jenkins CICD Pipeline with Sonar Qube / Git Lab Integration to perform a static analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.	LO1, LO4
9	To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.	LO1, LO5
10	To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.	LO1, LO5
11	To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.	LO1, LO6
12	To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.	LO1, LO6

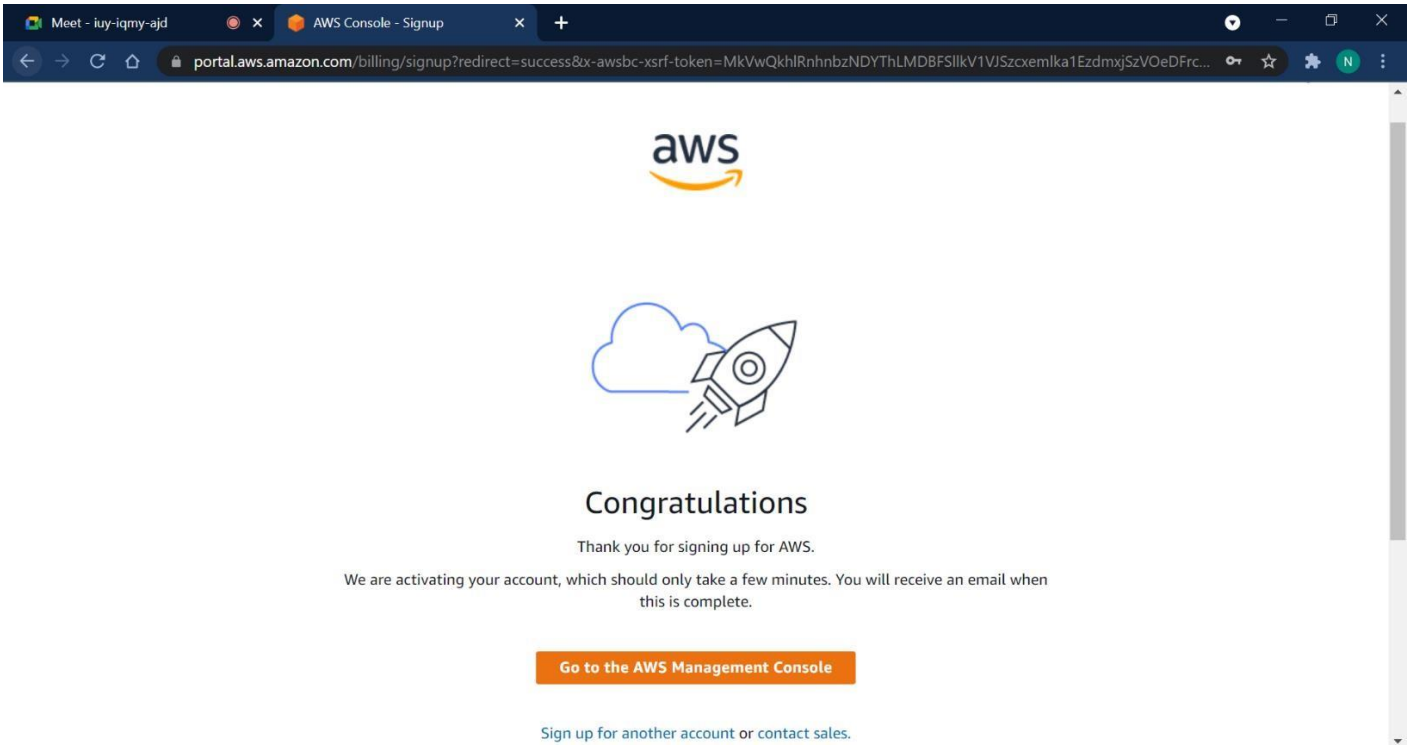
Experiment no. 1

Aim: To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.

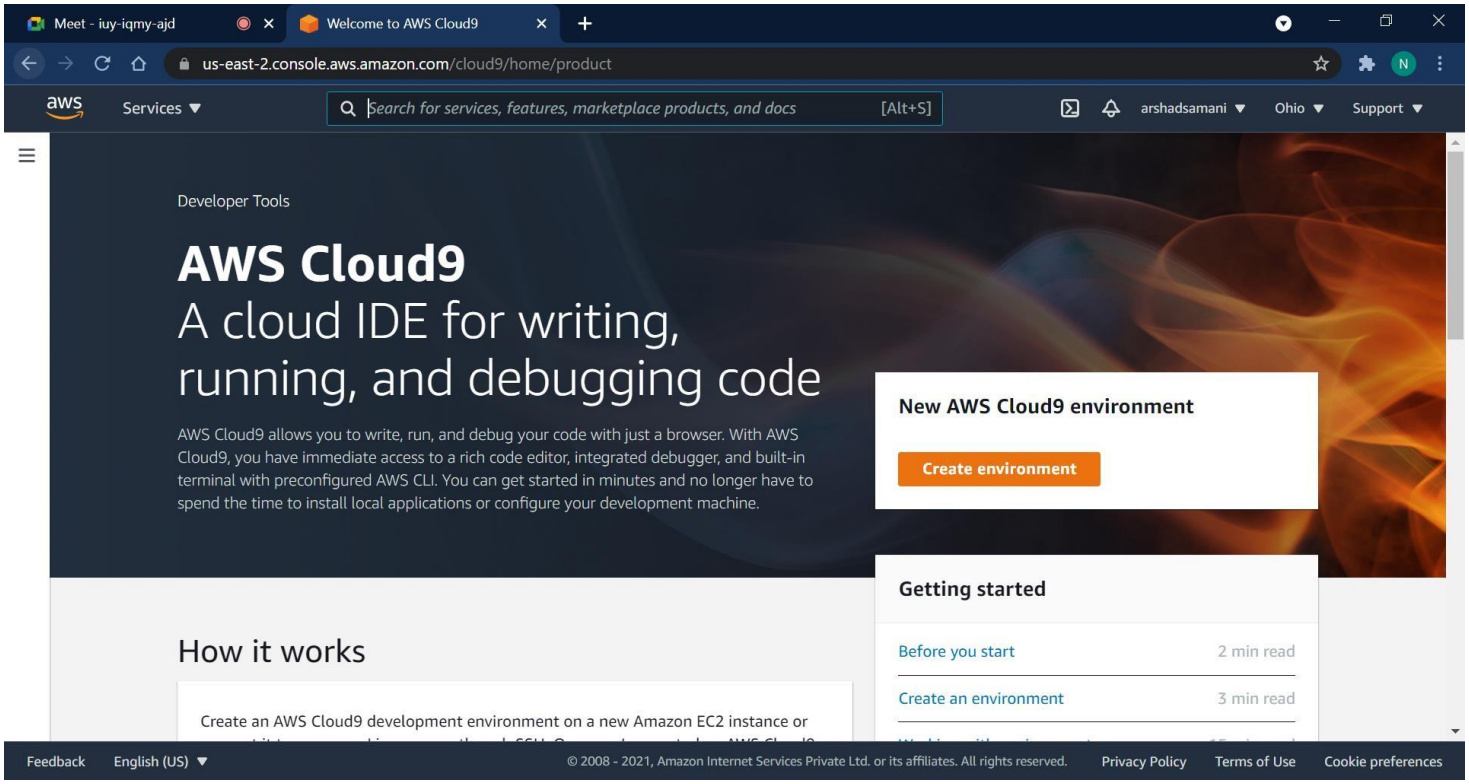
Theory:

Cloud infrastructure is a term used to describe the components needed for cloud computing, which includes hardware, abstracted resources, storage, and network resources. Think of cloud infrastructure as the tools needed to build a cloud. In order to host services and applications in the cloud, you need cloud infrastructure. Cloud infrastructure is made up of several components, each integrated with one another into a single architecture supporting business operations. A typical solution may be composed of hardware, virtualization, storage, and networking components. As a term, cloud infrastructure can be used to describe a complete cloud computing system - once all the pieces are put together - as well as the individual technologies themselves.

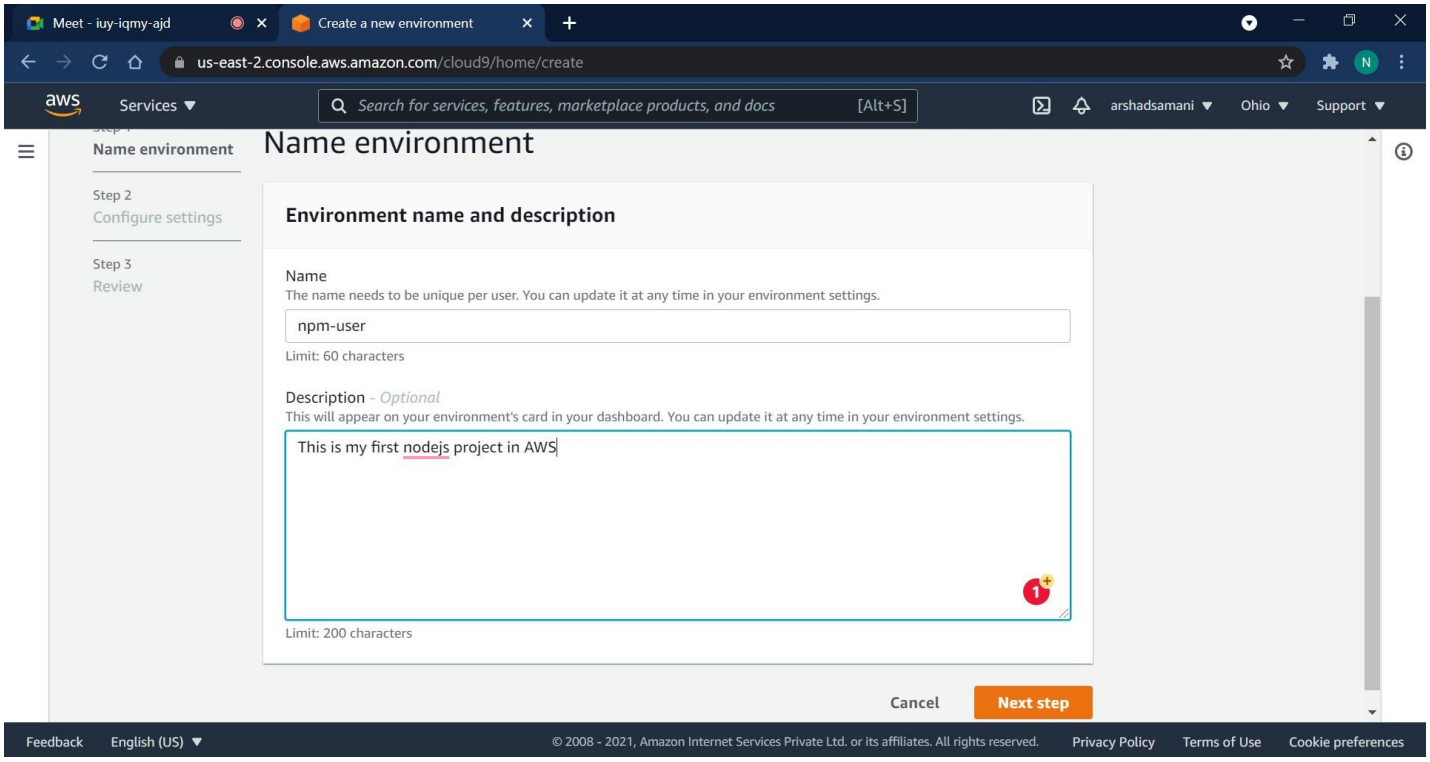
Results: Account created

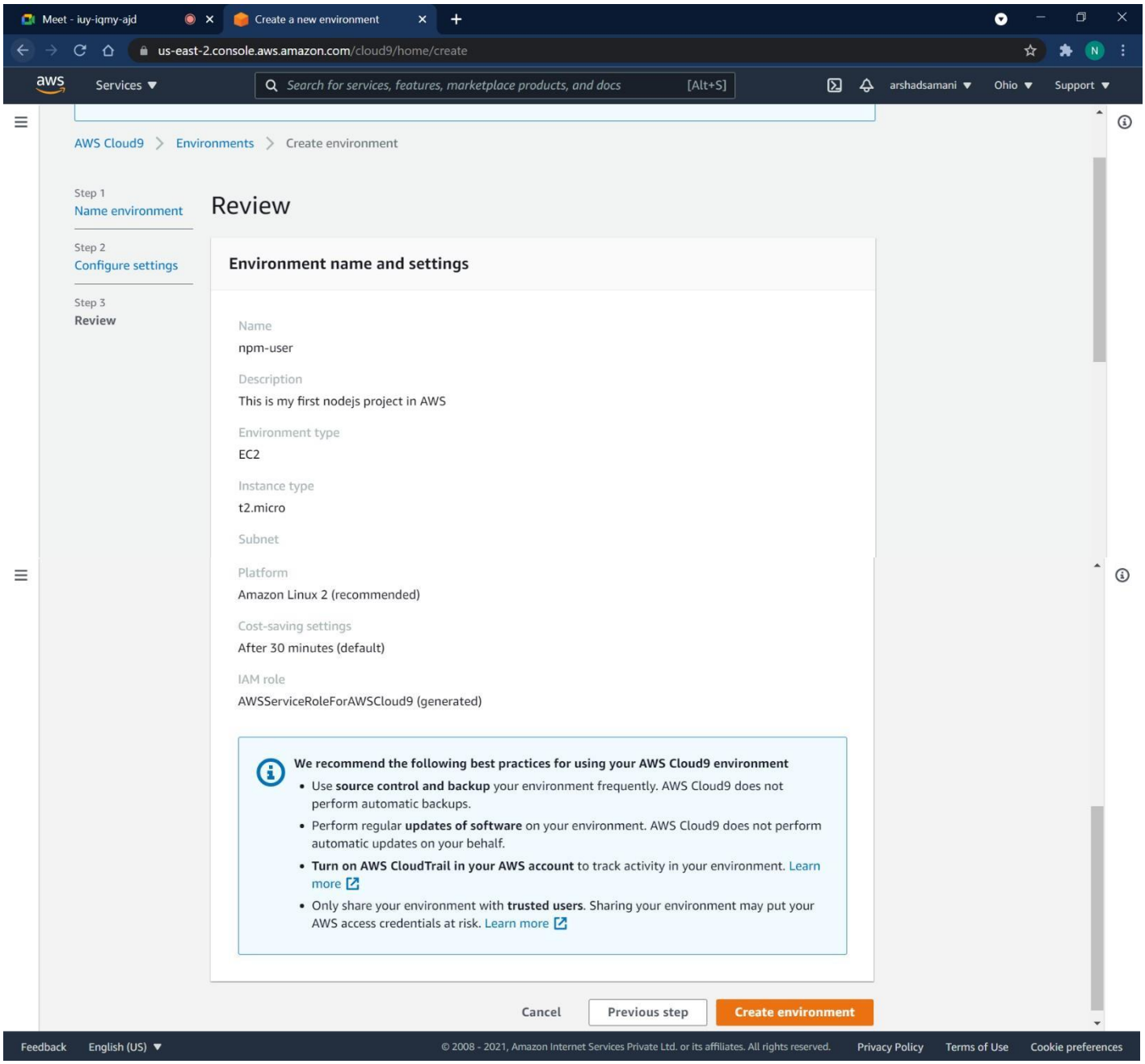


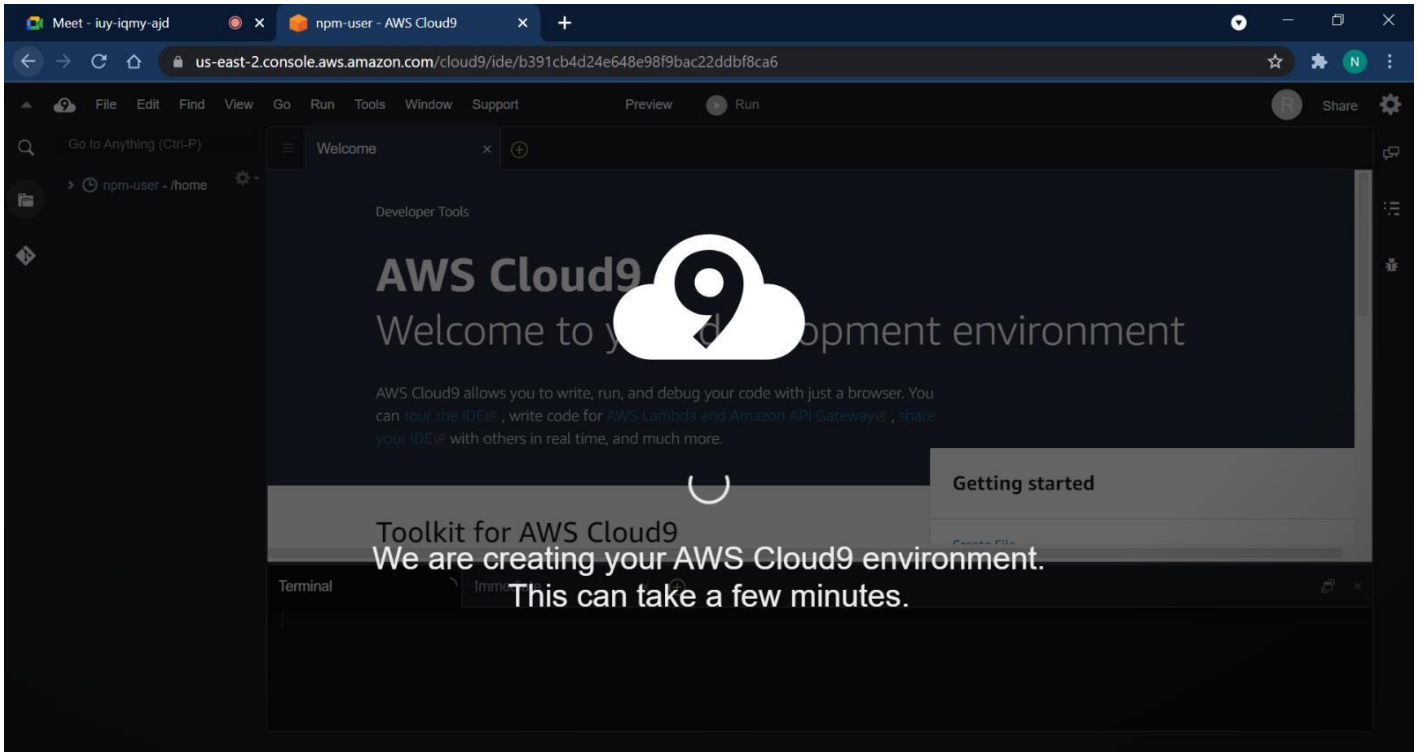
2. AWS Cloud9

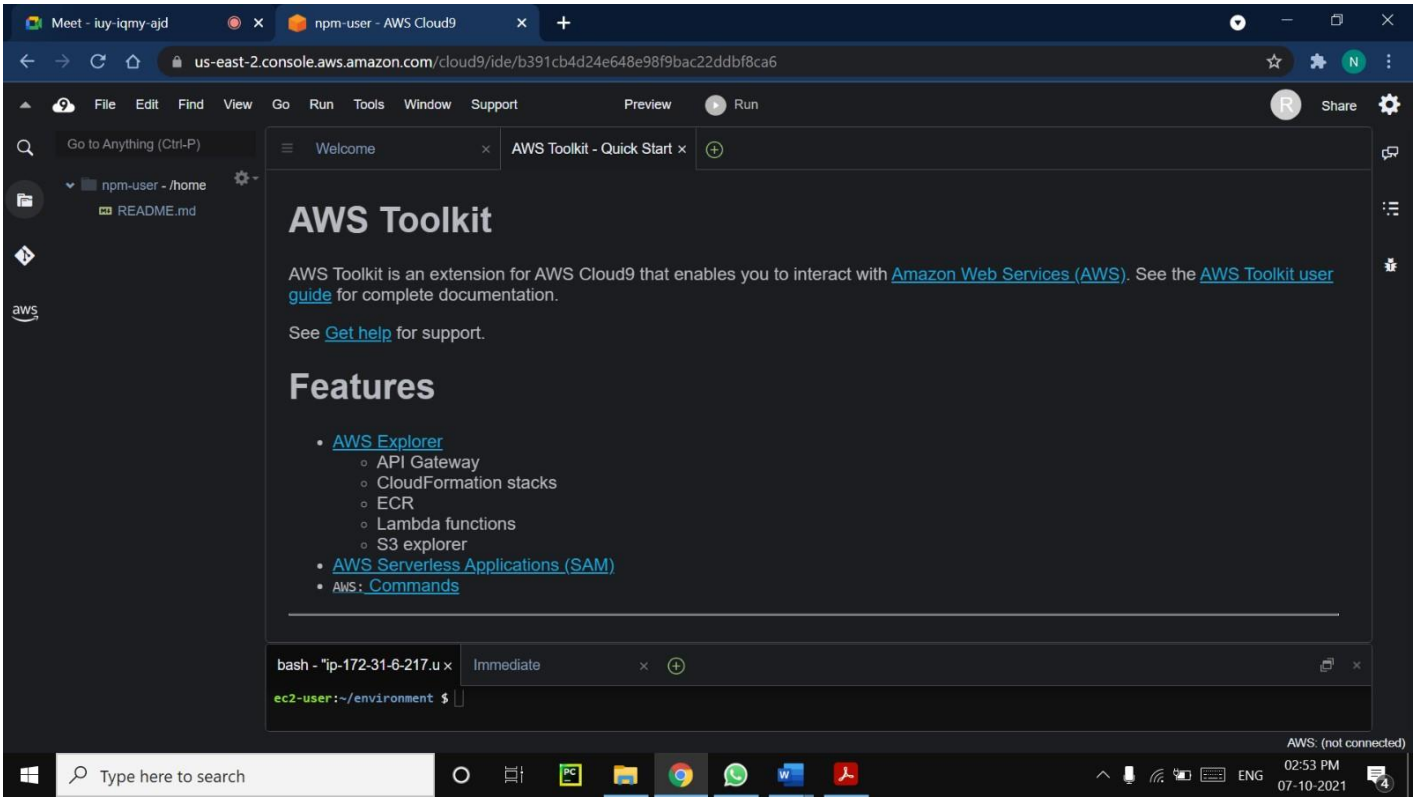


3. Creating Environment




Meet - iuy-iqmy-ajd
Create a new environment
us-east-2.console.aws.amazon.com/cloud9/home/create
aws
Services
Search for services, features, marketplace products, and docs [Alt+S]
arshadsamani
Ohio
Support
AWS Cloud9 > Environments > Create environment
Step 1
Name environment
Step 2
Configure settings
Step 3
Review
Review
Environment name and settings
Name
npm-user
Description
This is my first nodejs project in AWS
Environment type
EC2
Instance type
t2.micro
Subnet
Platform
Amazon Linux 2 (recommended)
Cost-saving settings
After 30 minutes (default)
IAM role
AWSServiceRoleForAWSCloud9 (generated)
We recommend the following best practices for using your AWS Cloud9 environment
Use source control and backup your environment frequently. AWS Cloud9 does not perform automatic backups.
Perform regular updates of software on your environment. AWS Cloud9 does not perform automatic updates on your behalf.
Turn on AWS CloudTrail in your AWS account to track activity in your environment. Learn more
Only share your environment with trusted users. Sharing your environment may put your AWS access credentials at risk. Learn more
Cancel
Previous step
Create environment
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences





Conclusion: We successfully created an account on AWS and also created an environment on Cloud9 IDE.

Experiment No :2

Title: To Build Your Application using AWS Code Build and Deploy on S3 / SEBS using AWS Code Pipeline, deploy Sample Application on EC2 instance using AWS Code Deploy.

Solution: In this experiment I am Building a Java application.

Step 1: Create the source code

1. In an empty directory on your local computer or instance, create this directory structure.

   (root directory name)

   ```
   `-- src
       |-- main
       |    `-- java
       `-- test
            `-- java
   ```

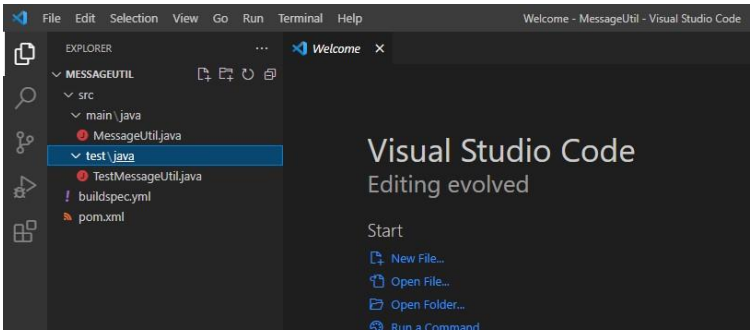
2. Using a text editor of your choice, create this file, name it MessageUtil.java, and then save it in the src/main/java directory.

This class file creates as output the string of characters passed into it. The `MessageUtil` constructor sets the string of characters. The `printMessage` method creates the output. The `salutationMessage` method outputs `Hi!` followed by the string of characters.

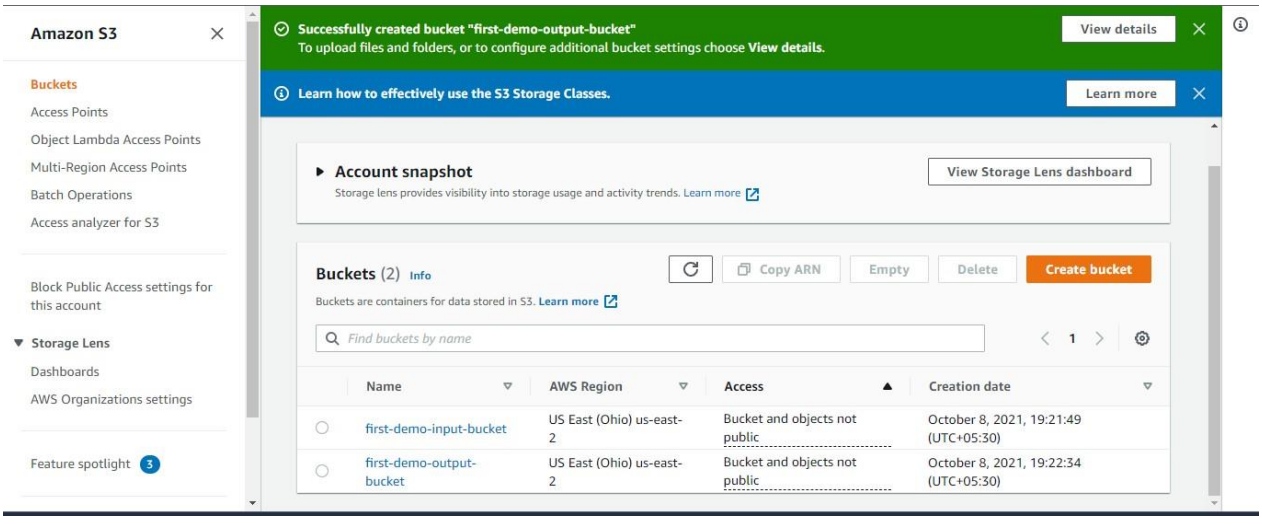
3. Create this file, name it TestMessageUtil.java, and then save it in the /src/test/java directory.
4. Create this file, name it pom.xml, and then save it in the root (top level) directory.

Step 2: Create the buildspec file.

Create this file, name it buildspec.yml, and then save it in the root (top level) directory.



Step 3: Create two S3 buckets



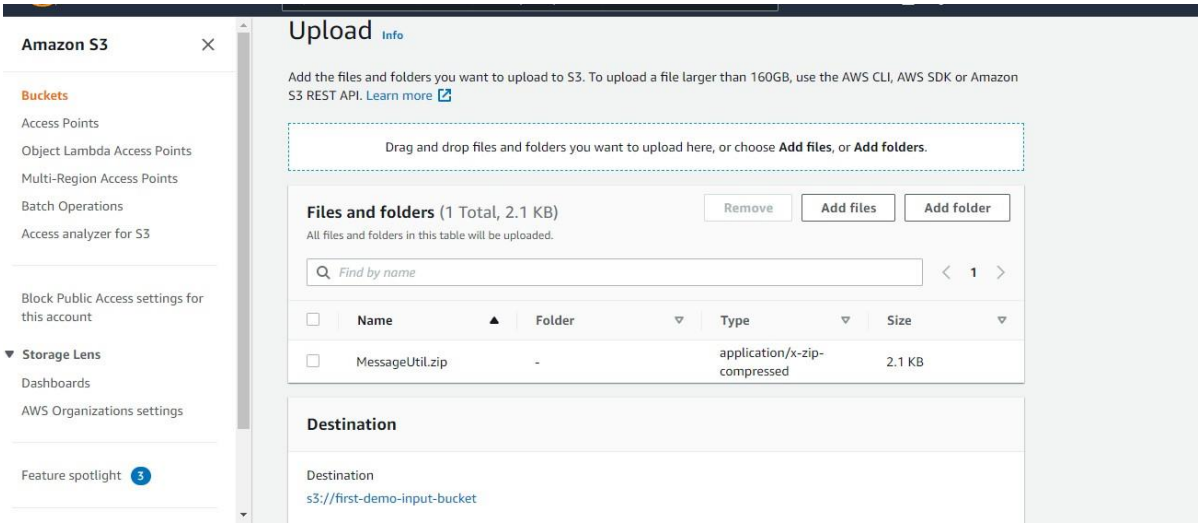
Step 4: Upload the source code and the buildspec file

In this step, you add the source code and build spec file to the input bucket.

Using your operating system's zip utility, create a file named MessageUtil.zip that includes MessageUtil.java, TestMessageUtil.java, pom.xml, and buildspec.yml.

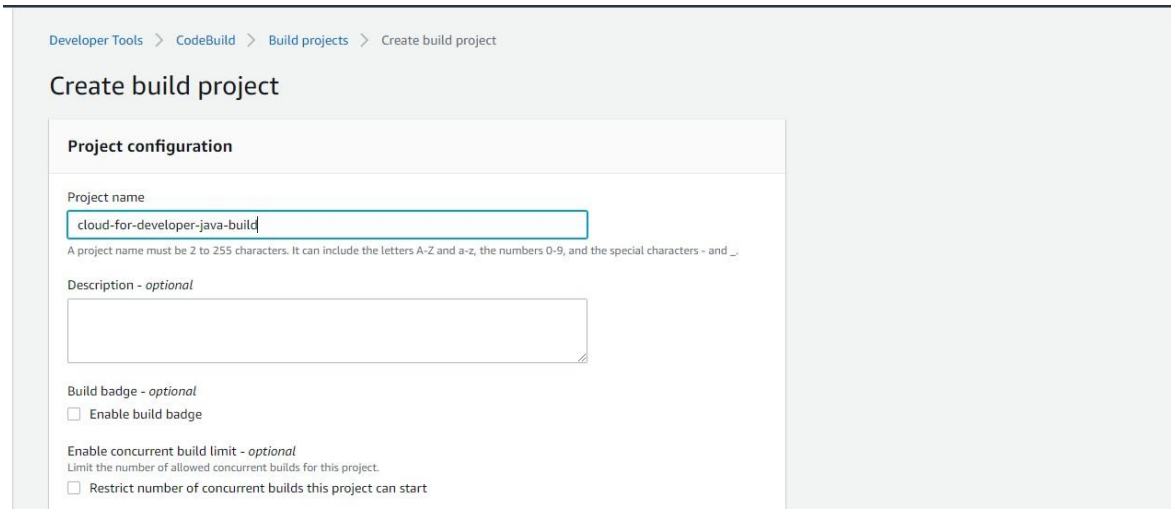
The MessageUtil.zip file's directory structure must look like this.

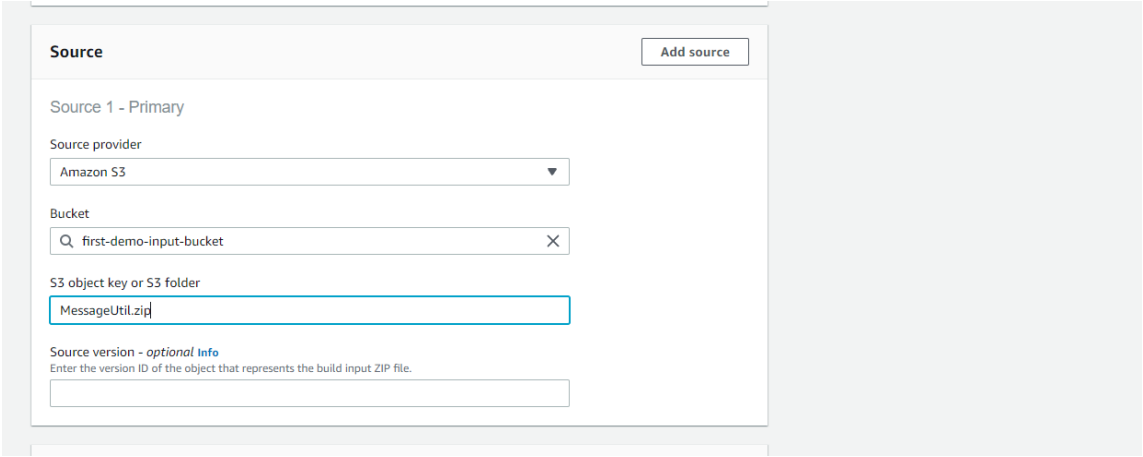
```
MessageUtil.zip
    |-- pom.xml
    |-- buildspec.yml
    `-- src
        |-- main
        |     `-- java
        |           `-- MessageUtil.java
        `-- test
              `-- java
                    `-- TestMessageUtil.java
```

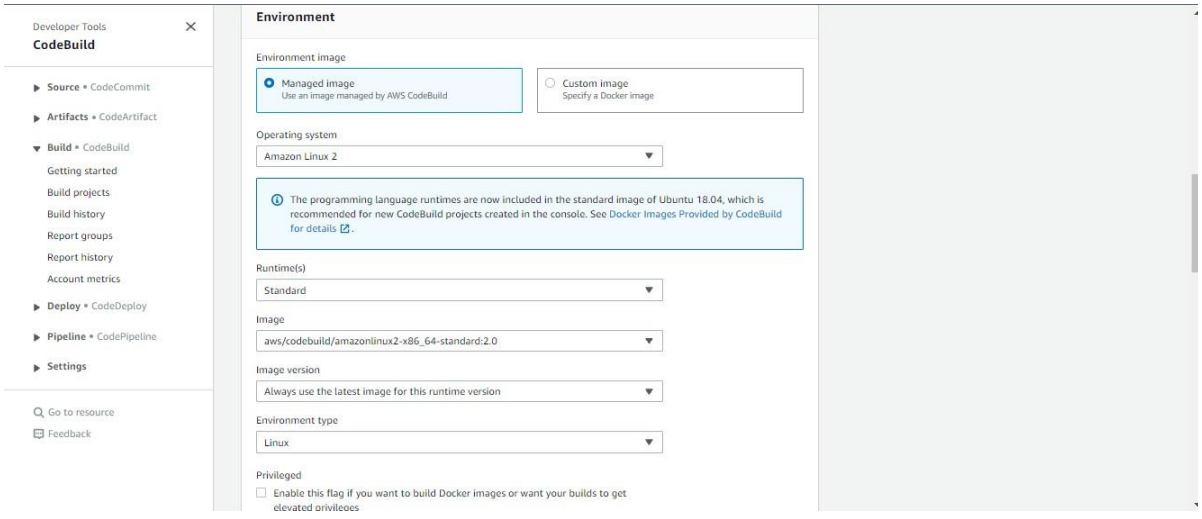


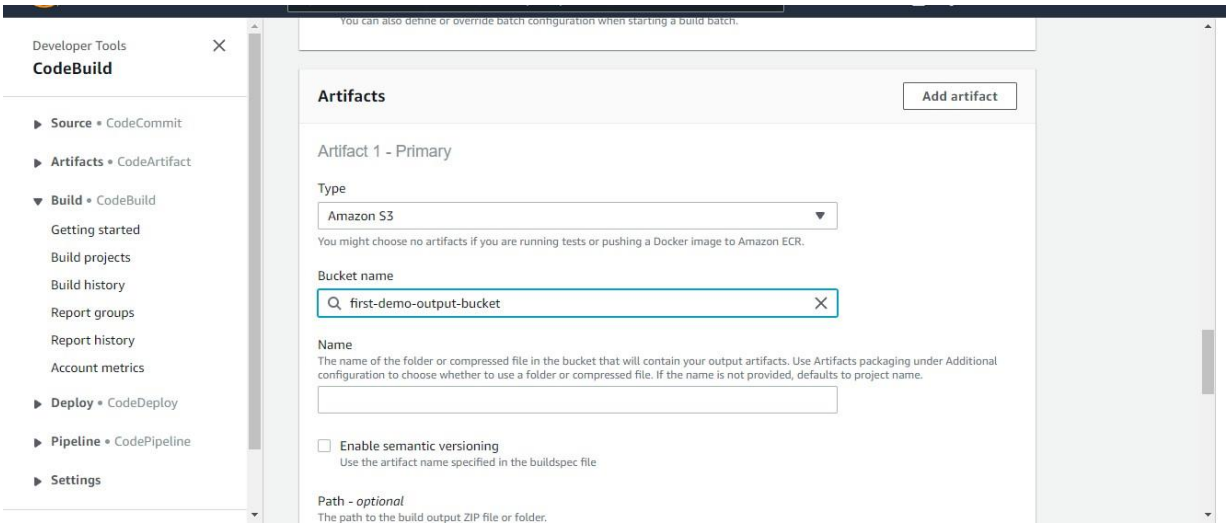
Step 5: Create the build project

In the CodeBuild service create new build Project

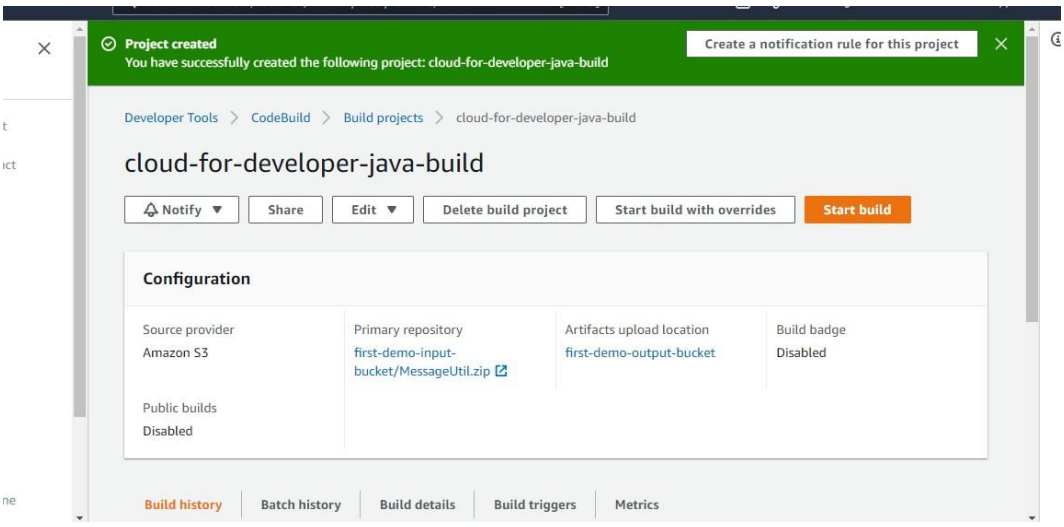

Developer Tools > CodeBuild > Build projects > Create build project
Create build project
Project configuration
Project name
cloud-for-developer-java-build
A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and _.
Description - optional
Build badge - optional
Enable build badge
Enable concurrent build limit - optional
Limit the number of allowed concurrent builds for this project.
Restrict number of concurrent builds this project can start


Source
Add source
Source 1 - Primary
Source provider
Amazon S3
Bucket
first-demo-input-bucket
S3 object key or S3 folder
MessageUtil.zip
Source version - optional Info
Enter the version ID of the object that represents the build input ZIP file.

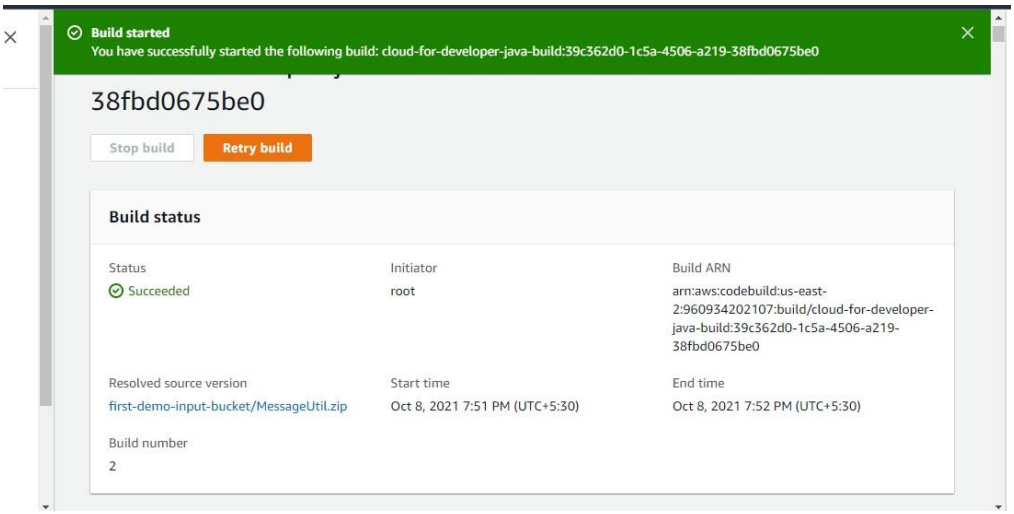

Developer Tools
CodeBuild
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build history
Report groups
Report history
Account metrics
Deploy • CodeDeploy
Pipeline • CodePipeline
Settings
Go to resource
Feedback
Environment
Environment image
Managed image
Use an image managed by AWS CodeBuild
Custom image
Specify a Docker image
Operating system
Amazon Linux 2
The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See Docker Images Provided by CodeBuild for details.
Runtime(s)
Standard
Image
aws/codebuild/amazonlinux2-x86_64-standard:2.0
Image version
Always use the latest image for this runtime version
Environment type
Linux
Privileged
Enable this flag if you want to build Docker images or want your builds to get elevated privileges



Step 6: Start Build by clicking on start Build

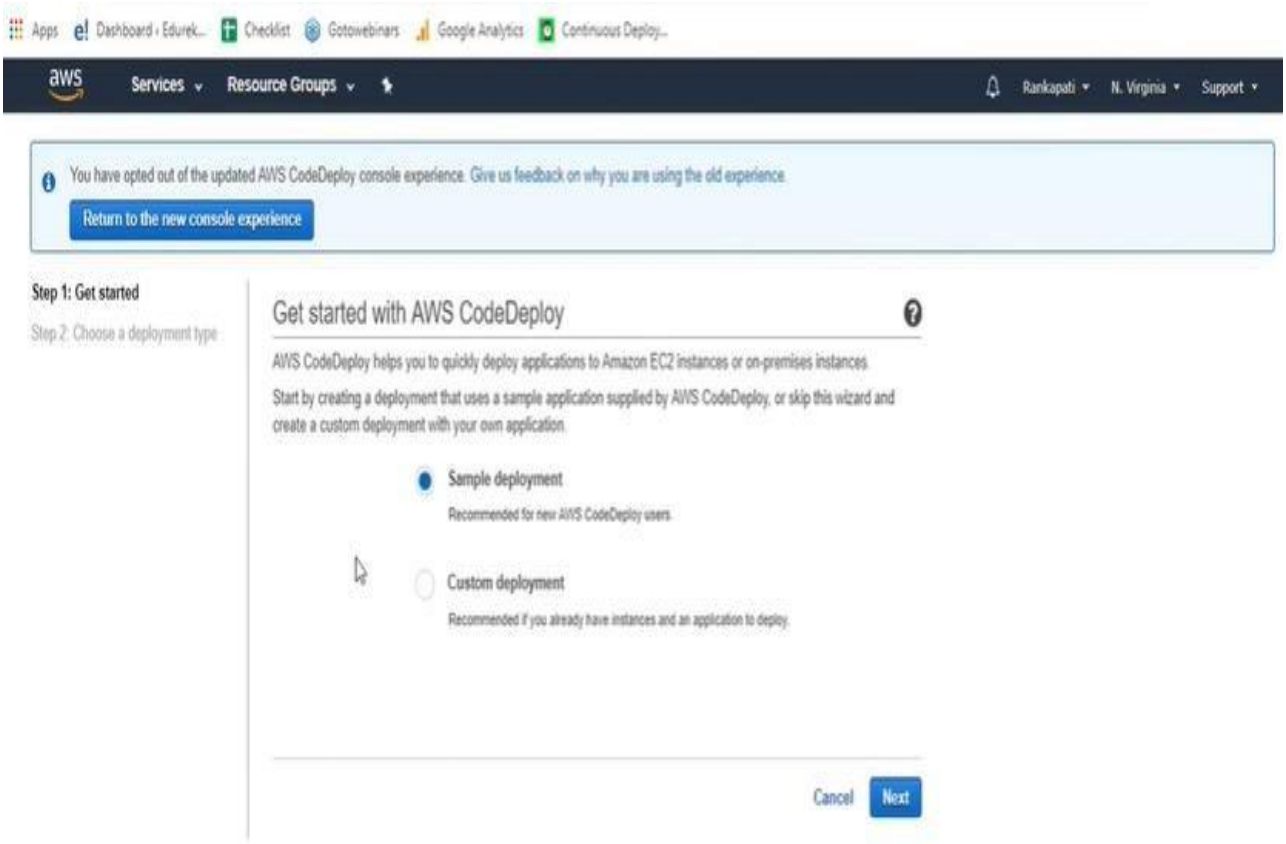


Step 7: Check Status success or fail

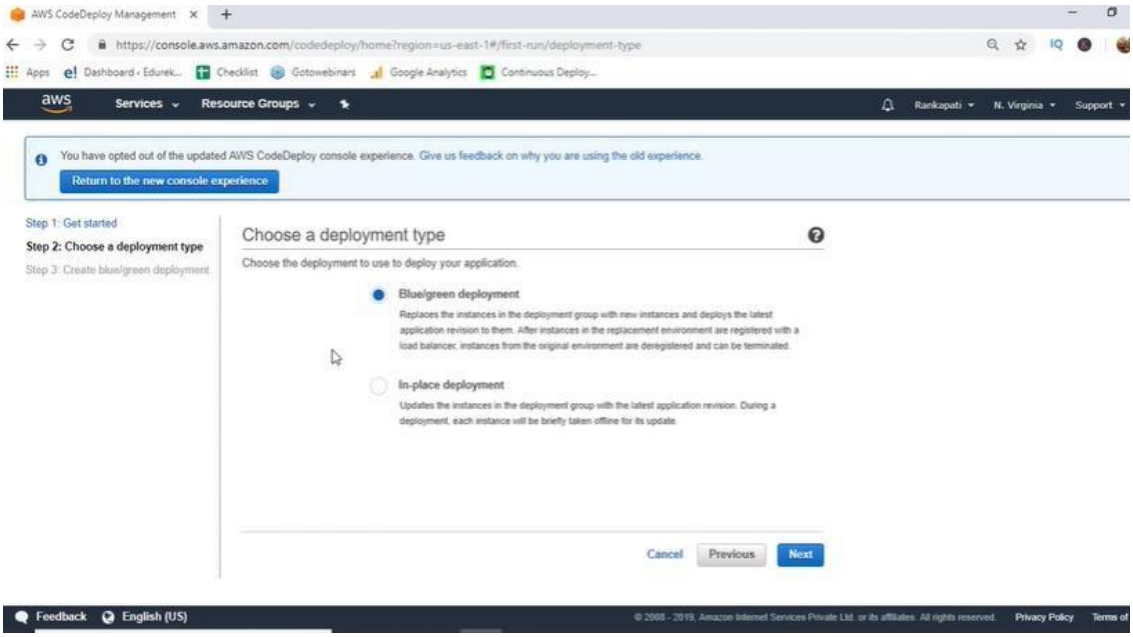


- **Deploy code using CodeDeploy**

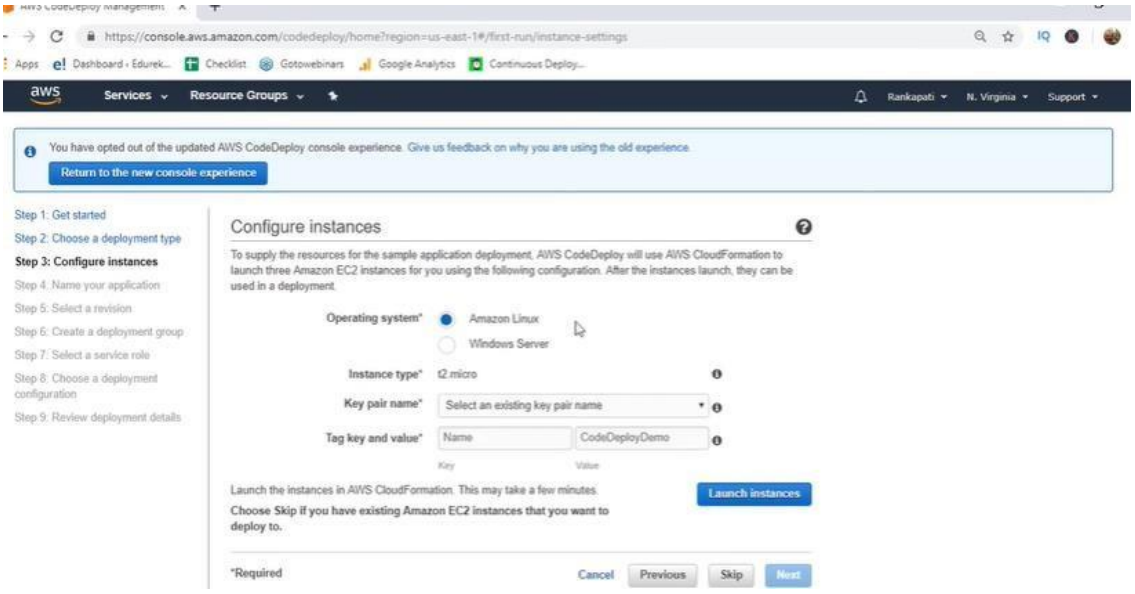
Step 1: Select Sample Deployment.



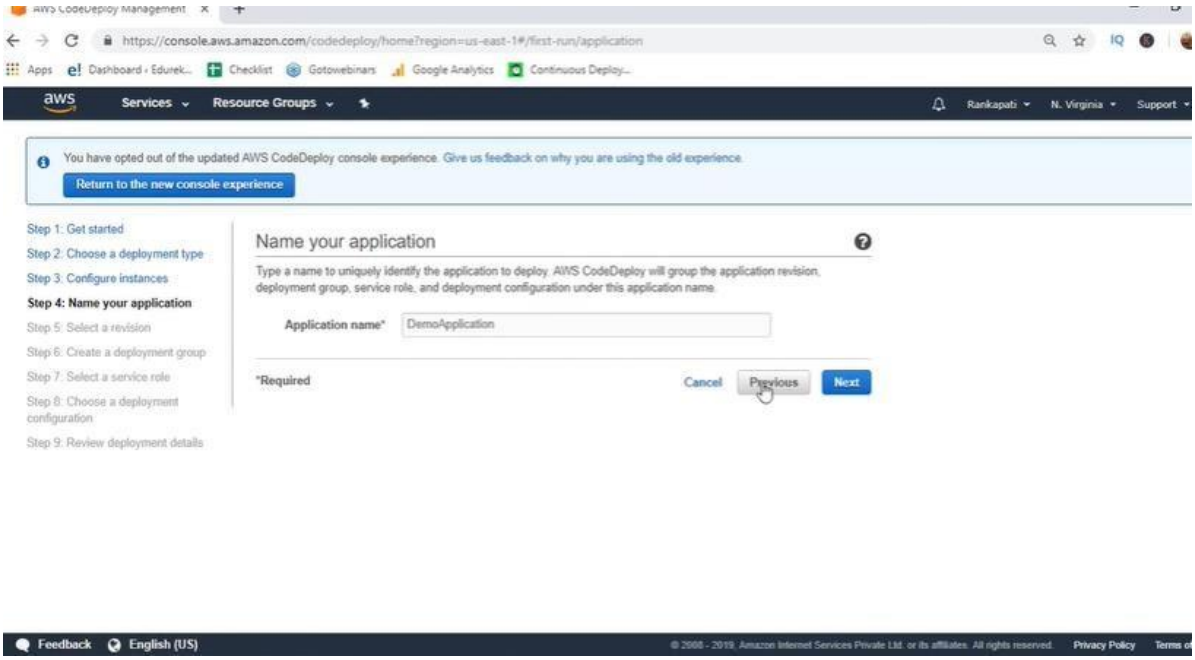
Step 2: Choose Deployment Type.



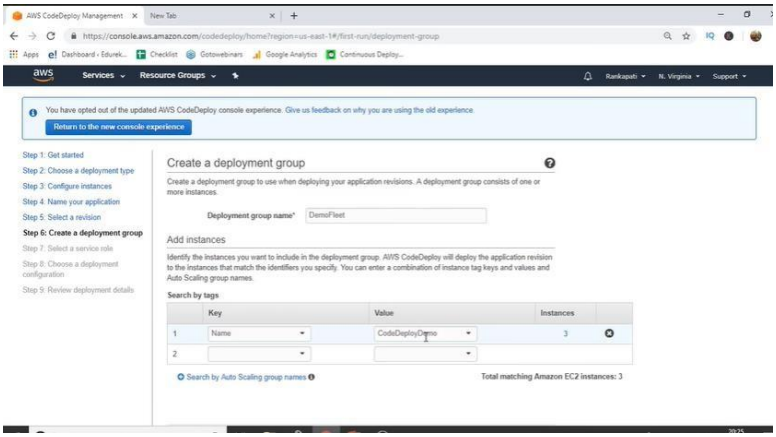
Step 3: Configure Instances



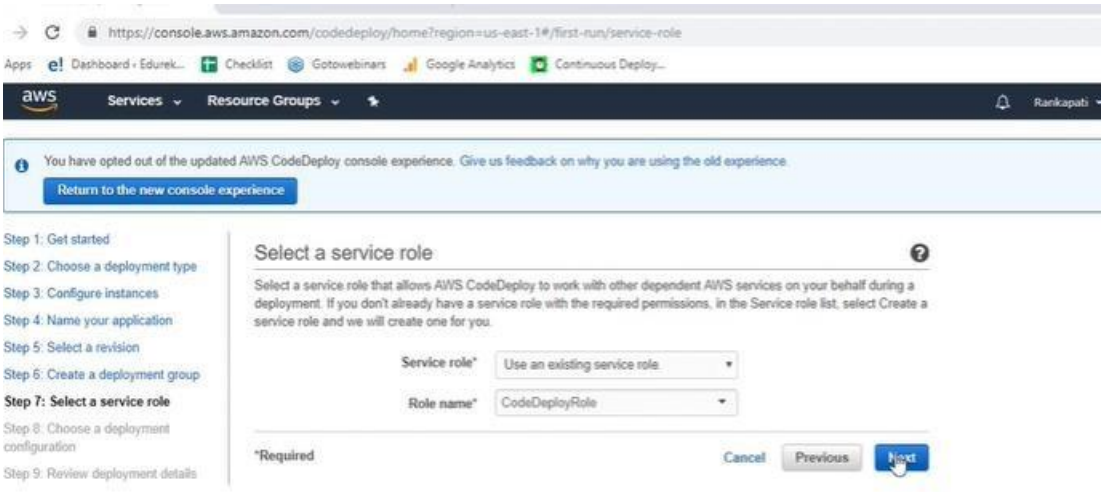
Step 4: Name Your Application.



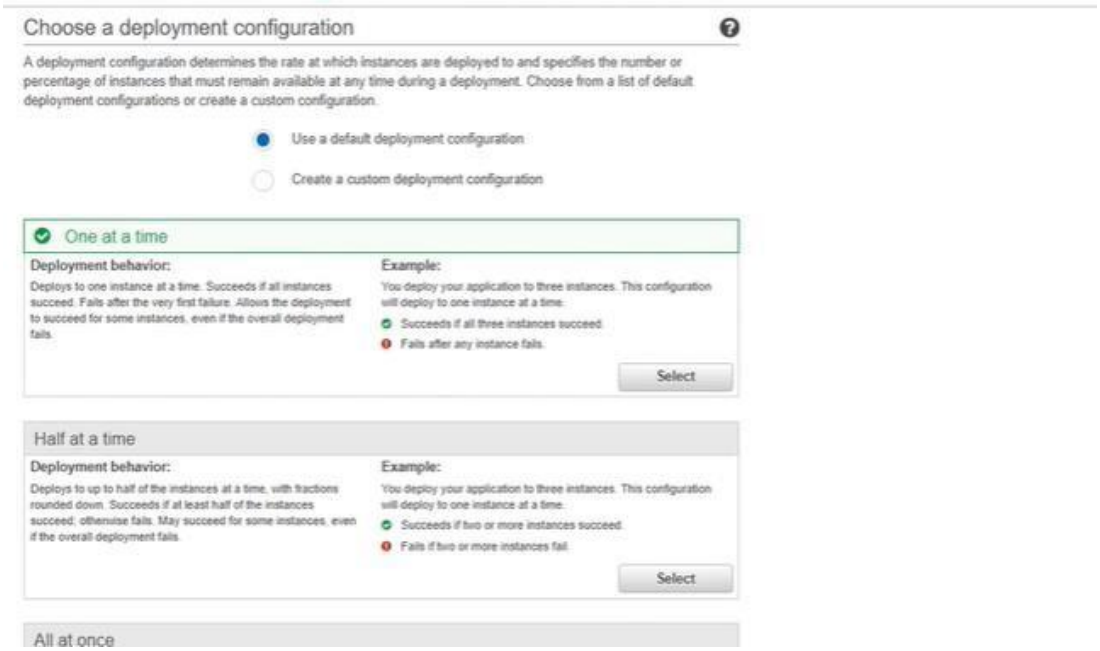
Step 5: Create Deployment Group



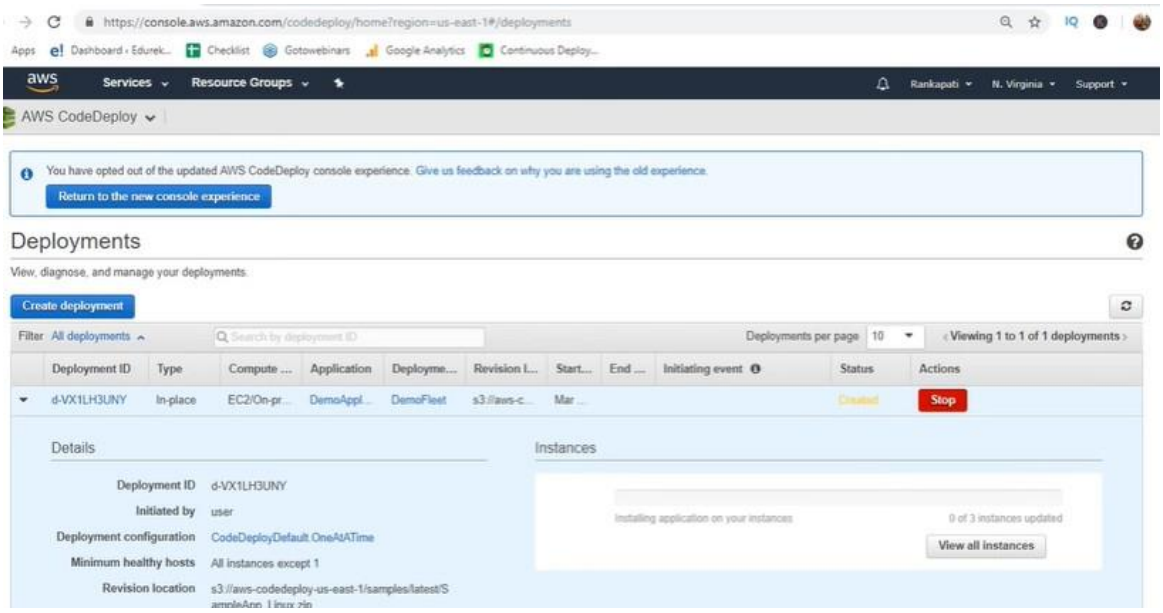
Step 6: Select Service Role



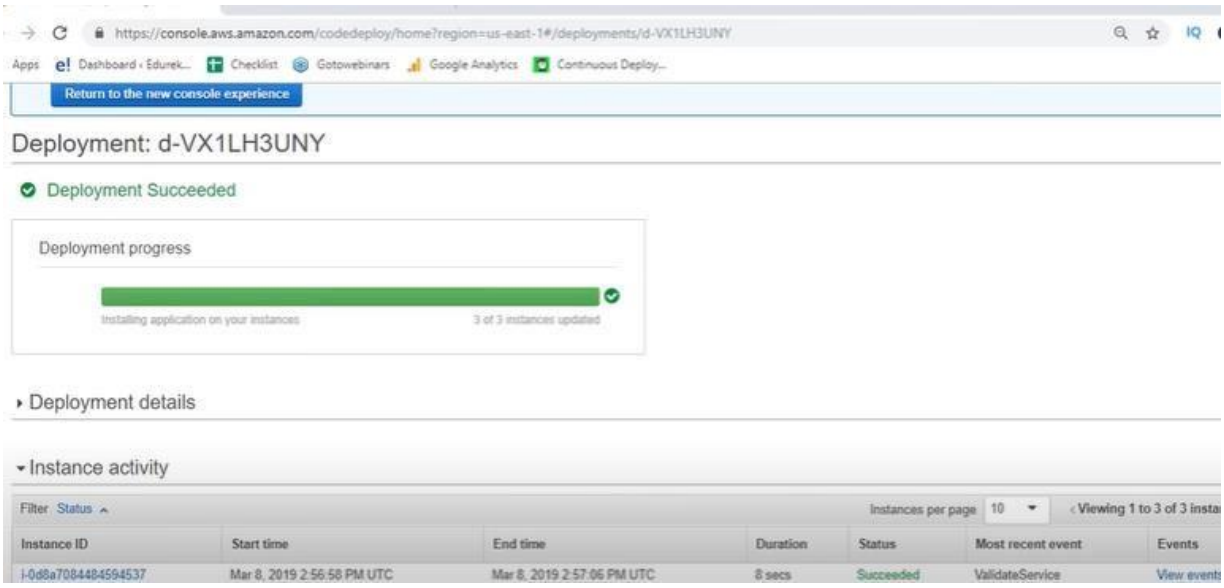
Step 7: Choose Deployment Configuration



Step 8: Deploy Application



Step 9: Check Status



Result: The CodeBuild and CodeDeploy operation is successfully performed. This Experiment helped me to understand the basic of AWS cloud9 and other services of AWS.

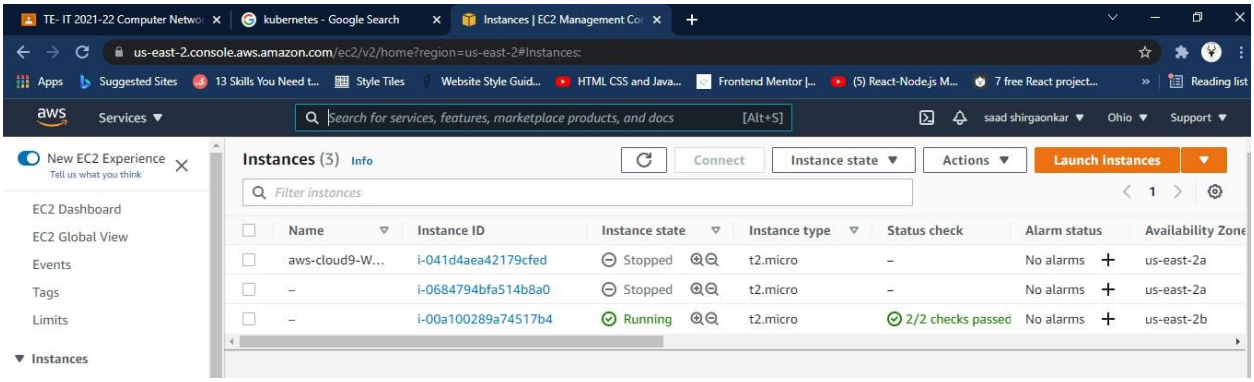
Conclusion: All instance to demonstrate the CodeBuild and CodeDeploy were successfully created and executed.

Experiment No: 3

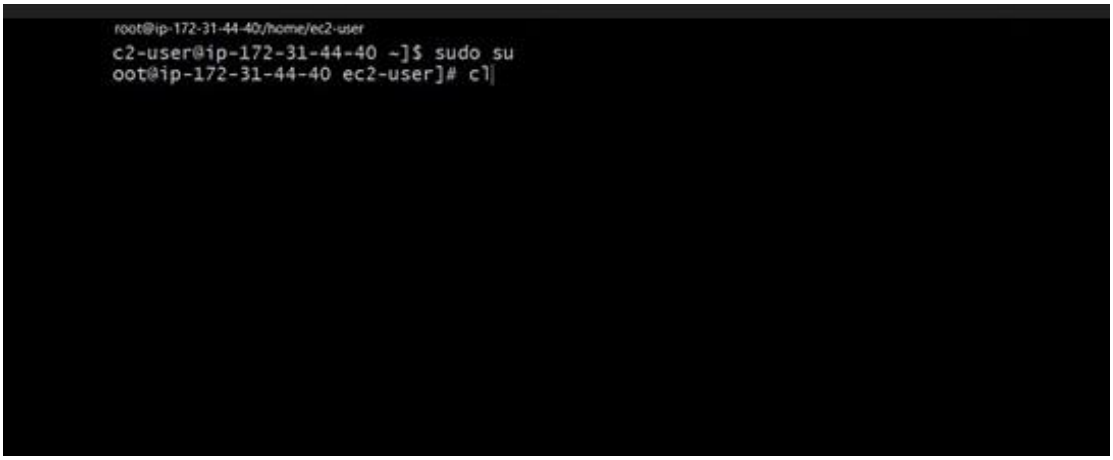
Title: To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.

Solution:

Step 1: Create a new or use an existing Cloud instance.



Step 2: Take Super user Access. Type Sudo su.



Step 3: Type the shown Command

```
root@ip-172-31-44-40/home/ec2-user
oot@ip-172-31-44-40 ec2-user]# free -h
             total        used        free      shared  buff/cache   available
m:            3.9G         89M        3.6G        408K        184M        3.6G
ap:             0B          0B          0B
oot@ip-172-31-44-40 ec2-user]# swapoff -a
oot@ip-172-31-44-40 ec2-user]# cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
```

Step 4: Installation of Docker and enable docker

```
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
oot@ip-172-31-44-40 ec2-user]# yum install docker
aded plugins: extras_suggestions, langpacks, priorities, update-motd
solving Dependencies
> Running transaction check
-> Package docker.x86_64 0:19.03.13ce-1.amzn2 will be installed
> Processing Dependency: runc >= 1.0.0 for package: docker-19.03.13ce-1.amzn2.x86_64
> Processing Dependency: containerd >= 1.3.2 for package: docker-19.03.13ce-1.amzn2.x86_64
> Processing Dependency: pigz for package: docker-19.03.13ce-1.amzn2.x86_64
> Processing Dependency: libcgroup for package: docker-19.03.13ce-1.amzn2.x86_64
> Running transaction check
-> Package containerd.x86_64 0:1.4.1-2.amzn2 will be installed
-> Package libcgroup.x86_64 0:0.41-21.amzn2 will be installed
-> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
-> Package runc.x86_64 0:1.0.0-0.1.20200826.gitff819c7.amzn2 will be installed
> Finished Dependency Resolution

pendencies Resolved

================================================================================================================
ackage                   Arch              Version                                    Repository           Size
================================================================================================================
stalling:
ocker                    x86_64            19.03.13ce-1.amzn2                         amzn2extra-docker    37 M
stalling for dependencies:
ontainerd                x86_64            1.4.1-2.amzn2                              amzn2extra-docker    24 M
ibcgroup                 x86_64            0.41-21.amzn2                              amzn2-core           66 k
igz                      x86_64            2.3.4-1.amzn2.0.1                          amzn2-core           81 k
unc                      x86_64            1.0.0-0.1.20200826.gitff819c7.amzn2        amzn2extra-docker   3.7 M

ansaction Summary
================================================================================================================
stall  1 Package (+4 Dependent packages)

tal download size: 65 M
stalled size: 270 M
 this ok [y/d/N]: y
wnloading packages:
```

```
stalled size: 270 M
 this ok [y/d/N]: y
wnloading packages:
/5): libcgroup-0.41-21.amzn2.x86_64.rpm                                                       |  66 kB  00:00:00
/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm                                                        |  81 kB  00:00:00
/5): containerd-1.4.1-2.amzn2.x86_64.rpm                                                      |  24 MB  00:00:00
/5): runc-1.0.0-0.1.20200826.gitff819c7.amzn2.x86_64.rpm                                      | 3.7 MB  00:00:00
/5): docker-19.03.13ce-1.amzn2.x86_64.rpm                                                     |  37 MB  00:00:01
----------------------------------------------------------------------------------------------------------------
tal                                                                                  61 MB/s |  65 MB  00:00:01
nning transaction check
nning transaction test
ansaction test succeeded
nning transaction
Installing : runc-1.0.0-0.1.20200826.gitff819c7.amzn2.x86_64
Installing : containerd-1.4.1-2.amzn2.x86_64
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64
Installing : libcgroup-0.41-21.amzn2.x86_64
Installing : docker-19.03.13ce-1.amzn2.x86_64
Verifying  : docker-19.03.13ce-1.amzn2.x86_64
Verifying  : libcgroup-0.41-21.amzn2.x86_64
Verifying  : pigz-2.3.4-1.amzn2.0.1.x86_64
Verifying  : containerd-1.4.1-2.amzn2.x86_64
Verifying  : runc-1.0.0-0.1.20200826.gitff819c7.amzn2.x86_64

stalled:
docker.x86_64 0:19.03.13ce-1.amzn2

pendency Installed:
containerd.x86_64 0:1.4.1-2.amzn2   libcgroup.x86_64 0:0.41-21.amzn2   pigz.x86_64 0:2.3.4-1.amzn2.0.1   runc.x86_64 0:1.0.0-0.1.20200826.gitff819c7.am

mplete!
oot@ip-172-31-44-40 ec2-user]# systemctl enable docker
eated symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
```

Step 5: Configure Kubernetes packages.

```
root@ip-172-31-44-40:/home/ec2-user
oot@ip-172-31-44-40 ec2-user]# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
       https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
oot@ip-172-31-44-40 ec2-user]#
```

Step 6: installation of Kubelet, kubeadm, kubectl.

```
ot@ip-172-31-44-40 ec2-user]#
ot@ip-172-31-44-40 ec2-user]#
ot@ip-172-31-44-40 ec2-user]# yum install kubeadm kubectl kubelet
ded plugins: extras_suggestions, langpacks, priorities, update-motd
n2-core                                                                 | 3.7 kB  00:00:00
n2extra-docker                                                          | 3.0 kB  00:00:00
ernetes/signature                                                       |  844 B  00:00:00
rieving key from https://packages.cloud.google.com/yum/doc/yum-key.gpg
orting GPG key 0xA7317B0F:
erid     : "Google Cloud Packages Automatic Signing Key <gc-team@google.com>"
ngerprint: d0bc 747f d8ca f711 7500 d6fa 3746 c208 a731 7b0f
om       : https://packages.cloud.google.com/yum/doc/yum-key.gpg
this ok [y/N]: y
```

Step 7: initialize Kubelet.

```
ot@ip-172-31-44-40 ec2-user]#
ot@ip-172-31-44-40 ec2-user]# systemctl status kubelet
ubelet.service - kubelet: The Kubernetes Node Agent
 Loaded: loaded (/usr/lib/systemd/system/kubelet.service; disabled; vendor preset: disabled)
rop-In: /usr/lib/systemd/system/kubelet.service.d
         └─10-kubeadm.conf
 Active: inactive (dead)
   Docs: https://kubernetes.io/docs/
ot@ip-172-31-44-40 ec2-user]# kubeadm init
07 11:09:02.904822    5366 configset.go:348] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s

it] Using Kubernetes version: v1.19.4
eflight] Running pre-flight checks
      [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at http
kubernetes.io/docs/setup/cri/
      [WARNING FileExisting-tc]: tc not found in system path
      [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
eflight] Pulling images required for setting up a Kubernetes cluster
eflight] This might take a minute or two, depending on the speed of your internet connection
eflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

Step 8:

This commend used to make master as a worker in Single node cluster

kubectl taint nodes --all node-role.kubernetes.io/master-

To apply pods network communication, we use calico network

kubectl apply

-f https://docs.projectcalico.org/v3.9/manifests/calico.yaml

to see the list of Pods we will use this command

kubectl get pods --all-namespaces

Result: The Installation of Kubernetes on EC2 cluster is successfully performed. The installed version of Kubernetes is Kubernetes version: v1.22.2

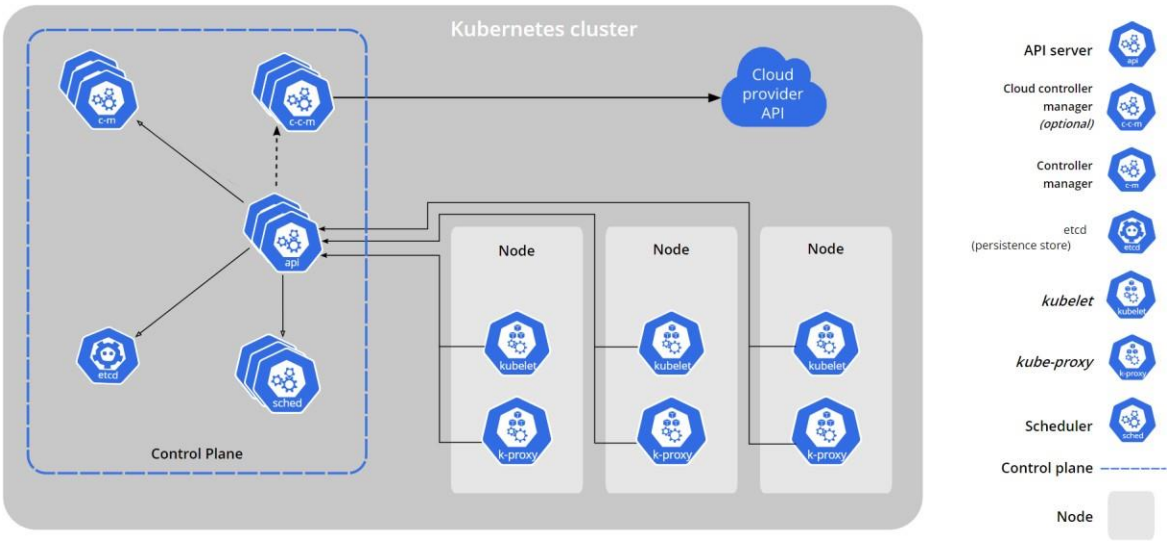
Conclusion: This experiment helped me to get a knowledge of Kubernetes. Kubernetes was successfully installed.

Experiment no. 4

Aim: To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.

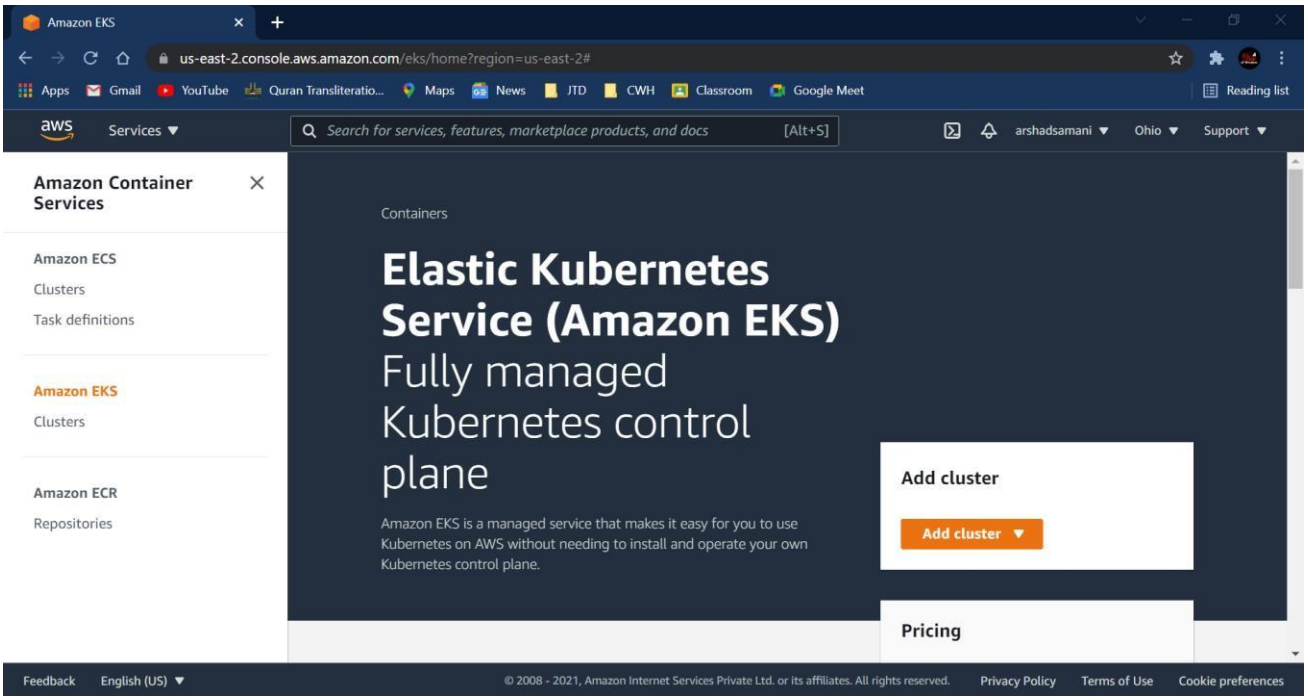
Kubernetes Components:

When you deploy Kubernetes, you get a cluster.
A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.
The worker node(s) host the Pods that are the components of the application workload.
The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.
This document outlines the various components you need to have for a complete and working Kubernetes cluster.
Here's the diagram of a Kubernetes cluster with all the components tied together.

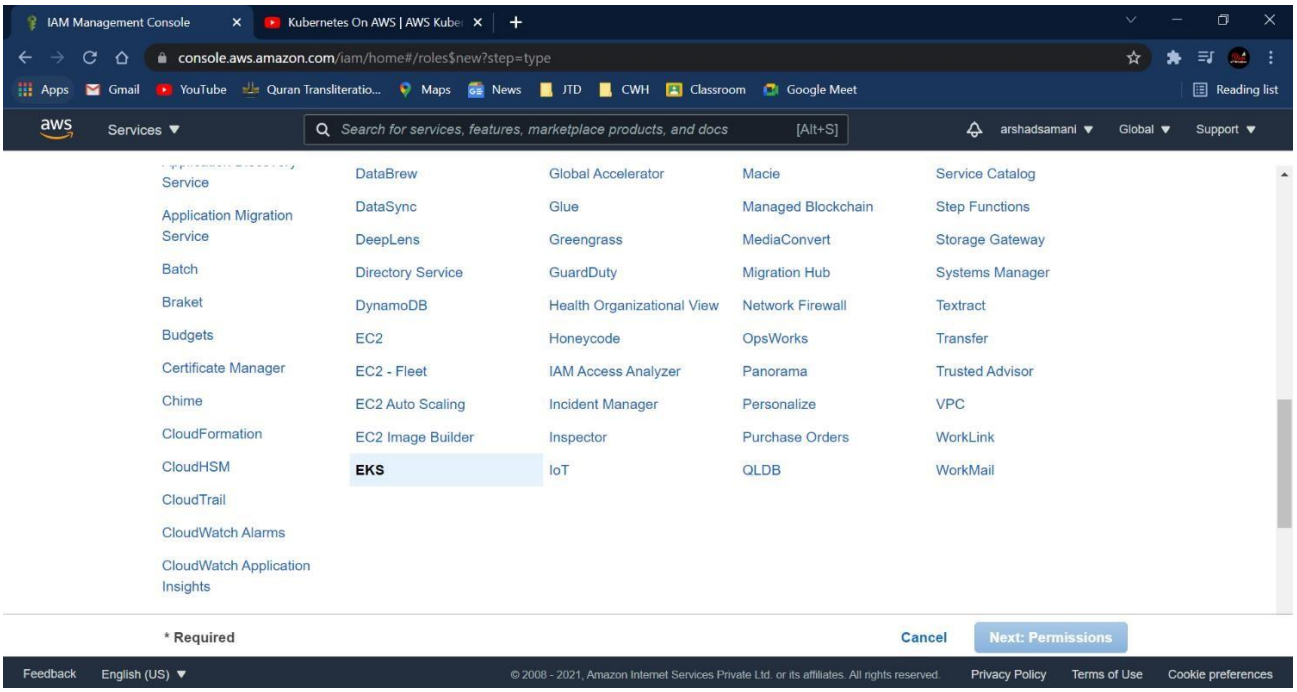


Outputs:

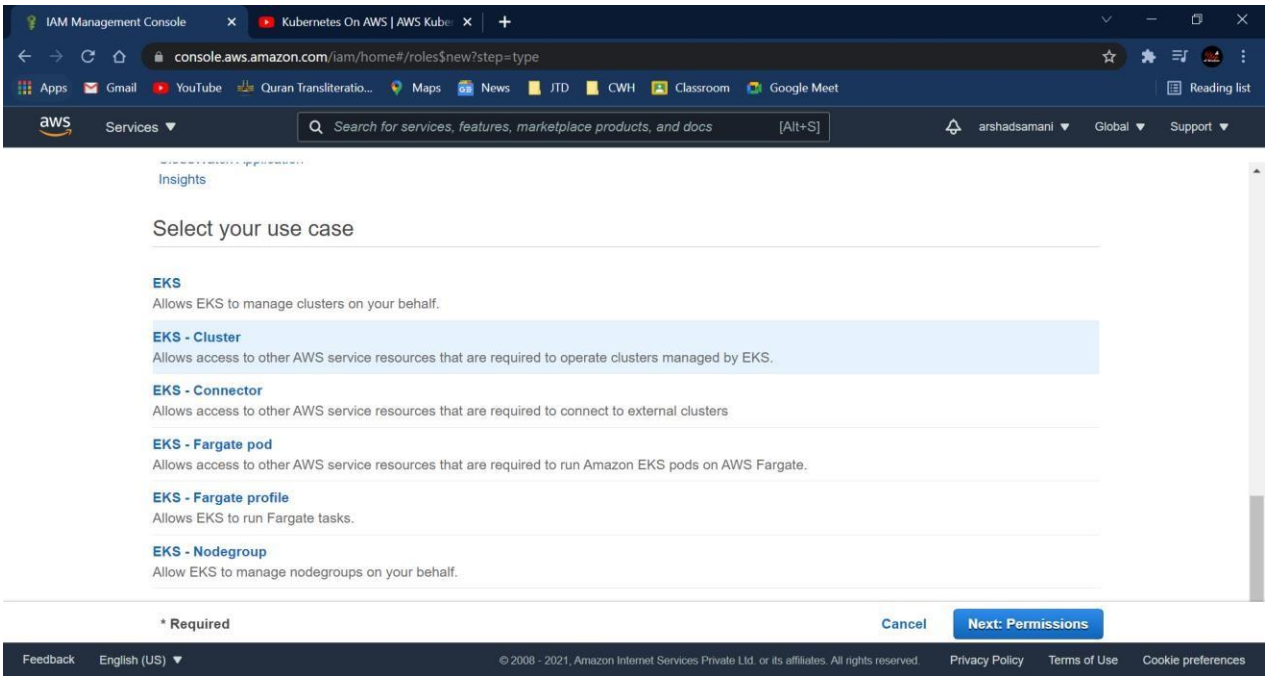
1. Starting with Kubernetes.



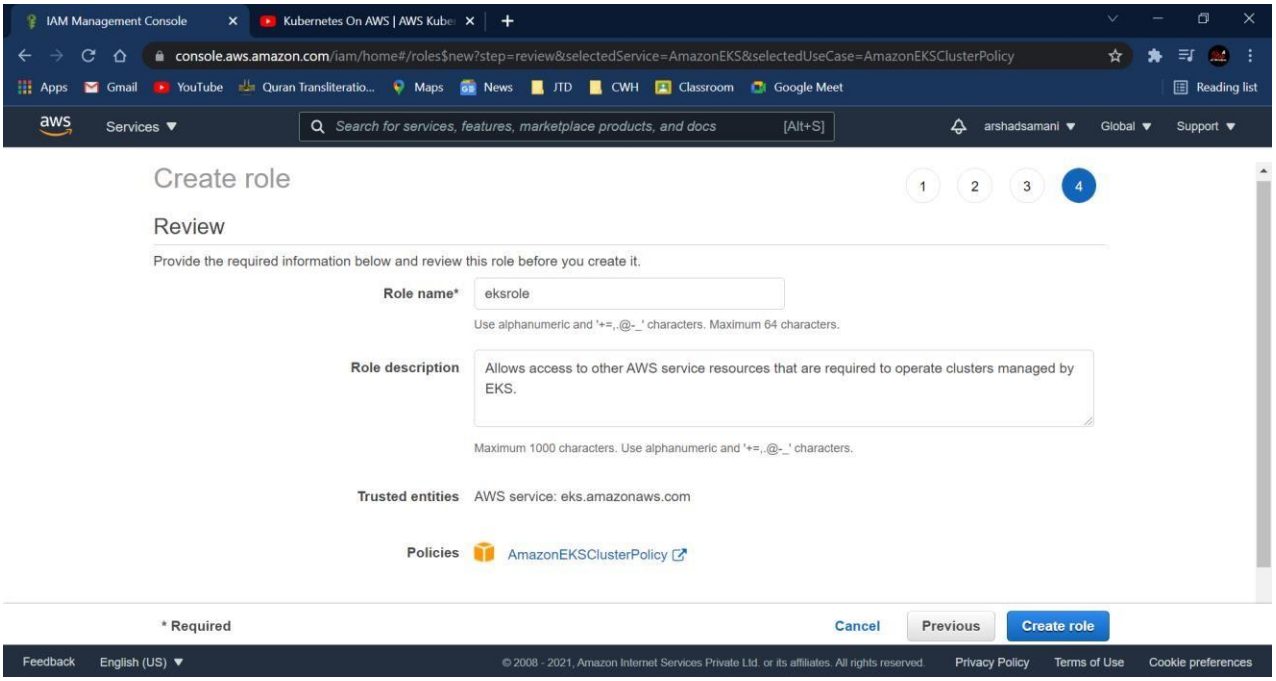
2. Selecting EKS



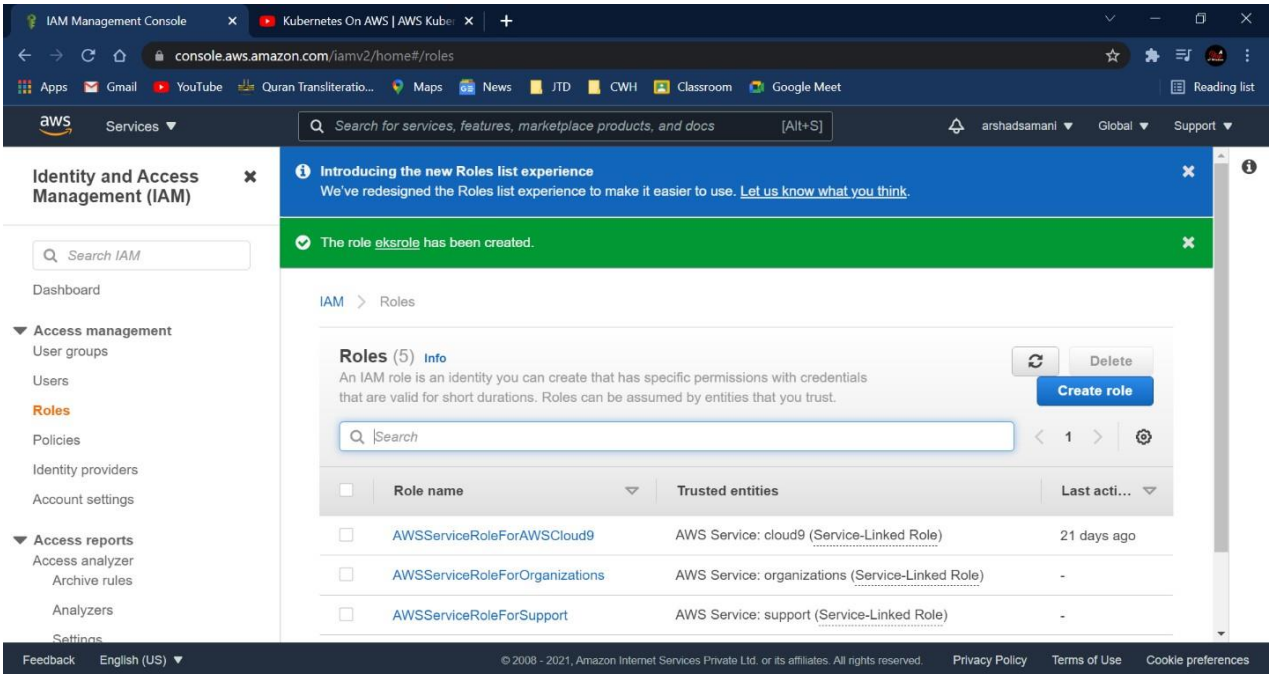
3. Select EKS - Cluster



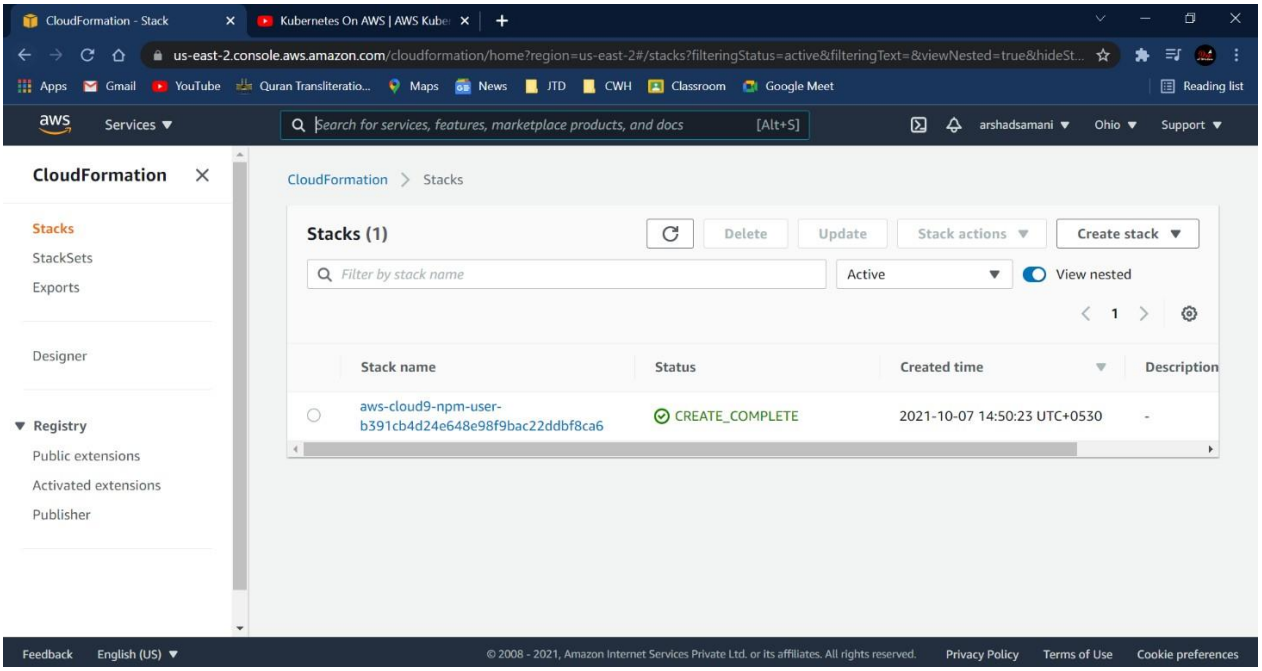
4. Creating Role

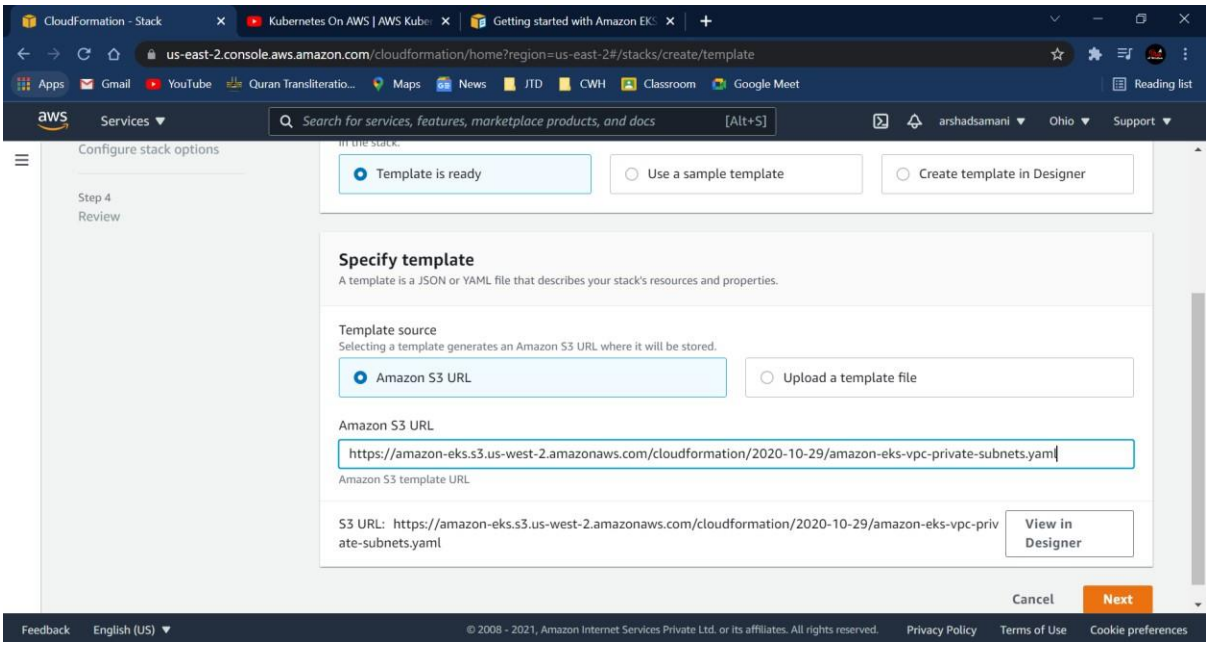


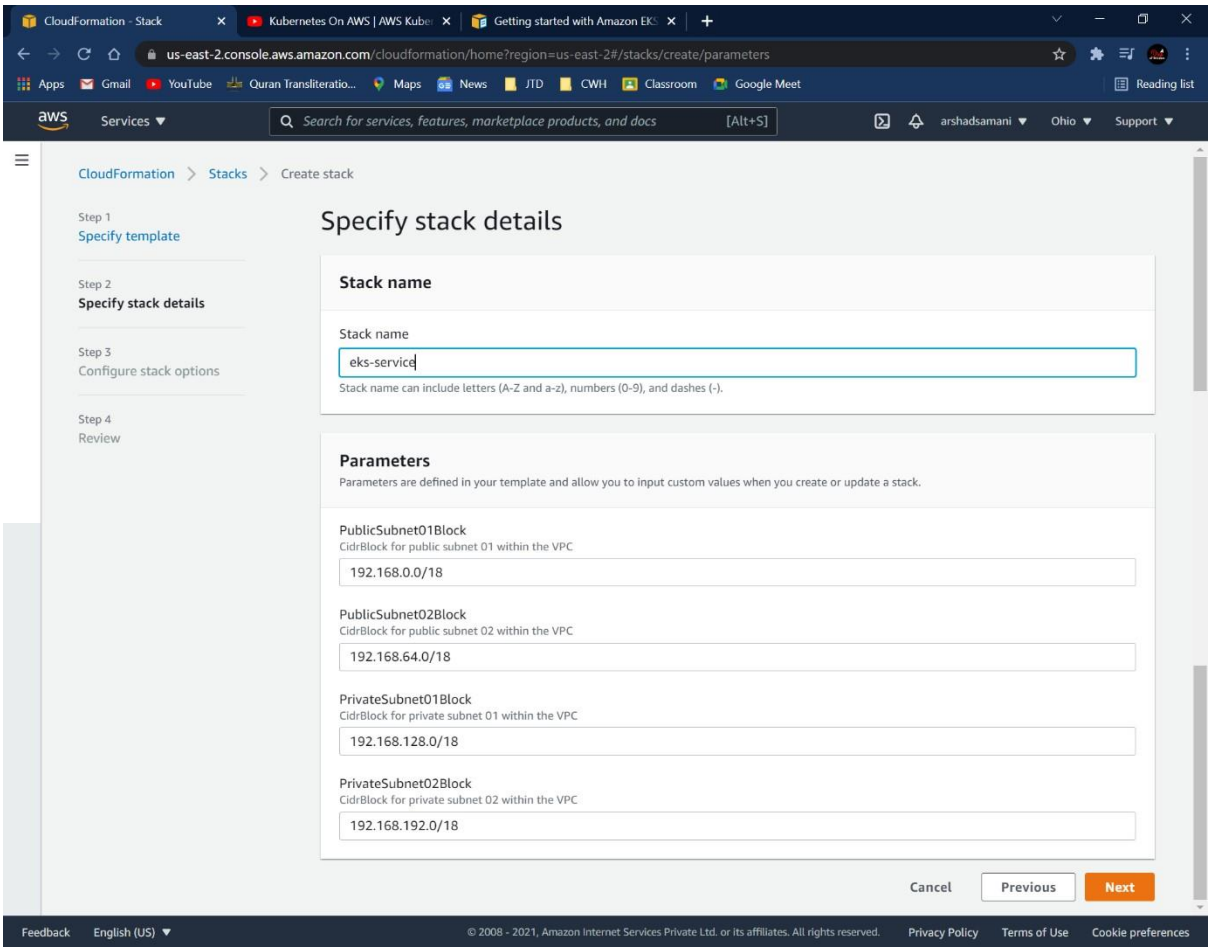
5. Role has been created



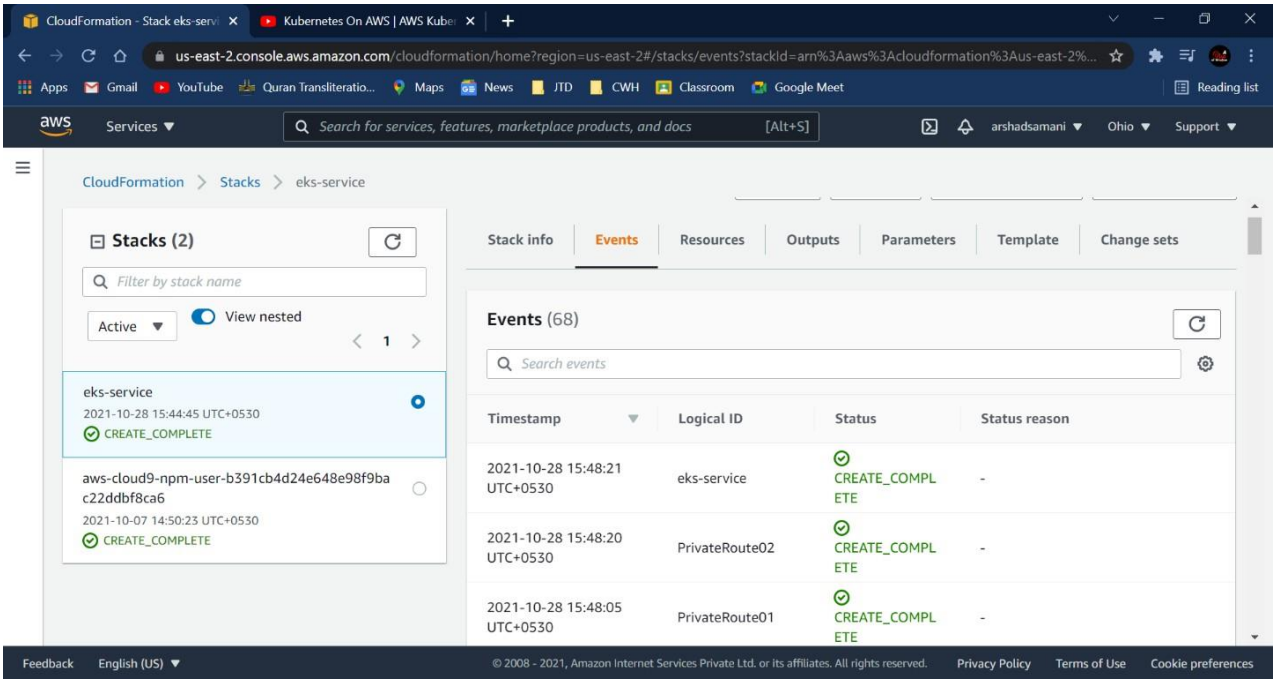
6. Cloud Formation



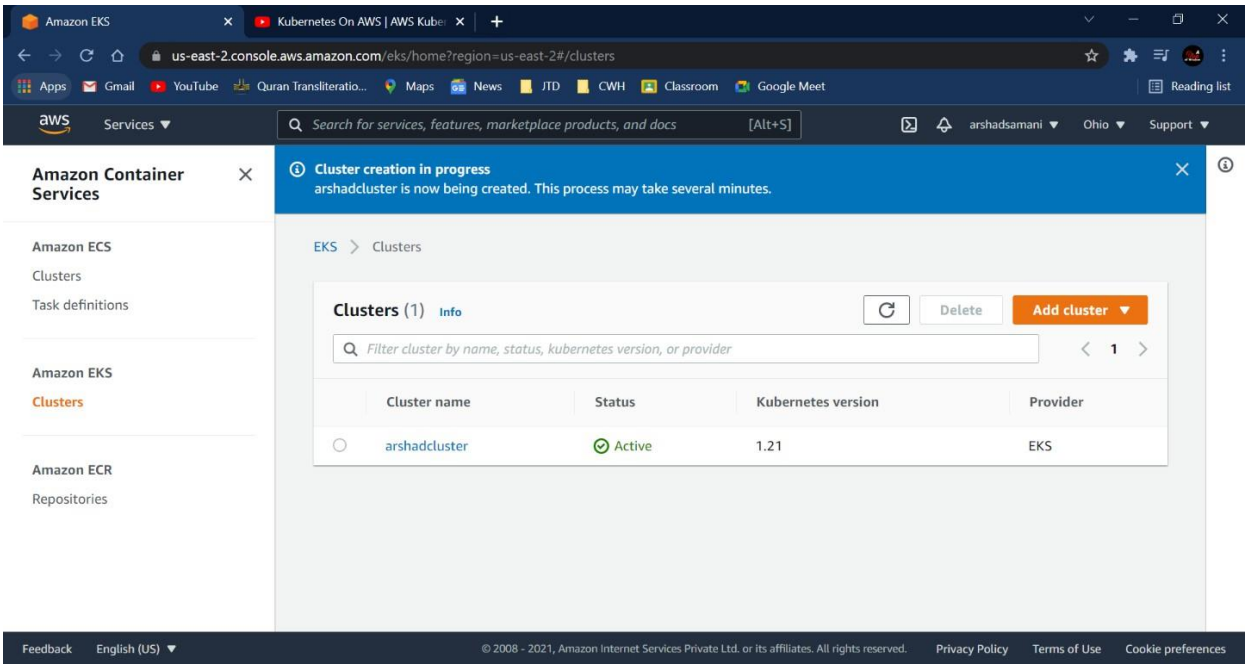

CloudFormation - Stack
Kubernetes On AWS | AWS Kube
Getting started with Amazon EK
us-east-2.console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/create/template
Apps
Gmail
YouTube
Quran Transliteratio...
Maps
News
JTD
CWH
Classroom
Google Meet
Reading list
aws
Services
Search for services, features, marketplace products, and docs
[Alt+S]
arshadsamani
Ohio
Support
Configure stack options
Step 4
Review
Template is ready
Use a sample template
Create template in Designer
Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.
Template source
Selecting a template generates an Amazon S3 URL where it will be stored.
Amazon S3 URL
Upload a template file
Amazon S3 URL
https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
Amazon S3 template URL
S3 URL: https://amazon-eks.s3.us-west-2.amazonaws.com/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
View in Designer
Cancel
Next
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences


CloudFormation - Stack
Kubernetes On AWS | AWS Kube
Getting started with Amazon EK
us-east-2.console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/create/parameters
Apps
Gmail
YouTube
Quran Transliteratio...
Maps
News
JTD
CWH
Classroom
Google Meet
Reading list
aws
Services
Search for services, features, marketplace products, and docs
[Alt+S]
arshadsamani
Ohio
Support
CloudFormation
Stacks
Create stack
Step 1
Specify template
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review
Specify stack details
Stack name
Stack name
eks-service
Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
PublicSubnet01Block
CidrBlock for public subnet 01 within the VPC
192.168.0.0/18
PublicSubnet02Block
CidrBlock for public subnet 02 within the VPC
192.168.64.0/18
PrivateSubnet01Block
CidrBlock for private subnet 01 within the VPC
192.168.128.0/18
PrivateSubnet02Block
CidrBlock for private subnet 02 within the VPC
192.168.192.0/18
Cancel
Previous
Next
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

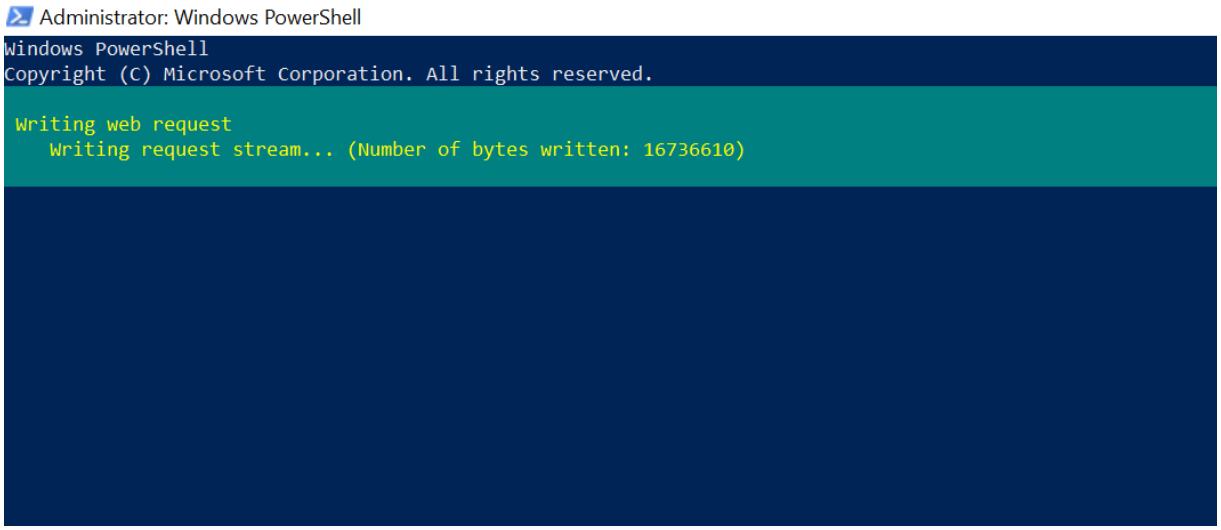
7. Stack is created

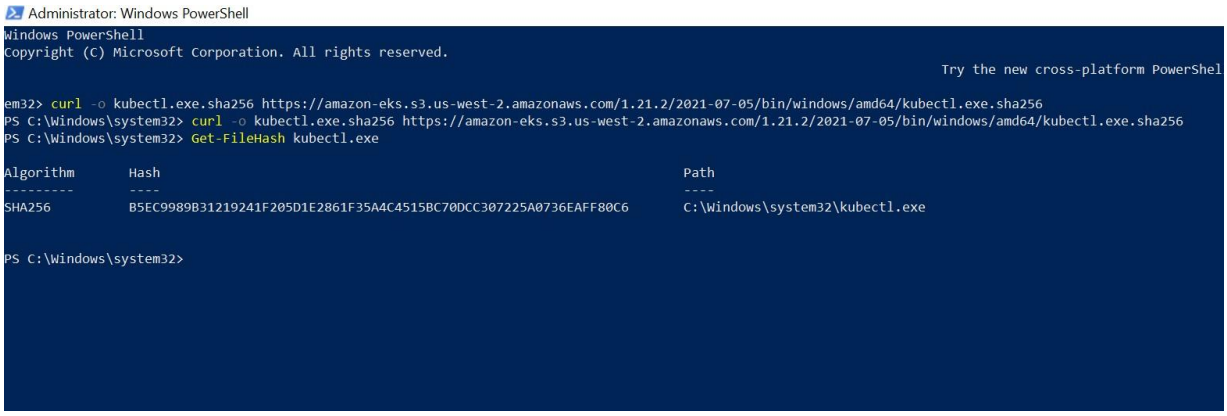


8. Cluster created

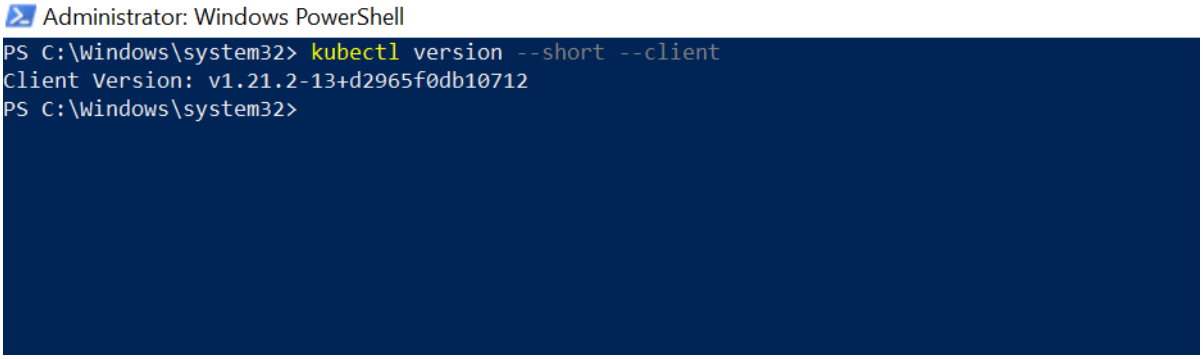


9. Installing kubectl





10. Succesfully Installed



Administrator: Windows PowerShell

```
PS C:\Windows\system32> C:\Windows\system32\kubectl.exe
kubectl controls the Kubernetes cluster manager.

 Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create        Create a resource from a file or from stdin.
  expose        Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
  run           Run a particular image on the cluster
  set           Set specific features on objects

Basic Commands (Intermediate):
  explain       Documentation of resources
  get           Display one or many resources
  edit          Edit a resource on the server
  delete        Delete resources by filenames, stdin, resources and names, or by resources and label selector

Deploy Commands:
  rollout       Manage the rollout of a resource
  scale         Set a new size for a Deployment, ReplicaSet or Replication Controller
  autoscale     Auto-scale a Deployment, ReplicaSet, StatefulSet, or ReplicationController

Cluster Management Commands:
  certificate   Modify certificate resources.
  cluster-info  Display cluster info
  top           Display Resource (CPU/Memory) usage.
  cordon        Mark node as unschedulable
  uncordon      Mark node as schedulable
  drain         Drain node in preparation for maintenance
  taint         Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
  describe      Show details of a specific resource or group of resources
  logs          Print the logs for a container in a pod
  attach        Attach to a running container
  exec          Execute a command in a container
  port-forward  Forward one or more local ports to a pod
  proxy         Run a proxy to the Kubernetes API server
  cp            Copy files and directories to and from containers.
  auth          Inspect authorization
  debug         Create debugging sessions for troubleshooting workloads and nodes

Advanced Commands:
  diff          Diff live version against would-be applied version
```

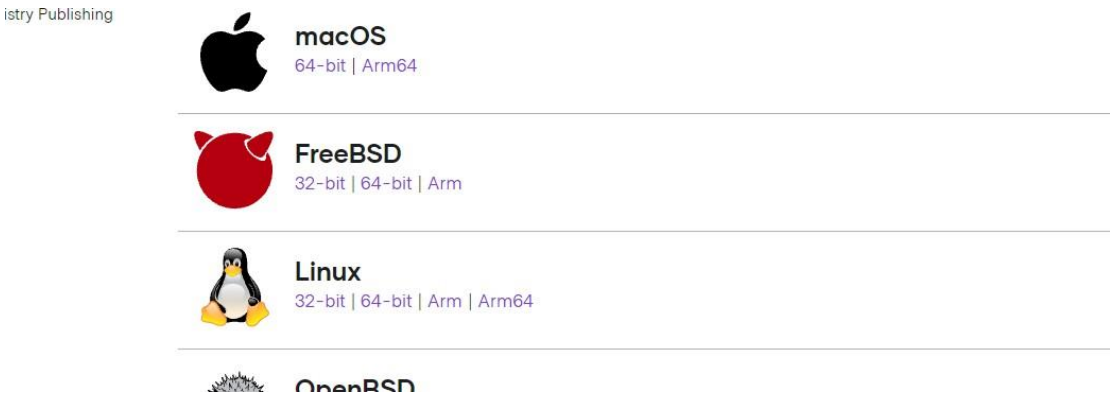
```
                                        NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   22m
PS C:\Users\snbha> kubectl get pods

                                        No resources found in default namespace.
PS C:\Users\snbha> kubectl get all

                                        NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PO
RT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   22m
PS C:\Users\snbha> kubectl get nods --watch
error: the server doesn't have a resource type "nods"
PS C:\Users\snbha> kubectl get nodes --watch
NAME                          STATUS     ROLES    AGE   VERSION
ip-172-31-79-10.ec2.internal   NotReady   <none>   0s    v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal   NotReady   <none>   0s    v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal   NotReady   <none>   0s    v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal   NotReady   <none>   0s    v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal   NotReady   <none>   0s    v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal   NotReady   <none>   0s    v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal    Ready      <none>   20s   v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal    Ready      <none>   20s   v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal   Ready      <none>   20s   v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal   Ready      <none>   20s   v1.18.9-eks-d1db3c
ip-172-31-79-10.ec2.internal    Ready      <none>   30s   v1.18.9-eks-d1db3c
ip-172-31-40-253.ec2.internal   Ready      <none>   30s   v1.18.9-eks-d1db3c
PS C:\Users\snbha>
```

Experiment No: 5

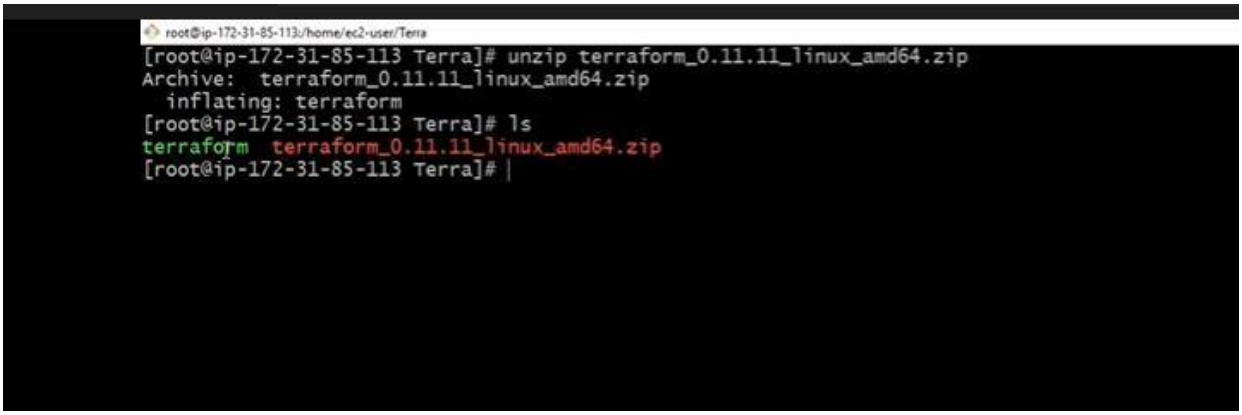
Title: To understand terraform lifecycle, core concepts/ terminologies and install it on a Linux machine.

Solution:

Step 1: Download the compatible installation file from the official website.



Step 2: Unzip the file.



Step 3: Create the Environment variable path for the Terraform.

Terraform is Successfully installed.

Result: The Experiment helped us to terraform lifecycle, core concepts/ terminologies. Terraform was successfully installed on a Linux machine.

Conclusion: The Terraform Installation is Successfully performed.

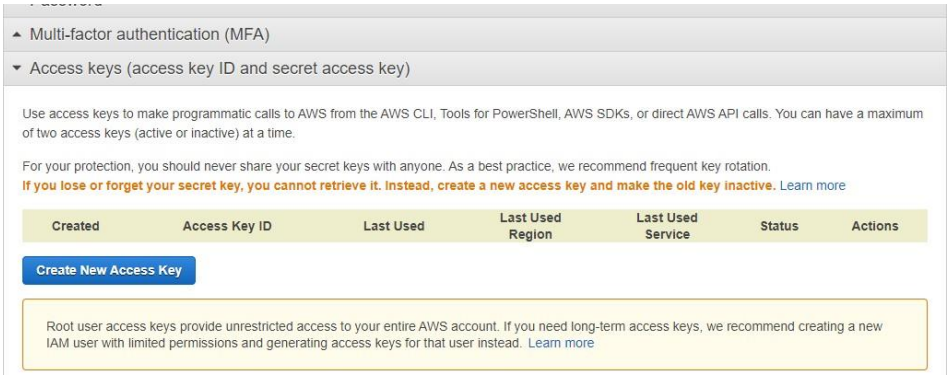
Experiment No: 6

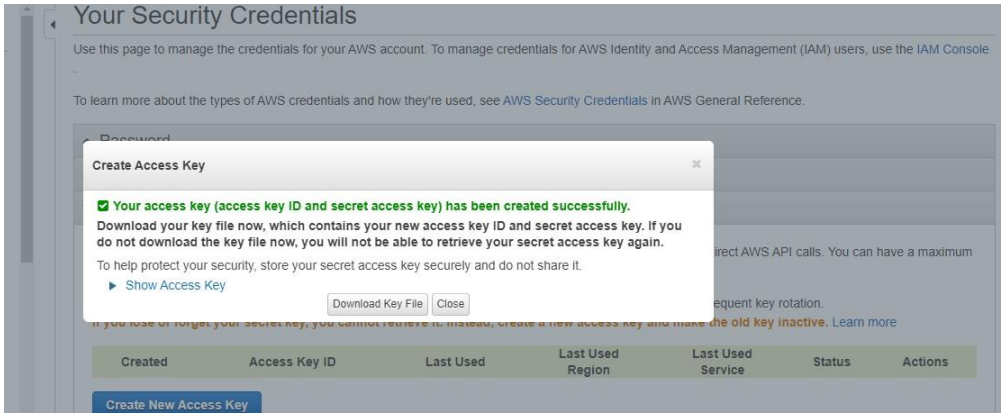
Title: To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.

Solution:

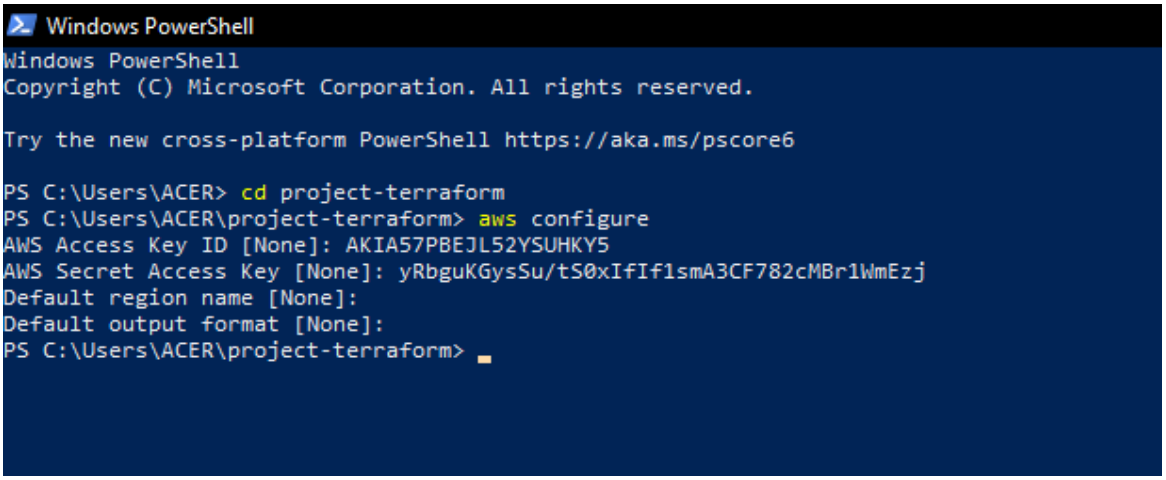
1. Build AWS Infrastructure.

Step 1: Create a new AWS Access Key

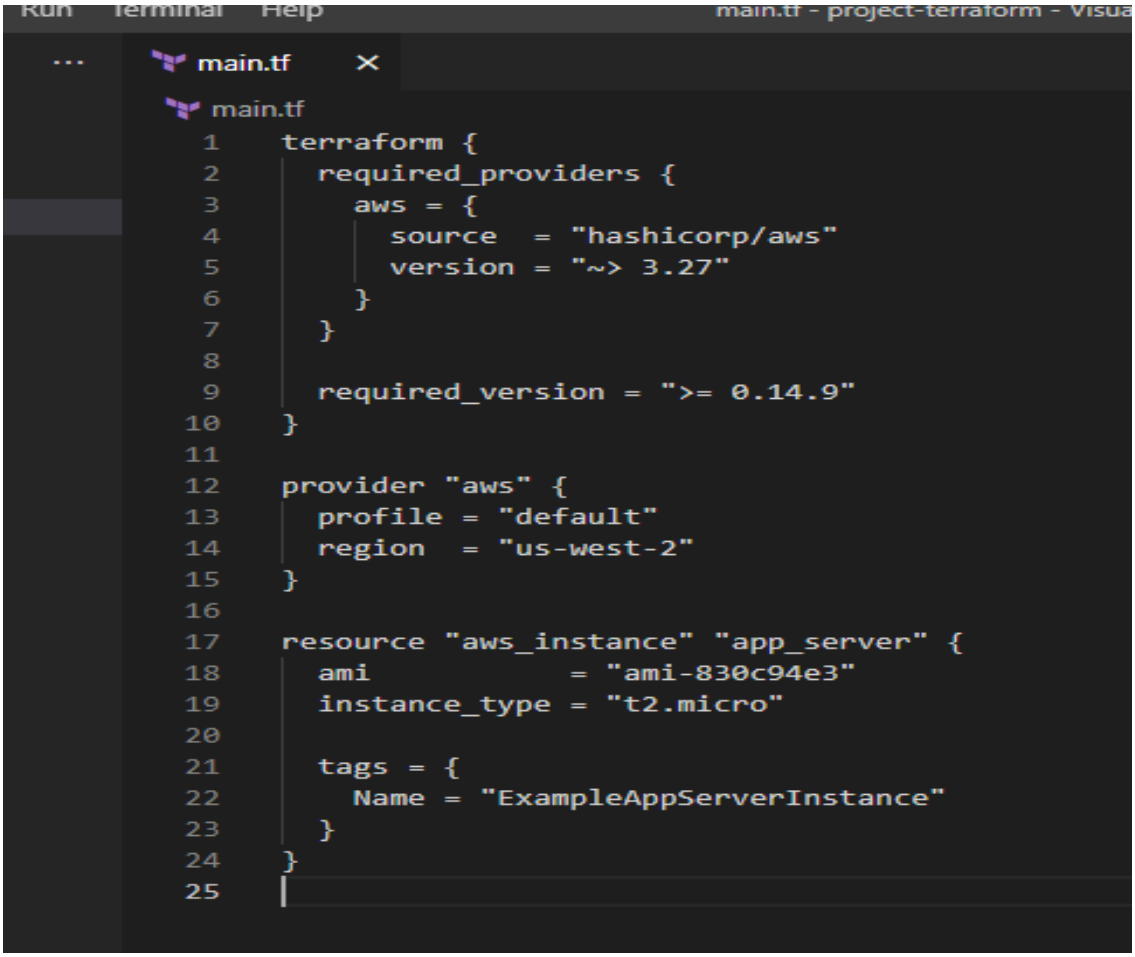




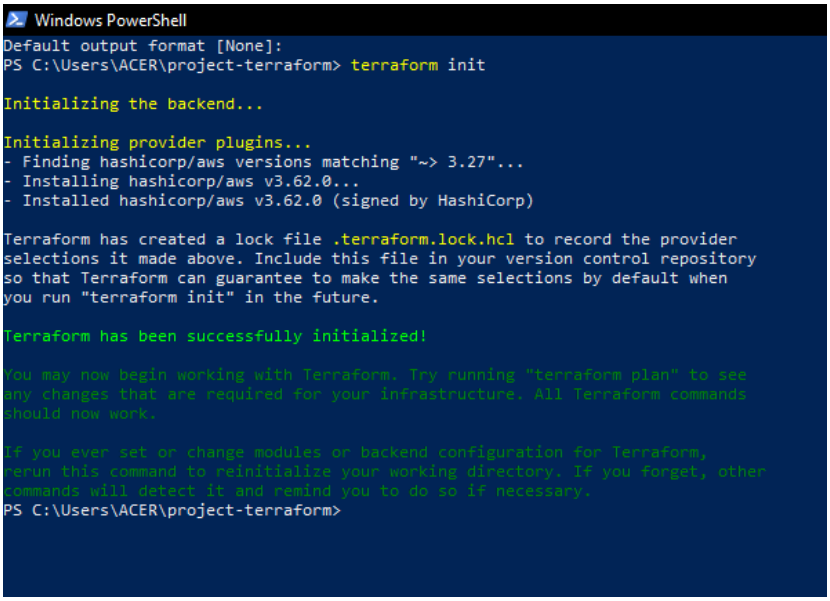
Step 2: Create a new Terraform project



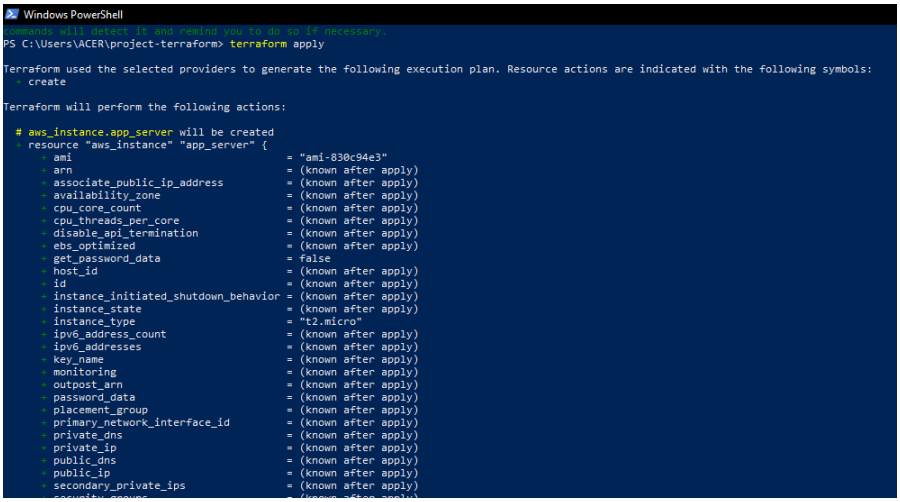
Step 3: Create a new .tf file and enter the following code.



Step 4: Initialize Terraform.



Step 5: Apply the setting



```
          + kms_key_id            = (known after apply)
          + tags                  = (known after apply)
          + throughput            = (known after apply)
          + volume_id             = (known after apply)
          + volume_size           = (known after apply)
          + volume_type           = (known after apply)
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.app_server: Creating...
aws_instance.app_server: Still creating... [10s elapsed]
aws_instance.app_server: Still creating... [20s elapsed]
aws_instance.app_server: Still creating... [30s elapsed]
aws_instance.app_server: Still creating... [40s elapsed]
aws_instance.app_server: Still creating... [50s elapsed]
aws_instance.app_server: Creation complete after 57s [id=i-0324ff4ec22dbe076]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\ACER\project-terraform>
```

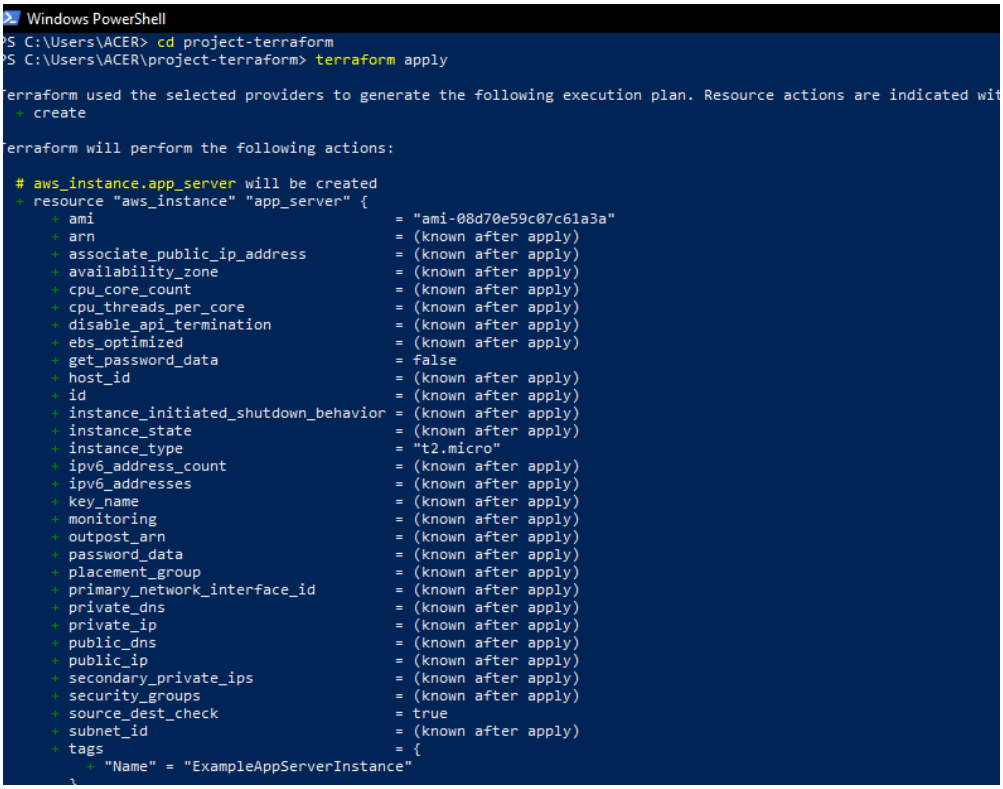
AWS Instance is successfully created using Terraform

2. Change Infrastructure.

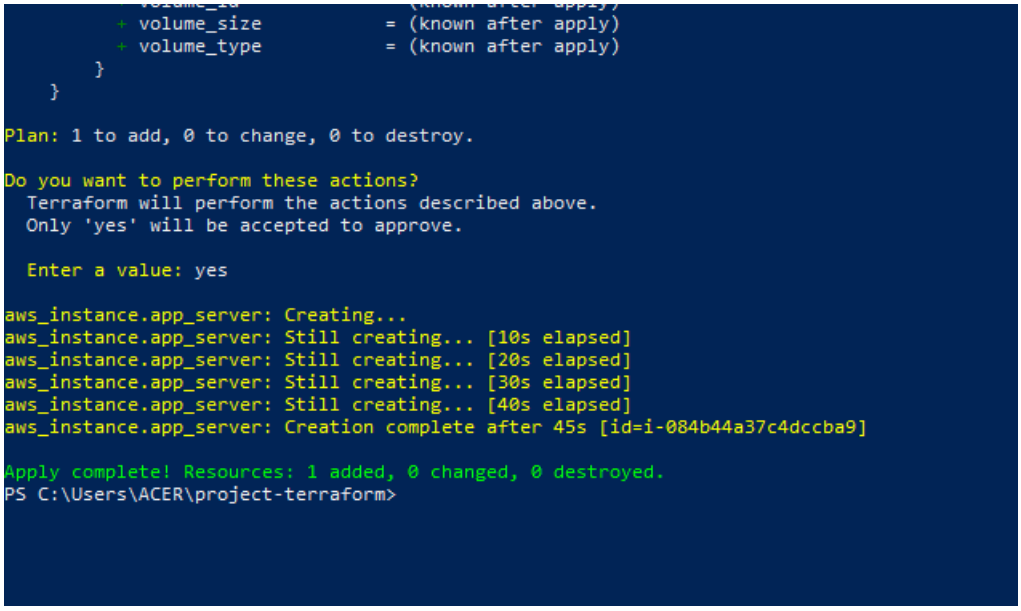
Step 1: Change the AMI in your .tf file you created before as shown below.

```
resource "aws_instance" "app_server" {
  ami           = "ami-08d70e59c07c61a3a"
  instance_type = "t2.micro"

  tags = {
    Name = "ExampleAppServerInstance"
  }
}
```

Step 2: Apply the Settings.



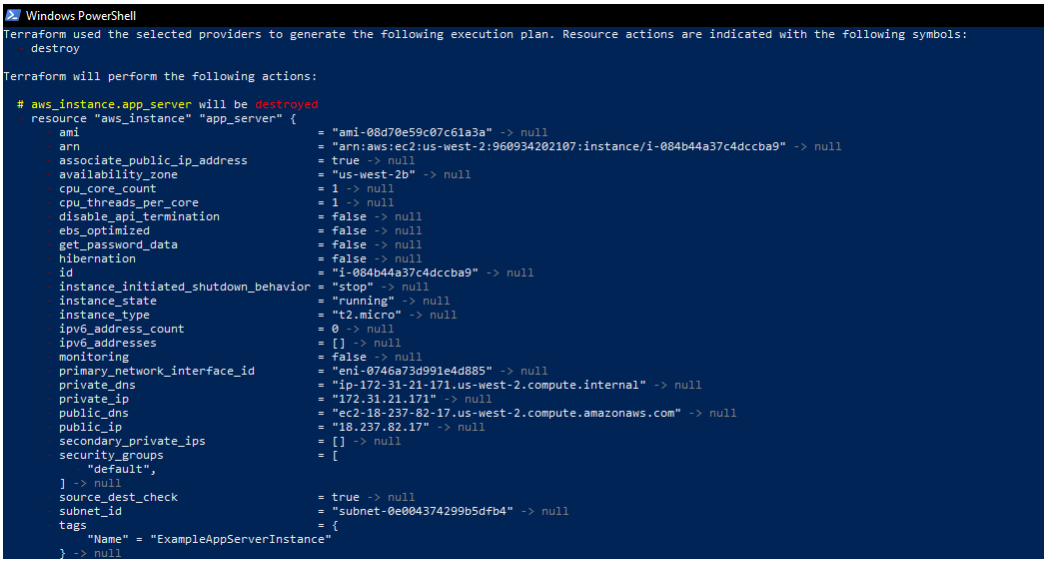
Step 3: Check Result



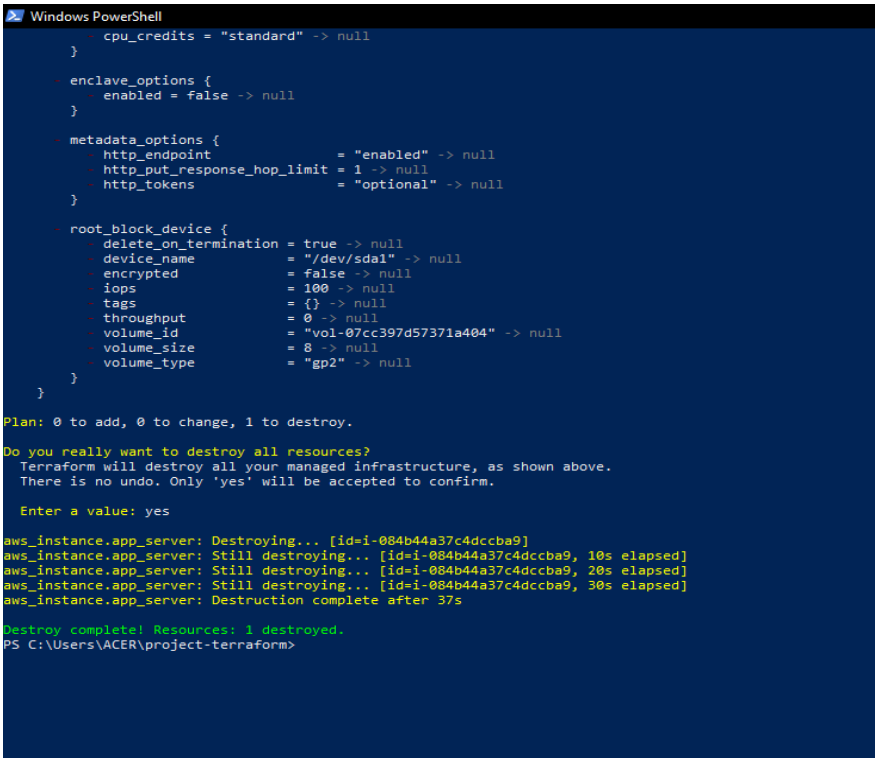
AWS Instance is successfully updated using Terraform

3. Delete the Instance.

Step 1: Type Destroy command



Step 2: Confirm the destruction



Result: The AWS Instance using Terraform was successfully created, changed and destroyed.

Conclusion: The Experiment helped me to understand the use and importance of terraform. The modules of Experiments were successfully performed.

Experiment No : 7

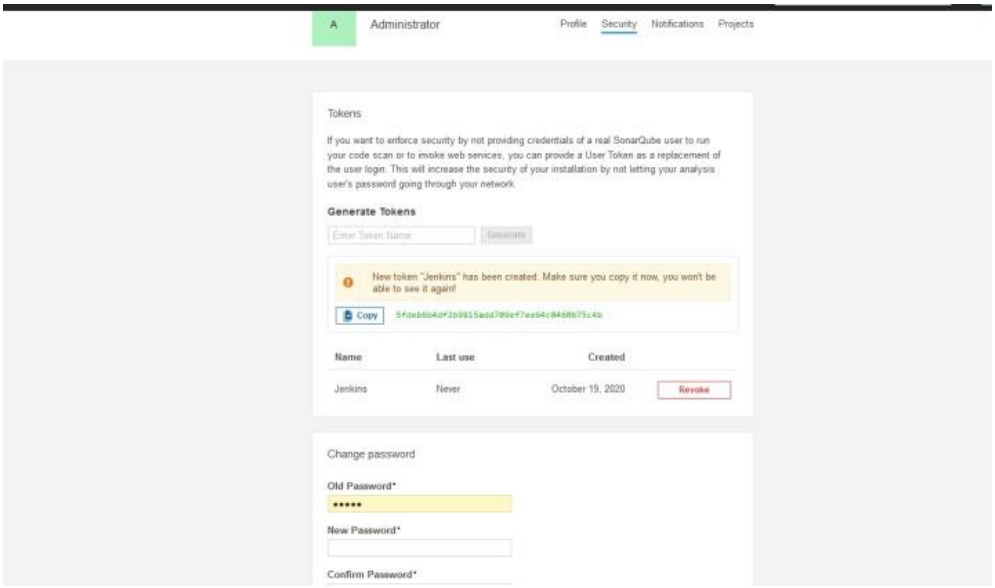
Title: To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab.

Solution:

Step 1: SonarQube Instance (Type the shown command)

$ docker run -d -p 9000:9000 sonarqube

Step 2: Generate User Token



Step 3: Add necessary plugin

Step 4: Configuring SonarQube in Codebase (Type the shown command)

sonar.projectKey=movie-crud-flask

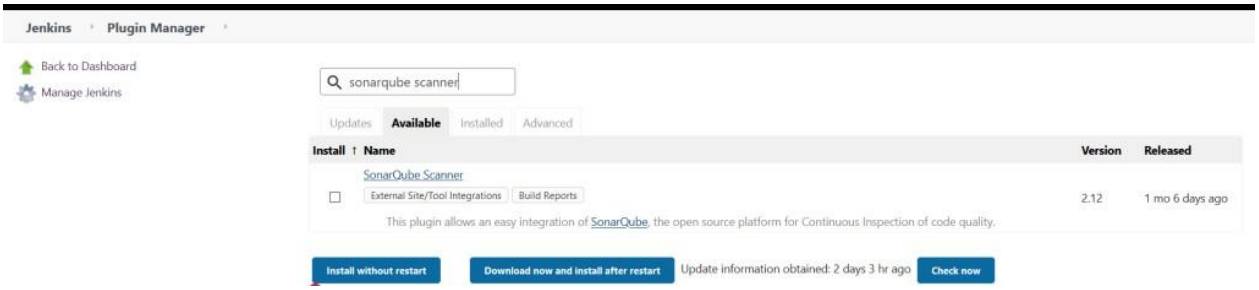
sonar.projectName=movie-crud-flask

sonar.projectVersion=1.0

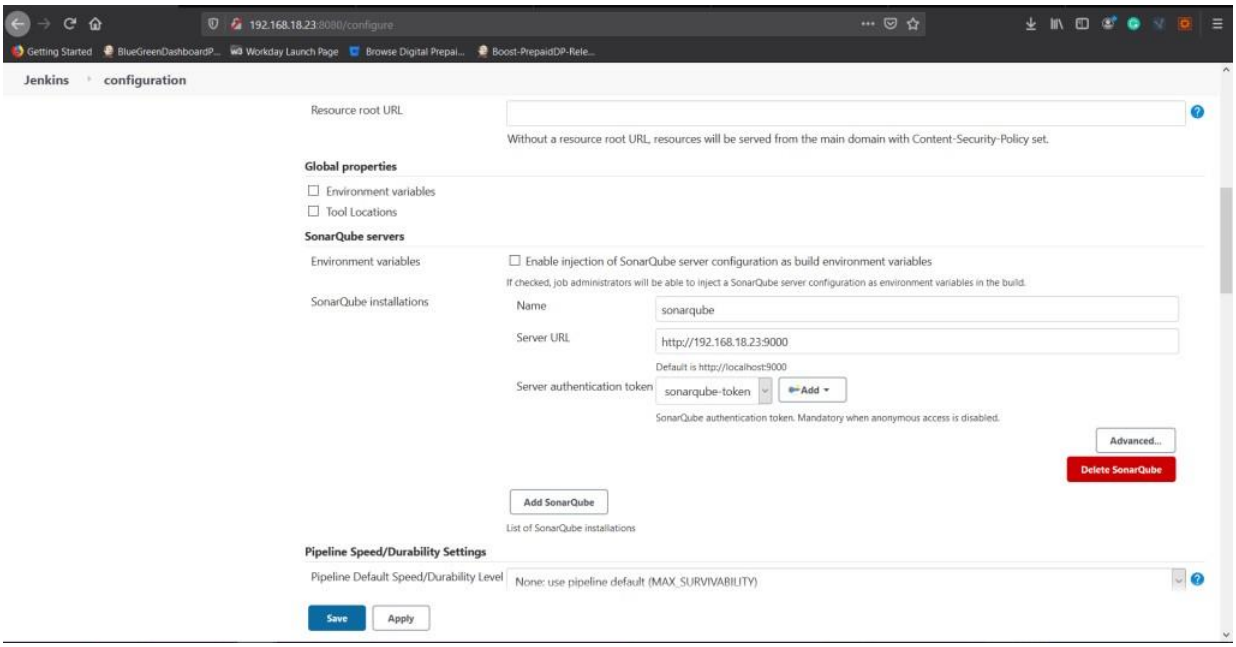
sonar.projectBaseDir=.

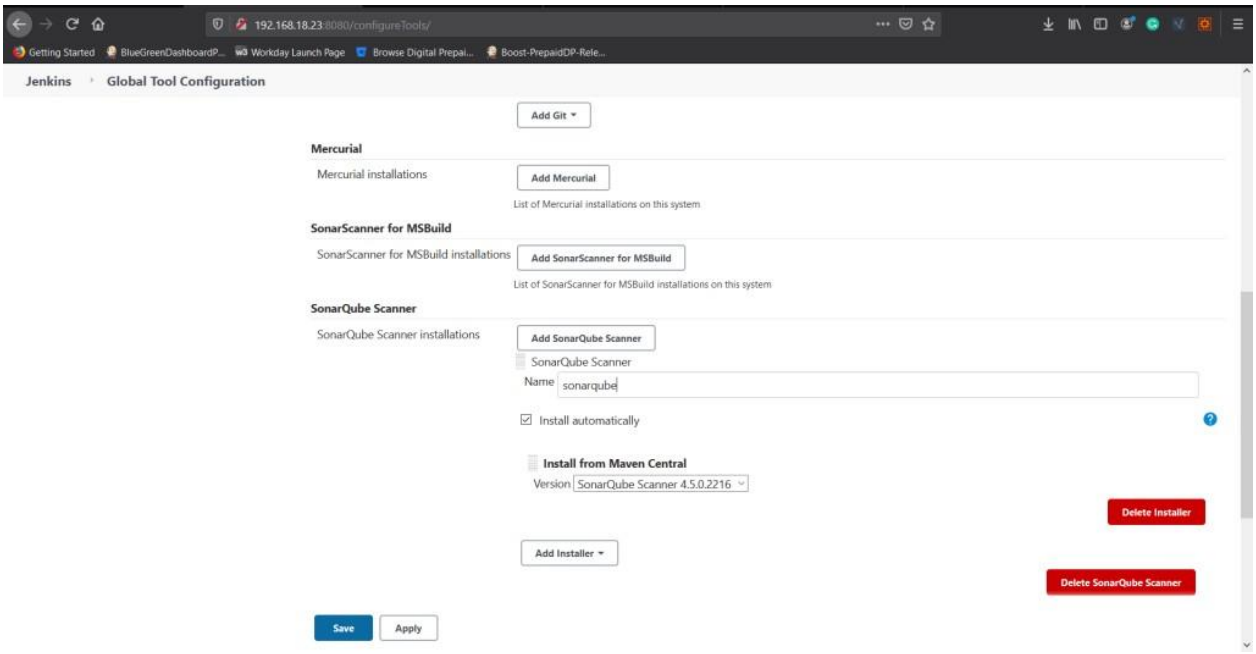
sonar.python.bandit.reportPaths=/report/banditResult.json

Step 5: Install SonarQube plugin in Jenkins.



Step 6: Tool Configuration SonarQube Scanner



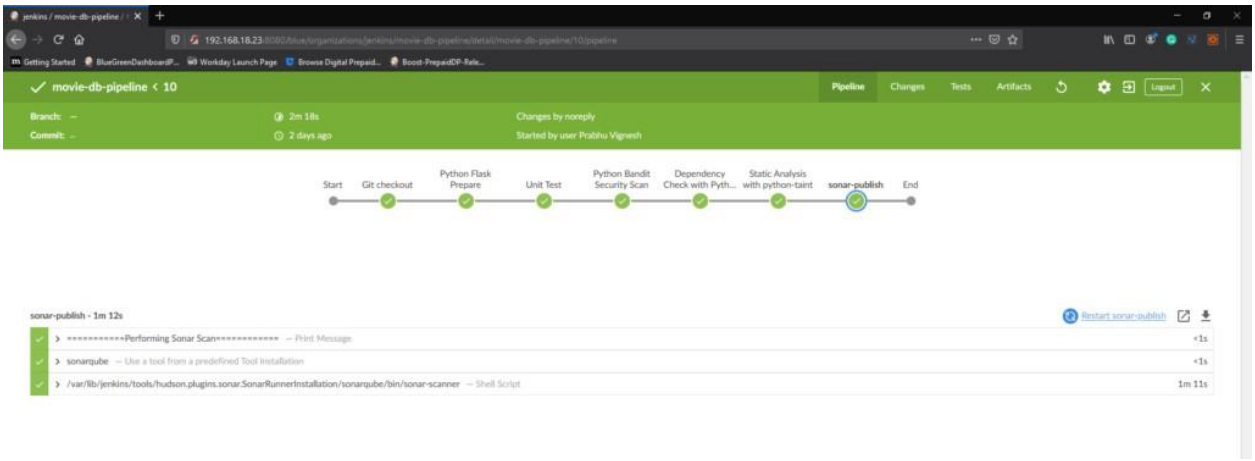


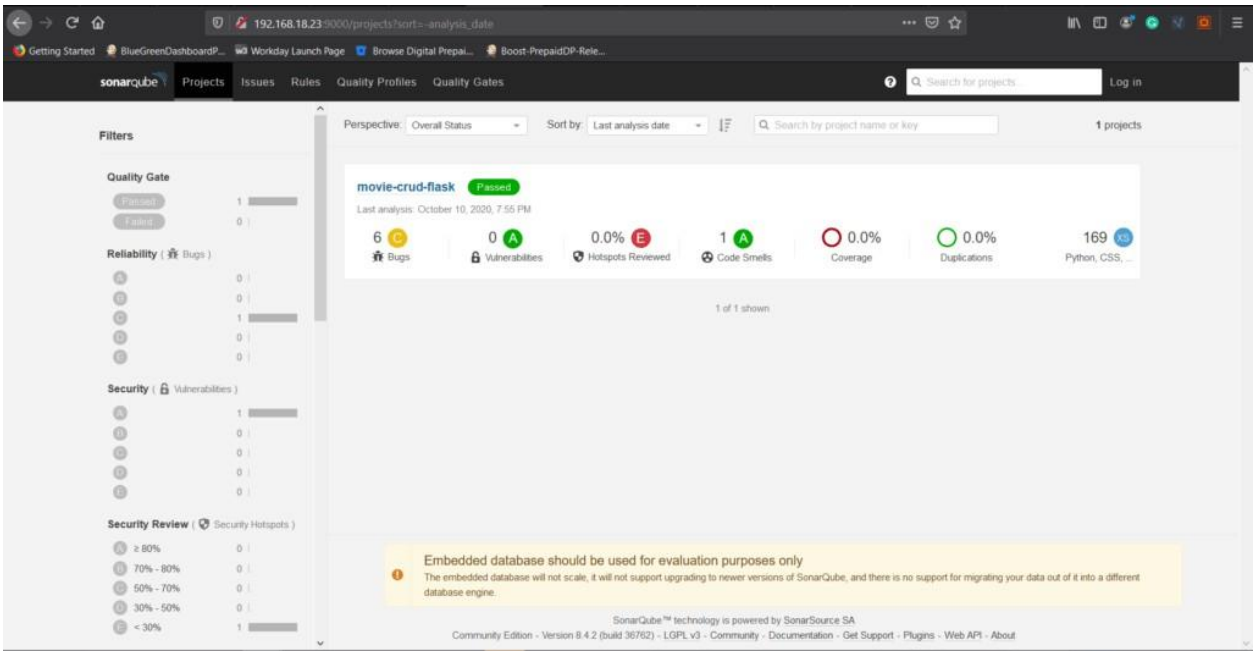
Step 7: SonarQube Scanner in Jenkins Pipeline

Add following code.

```
stage ("sonar-publish"){
        steps {
                echo "============Performing Sonar Scan============"
                sh "${tool("sonarqube")}/bin/sonar-scanner"
        }
}
```

Step 8: Execute the code.





Result: The Experiment is successfully performed and the integration of Jenkins SAST to SonarQube/GitLab was done successfully.

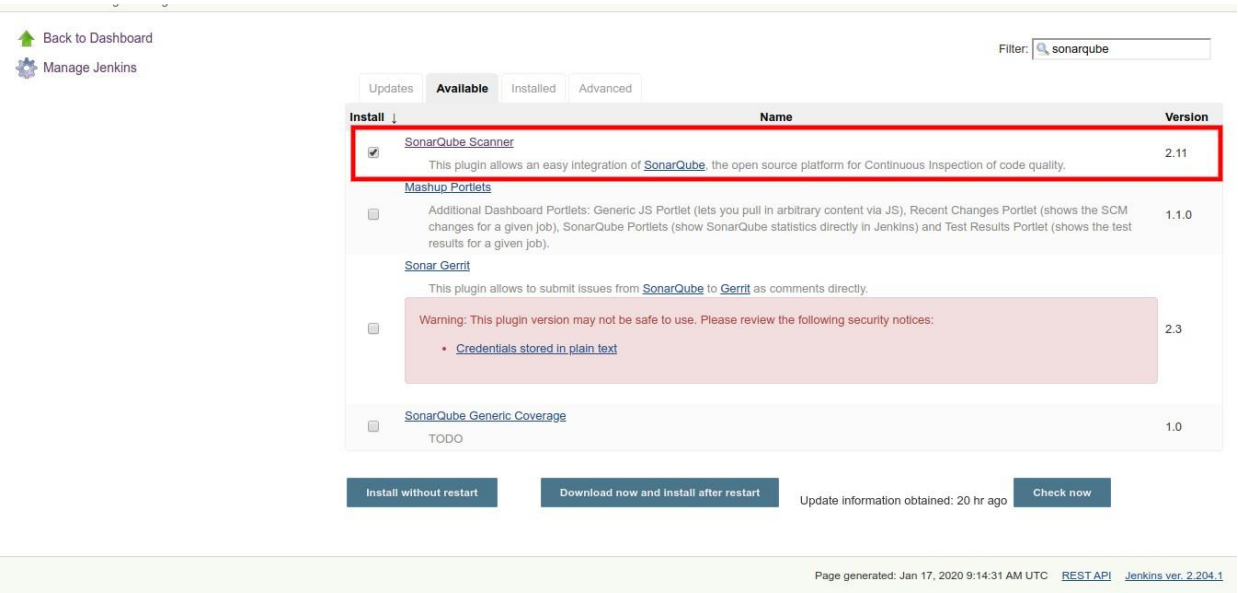
Experiment No: 8

Title: Create a Jenkins CICD Pipeline with SonarQube / GitLab Integration to perform a static Analysis of the code to detect bugs, code smells, and security vulnerabilities on a sample Web / Java / Python application.

Solution:

Step 1: Integrating SonarQube and GitLabs in Jenkins

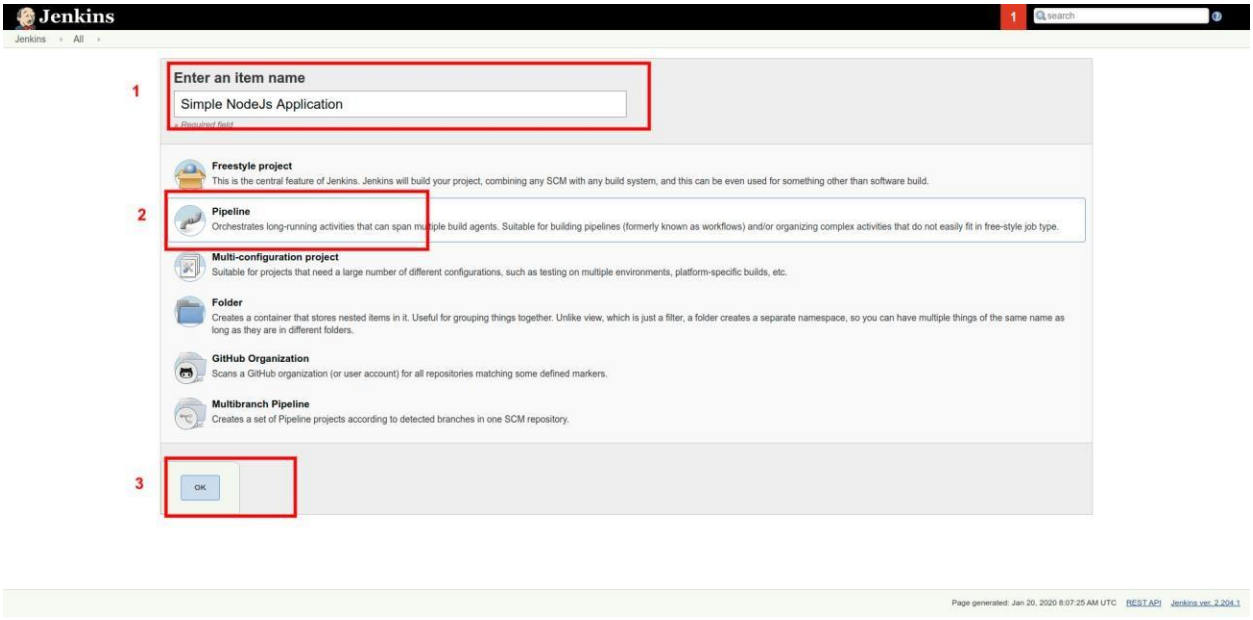
to integrate SonarQube. Go Dashboard > Manage Jenkins > Manage Plugins and search for SonarQube



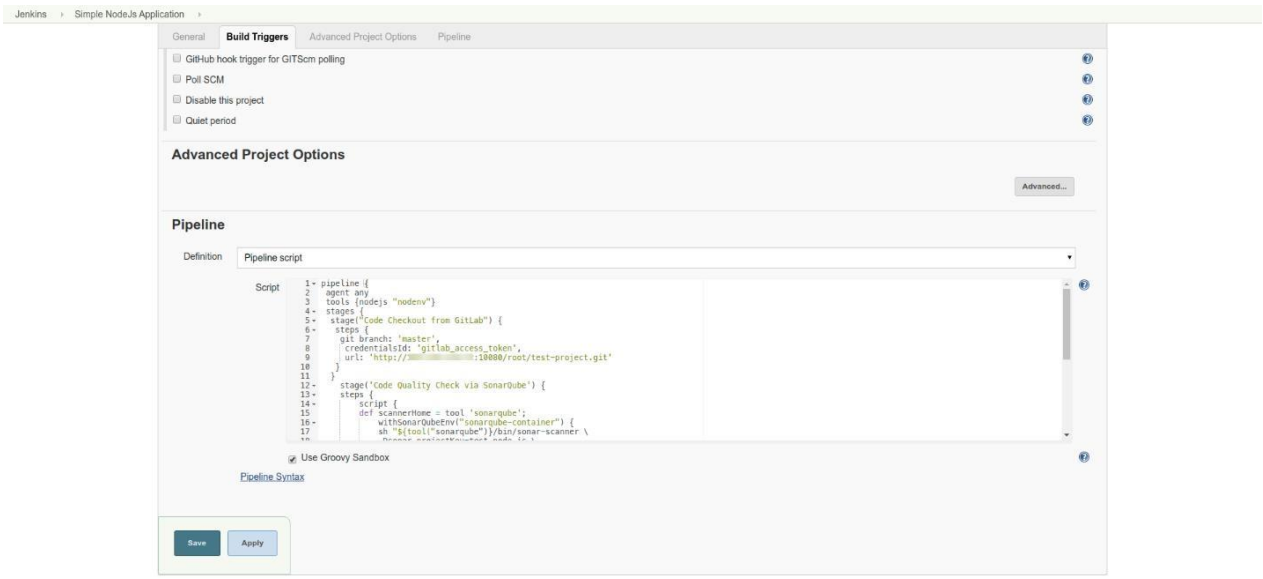
Make sure you restart Jenkins once the plugin is successfully installed.

Step 2: Create a New Job

Go to Jenkins' Dashboard and click on the "New Item" link. Now, enter the item name and select *Pipeline* option as shown in the figure:

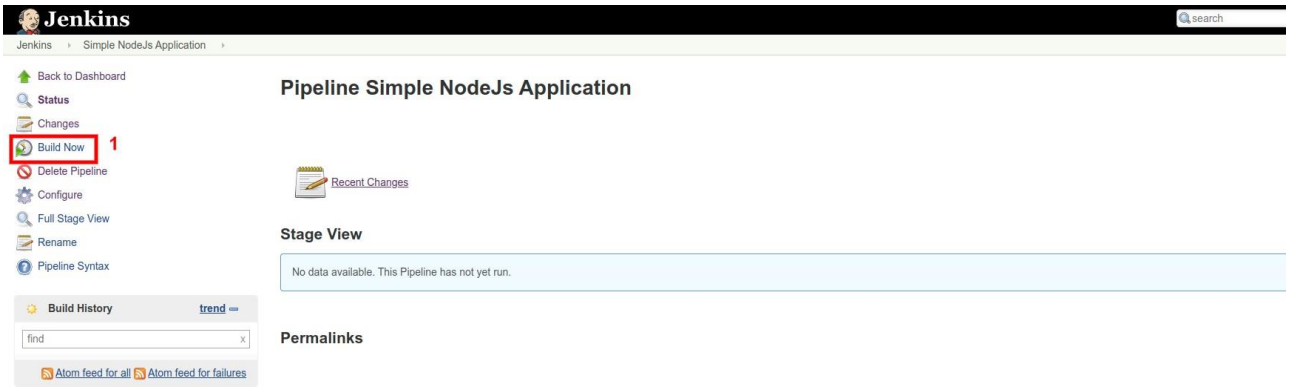


Step 3: Pipeline



Step 4 - Run Pipeline

Now, as you have saved the pipeline script, it's time to build your application in Jenkins. Go to Dashboard > YOUR PROJECT > Build Now.



If your build runs successfully, you will be able to see the time taken by each stage, in Stage View

Result: The Creation of a Jenkins CICD Pipeline with SonarQube Integration was successfully created.

Conclusion: The Experiment was successfully performed.

Experiment No: 9

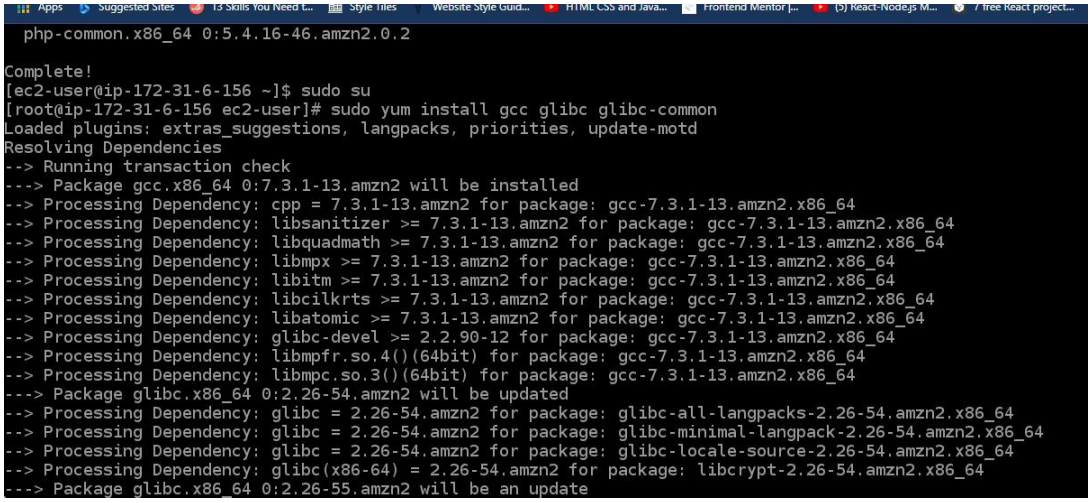
Aim: To Understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor) on Linux Machine.

Solution:

Step by Step method for installing Nagios in Amazon Linux

Step 1: Install Prerequisite Software
Nagios requires the following packages are installed on your server prior to installing Nagios:

Apache
PHP
GCC compiler
GD development libraries



```
  libitm.x86_64 0:7.3.1-13.amzn2                    libmpc.x86_64 0:1.0.1-3.amzn2.0.2
  libmpx.x86_64 0:7.3.1-13.amzn2                    libquadmath.x86_64 0:7.3.1-13.amzn2
  libsanitizer.x86_64 0:7.3.1-13.amzn2              mpfr.x86_64 0:3.1.1-4.amzn2.0.2

Updated:
  glibc.x86_64 0:2.26-55.amzn2                        glibc-common.x86_64 0:2.26-55.amzn2

Dependency Updated:
  glibc-all-langpacks.x86_64 0:2.26-55.amzn2              glibc-locale-source.x86_64 0:2.26-55.amzn2
  glibc-minimal-langpack.x86_64 0:2.26-55.amzn2           libcrypt.x86_64 0:2.26-55.amzn2

Complete!
[root@ip-172-31-6-156 ec2-user]# sudo yum install gd gd-devel
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package gd.x86_64 0:2.0.35-27.amzn2 will be installed
--> Processing Dependency: libfontconfig.so.1()(64bit) for package: gd-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libXpm.so.4()(64bit) for package: gd-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libX11.so.6()(64bit) for package: gd-2.0.35-27.amzn2.x86_64
---> Package gd-devel.x86_64 0:2.0.35-27.amzn2 will be installed
--> Processing Dependency: zlib-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libpng-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libjpeg-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
--> Processing Dependency: libXpm-devel for package: gd-devel-2.0.35-27.amzn2.x86_64
```

Step 2: Create Account Information
You need to set up a Nagios user. Run the following commands:

- sudo adduser -m nagios
- sudo passwd nagios

Type the new password twice.

```
  expat-devel.x86_64 0:2.1.0-12.amzn2                fontconfig.x86_64 0:2.13.0-4.3.amzn2
  fontconfig-devel.x86_64 0:2.13.0-4.3.amzn2         fontpackages-filesystem.noarch 0:1.44-8.amzn2
  freetype-devel.x86_64 0:2.8-14.amzn2.1             libICE.x86_64 0:1.0.9-9.amzn2.0.2
  libSM.x86_64 0:1.2.2-2.amzn2.0.2                   libX11.x86_64 0:1.6.7-3.amzn2.0.2
  libX11-common.noarch 0:1.6.7-3.amzn2.0.2           libX11-devel.x86_64 0:1.6.7-3.amzn2.0.2
  libXau.x86_64 0:1.0.8-2.1.amzn2.0.2                libXau-devel.x86_64 0:1.0.8-2.1.amzn2.0.2
  libXext.x86_64 0:1.3.3-3.amzn2.0.2                 libXpm.x86_64 0:3.5.12-1.amzn2.0.2
  libXpm-devel.x86_64 0:3.5.12-1.amzn2.0.2           libXt.x86_64 0:1.1.5-3.amzn2.0.2
  libjpeg-turbo-devel.x86_64 0:2.0.90-2.amzn2.0.1    libpng-devel.x86_64 2:1.5.13-8.amzn2
  libuuid-devel.x86_64 0:2.30.2-2.amzn2.0.5          libxcb.x86_64 0:1.12-1.amzn2.0.2
  libxcb-devel.x86_64 0:1.12-1.amzn2.0.2             xorg-x11-proto-devel.noarch 0:2018.4-1.amzn2.0.2
  zlib-devel.x86_64 0:1.2.7-18.amzn2

Complete!
[root@ip-172-31-6-156 ec2-user]# sudo adduser -m nagios
[root@ip-172-31-6-156 ec2-user]# sudo passwd nagios
Changing password for user nagios.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-6-156 ec2-user]#
```

- sudo groupadd nagcmd
- sudo usermod -a -G nagcmd nagios
- sudo usermod -a -G nagcmd apache

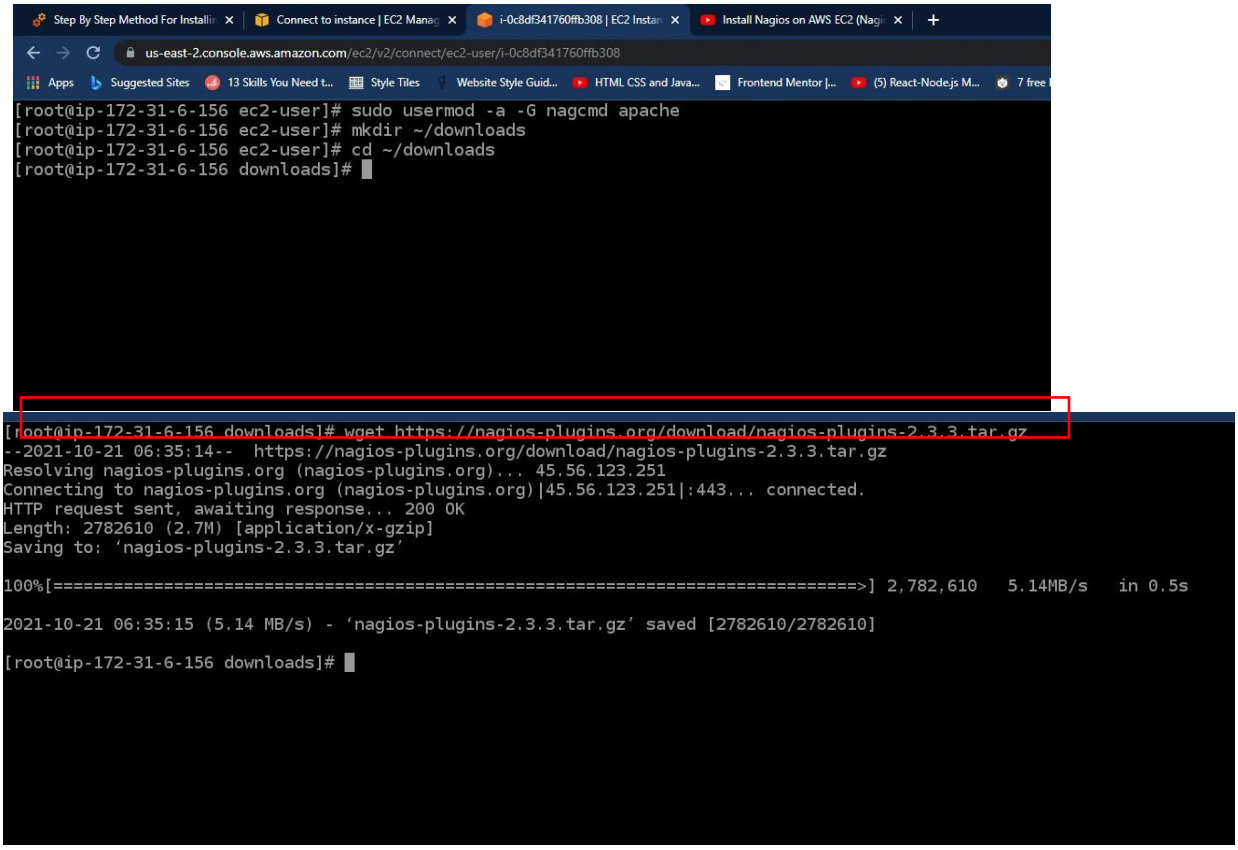
Step 3: Download Nagios Core and the Plugins
Create a directory for storing the downloads.

mkdir ~/downloads

cd ~/downloads

wget http://prdownloads.sourceforge.net/sourceforge/nagios/nagios-4.0.8.tar.gz

wget http://nagios-plugins.org/download/nagios-plugins-2.0.3.tar.gz



Step 4: Compile and Install Nagios

- Extract the Nagios source code tarball.

➢ tar zxvf nagios-4.0.8.tar.gz

➢ cd nagios-4.0.8

```
Saving to: 'nagios-plugins-2.0.3.tar.gz'

100%[======================================================================================>] 2,659,772   7.96MB/s   in 0.3s

2021-10-21 06:36:22 (7.96 MB/s) - 'nagios-plugins-2.0.3.tar.gz' saved [2659772/2659772]

[root@ip-172-31-6-156 downloads]# tar zxvf nagios-4.0.8.tar.gz
nagios-4.0.8/
nagios-4.0.8/.gitignore
nagios-4.0.8/Changelog
nagios-4.0.8/INSTALLING
nagios-4.0.8/LEGAL
nagios-4.0.8/LICENSE
nagios-4.0.8/Makefile.in
nagios-4.0.8/README
nagios-4.0.8/README.asciidoc
nagios-4.0.8/THANKS
nagios-4.0.8/UPGRADING
nagios-4.0.8/base/
nagios-4.0.8/base/.gitignore
nagios-4.0.8/base/Makefile.in
nagios-4.0.8/base/broker.c
nagios-4.0.8/base/checks.c
nagios-4.0.8/base/commands.c
nagios-4.0.8/base/config.c
```

```
nagios-4.0.8/worker/Makefile.in
nagios-4.0.8/worker/ping/
nagios-4.0.8/worker/ping/.gitignore
nagios-4.0.8/worker/ping/Makefile.in
nagios-4.0.8/worker/ping/worker-ping.c
nagios-4.0.8/xdata/
nagios-4.0.8/xdata/.gitignore
nagios-4.0.8/xdata/Makefile.in
nagios-4.0.8/xdata/xcddefault.c
nagios-4.0.8/xdata/xcddefault.h
nagios-4.0.8/xdata/xodtemplate.c
nagios-4.0.8/xdata/xodtemplate.h
nagios-4.0.8/xdata/xpddefault.c
nagios-4.0.8/xdata/xpddefault.h
nagios-4.0.8/xdata/xrddefault.c
nagios-4.0.8/xdata/xrddefault.h
nagios-4.0.8/xdata/xsddefault.c
nagios-4.0.8/xdata/xsddefault.h
[root@ip-172-31-6-156 downloads]# cd nagios-4.0.8
[root@ip-172-31-6-156 nagios-4.0.8]#
```

- Run the configuration script with the name of the group which you have created in the above step.

➢ ./configure --with-command-group=nagcmd

```
nagios-4.0.8/tap/tests/todo/Makefile.in
nagios-4.0.8/tap/tests/todo/test.c
nagios-4.0.8/tap/tests/todo/test.pl
nagios-4.0.8/tap/tests/todo/test.t
[root@ip-172-31-6-156 nagios-4.0.8]# ./configure --with-command-group=nagcmd
checking for a BSD-compatible install... /bin/install -c
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking whether make sets $(MAKE)... yes
checking for strip... /bin/strip
```

- Compile the Nagios source code.

➢ make all

```
Web Interface Options:
------------------------
               HTML URL:  http://localhost/nagios/
                CGI URL:  http://localhost/nagios/cgi-bin/
Traceroute (used by WAP):  /bin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.

[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# make all
cd ./base && make
make[1]: Entering directory `/root/downloads/nagios-4.0.8/base'
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o nagios.o nagios.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o broker.o broker.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nebmods.o nebmods.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o ../common/shared.o ../common/shared.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o nerd.o nerd.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o query-handler.o query-handler.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE -c -o workers.o workers.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o checks.o checks.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o config.o config.c
gcc -Wall -I.. -g -O2 -DHAVE_CONFIG_H -DNSCORE   -c -o commands.o commands.c
```

- Install binaries, init script, sample config files and set permissions on the external command directory.

- sudo make install
- sudo make install-init
- sudo make install-config
- sudo make install-commandmode

```
       web interface


*** Support Notes *******************************************

If you have questions about configuring or running Nagios,
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install
cd ./base && make install
make[1]: Entering directory `/root/downloads/nagios-4.0.8/base'
make install-basic
make[2]: Entering directory `/root/downloads/nagios-4.0.8/base'
/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/bin
/bin/install -c -m 774 -o nagios -g nagios nagios /usr/local/nagios/bin
/bin/install -c -m 774 -o nagios -g nagios nagiostats /usr/local/nagios/bin
make[2]: Leaving directory `/root/downloads/nagios-4.0.8/base'
make strip-post-install
make[2]: Entering directory `/root/downloads/nagios-4.0.8/base'
/bin/strip /usr/local/nagios/bin/nagios
/bin/strip /usr/local/nagios/bin/nagiostats
make[2]: Leaving directory `/root/downloads/nagios-4.0.8/base'
make[1]: Leaving directory `/root/downloads/nagios-4.0.8/base'
cd ./cgi && make install
```

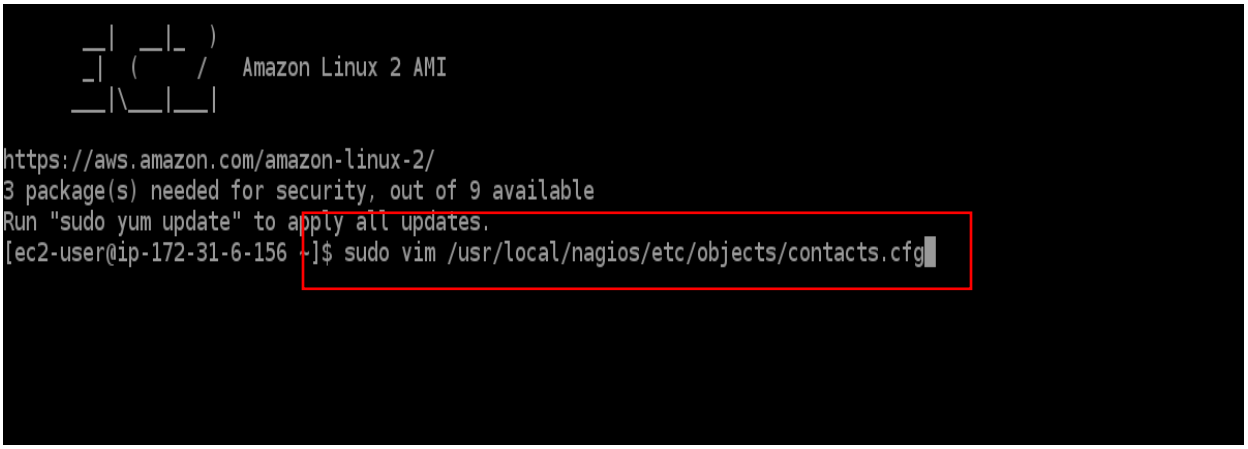
```
cp -rf contrib/exfoliation/images/* /usr/local/nagios/share/images

*** Exfoliation theme installed ***
NOTE: Use 'make install-classicui' to revert to classic Nagios theme

make[1]: Leaving directory `/root/downloads/nagios-4.0.8'
make install-basic
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-init
/bin/install -c -m 755 -d -o root -g root /etc/rc.d/init.d
/bin/install -c -m 755 -o root -g root daemon-init /etc/rc.d/init.d/nagios

*** Init script installed ***

[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-config
/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/objects/te
mplates.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objects/com
mands.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objects/con
tacts.cfg
```

```
calhost.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/objects/wind
ows.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/objects/prin
ter.cfg
/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects/switc
h.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-commandmode
/bin/install -c -m 775 -o nagios -g nagcmd -d /usr/local/nagios/var/rw
chmod g+s /usr/local/nagios/var/rw

*** External command directory configured ***

[root@ip-172-31-6-156 nagios-4.0.8]#
```

Step 5: Customize Configuration

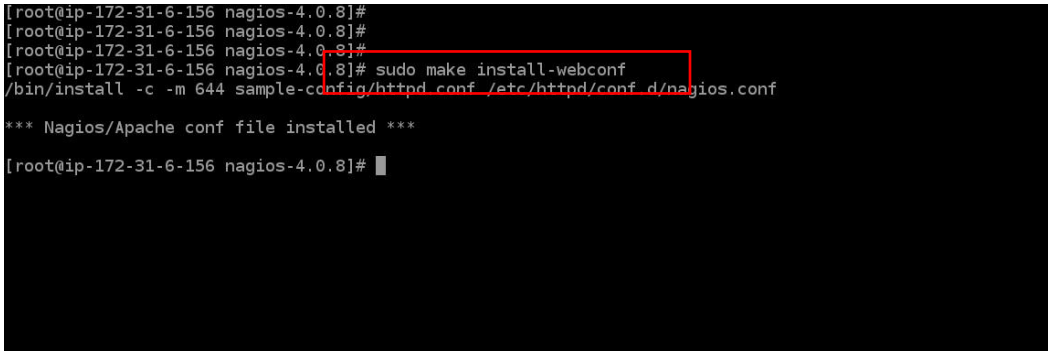
Change E-Mail address with nagiosadmin contact definition you'd like to use for receiving Nagios alerts.

- sudo vim /usr/local/nagios/etc/objects/contacts.cfg



Step 6: Configure the Web Interface

- sudo make install-webconf



Create a nagiosadmin account for logging into the Nagios web interface. Note the password you need it while login to Nagios web console.

- sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin // Type Password Twice
- sudo service httpd restart //Restart Service

```
[root@ip-172-31-6-156 nagios-4.0.8]#
[root@ip-172-31-6-156 nagios-4.0.8]# sudo make install-webconf
/bin/install -c -m 644 sample-config/httpd.conf /etc/httpd/conf.d/nagios.conf

*** Nagios/Apache conf file installed ***

[root@ip-172-31-6-156 nagios-4.0.8]# sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[root@ip-172-31-6-156 nagios-4.0.8]# sudo service httpd restart
Redirecting to /bin/systemctl restart         httpd.service
Failed to restart \xc2\xa0\xc2\xa0\xc2\xa0\xc2\xa0\xc2\xa0\xc2\xa0\xc2\xa0\xc2\xa0.service: Unit not found.
[root@ip-172-31-6-156 nagios-4.0.8]# cd ~/downloads
[root@ip-172-31-6-156 downloads]#
```

Step 7: Compile and Install the Nagios Plugins

- Extract the Nagios plugins source code tarball.

➢ cd ~/downloads

➢ tar zxvf nagios-plugins-2.0.3.tar.gz

➢ cd nagios-plugins-2.0.3

- Compile and install the plugins.

➢ ./configure --with-nagios-user=nagios --with-nagios-group=nagios

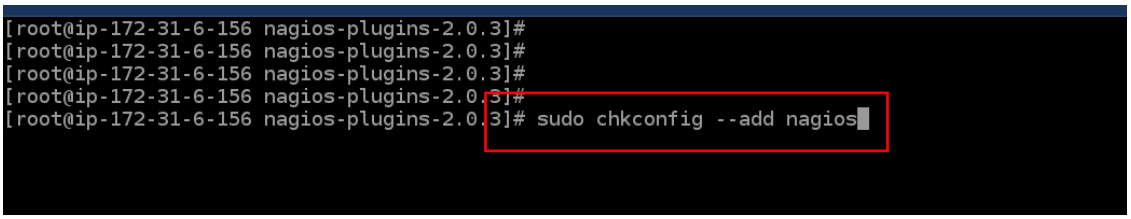
➢ make

➢ sudo make install

➢

```
config.status: creating po/Makefile
            --with-apt-get-command:
              --with-ping6-command: /sbin/ping6 -n -U -w %d -c %d %s
               --with-ping-command: /bin/ping -n -U -w %d -c %d %s
                       --with-ipv6: yes
                      --with-mysql: no
                   --with-openssl: no
                    --with-gnutls: no
               --enable-extra-opts: yes
                      --with-perl: /bin/perl
             --enable-perl-modules: no
                     --with-cgiurl: /nagios/cgi-bin
               --with-trusted-path: /bin:/sbin:/usr/bin:/usr/sbin
                   --enable-libtap: no
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# ./configure --with-nagios-user=nagios --with-nagios-group=nagios
```

```
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#
[root@ip-172-31-6-156 nagios-plugins-2.0.3]#
[root@ip-172-31-6-156 nagios-plugins-2.0.3]# sudo make install
```

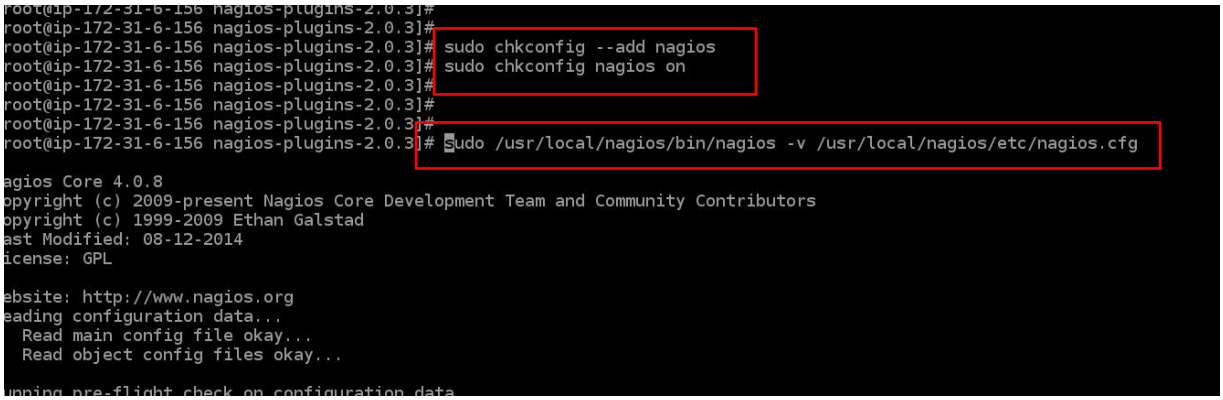


Step 8: Start Nagios
Add Nagios to the list of system services and have it automatically start when the system boots.

- sudo chkconfig --add nagios
- sudo chkconfig nagios on

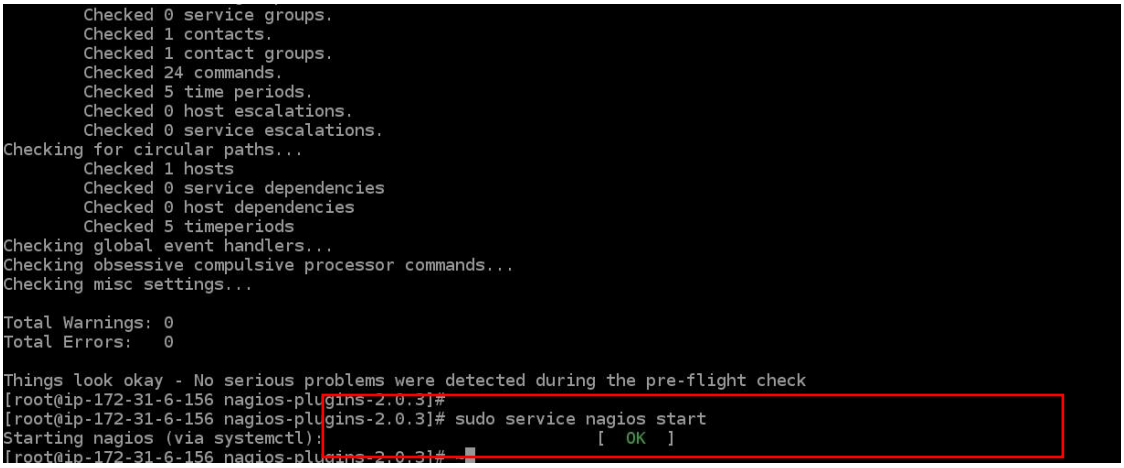
Verify the sample Nagios configuration files.

- sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg



If there are no errors, start Nagios.

- sudo service nagios start



- The Nagios service has been started.

Result: This Experiment helped me to understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE. The Experiment was successfully completed.

Conclusion: The modules of the experiment were successfully executed. The Installation of Nagios core was performed.

Experiment No: 10

Aim: To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Solution:

1. **Adding a remote Linux Host to Nagios Server**

Step1: Install Nagios NRPE

sudo apt install nagios-nrpe-server nagios-plugins

```
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# sudo apt-get install nagios-nrpe-server nagios-plugins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  grub-pc-bin libnuma1
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libarchive13 libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdbi1 libjansson4 libldb1
  libmysqlclient20 libnet-snmp-perl libpq5 libpython-stdlib libpython2.7 libpython2.7-minimal libpython2.7-stdlib
  libsensors4 libsmbclient libsnmp-base libsnmp30 libtalloc2 libtdb1 libtevent0 libtirpc1 libwbclient0
  monitoring-plugins monitoring-plugins-basic monitoring-plugins-common monitoring-plugins-standard mysql-common
  python python-crypto python-ldb python-minimal python-samba python-talloc python-tdb python2.7
  python2.7-minimal rpcbind samba-common samba-common-bin samba-libs smbclient snmp
Suggested packages:
  lrzip cups-common libcrypt-des-perl libdigest-hmac-perl libio-socket-inet6-perl lm-sensors snmp-mibs-downloader
  icinga | icinga2 nagios-plugins-contrib fping postfix | sendmail-bin | exim4-daemon-heavy | exim4-daemon-light
  qstat xinetd | inetd python-doc python-tk python-crypto-doc python-gpgme python2.7-doc binutils binfmt-support
  heimdal-clients cifs-utils
```

Step 2: Configure NRPE by opening its configuration file in /etc/nagios/nrpe.cfg

```
# SERVER ADDRESS
# Address that nrpe should bind to in case there are more than one interface
# and you do not want nrpe to bind on all interfaces.
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

server_address=10.128.0.69
```

Step 3: Add Nagios server IP address in the 'allowed_hosts' attribute,

```
# Note: The daemon only does rudimentary checking of the client's IP
# address.  I would highly recommend adding entries in your /etc/hosts.allow
# file to allow only the specified host to connect to the port
# you are running this daemon on.
#
# NOTE: This option is ignored if NRPE is running under either inetd or xinetd

allowed_hosts=127.0.0.1, 10.128.0.50
```

Save and exit the configuration file.

Step 4: Restart NRPE service and verify its status

- # systemctl restart nagios-nrpe-server

```
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# systemctl restart nagios-nrpe-server
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# systemctl status nagios-nrpe-server
● nagios-nrpe-server.service - Nagios Remote Plugin Executor
   Loaded: loaded (/lib/systemd/system/nagios-nrpe-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-10-31 21:13:42 UTC; 13s ago
     Docs: http://www.nagios.org/documentation
  Process: 6592 ExecStopPost=/bin/rm -f /var/run/nagios/nrpe.pid (code=exited, status=0/SUCCESS)
 Main PID: 6595 (nrpe)
    Tasks: 1 (limit: 4395)
   CGroup: /system.slice/nagios-nrpe-server.service
           └─6595 /usr/sbin/nrpe -c /etc/nagios/nrpe.cfg -f
```

2. **Configure Nagios Server to monitor Linux host**

Step 1: Log in to Nagios Server and install EPEL (Extra packages for Enterprise Linux) package.

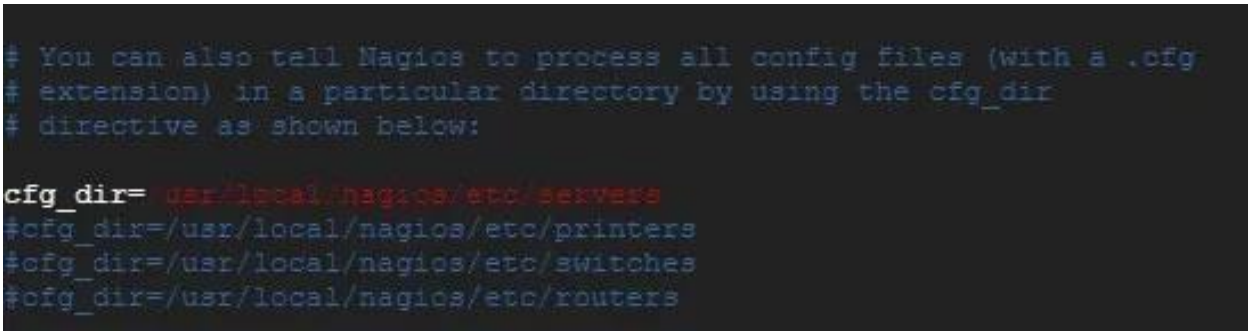
- # dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

Install NRPE plugin on the server

- # dnf install nagios-plugins-nrpe -y

Step 2: Uncomment the line below in the configuration file

- cfg_dir=/usr/local/nagios/etc/servers



Step 3: Create a configuration directory

- # mkdir /usr/local/nagios/etc/servers

Step 4: create client configuration file

- # vim /usr/local/nagios/etc/servers/ubuntu-host.cfg

Copy and paste the configuration below to the file.

define host{

use linux-server

host_name ubuntu-nagios-client

alias ubuntu-nagios-client

address 10.128.0.53

}

```
define hostgroup{
        hostgroup_name          linux-server
        alias                   Linux Servers
        members                 ubuntu-nagios-client
}

define service{
        use                     local-service
        host_name               ubuntu-nagios-client
        service_description     SWAP Usage
        check_command           check_nrpe!check_swap

}

define service{
        use                     local-service
        host_name               ubuntu-nagios-client
        service_description     Root / Partition
        check_command           check_nrpe!check_root
```

```
}
        define service{
                use                     local-service
                host_name               ubuntu-nagios-client
                service_description     Current Users
                check_command           check_nrpe!check_users
        }
        define service{
                use                     local-service
                host_name               ubuntu-nagios-client
                service_description     Total Processes
                check_command           check_nrpe!check_total_procs
        }
        define service{
                use                     local-service
                host_name               ubuntu-nagios-client
                service_description     Current Load
                check_command           check_nrpe!check_load
        }
```

Step 5: Save and exit the configuration file.

Verify that there are no errors in Nagios configuration

- # /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Step 6: Open port 5666 which is used by NRPE plugin on the firewall of the Nagios server.

```
[root@centos-8 servers]#
[root@centos-8 servers]# firewall-cmd --permanent --add-port=5666/tcp
success
[root@centos-8 servers]# firewall-cmd --reload
success
[root@centos-8 servers]#
```

Step 7: Go to your Linux host (Ubuntu 18.04 LTS) and allow the port on UFW firewall.

```
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# ufw allow 5666/tcp
Rule added
Rule added (v6)
root@ubuntu-nagios-client:~# ufw reload
Firewall reloaded
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~#
root@ubuntu-nagios-client:~# ufw status
Status: active

To                         Action      From
--                         ------      ----
5666/tcp                   ALLOW       Anywhere
5666/tcp (v6)              ALLOW       Anywhere (v6)
```

Result: The Setup to perform monitoring of Port, Service monitoring, Windows/Linux server monitoring using Nagios is successfully done.

This experiment helped me to get knowledge of Working of Nagios.

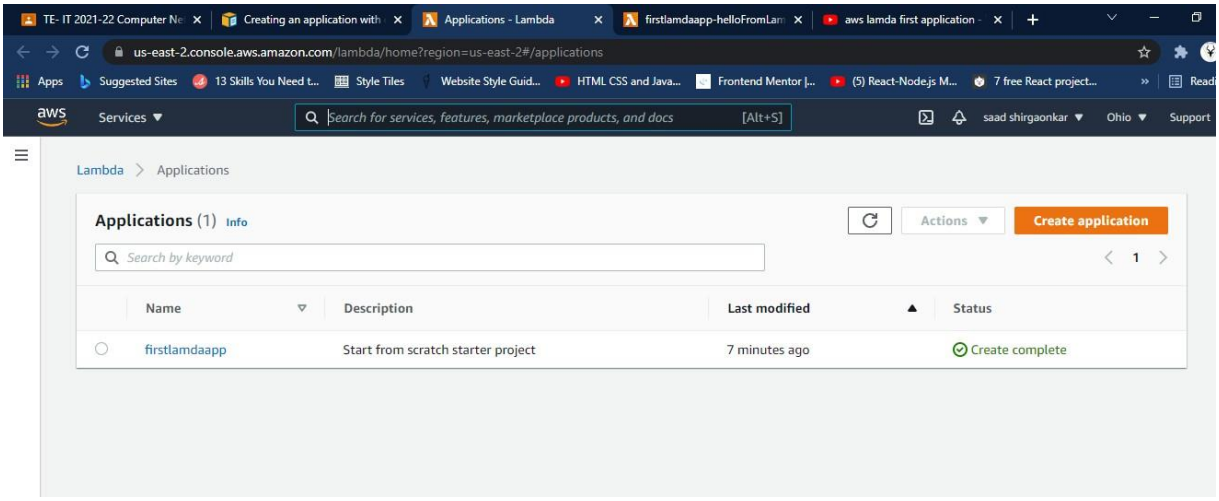
Conclusion: The modules of the experiment were successfully performed. The monitoring setup was successfully done.

Experiment No: 11

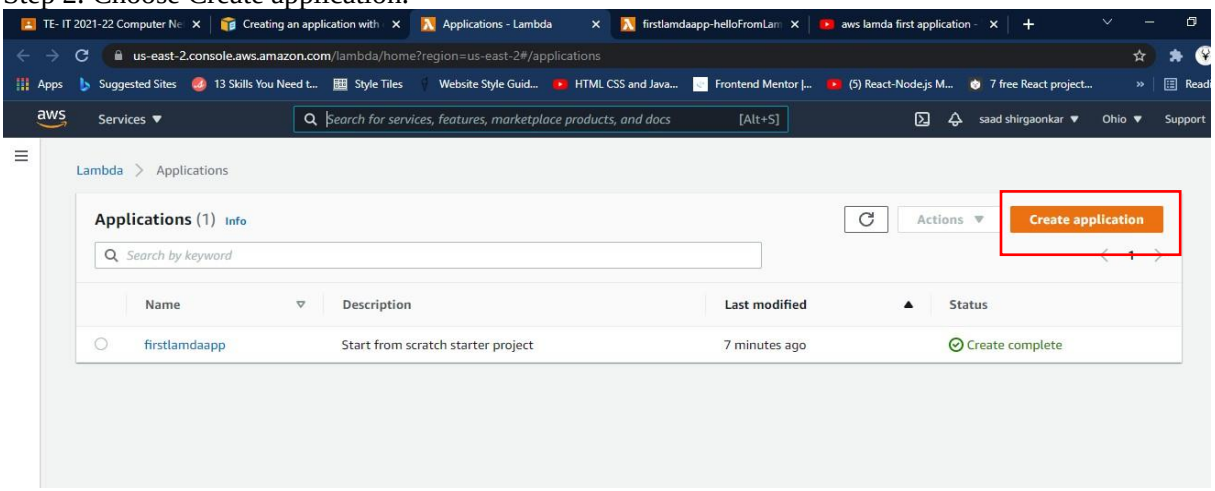
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Solution:

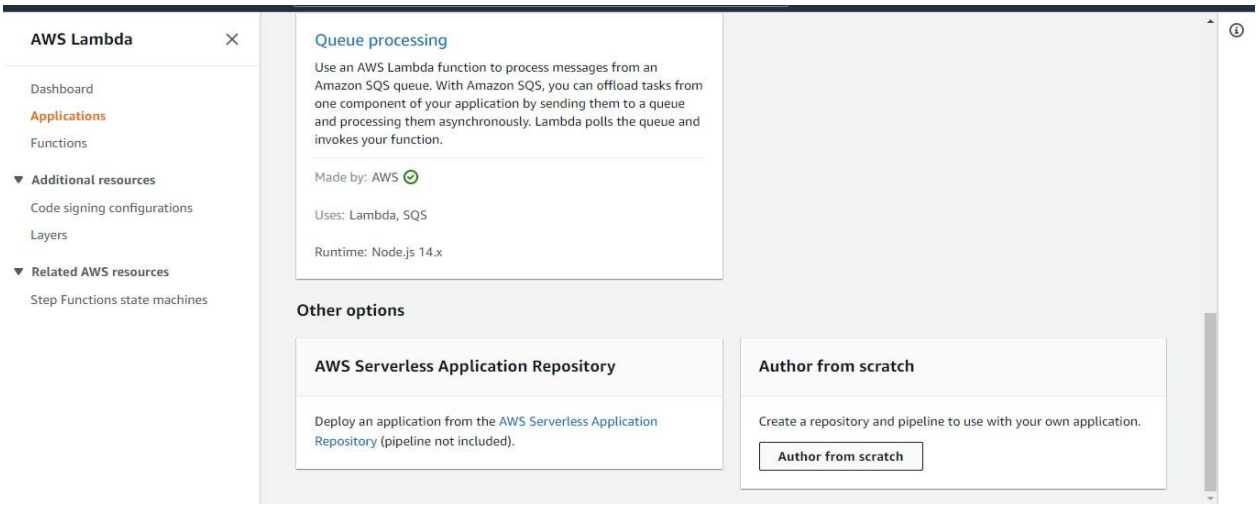
Step 1: Open the Lambda console



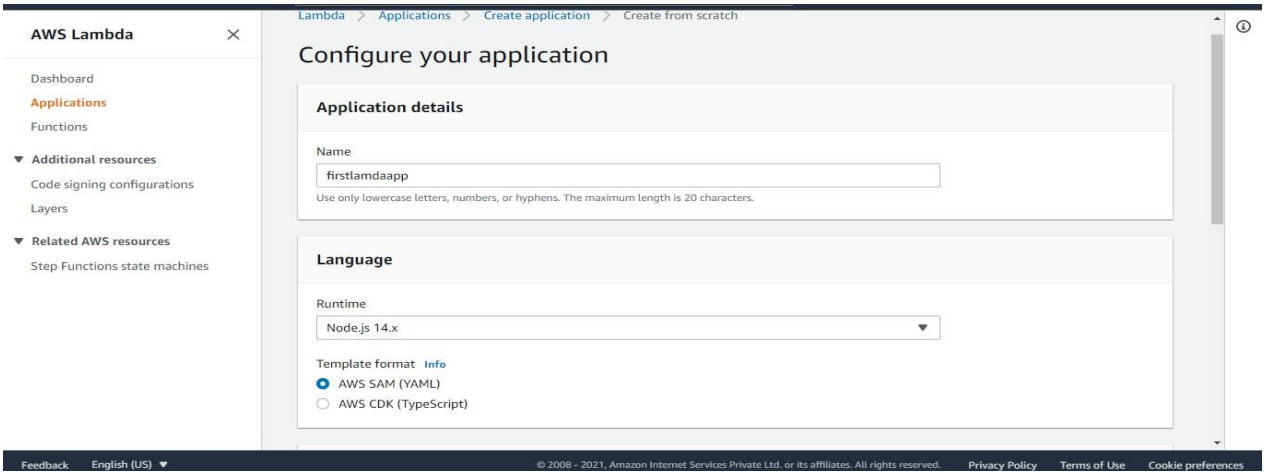
Step 2: Choose Create application.

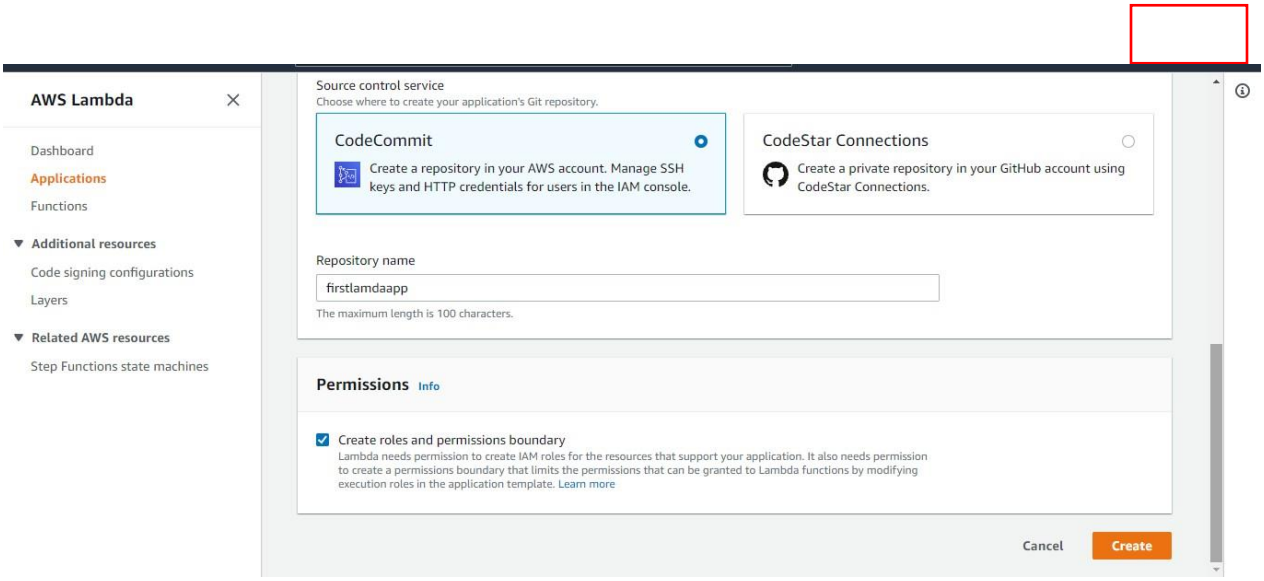


Step 3: Choose **Author from scratch**.

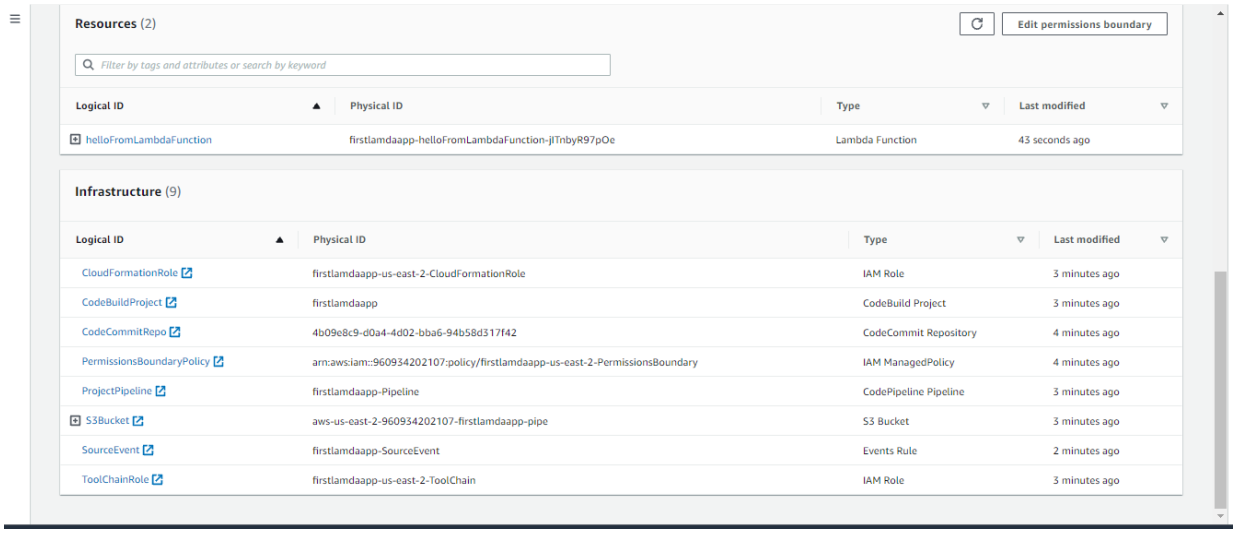


Step 4: Configure application settings and Choose Create..

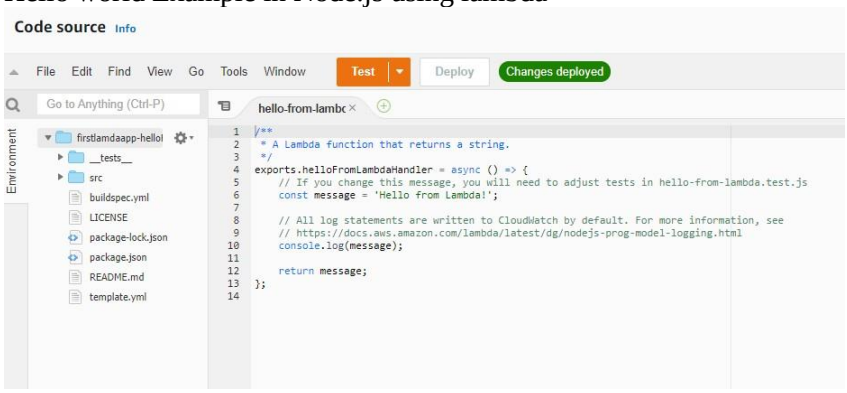




Your Lambda Application is Successfully created



Hello world Example in Node.js using lambda



Result: The Creation of lambda Application is successfully done.

Conclusion: The importance of AWS Lambda, its workflow, various functions were learned in this experiment and the first lambda application is successfully done.

Experiment No: 12

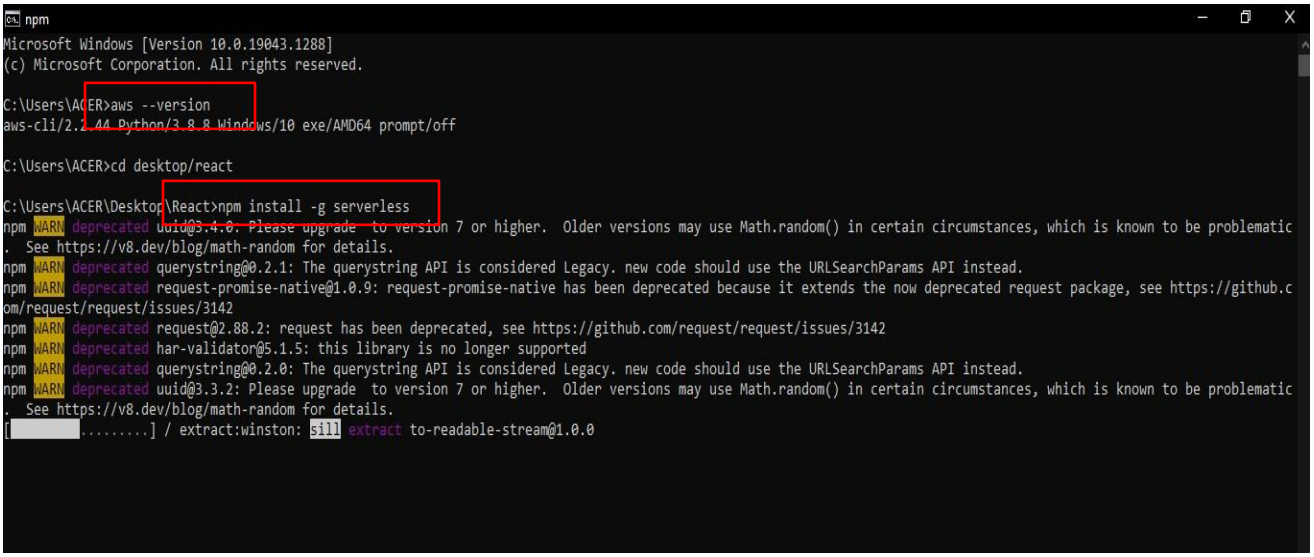
Aim: To create a Lambda function which will log “An Image has been added” once you add an object to a specific bucket in S3.

Solution:

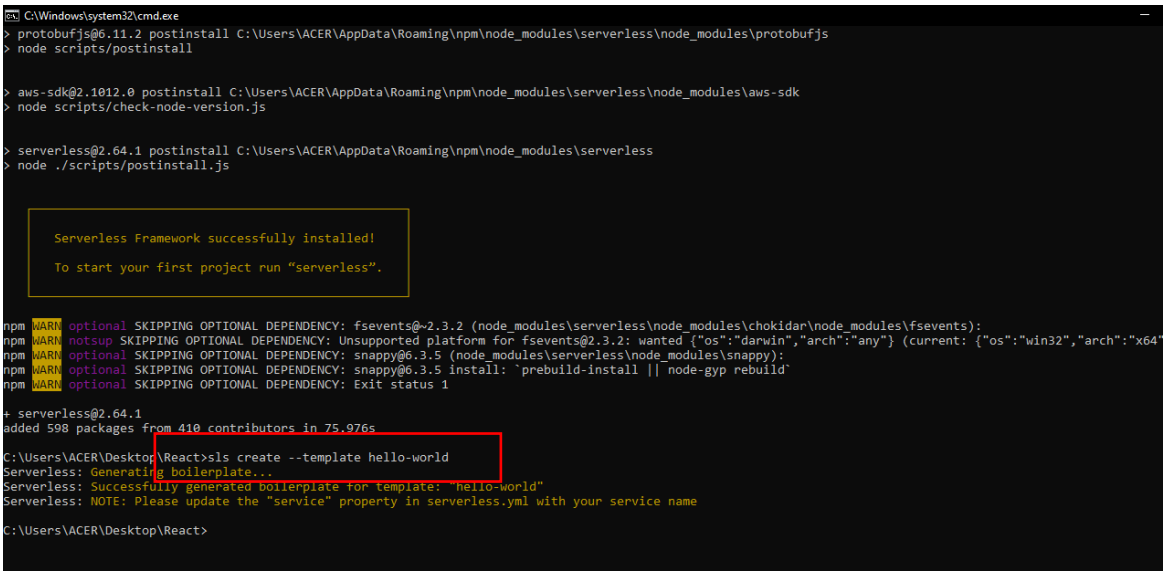
Required Tools

- Node JS 12
- Serverless
- AWS CLI

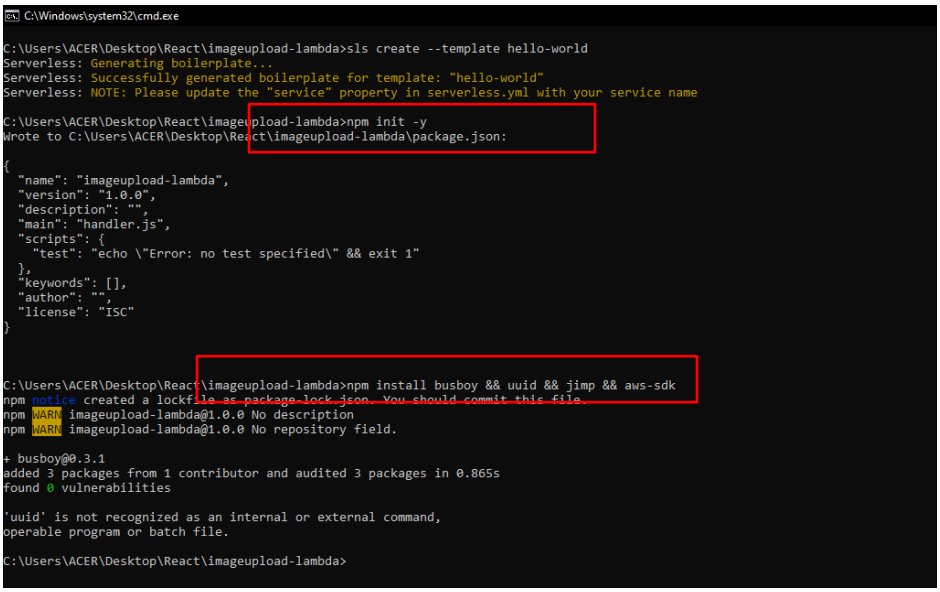
Step 1: Install AWS CLI and Serverless Framework.



Step 2: Run the following command to generate sample code with serverless.



Step 3: Install dependencies



Step 4: Create the function to decode the multipart/form-data

formParser.js - Notepad

File Edit Format View Help

```
const Busboy = require('busboy');

module.exports.parser = (event, fileZise) =>
    new Promise((resolve, reject) => {
    const busboy = new Busboy({
        headers: {
            'content-type':
            event.headers['content-type'] || event.headers['Content-Type']
        },
        limits: {
            fileZise
        }
    });

    const result = {
        files: []
    };

    busboy.on('file', (fieldname, file, filename, encoding, mimetype) => {
        const uploadFile = {}
        file.on('data', data => {
            uploadFile.content = data
        });
        file.on('end', () => {
            if (uploadFile.content) {
                uploadFile.filename = filename
                uploadFile.contentType = mimetype
                uploadFile.encoding = encoding
                uploadFile.fieldname = fieldname
                result.files.push(uploadFile)
             }
        })
    })

    busboy.on('field', (fieldname, value) => {
        result[fieldname] = value
```

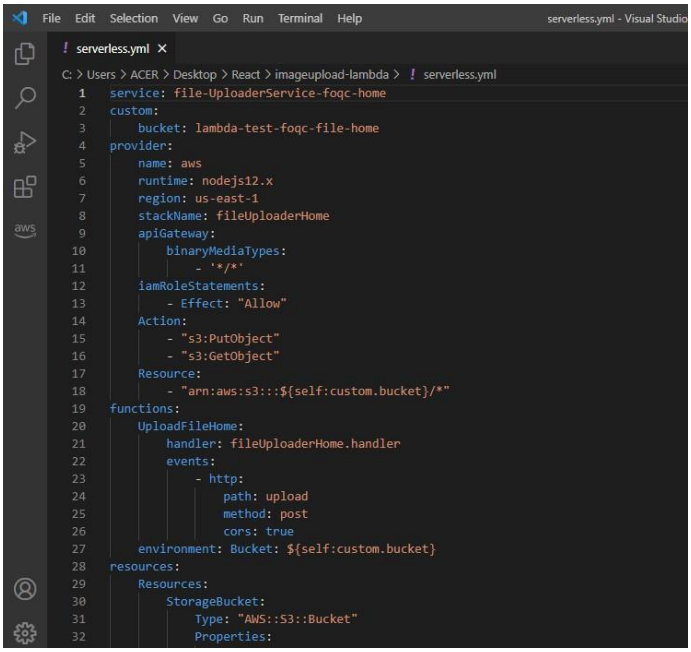
Step 5: Function that will process and upload the images to S3

fileUploaderHome.js - Notepad

File Edit Format View Help

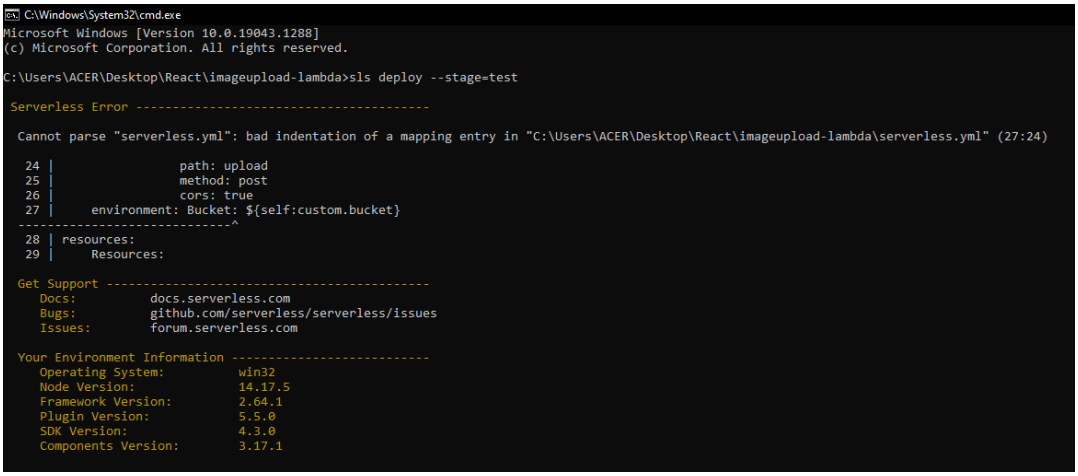
```
"use strict";
const AWS = require("aws-sdk")
const uuid = require("uuid/v4")
const Jimp = require("jimp")
const s3 = new AWS.S3()
const formParser = require("./formParser")

const bucket = process.env.Bucket
const MAX_SIZE = 4000000 // 4MB
const PNG_MIME_TYPE = "image/png"
const JPEG_MIME_TYPE = "image/jpeg"
const JPG_MIME_TYPE = "image/jpg"
const MIME_TYPES = [PNG_MIME_TYPE, JPEG_MIME_TYPE, JPG_MIME_TYPE]

module.exports.handler = async event => {
const getErrorMessage = message => ({ statusCode: 500, body: JSON.stringify( message })})

const isAllowedFile = (size, mimeType) => { // some validation code }

const uploadToS3 = (bucket, key, buffer, mimeType) =>
    new Promise((resolve, reject) => {
        s3.upload(
            { Bucket: bucket, Key: key, Body: buffer, ContentType: mimeType },
            function(err, data) {
                if (err) reject(err);
                resolve(data)
            })
    })

const resize = (buffer, mimeType, width) =>
    new Promise((resolve, reject) => {
        Jimp.read(buffer)
        .then(image => image.resize(width, Jimp.AUTO).quality(70).getBufferAsync(mimeType))
        .then(resizedBuffer => resolve(resizedBuffer))
        .catch(error => reject(error))
    })
```

Step 6: Build serverless.yml file.

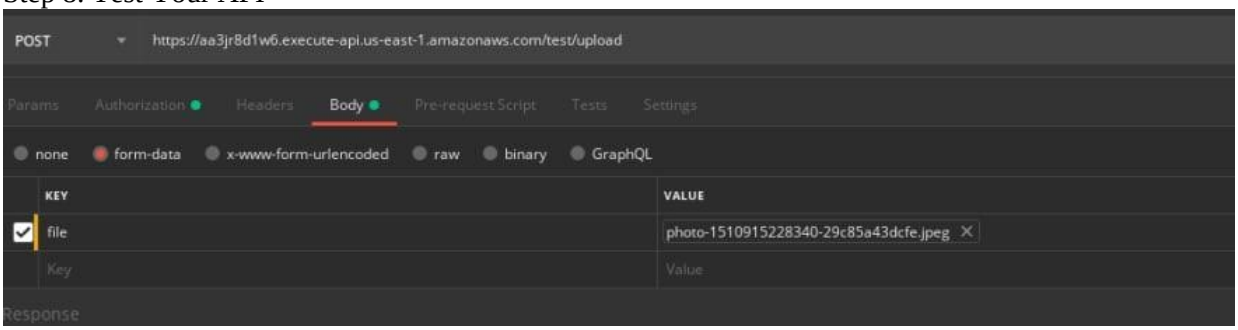


Step 7: Deploy the code using the following command.

- sls deploy --stage=test



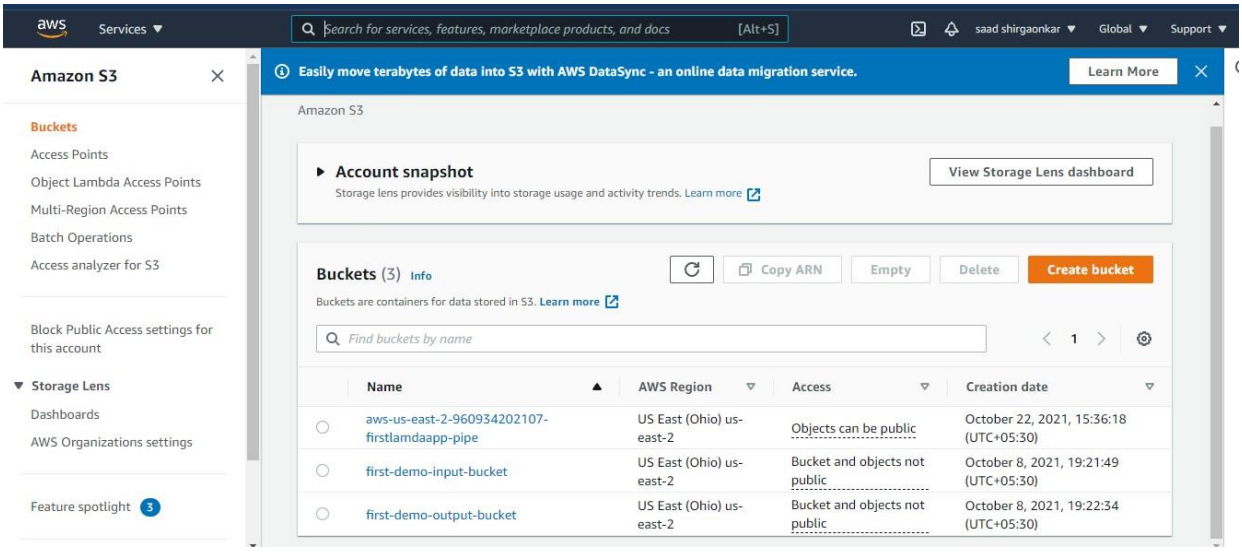
Step 8: Test Your API



Step 9: To conclude, in case you need to remove the service, run the following command.

- sls remove --stage=test

Step 10: Check the result in S3 Bucket.



Result: The Image upload with AWS lambda function is successfully completed.

Conclusion: The experiment helped me to understand the importance of AWS Lambda and its use. The modules of the experiment have been successfully completed.