Program Outcomes as defined by NBA (PO)

Engineering Graduates will be able to:

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**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
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.

Datta Meghe College of Engineering, Airoli, Navi Mumbai

DEPARTMENT OF INFORMATION TECHNOLOGY

Institute Vision : **To create value - based technocrats to fit in the world of work and research**

Institute Mission : **To adapt the best practices for creating competent human beings to work in the world of technology and research.**

Department Vision : **To develop and foster students for successful careers in the dynamic field of Information Technology.**

Department Mission :

M1:	**To create and disseminate knowledge through research, teaching & learning and to enhance society in meaningful and sustainable ways.**
M2:	**To impart a suitable environment for students and staff to showcase innovative ideas in the field of IT.**
M3:	**To bridge the curriculum gap by facilitating effective interaction among industry and Staff/Students.**

Program Educational Objectives (PEO)

PEO 1	**Develop proficiency as an IT technocrat with an ability to solve a wide range of computational problems in industry, government, or other work environments.**
PEO 2	**Attain the ability to adapt quickly to new environments and technologies, assimilate new information, and work in multi-disciplinary areas with a strong focus on innovation and entrepreneurship.**
PEO 3	**Prepare graduates with the ability of life-long learning to innovate in ever-changing global economic and technological environments of the current era.**
PEO 4	**Possess the ability to function ethically and responsibly with good cultural values and integrity to apply the best principles and practices of Information Technology towards the society.**

Program Specific Outcomes (PSO)

PSO1	**Apply Core Information Technology knowledge to develop stable and secure IT system**
PSO2	**Design, IT infrastructures for an enterprise using concepts of best practices in Information Technology and security domain.**
PSO3	**Ability to work in multidisciplinary IT enabled projects for industry and society by adapting latest trends and technologies like Analytics, Blockchain, Cloud, Data science.**

Datta Meghe College of Engineering, Airoli

Department of Information Technology

Course Name: Advanced Devops Lab (R-19)

Course Code: ITL504

Year of Study:2021 Semester: V

Course Outcomes

ITL504.1	To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements
ITL504.2	To deploy single and multiple container applications and manage application deployments with rollouts in Kubernetes
ITL504.3	To apply best practices for managing infrastructure as code environments and use terraform to define and deploy cloud infrastructure
ITL504.4	To identify and remediate application vulnerabilities earlier and help integrate security in the development process using SAST Technique
ITL504.5	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
ITL504.6	To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework

DATTA MEGHE COLLEGE OF ENGINEERING

AIROLI, NAVI MUMBAI - 400708

C E R T I F I C A T E

This is to certify that Mr. / ~~Miss~~ Parmar Jay Rohit

Of ____TE_____Class ____IT-B_____Roll No. 3_____

Subject_____Advanced Devops Lab__has performed the experiments /

Sheets mentioned in the index, in the premises of this institution.

S. P. Nehete	Dr. S. R. Kolhe	Dr. S. D. Sawarkar
Practical Incharge	**Head of Dept.**	**Principal**

Date 21.10.21

Examined on

Examiner 1_____**Examiner 2** _____

Datta Meghe College of Engineering, Airoli, Navi Mumbai	DEPARTMENT OF INFORMATION TECHNOLOGY
	List of Experiments **Subject: Advanced Devops Lab** **Code: ITL504**

Sr.No.	Name of the Experiment	CO Covered	Page No.	Date	Signature
1	To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collaboration Demonstration.	LO1	1-15	7.7.21	
2	To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy	LO1	16-52	14.7.21	
3	To understand the Kubernetes Cluster Architecture, install and Spin Up a Kubernetes Cluster on Linux Machines/Cloud Platforms.	LO1,LO2	53-55	14.7.21	
4	To install Kubectl and execute Kubectl commands to manage the Kubernetes cluster and deploy Your First Kubernetes Application.	LO1,LO2	56-60	28.7.21	
5	To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine	LO1,LO3	61-65	11.8.21	
6	To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform	LO1,LO3	66-75	11.8.21	
7	To understand Static Analysis SAST process and learn to integrate Jenkins SAST to SonarQube/GitLab	LO1,LO4	76-82	25.8.21	
8	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	LO1,LO4	83-90	8.9.21	
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	91-96	15.9.21	
10	To perform Port, Service monitoring, Windows/Linux server monitoring using Nagios	LO1,LO5	97-109	22.9.21	
	Assignment No1	LO1-LO6	110-118	27.7.21	
	Assignment No 2	LO1-LO6	119-130	7.9.21	

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:
AWS Cloud9 is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more, so you don't need to install files or configure your development machine to start new projects. Since your Cloud9 IDE is cloud-based, you can work on your projects from your office, home, or anywhere using an internet-connected machine. Cloud9 also provides a seamless experience for developing serverless applications enabling you to easily define resources, debug, and switch between local and remote execution of serverless applications. With Cloud9, you can quickly share your development environment with your team, enabling you to pair program and track each other's inputs in real time.

Benefits:

CODE WITH JUST A BROWSER
AWS Cloud9 gives you the flexibility to run your development environment on a managed Amazon EC2 instance or any existing Linux server that supports SSH. This means that you can write, run, and debug applications with just a browser, without needing to install or maintain a local IDE. The Cloud9 code editor and integrated debugger include helpful, time-saving features such as code hinting, code completion, and step-through debugging. The Cloud9 terminal provides a browser- based shell experience enabling you to install additional software, do a git push, or enter commands.

CODE TOGETHER IN REAL TIME
AWS Cloud9 makes collaborating on code easy. You can share your development environment with your team in just a few clicks and pair program together. While collaborating, your team members can see each other type in real time, and instantly chat with one another from within the IDE.

BUILD SERVERLESS APPLICATIONS WITH EASE

AWS Cloud9 makes it easy to write, run, and debug serverless applications. It preconfigures the development environment with all the SDKs, libraries, and plug-ins needed for serverless development. Cloud9 also provides an environment for locally testing and debugging AWS Lambda functions. This allows you to iterate on your code directly, saving you time and improving the quality of your code.

DIRECT TERMINAL ACCESS TO AWS

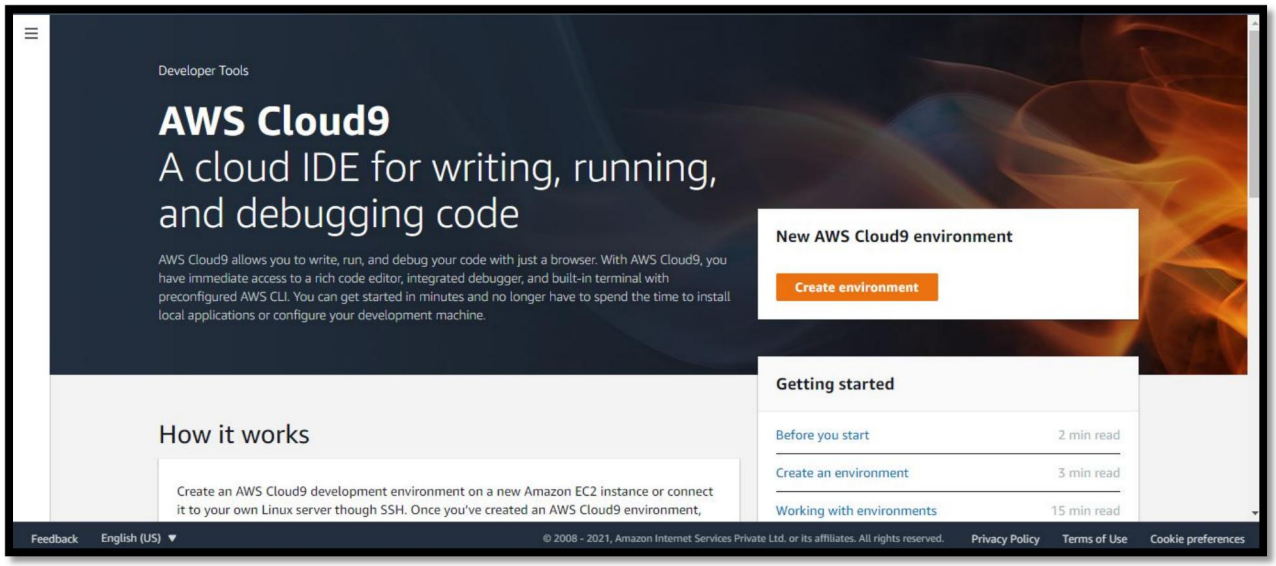
AWS Cloud9 comes with a terminal that includes sudo privileges to the managed Amazon EC2 instance that is hosting your development environment and a preauthenticated AWS Command Line Interface. This makes it easy for you to quickly run commands and directly access AWS services.

START NEW PROJECTS QUICKLY

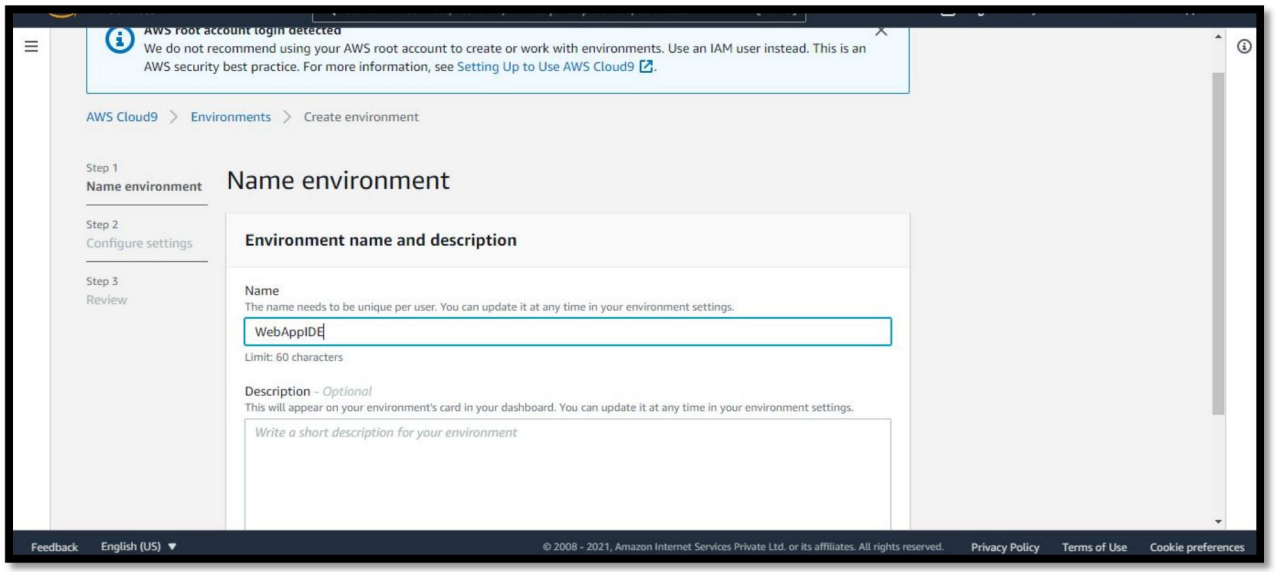
AWS Cloud9 makes it easy for you to start new projects. Cloud9's development environment comes prepackaged with tooling for over 40 programming languages, including Node.js, JavaScript, Python, PHP, Ruby, Go, and C++. This enables you to start writing code for popular application stacks within minutes by eliminating the need to install or configure files, SDKs, and plug-ins for your development machine. Because Cloud9 is cloud-based, you can easily maintain multiple development environments to isolate your project's resources.

Steps:

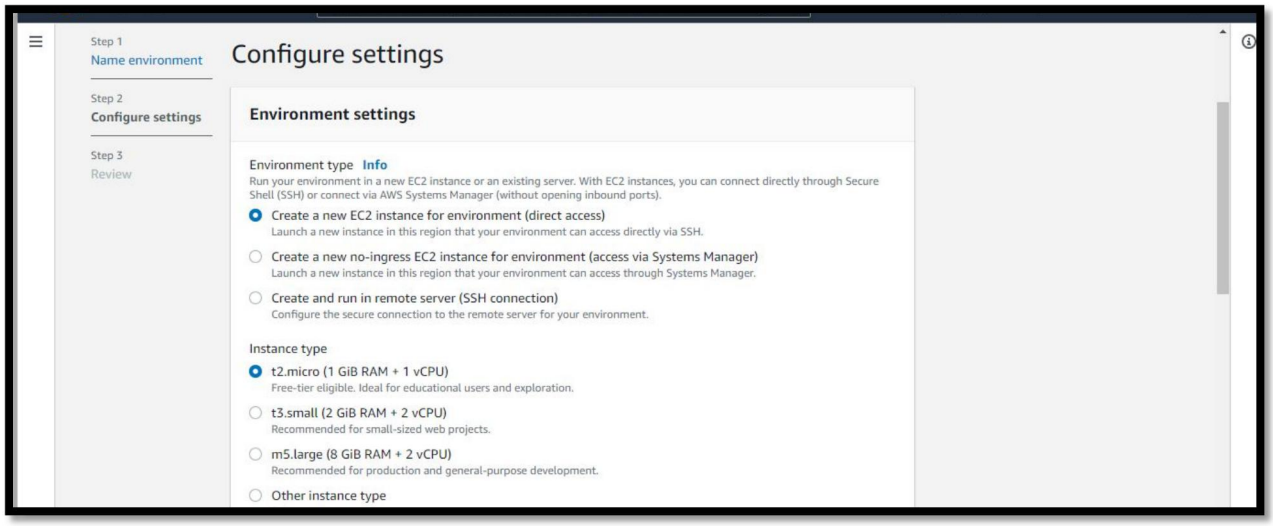
1. Click on Create Environment.

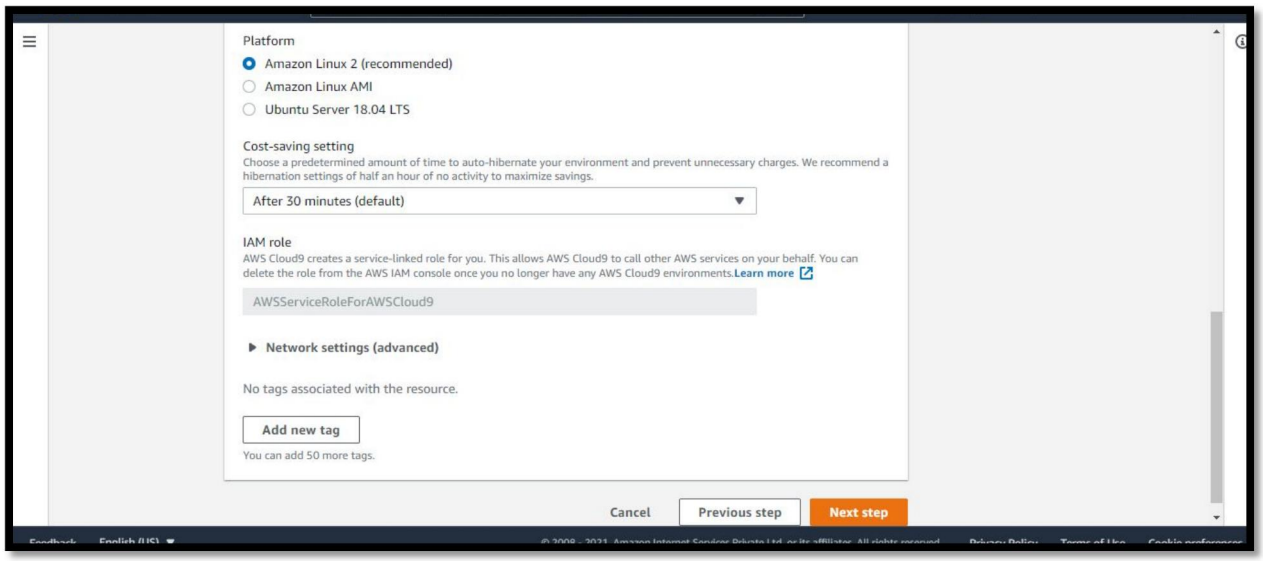


2. Provide name for the Environment and click on next.

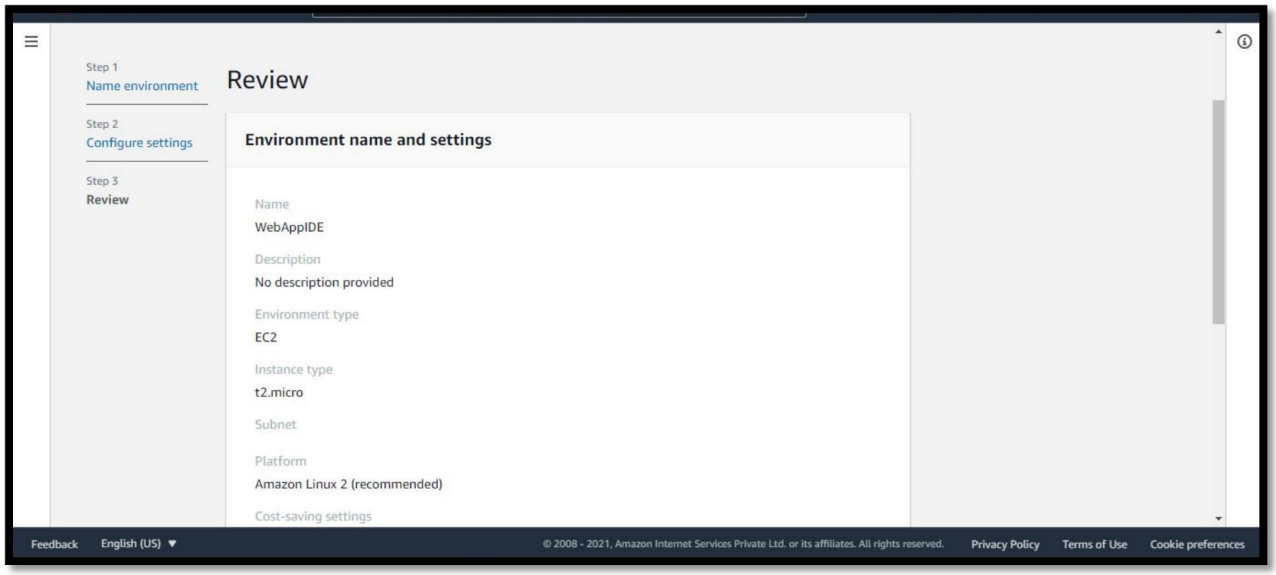


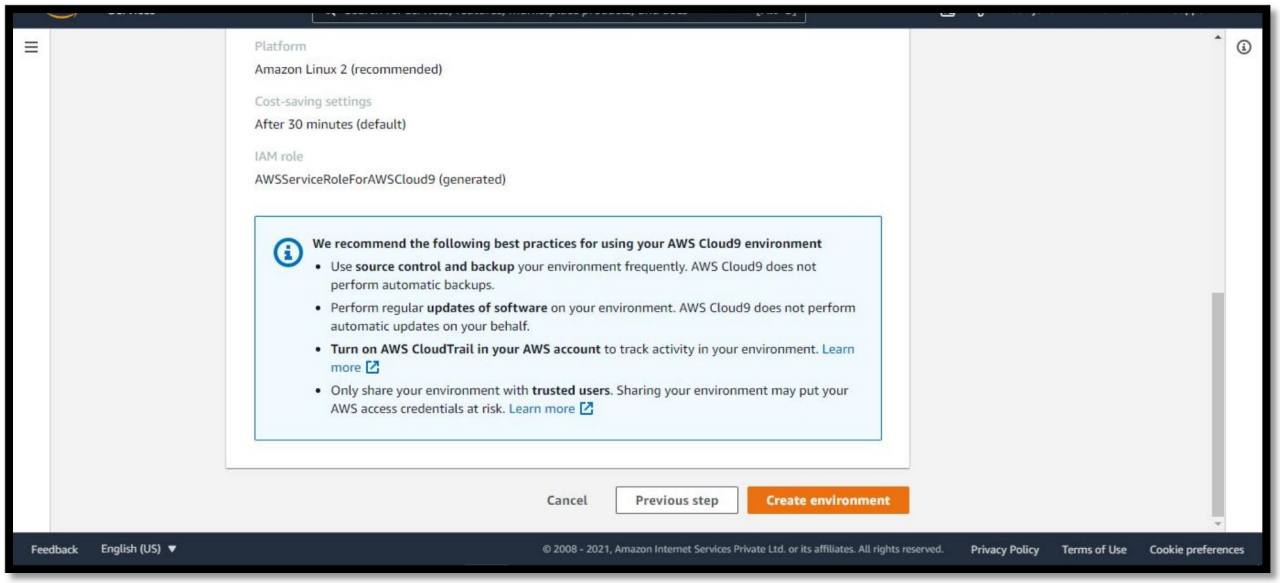
3. Keep all the Default settings as shown in below:





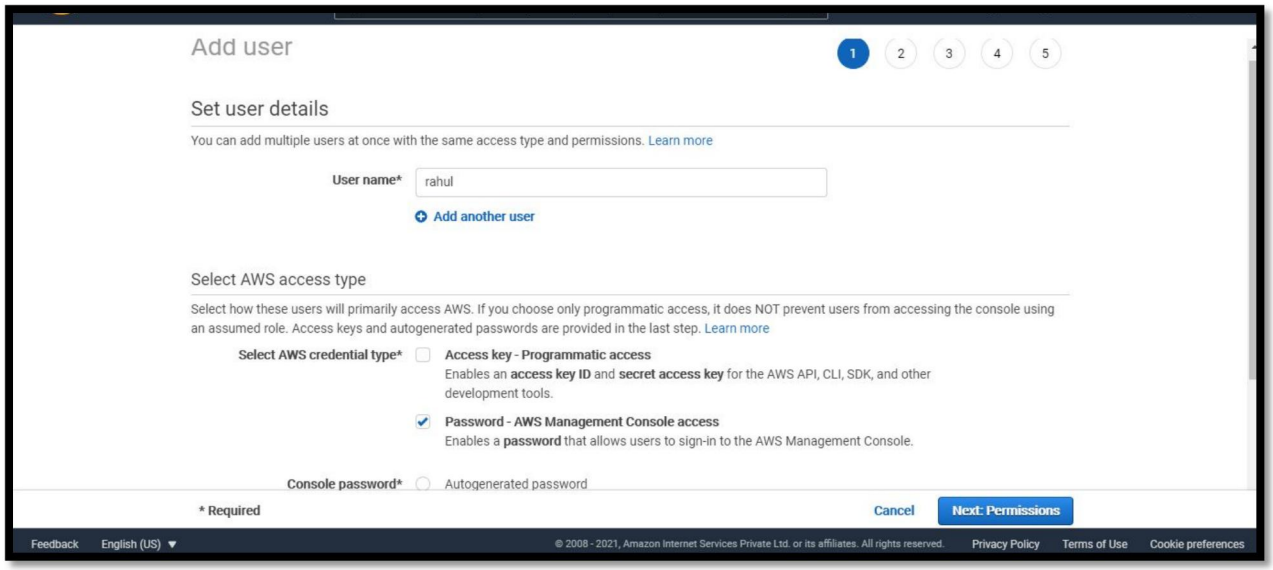
4. Review the Environment name and Settings and click on Create Environment:

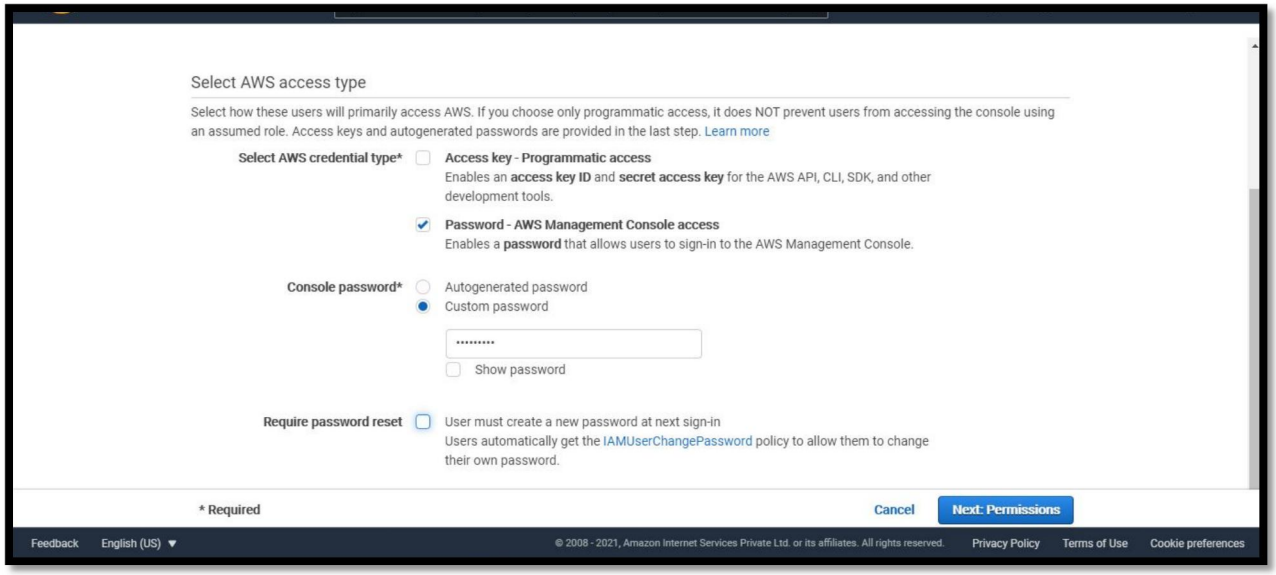

Step 1
Name environment
Step 2
Configure settings
Step 3
Review
Review
Environment name and settings
Name
WebAppIDE
Description
No description provided
Environment type
EC2
Instance type
t2.micro
Subnet
Platform
Amazon Linux 2 (recommended)
Cost-saving settings
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences


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

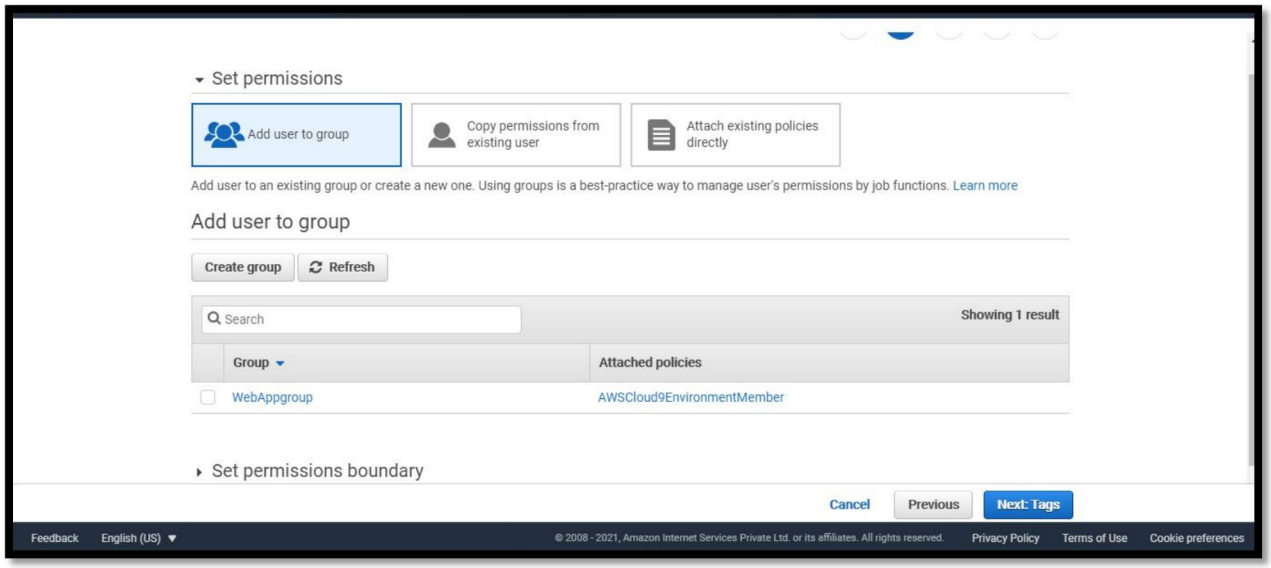
It will take few minutes to create aws instance for your Cloud 9 Environment.

5. Till that time open IAM Identity and Access Management in order to Add user In other tab.
6. Add user provide manual password if you want and click on Next permission tab.

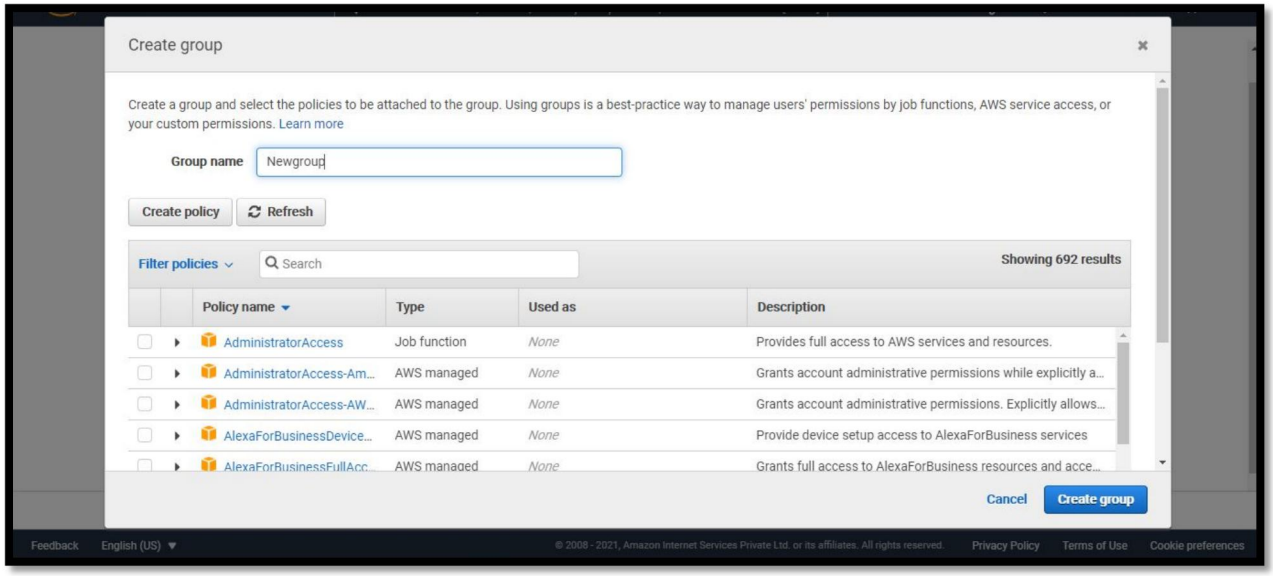




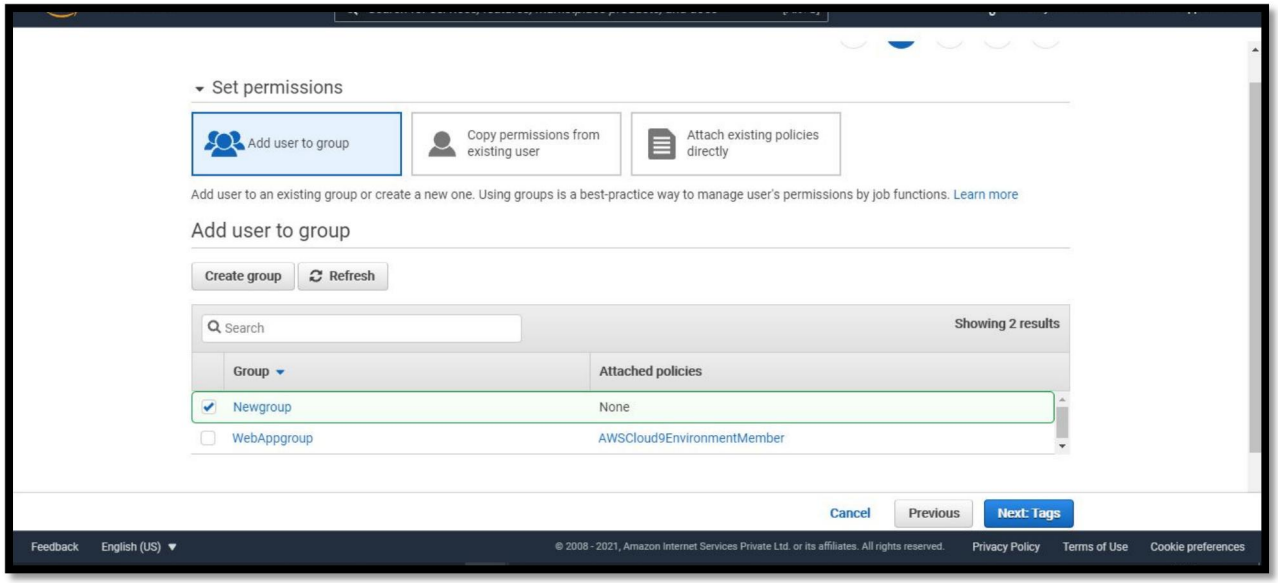
7. Click on Create group

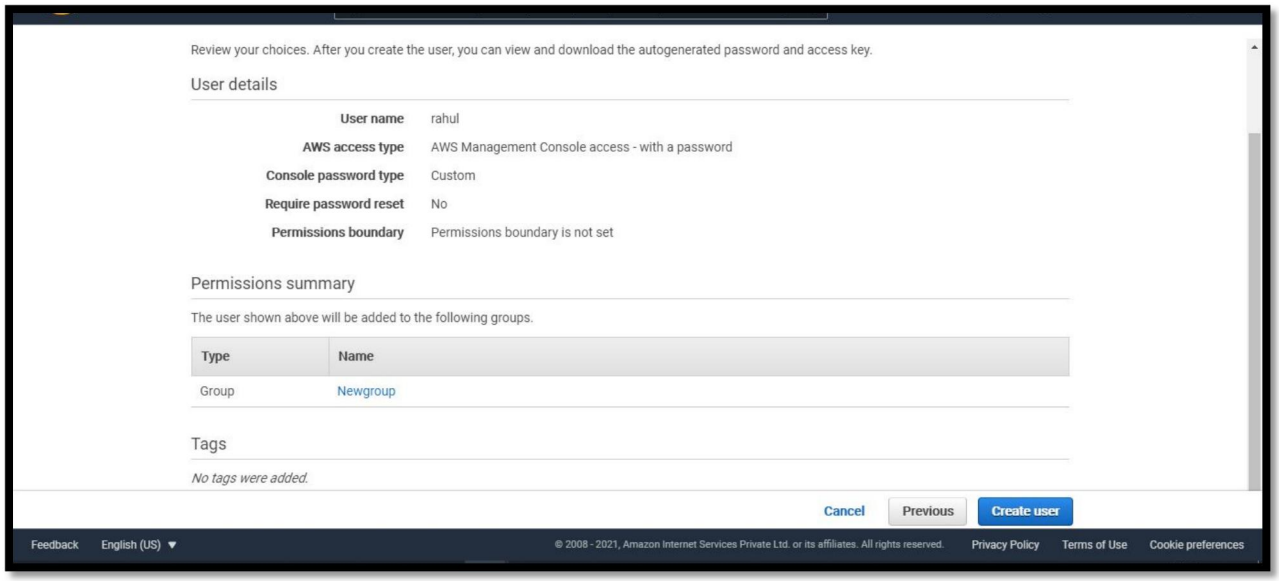


8. Provide group name and click on create group.

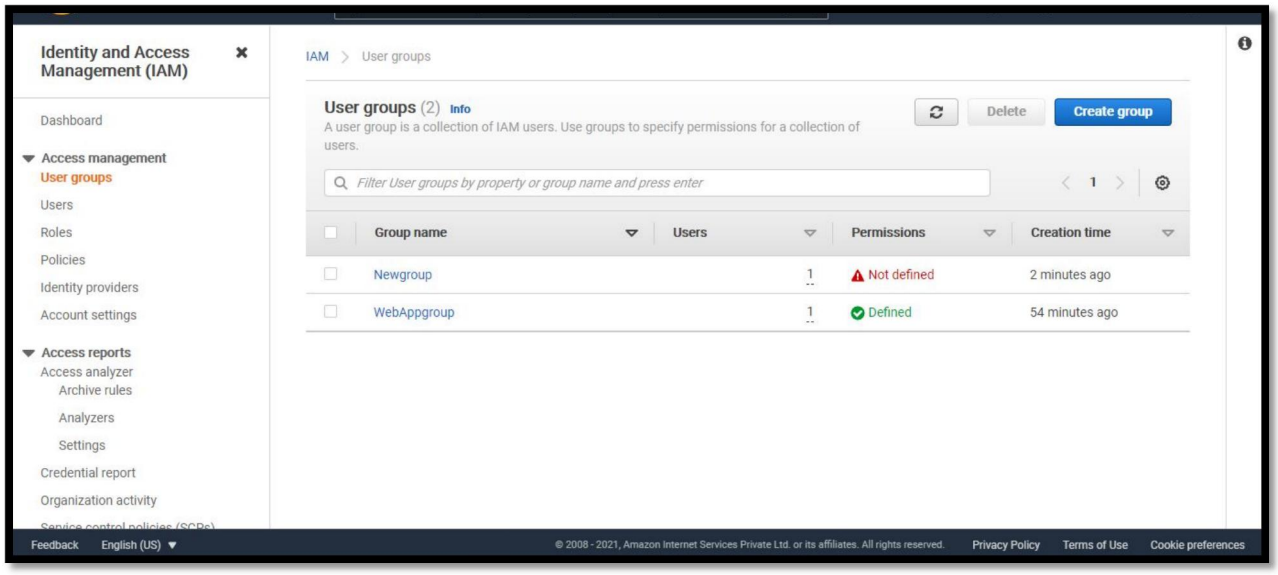


9. After that group is created click on next if u want to provide tag else click on Review for user settings and click on create user as shown in fig.

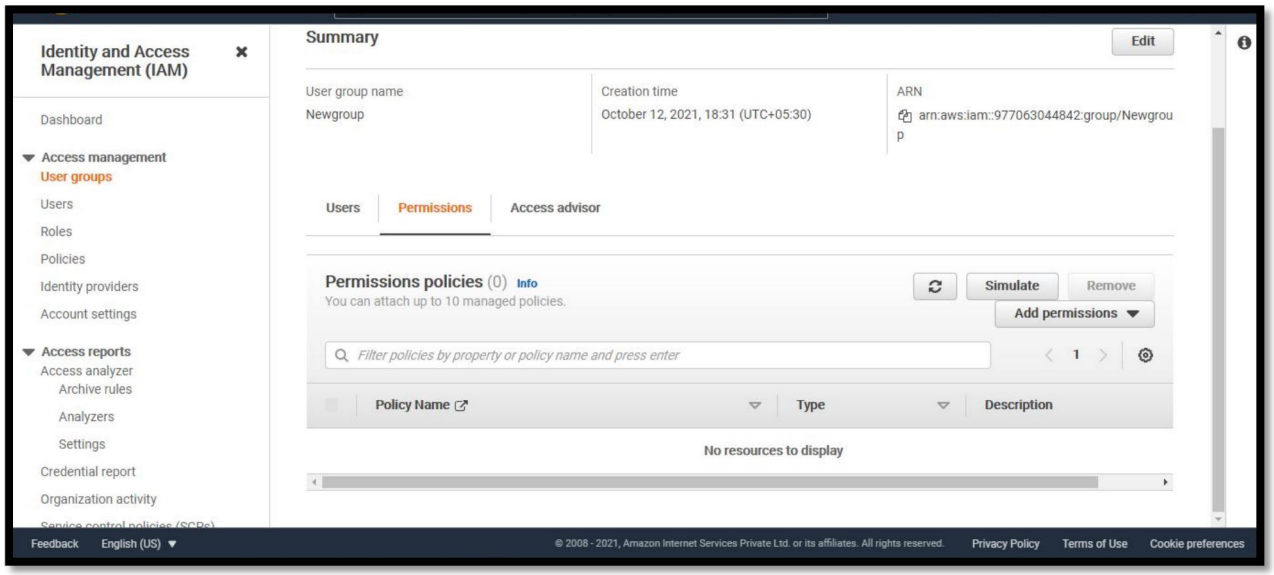




10. Now close that window and Navigate to user Groups from left pane in IAM.

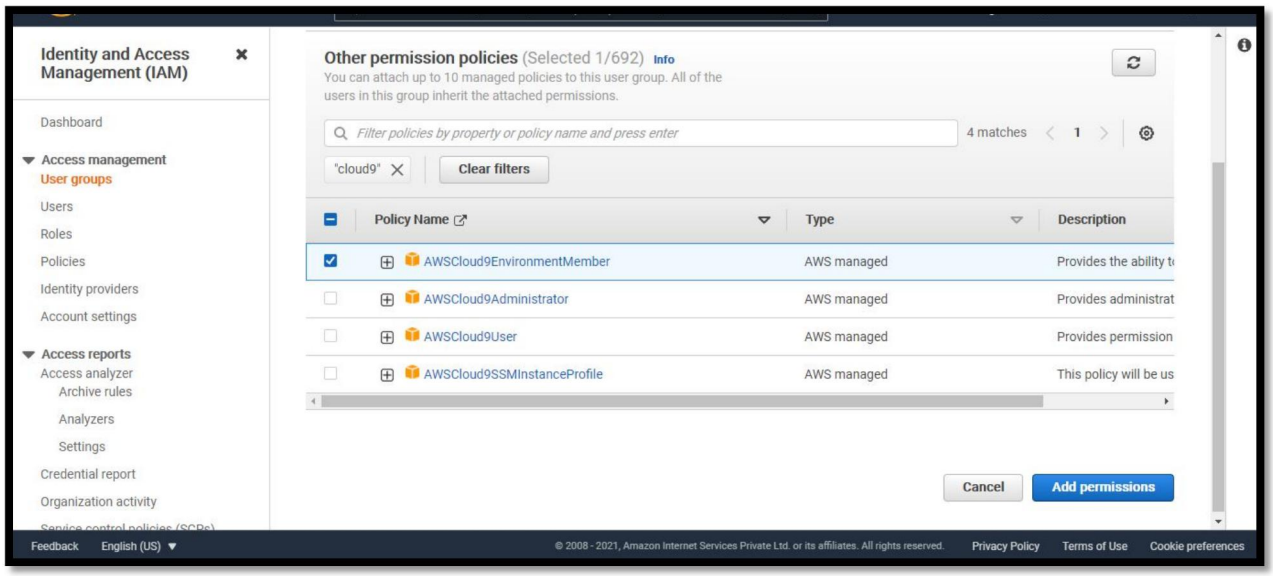


11. click on your group name which you have created and navigate to permission tab as shown:

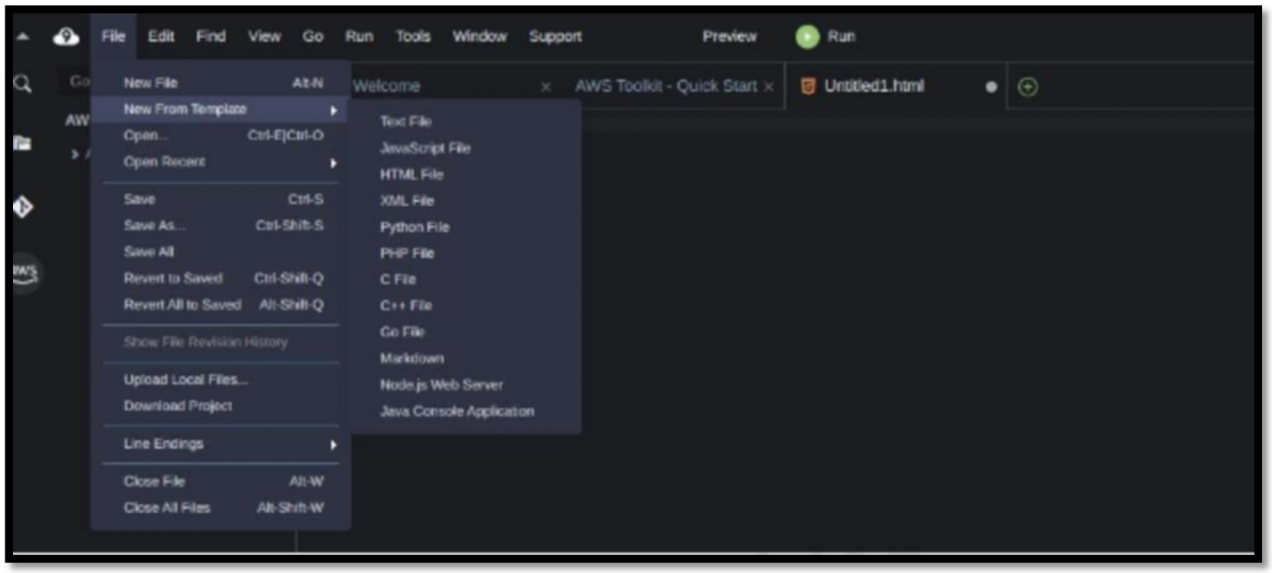


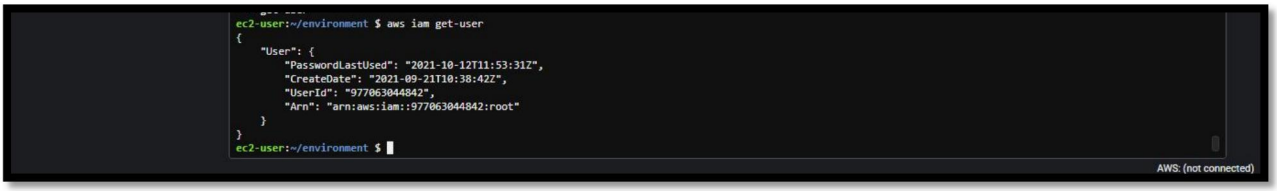
12. Now click on Add permission and select Attach Policy after that search for

Cloud9 related policy and select Awscloud9EnviornmentMember policy and add it.

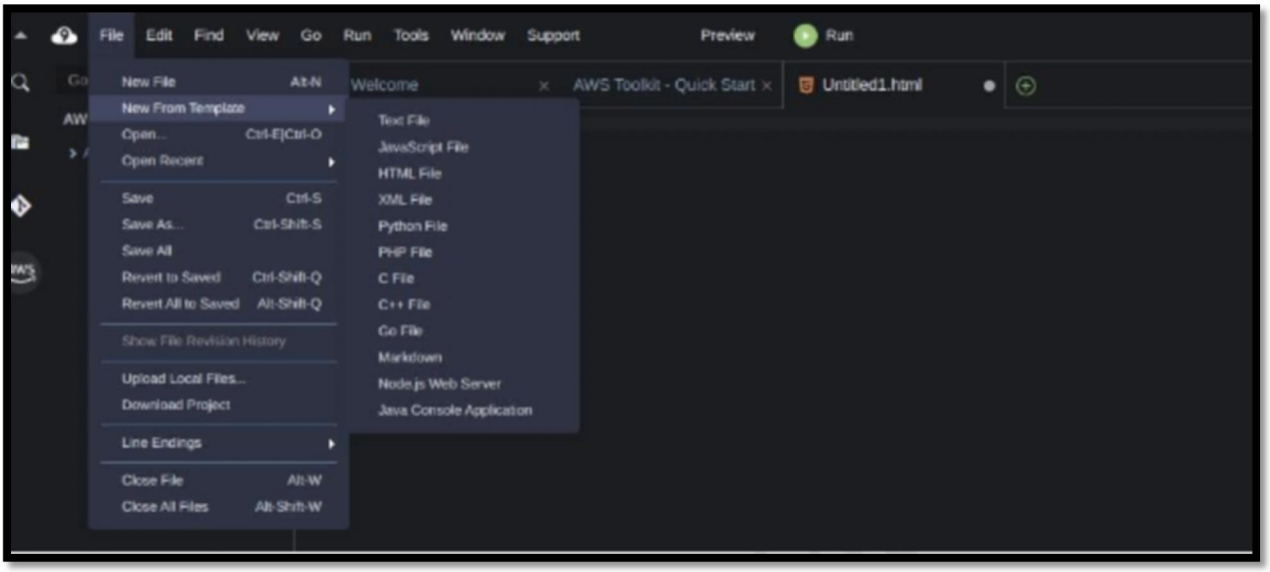


13. If you check at bottom side Cloud9 IDE also giving you and aws CLI for command operations: as we here checked git version, iam user details and

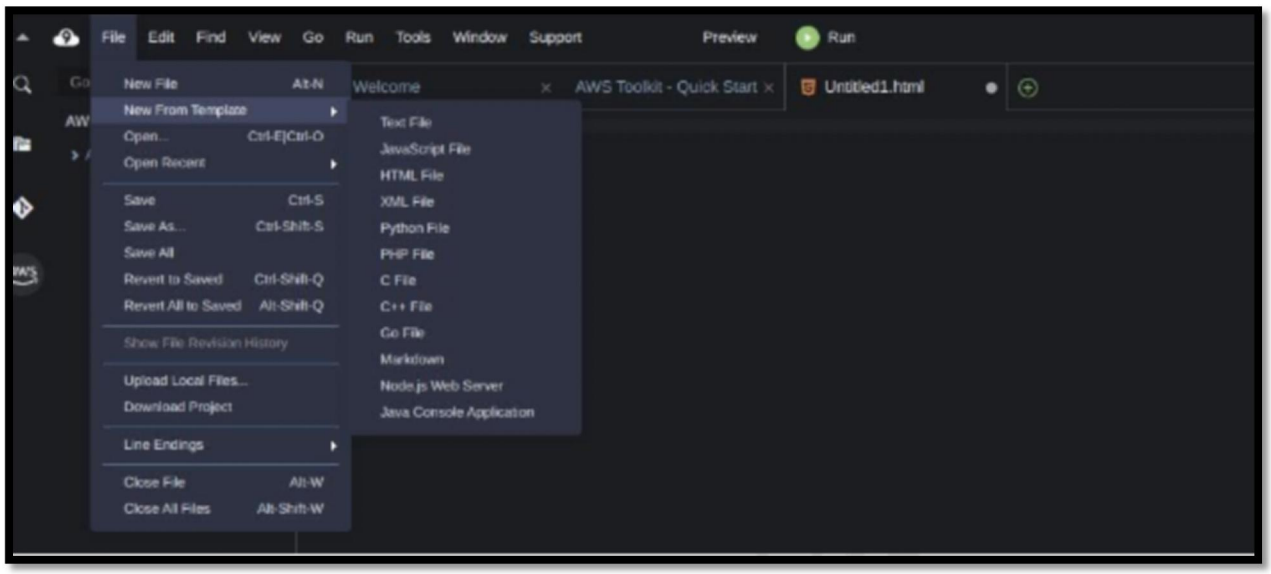




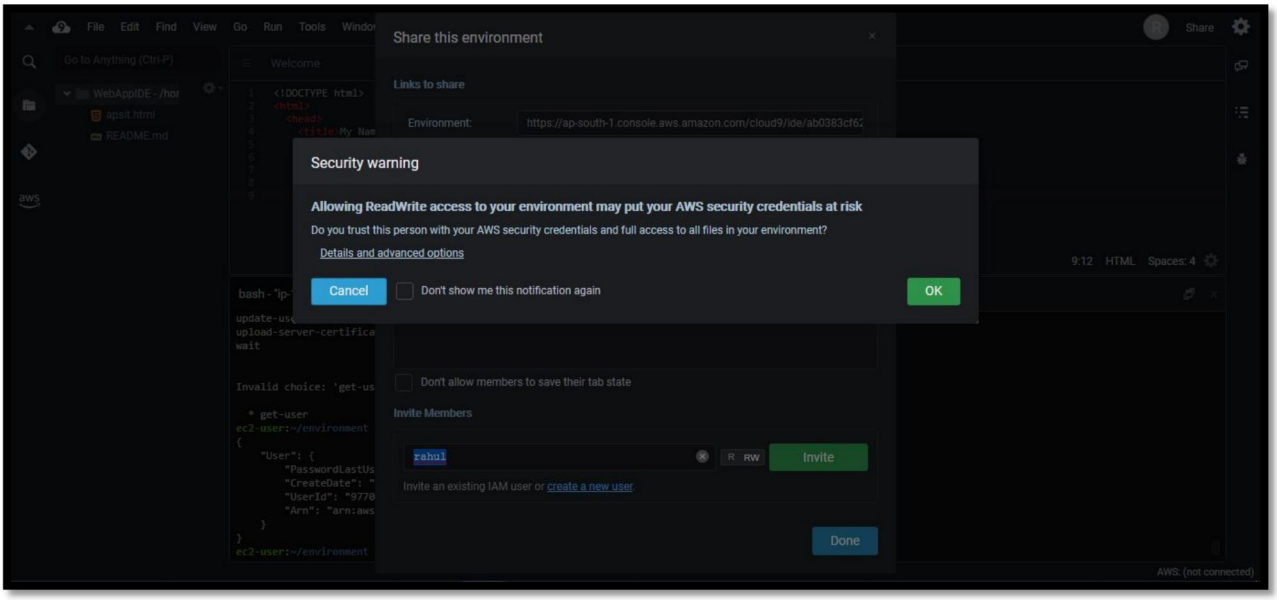
14. Now we will setup collaborative environment Click on File you can create new file or choose from template, here m opting HTML file to collaborate.

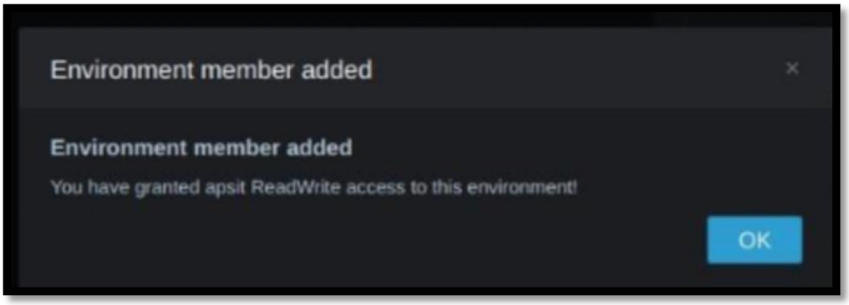


15. Edit html file and save it

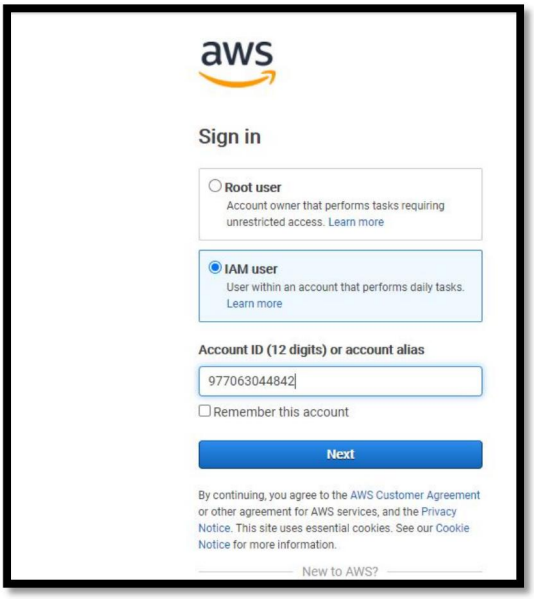


16. now in order to share this file to collaborate with other members of your team click on Share option on Right Pane and username which you created in IAM before into Invite members and enable permission as RW (Read and Write) and click on Done. Click OK for Security warning.

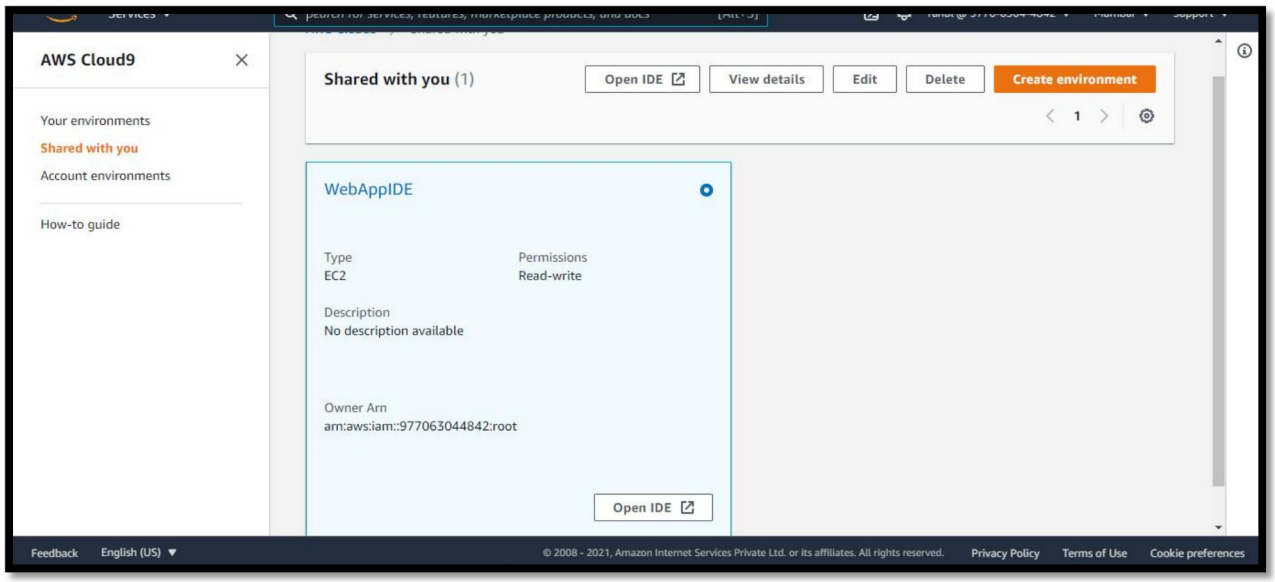




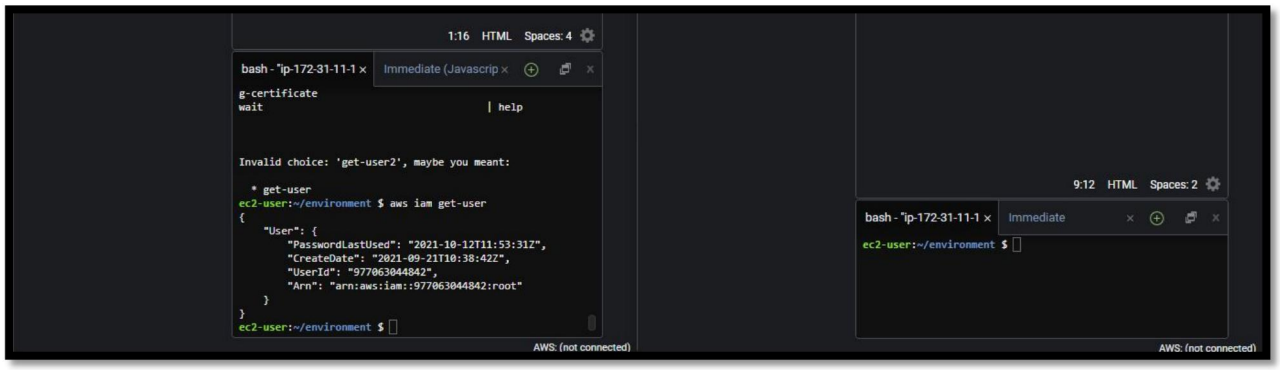
17. Now Open your Browsers Incognito Window and login with IAM user which you configured before.



18. After Successful login with IAM user open Cloud9 service from dashboard services and click on shared with you environment to collaborate.



19. Click on Open IDE you will same interface as your other member have to collaborate in real time, also you all within team can do group chats as shown below:



Conclusion:

Hence, We understood the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launched AWS Cloud9 IDE and Performed Collaboration Demonstration.

Experiment no. 2

Aim :- To Build Your Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Theory :-

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

Benefits

Fully managed build service

AWS CodeBuild eliminates the need to set up, patch, update, and manage your own build servers and software. There is no software to install or manage.

Continuous scaling

AWS CodeBuild scales up and down automatically to meet your build volume. It immediately processes each build you submit and can run separate builds concurrently, which means your builds are not left waiting in a queue.

Pay as you go

With AWS CodeBuild, you are charged based on the number of minutes it takes to complete your build. This means you no longer have to worry about paying for idle build server capacity.

Extensible

You can bring your own build tools and programming runtimes to use with AWS CodeBuild by creating customized build environments in addition to the prepackaged build tools and runtimes supported by CodeBuild.

Enables continuous integration and delivery

AWS CodeBuild belongs to a family of AWS Code Services, which you can use to create complete, automated software release workflows for continuous integration and delivery (CI/CD). You can also integrate CodeBuild into your existing CI/CD workflow. For example, you can use CodeBuild as a worker node for your existing Jenkins server setup for distributed builds.

Secure

With AWS CodeBuild, your build artifacts are encrypted with customer-specific keys that are managed by the AWS Key Management Service (KMS). CodeBuild is integrated with AWS Identity and Access Management (IAM), so you can assign user-specific permissions to your build projects.

AWS CodePipeline

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.

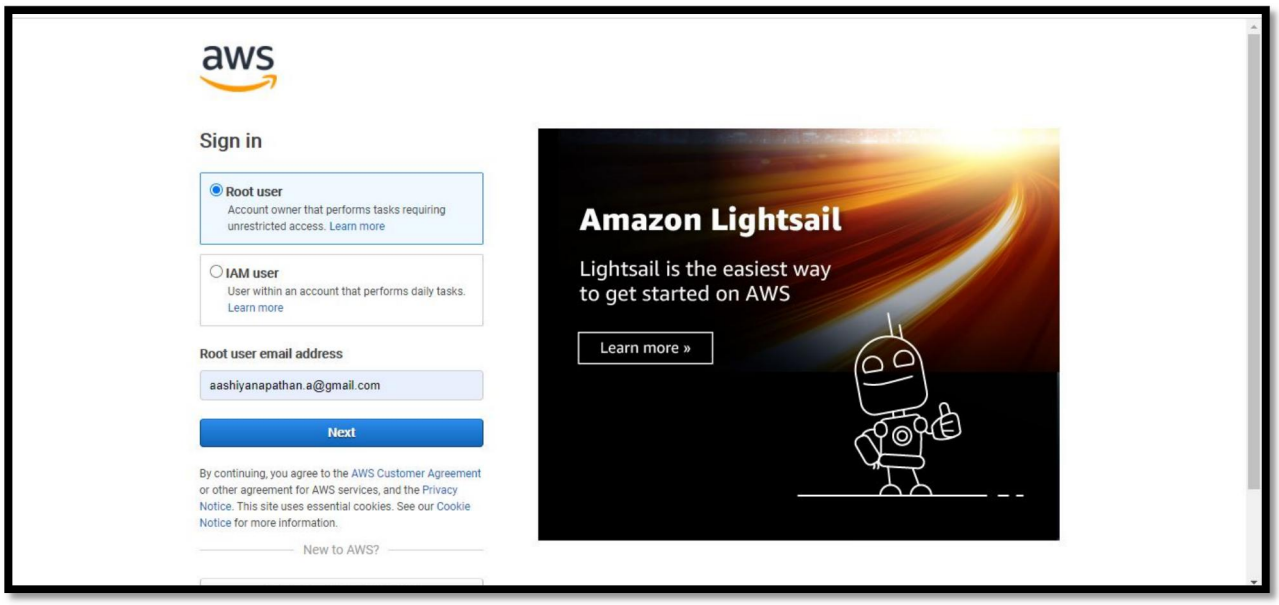
when developers commit changes to a source repository, CodePipeline automatically detects the changes. Those changes are built, and if any tests are configured, those tests are run. After the tests are complete, the built code is deployed to staging servers for testing. From the staging server, CodePipeline runs more tests, such as integration or load tests. Upon the successful completion of those tests, and after a manual approval action that was added to the pipeline is approved, CodePipeline deploys the tested and approved code to production instances.

CodePipeline can deploy applications to EC2 instances by using CodeDeploy, AWS Elastic Beanstalk, or AWS OpsWorks Stacks. CodePipeline can also deploy container-based applications to services by using Amazon ECS. Developers can also use the integration points provided with CodePipeline to plug in other tools or services, including build services, test providers, or other deployment targets or systems.

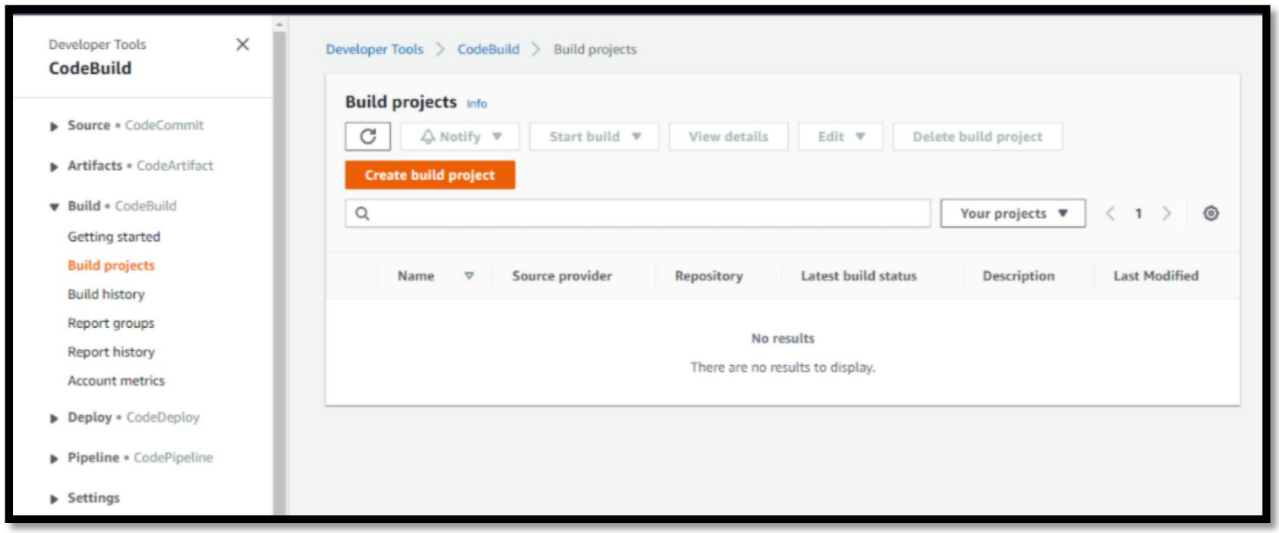
Steps:

1. Login with your AWS account.

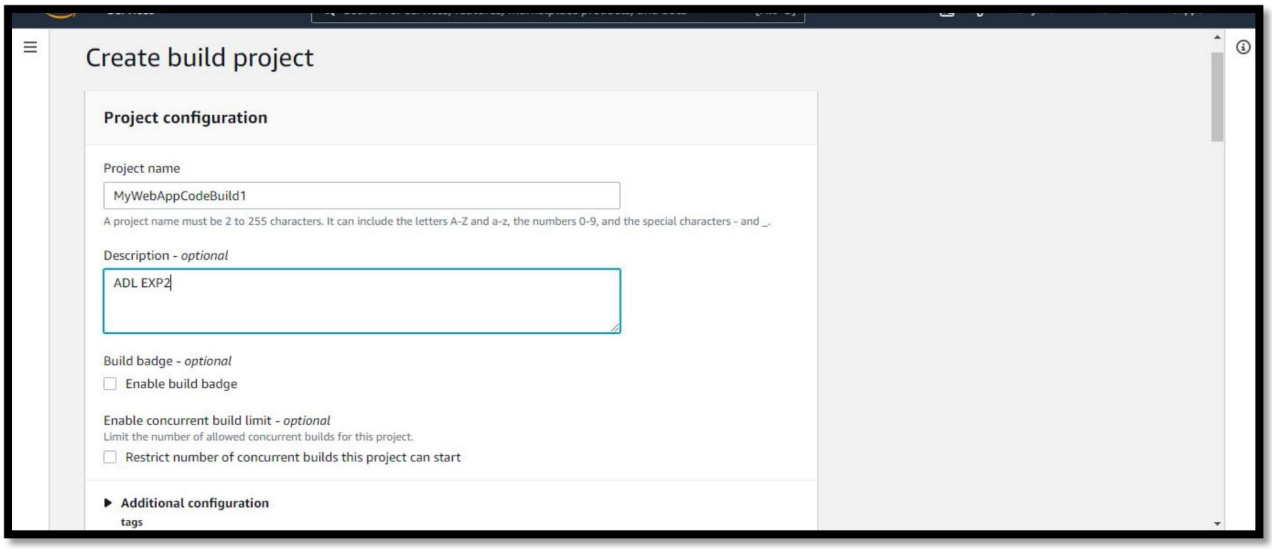
2. Navigate to CodeBuild service from Developer tools section as below:



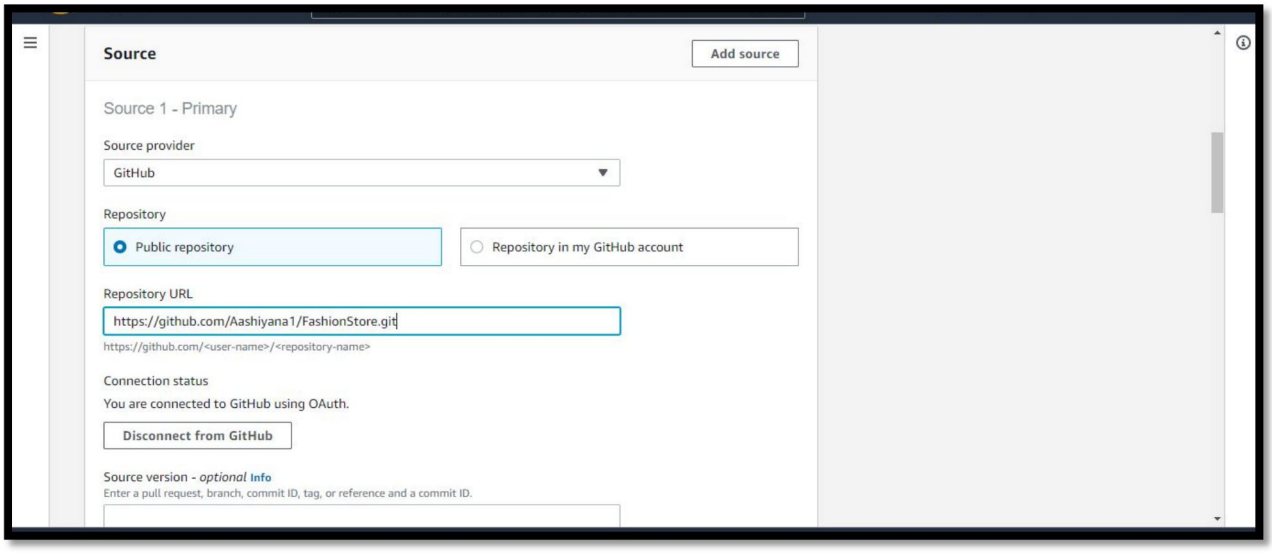
3. Click on Create build project :



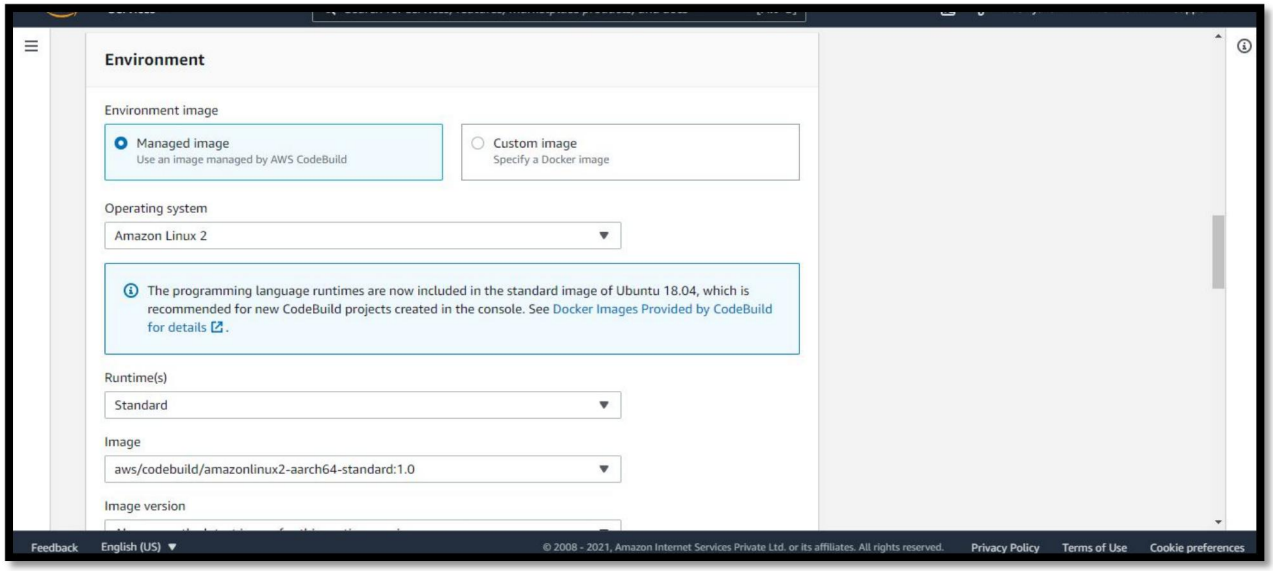
4. Provide the name for the build project(MyWebAppCodeBuild) and click on Next.

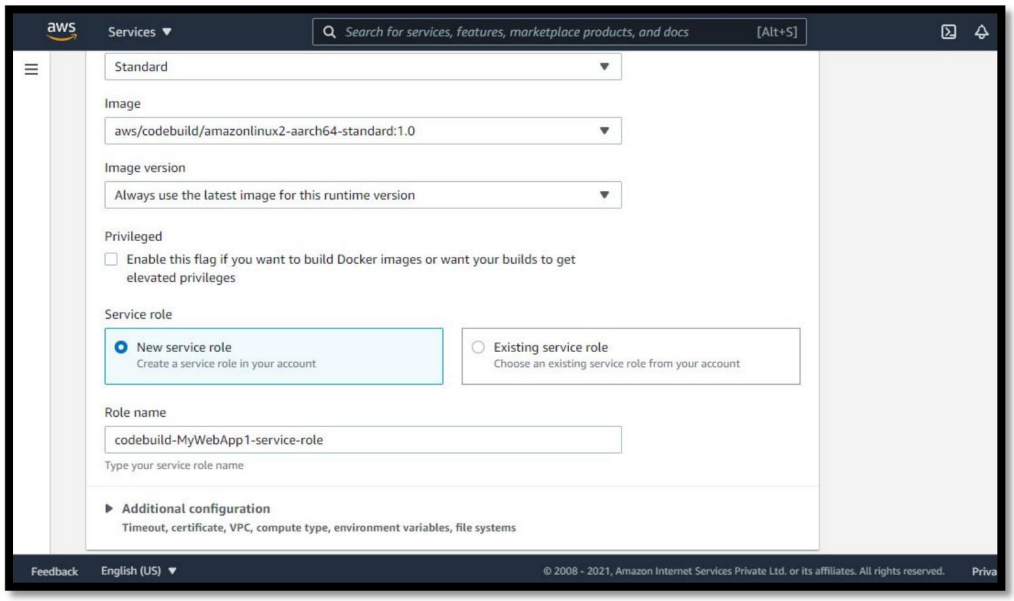


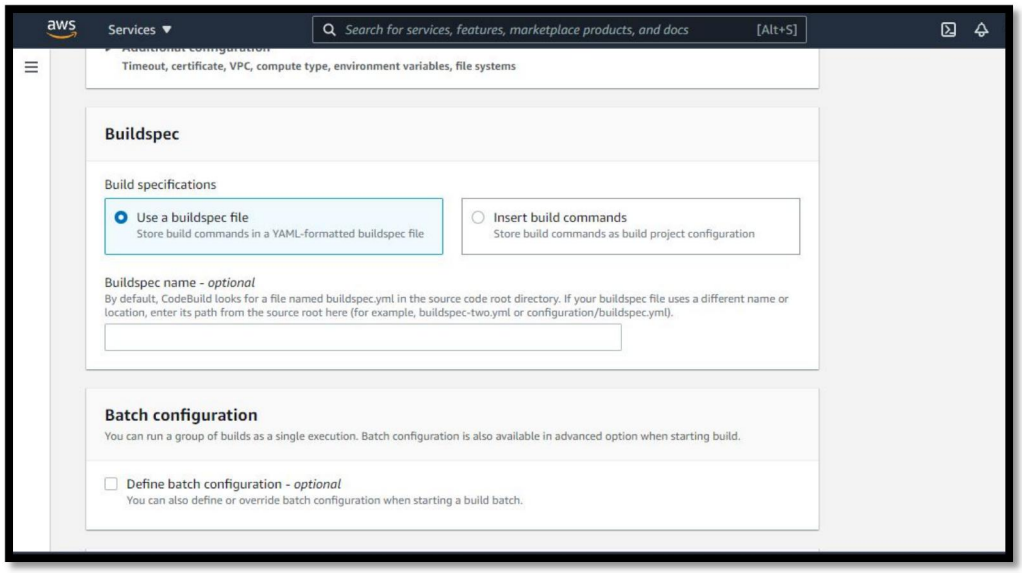
5. Add Source

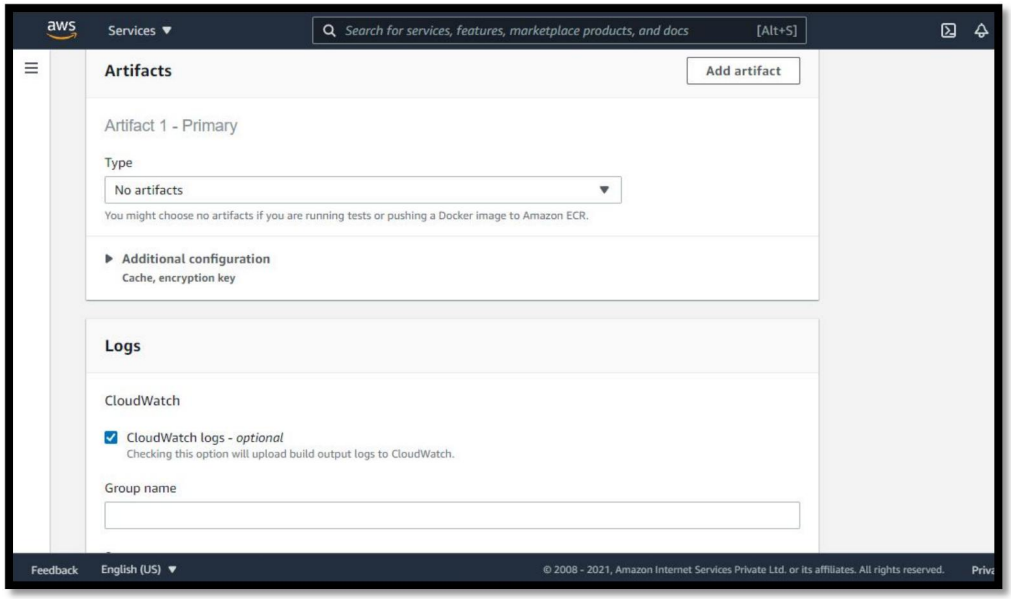


5. Add Environment image

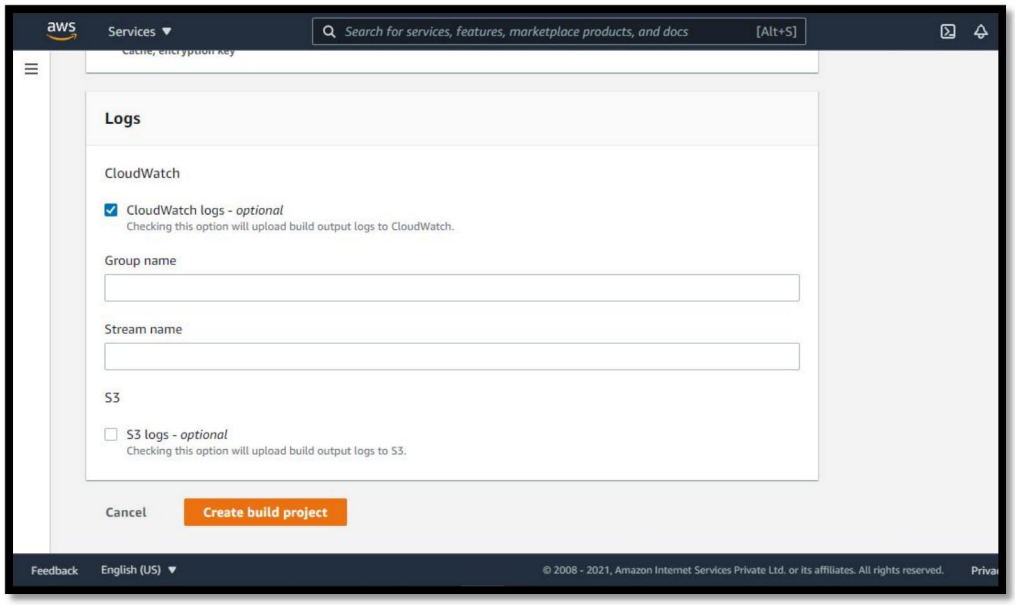

Environment
Environment image
Managed image
Use an image managed by AWS CodeBuild
Custom image
Specify a Docker image
Operating system
Amazon Linux 2
The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See Docker Images Provided by CodeBuild for details.
Runtime(s)
Standard
Image
aws/codebuild/amazonlinux2-aarch64-standard:1.0
Image version
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

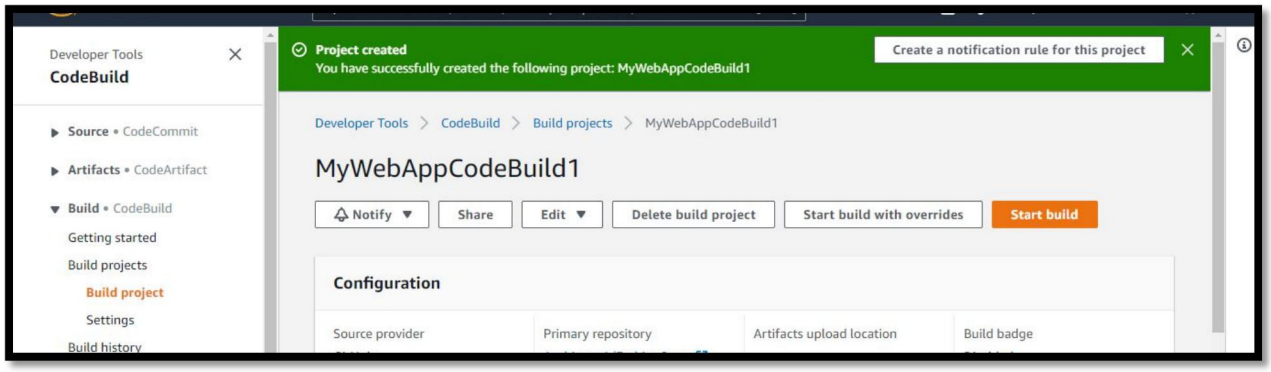

aws
Services
Search for services, features, marketplace products, and docs
[Alt+S]
Standard
Image
aws/codebuild/amazonlinux2-aarch64-standard:1.0
Image version
Always use the latest image for this runtime version
Privileged
Enable this flag if you want to build Docker images or want your builds to get elevated privileges
Service role
New service role
Create a service role in your account
Existing service role
Choose an existing service role from your account
Role name
codebuild-MyWebApp1-service-role
Type your service role name
Additional configuration
Timeout, certificate, VPC, compute type, environment variables, file systems
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.



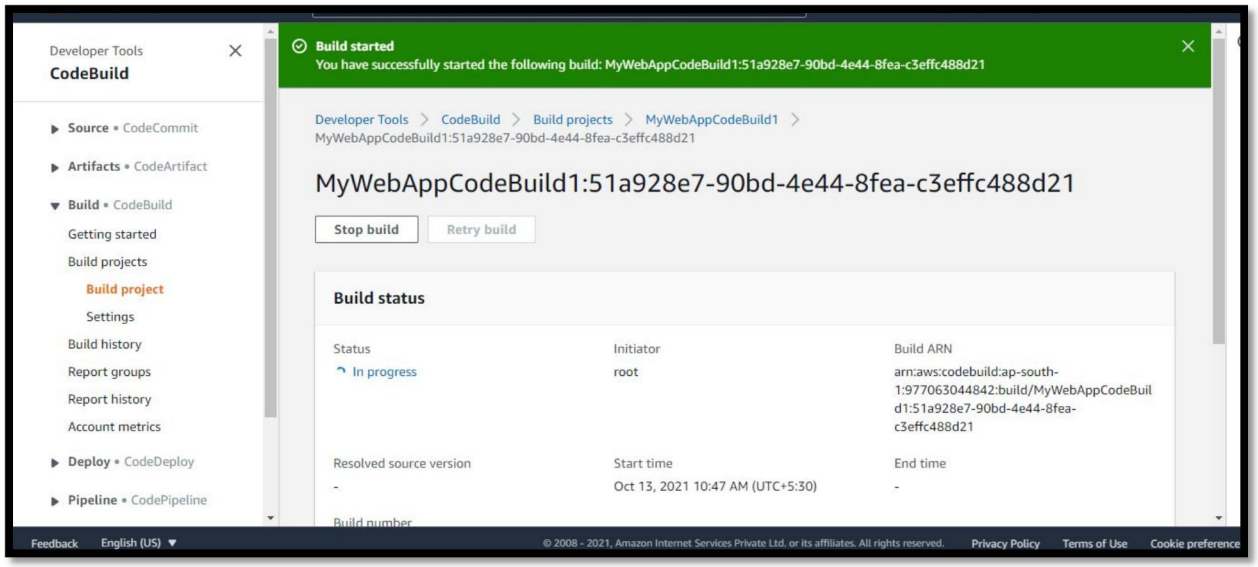


7. Click on Create build project





8. Click on Start build

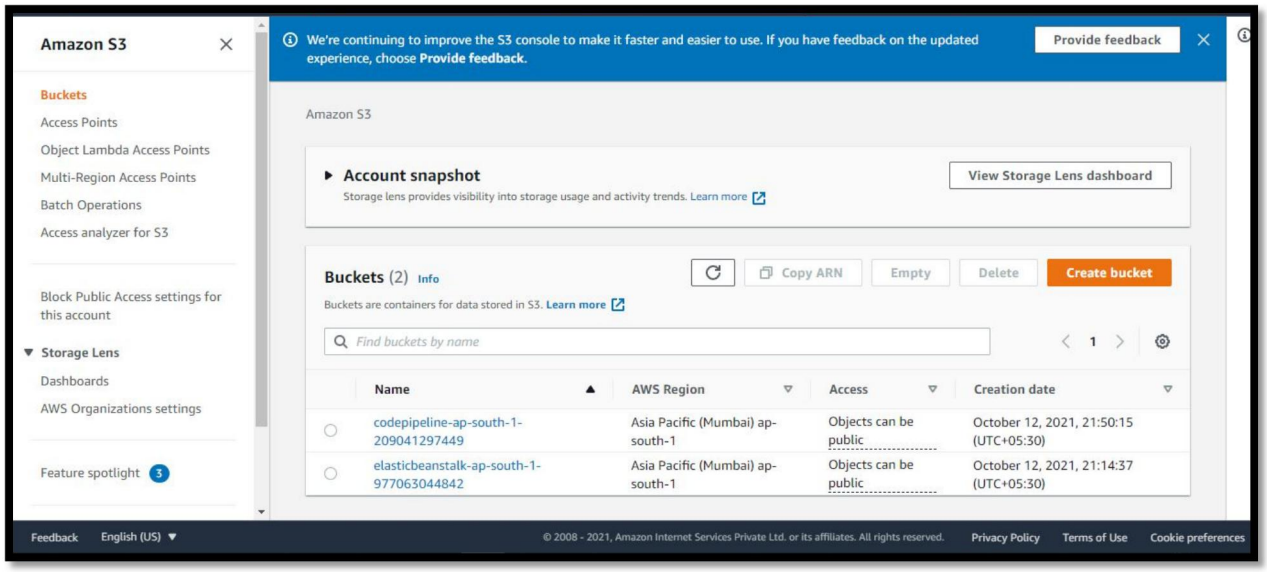


Deploy on S3 / SEBS using AWS CodePipeline

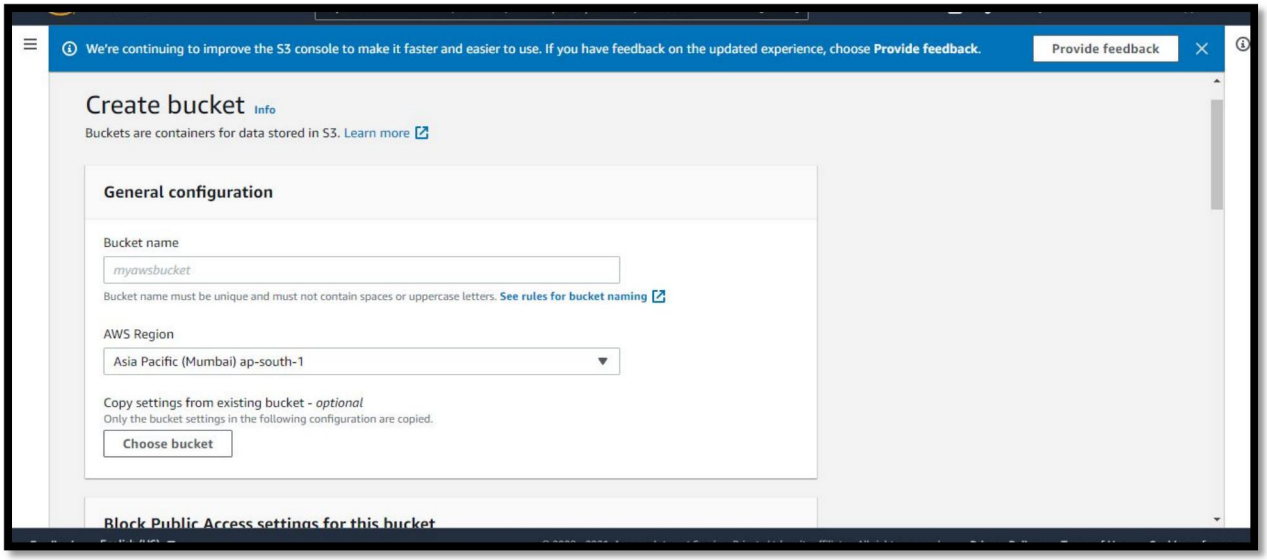
Create an S3 bucket for your application

To create an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

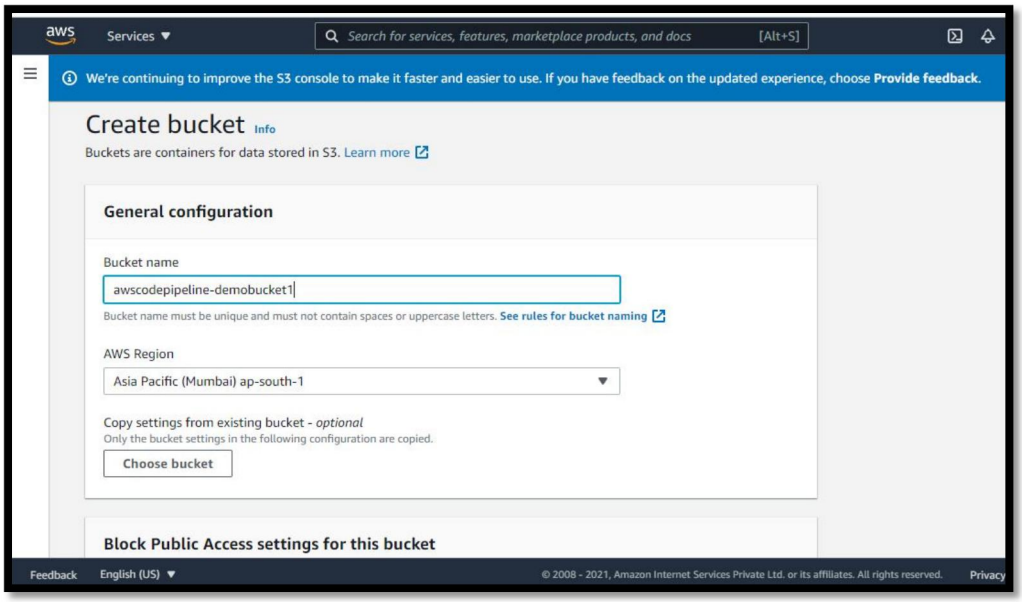


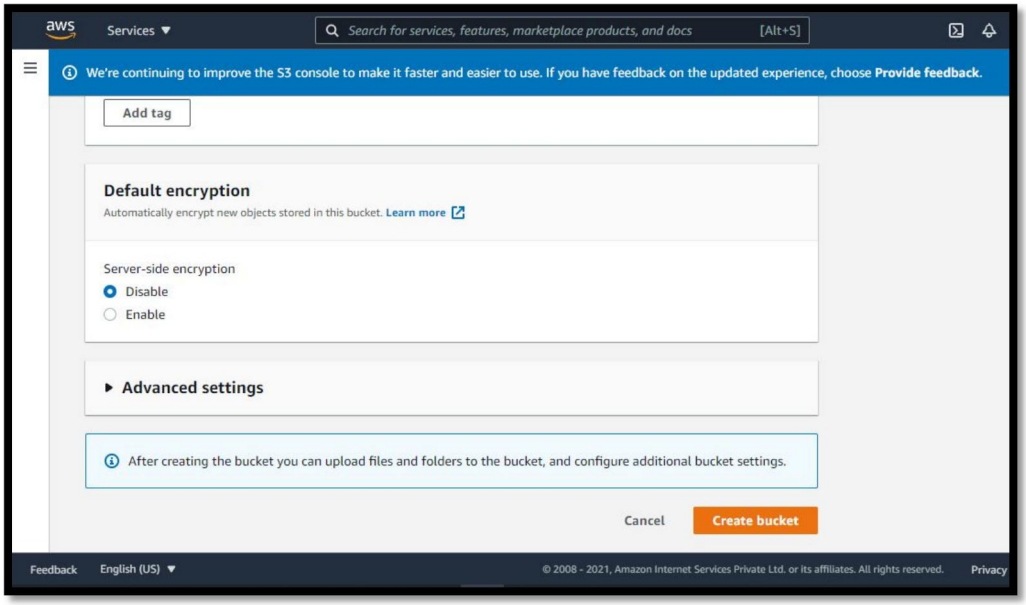
2. Choose Create bucket.



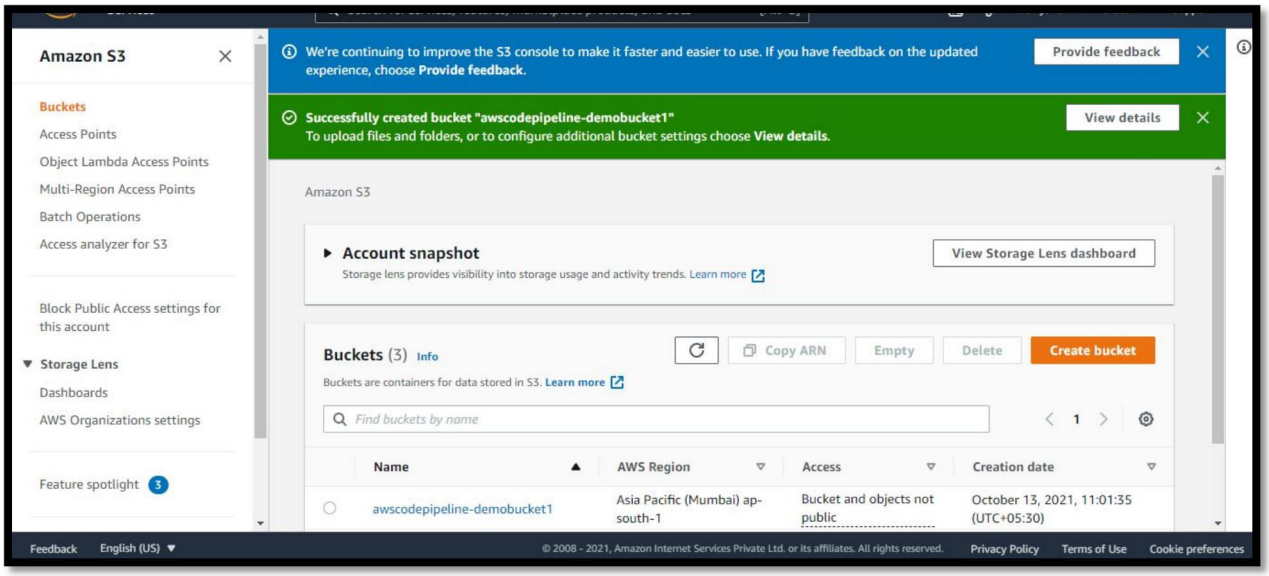
3. In Bucket name, enter a name for your bucket (for example, awscodepipeline-demobucket-example-date).

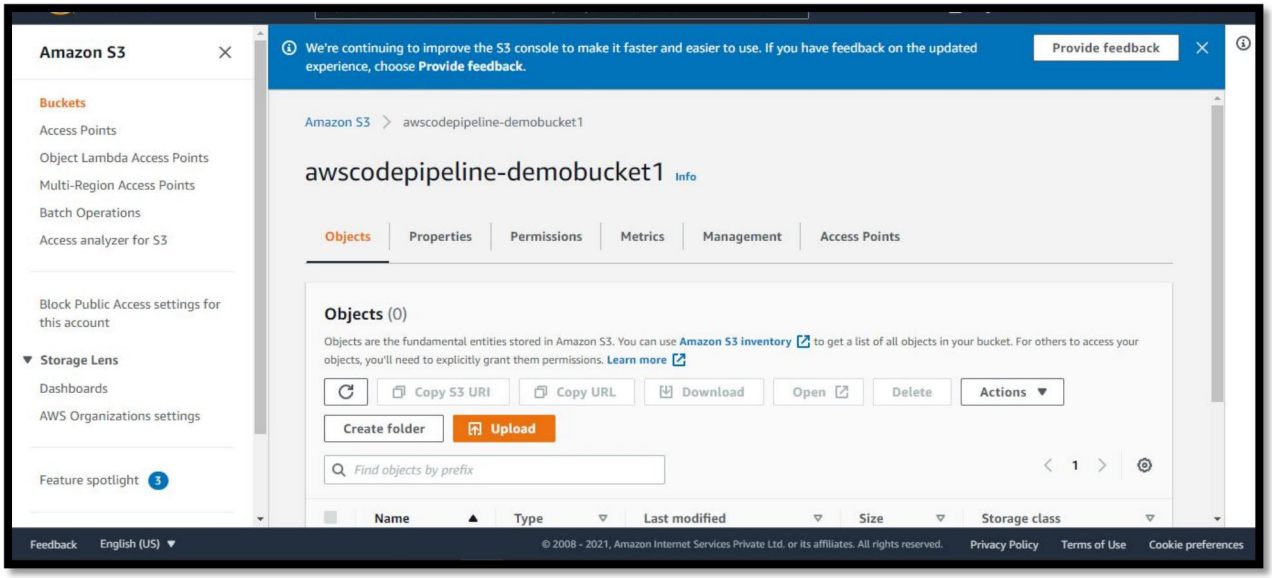
In Region, choose the Region where you intend to create your pipeline, such as Asia Pacific (Mumbai), and then choose Create bucket.





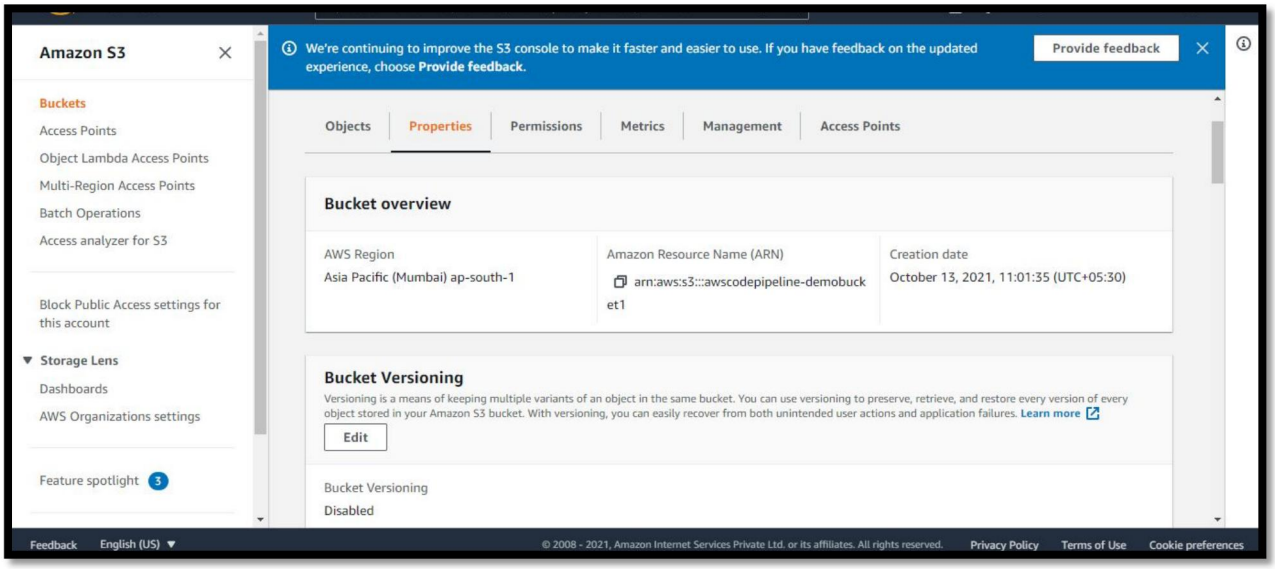
4. After the bucket is created, a success banner displays. Choose Go to bucket details.

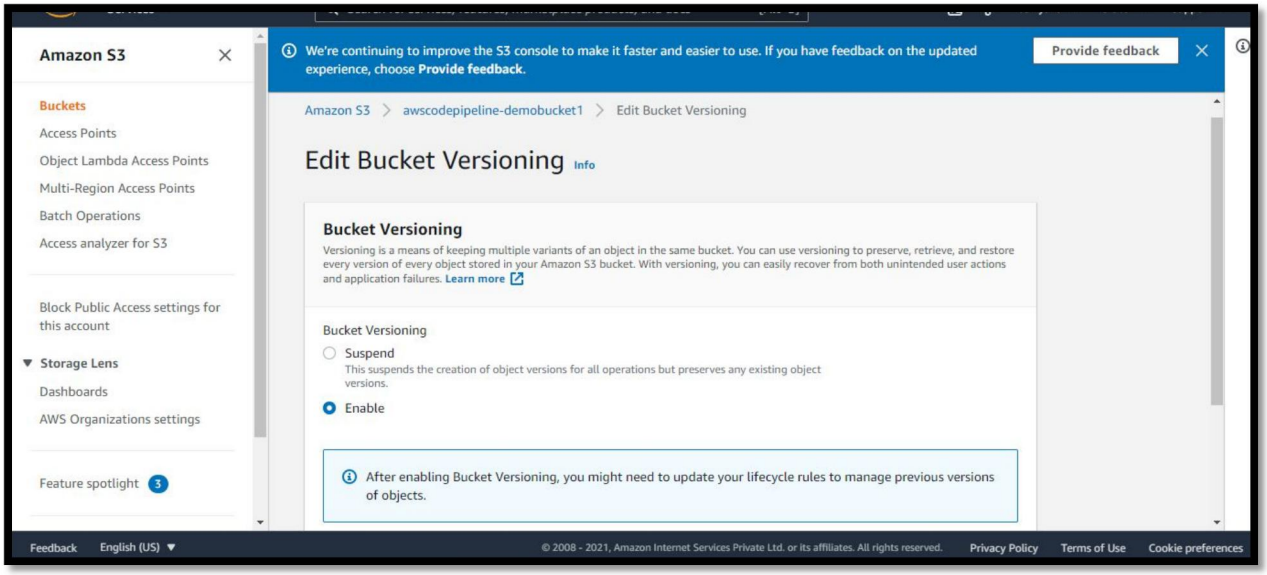




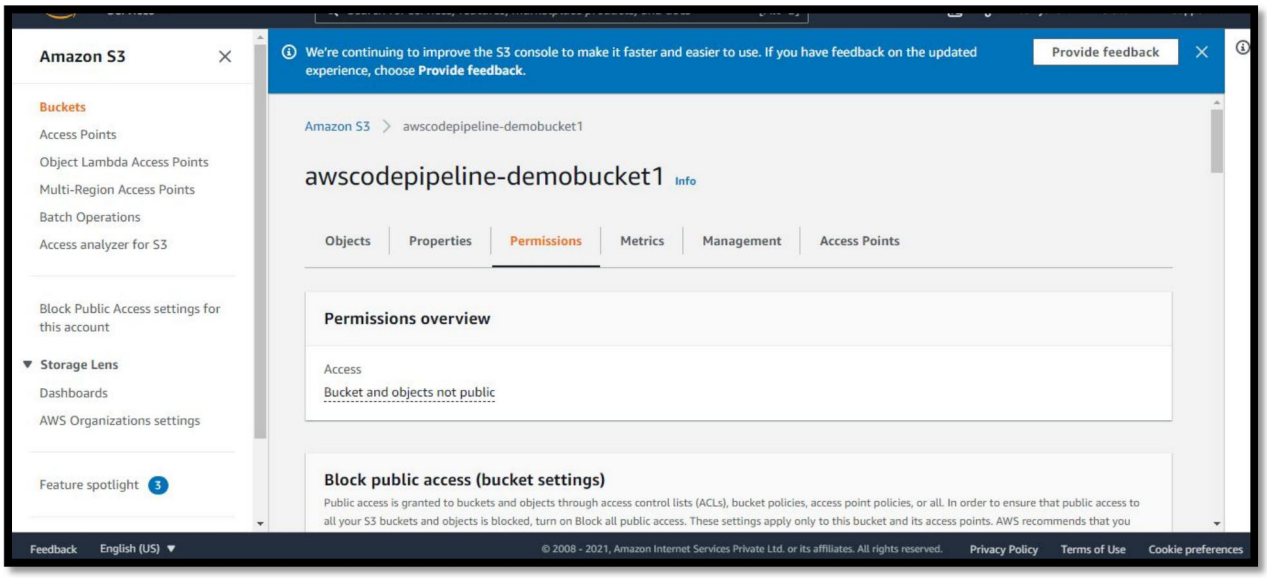
5. On the Properties tab, choose Versioning. Choose Enable versioning, and

then choose Save.
When versioning is enabled, Amazon S3 saves every version of every object in the bucket.





6. On the Permissions tab, leave the defaults. For more information about S3 bucket and object permissions, see Specifying Permissions in a Policy.



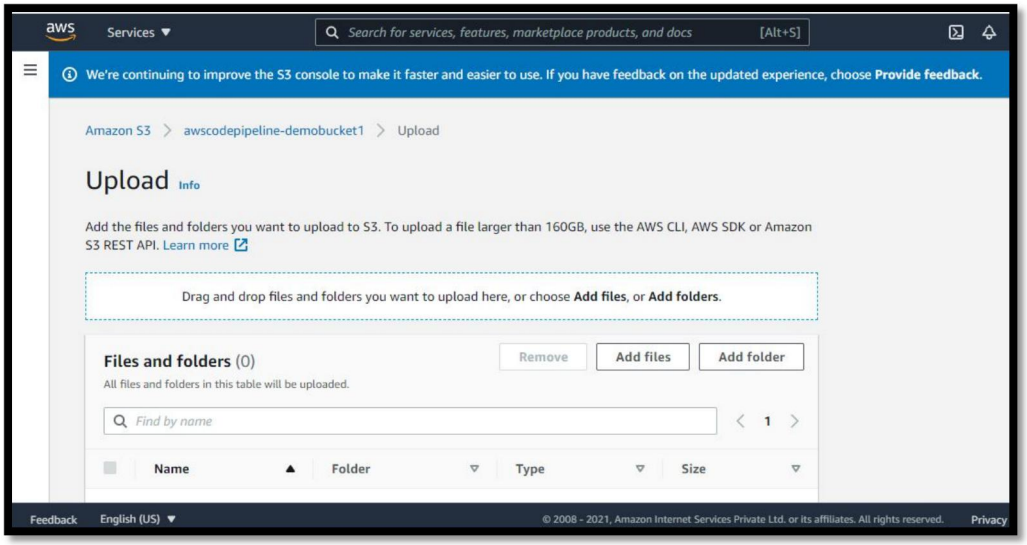
7. Next, download a sample and save it into a folder or directory on your local computer.
    a. Choose one of the following. Choose SampleApp_Windows.zip if you want to follow the steps in this tutorial for Windows Server instances.

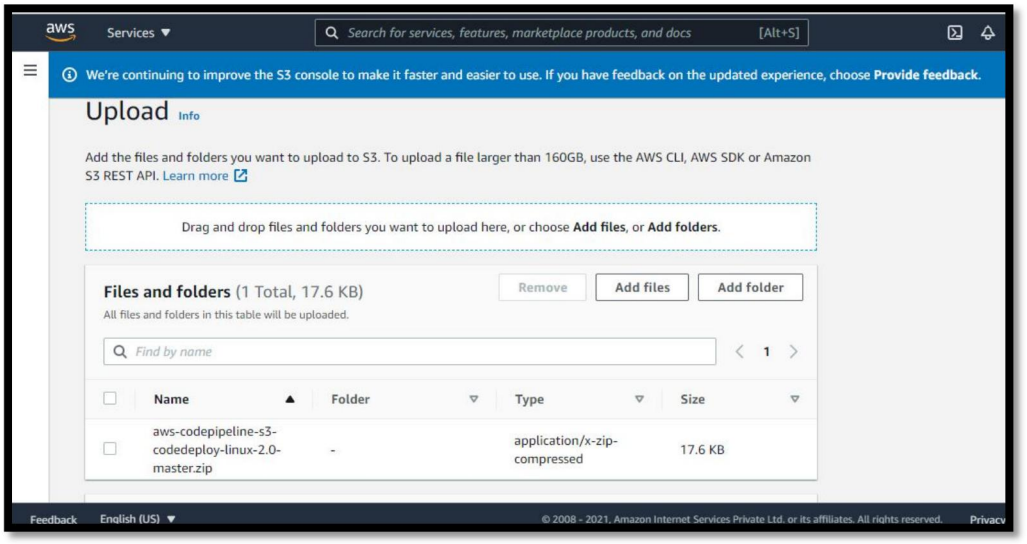
- If you want to deploy to Amazon Linux instances using CodeDeploy, download the sample application here: SampleApp_Linux.zip.
- If you want to deploy to Windows Server instances using CodeDeploy, download the sample application here: SampleApp_Windows.zip.
    b. Download the compressed (zipped) file. Do not unzip the file.

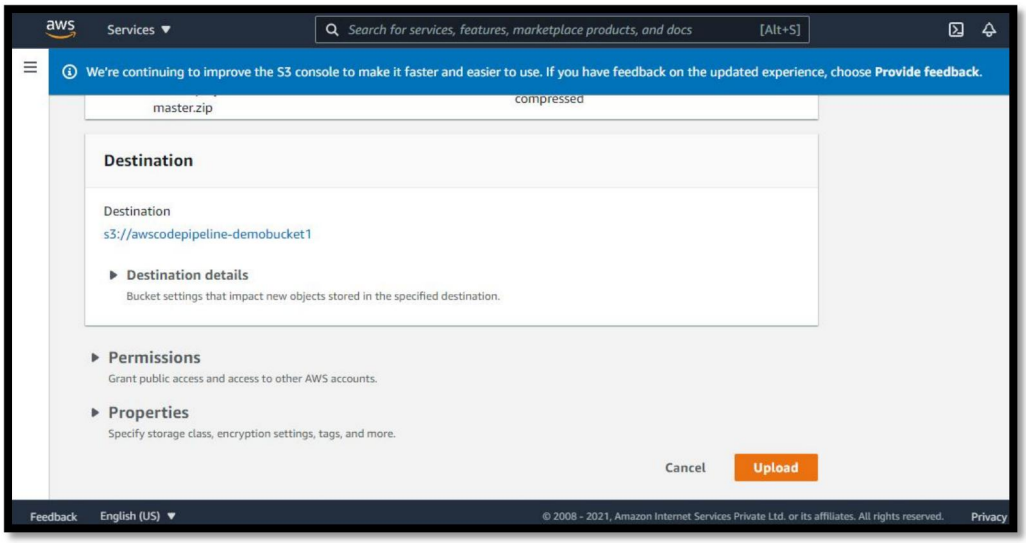
SampleApp_Windows.zip

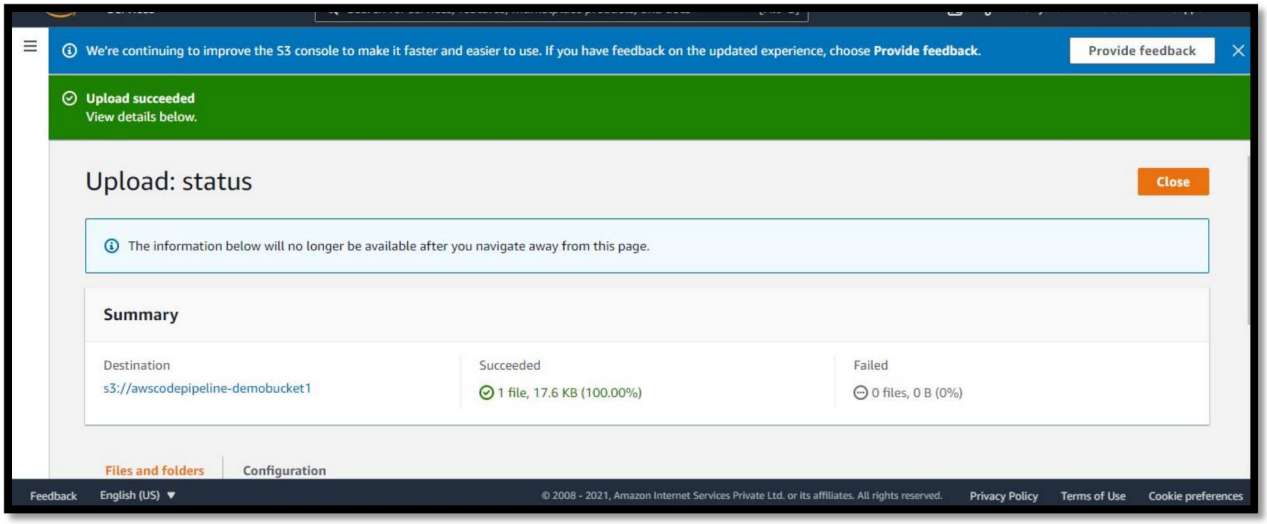
8. In the Amazon S3 console, for your bucket, upload the file:
    1. Choose **Upload**.

2. Drag and drop the file or choose **Add files** and browse for the file.
3. Choose **Upload**.





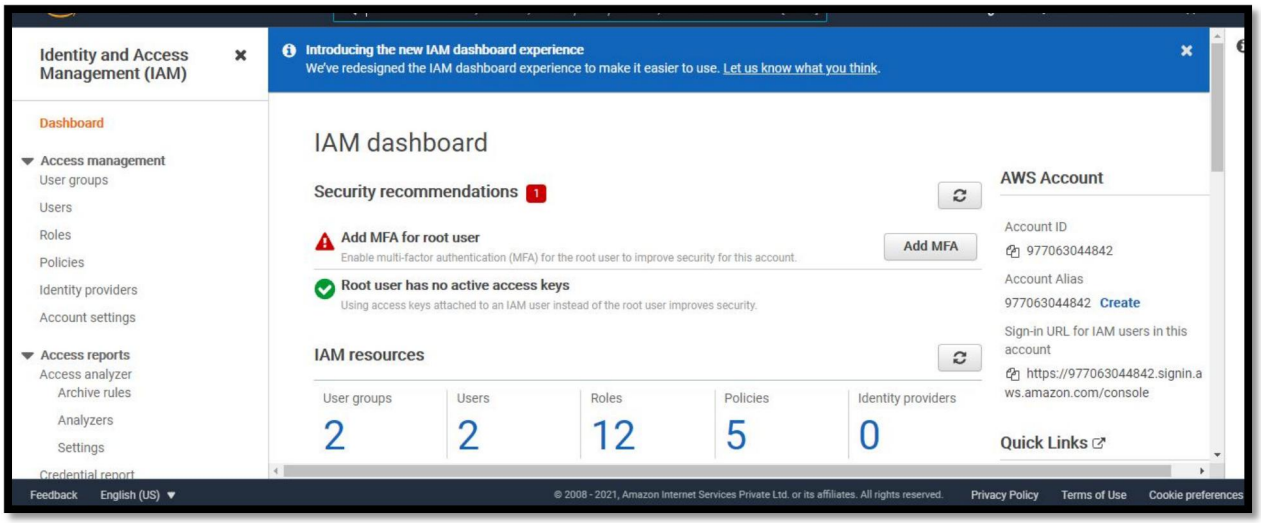




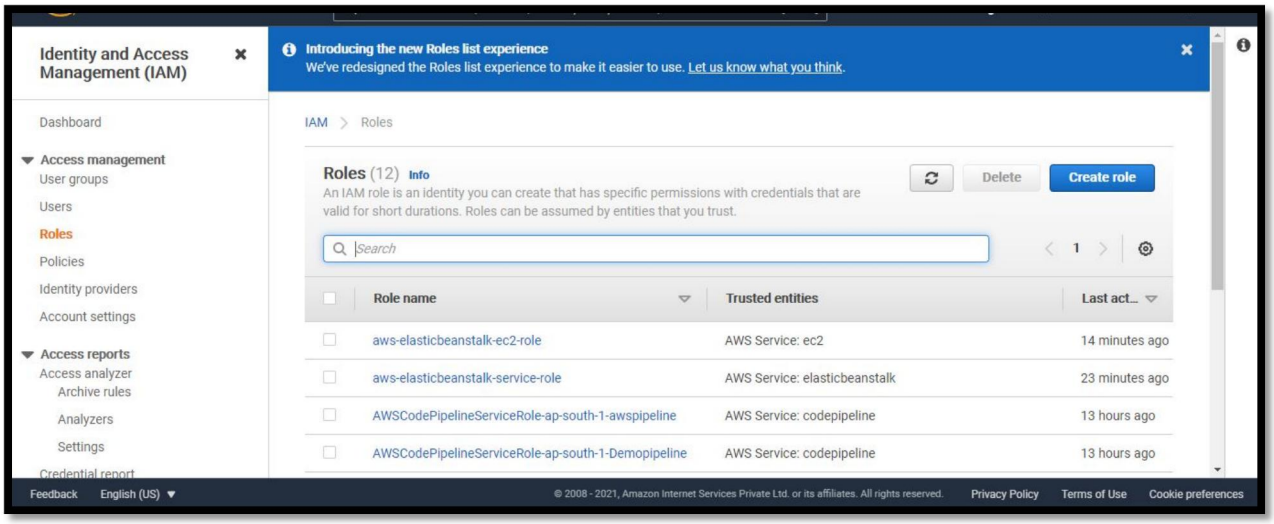
Deploy Sample Application on EC2 instance using AWS CodeDeploy

To create an instance role

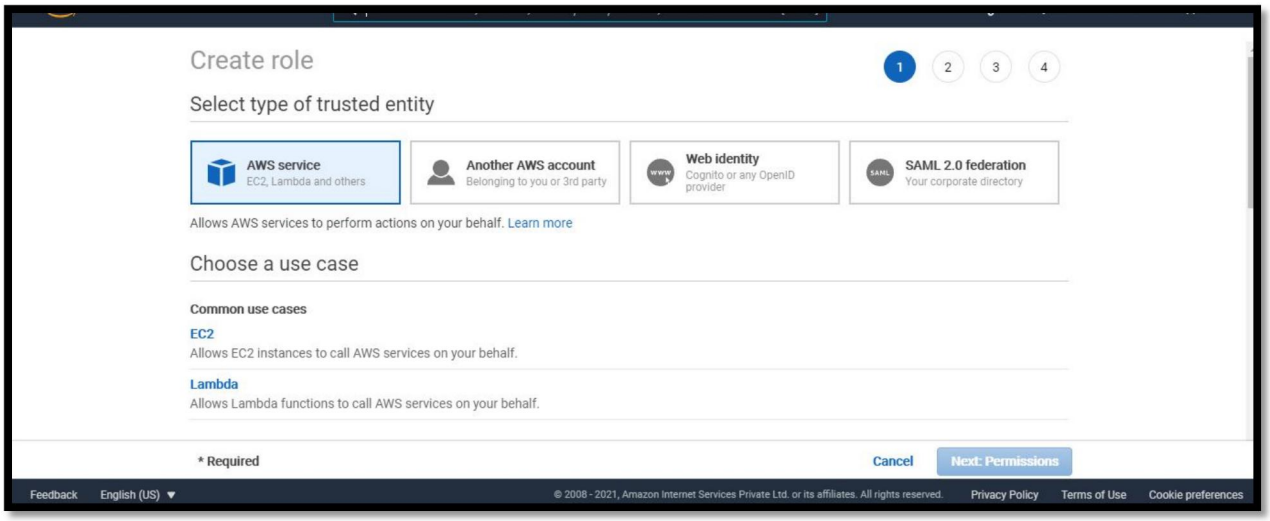
1. Open the IAM console at (https://console.aws.amazon.com/iam/).



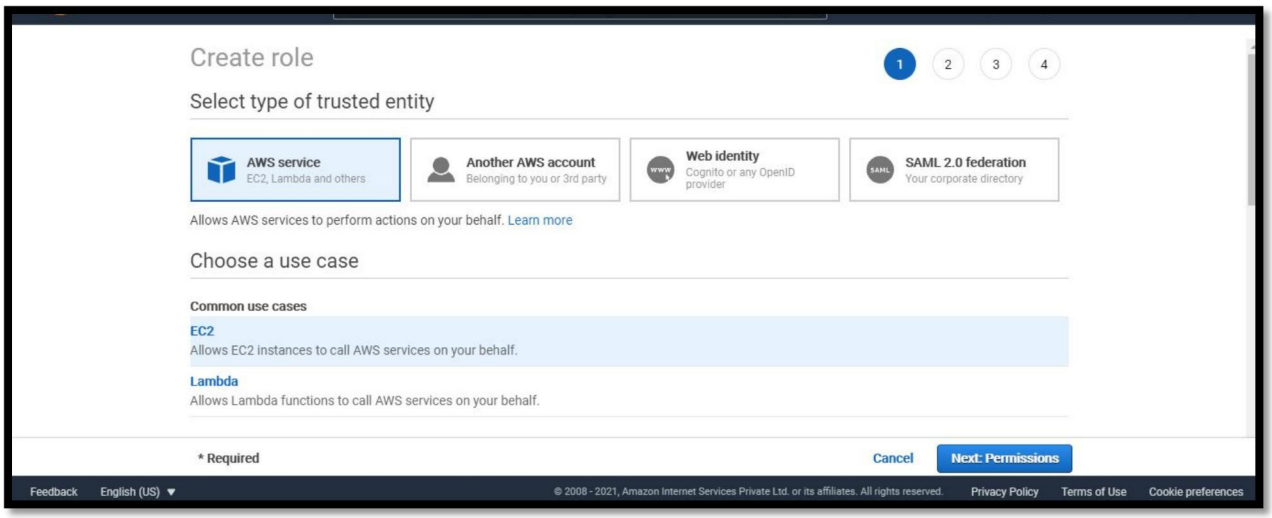
2. From the console dashboard, choose Roles.



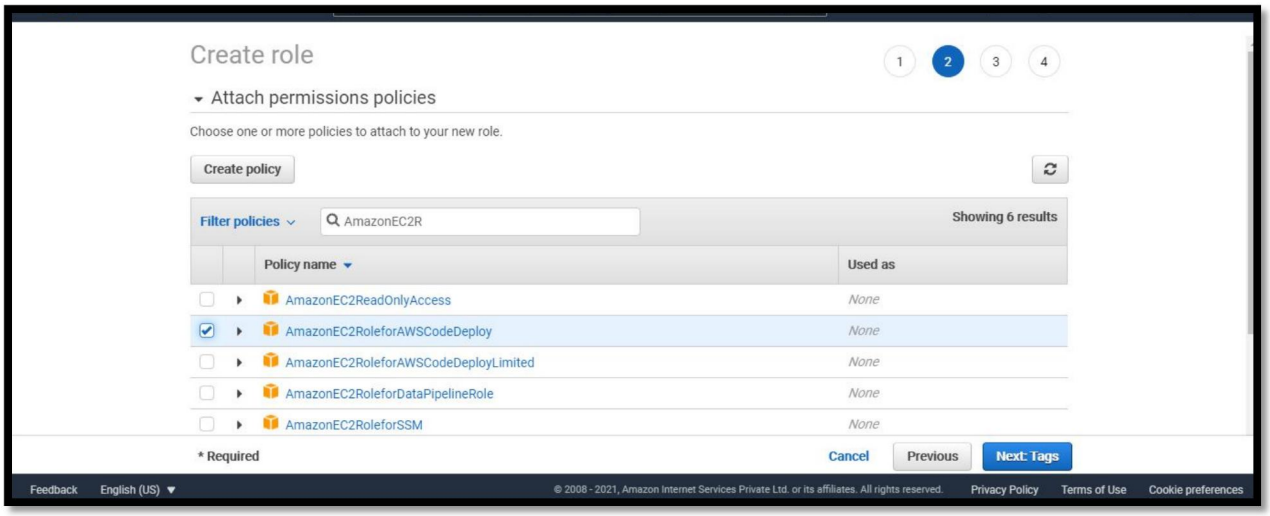
3. Choose Create role.



4. Under Select type of trusted entity, select AWS service. Under Choose a use case, select EC2, and then choose Next: Permissions.

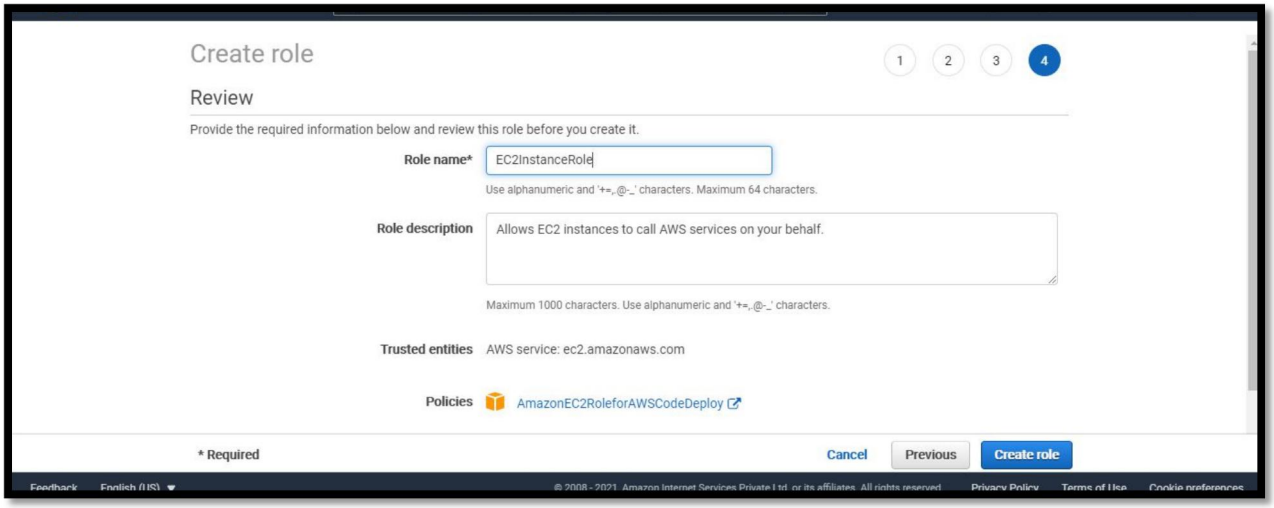


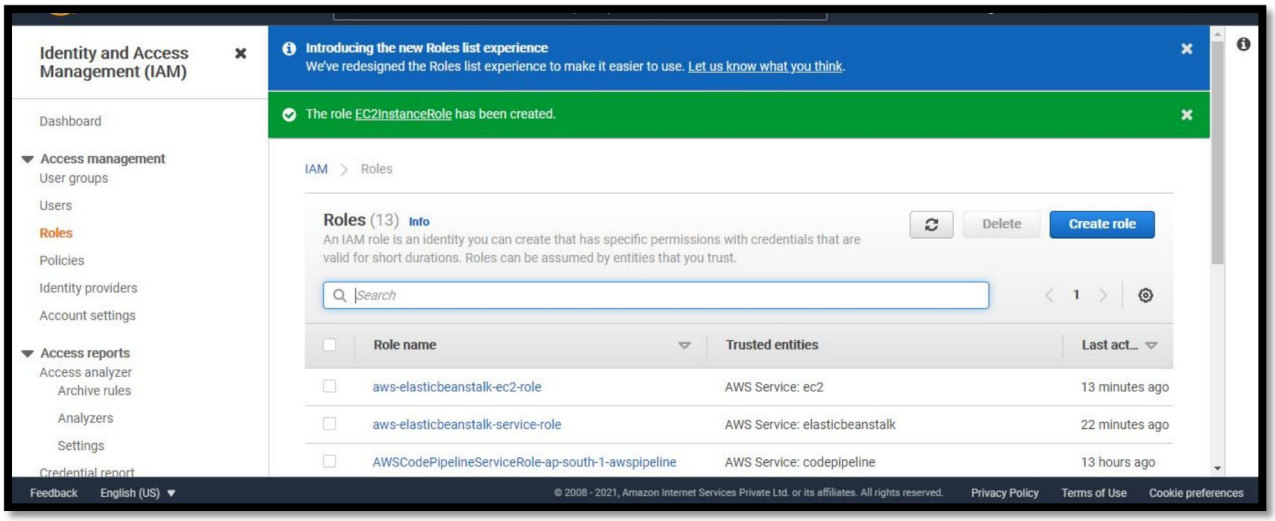
5. Search for and select the policy named AmazonEC2RoleforAWSCodeDeploy, and then choose Next: Tags.



6. Choose Next: Review. Enter a name for the role (for example, EC2InstanceRole).

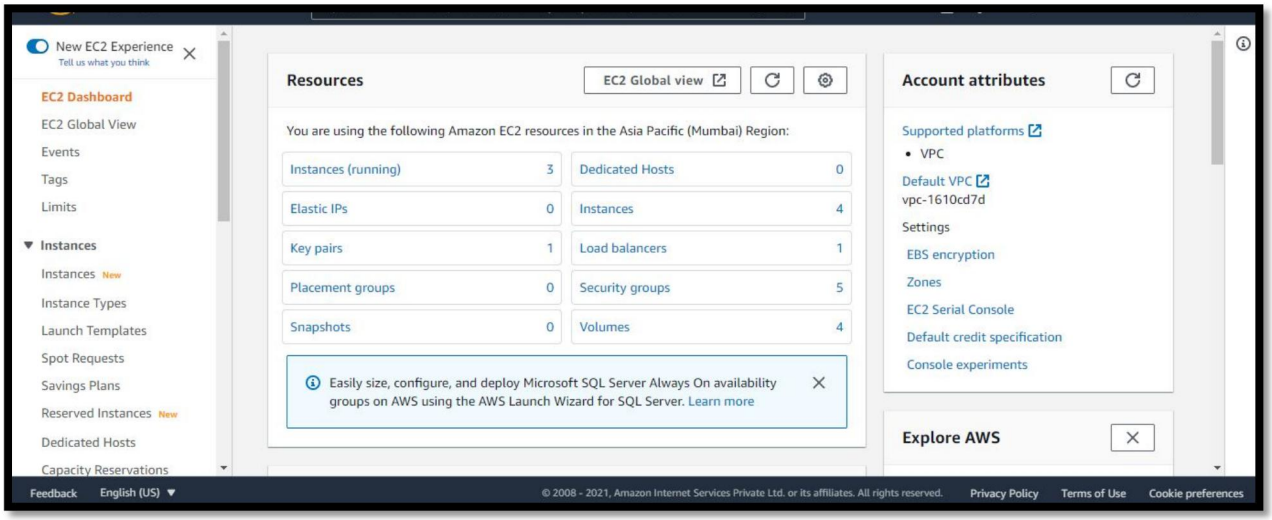
Choose Create role.



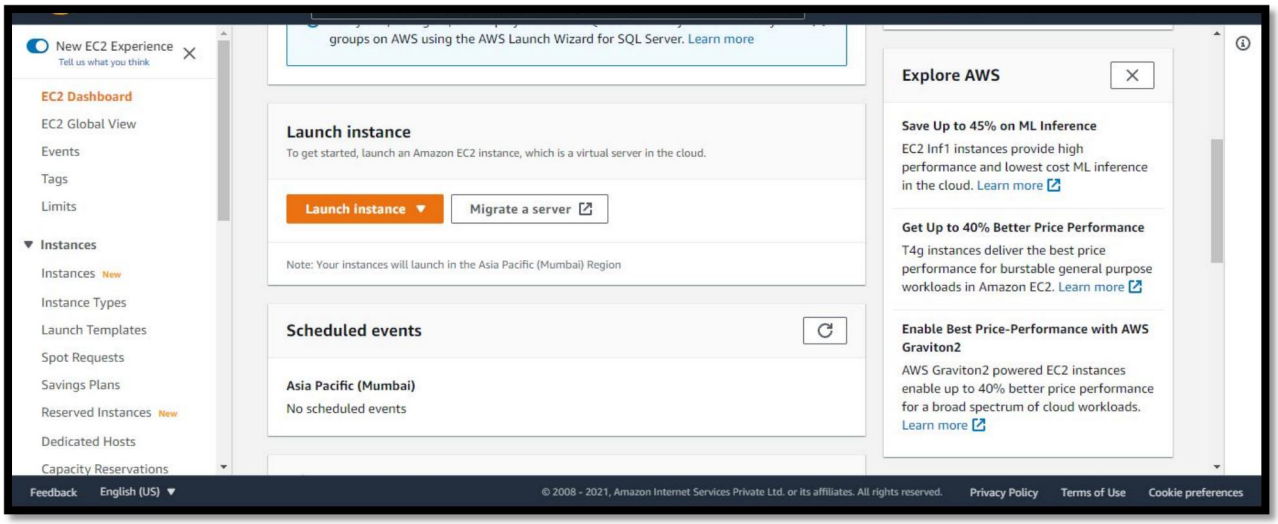


To launch instances

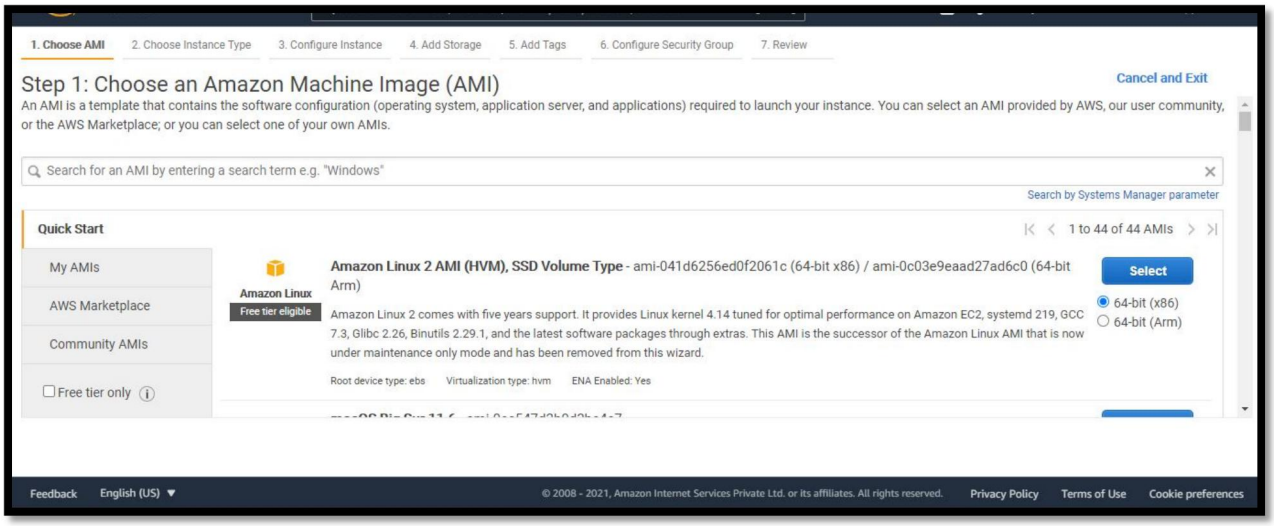
1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

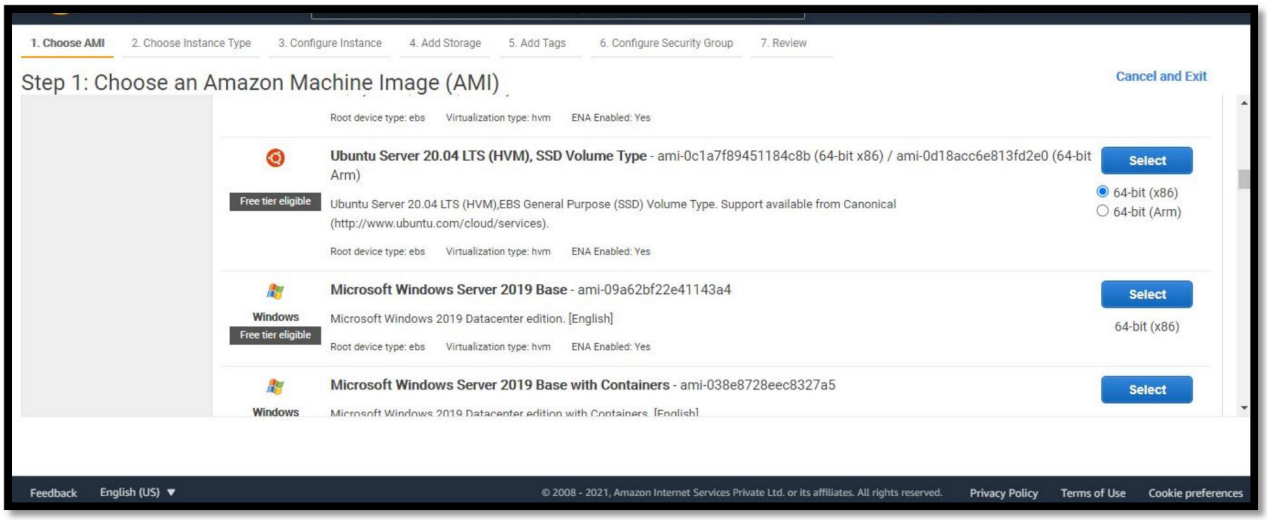


2. From the console dashboard, choose Launch instance, and select Launch instance from the options that pop up.

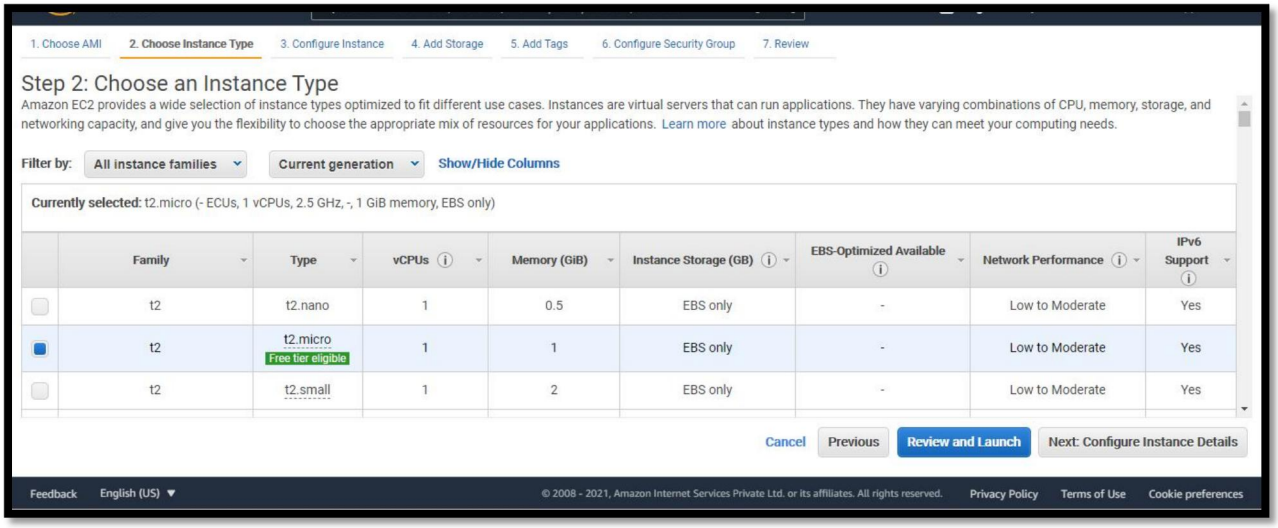


3. On the Step 1: Choose an Amazon Machine Image (AMI) page, locate the Microsoft Windows Server 2019 Base option, and then choose Select. (This AMI is labeled "Free tier eligible" and can be found at the top of the list.)





4. On the Step 2: Choose an Instance Type page, choose the free tier eligible t2.micro type as the hardware configuration for your instance, and then choose Next: Configure Instance Details.

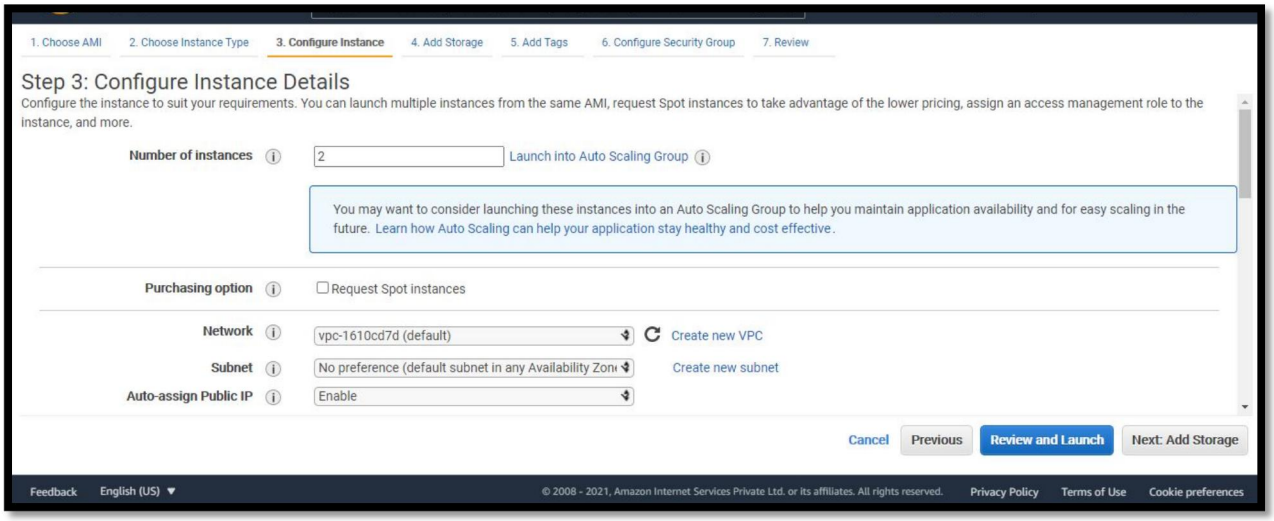


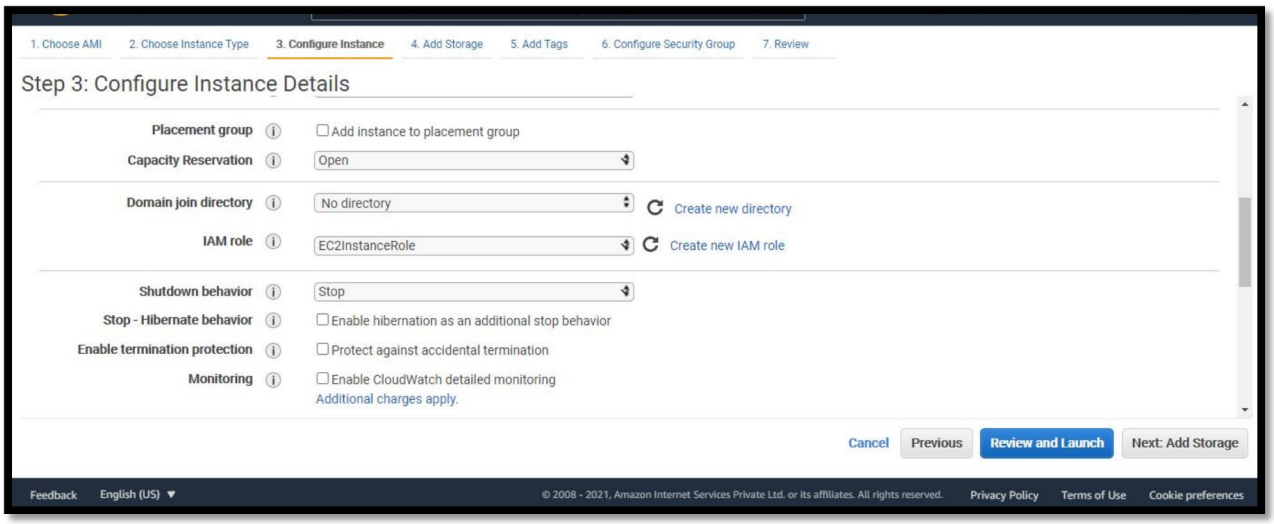
5. On the Step 3: Configure Instance Details page, do the following:
   - In Number of instances, enter 2.
   - In Auto-assign Public IP, choose Enable.
   - In IAM role, choose the IAM role you created in the previous procedure (for example, EC2InstanceRole).
   - Expand Advanced Details, and in User data, with As text selected, enter the following:

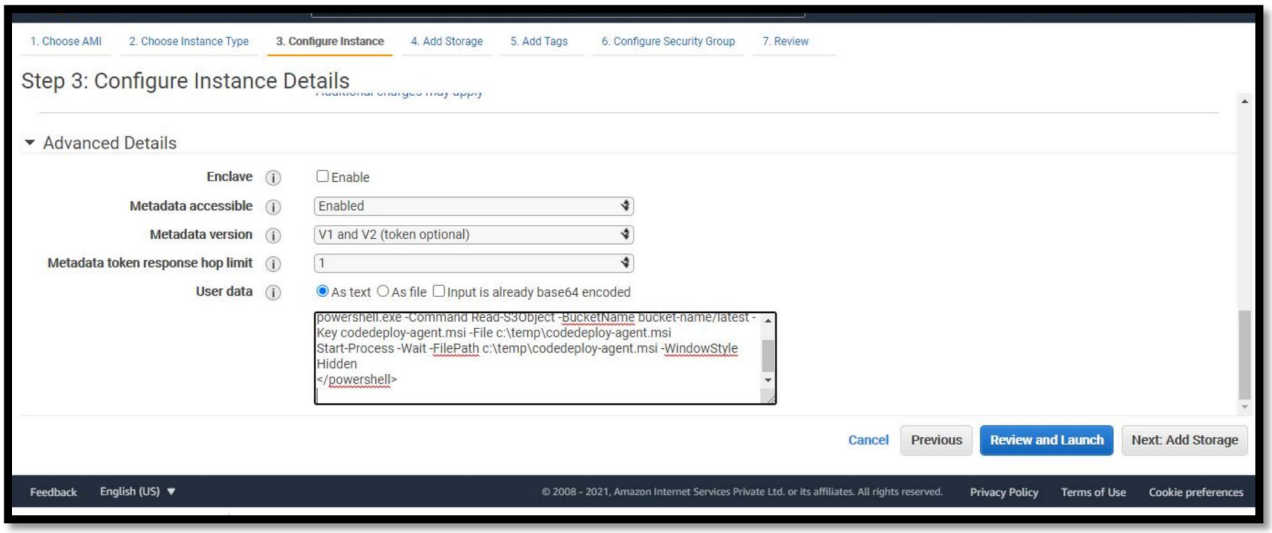
```
<powershell>
New-Item -Path c:\temp -ItemType "directory" -Force
powershell.exe -Command Read-S3Object -BucketName bucket-name/latest -Key codedeploy-agent.msi -File c:\temp\codedeploy-agent.msi
Start-Process -Wait -FilePath c:\temp\codedeploy-agent.msi -WindowStyle Hidden
</powershell>
```

   - *bucket-name* is the name of the S3 bucket that contains the CodeDeploy Resource Kit files for your Region. For example, for the US West (Oregon) Region, replace *bucket-name* with aws-codedeploy-us-west-2. For a list of bucket names, see Resource Kit Bucket Names by Region.
     This code installs the CodeDeploy agent on your instance as it is created. This script is written for Windows instances only.

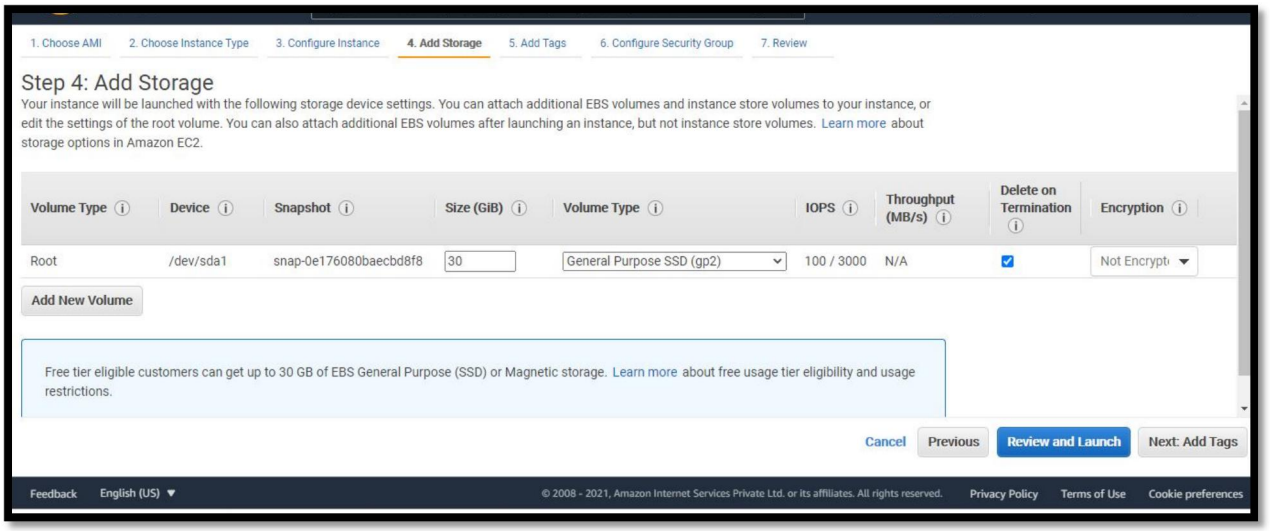
- Leave the rest of the items on the Step 3: Configure Instance Details page unchanged. Choose Next: Add Storage.



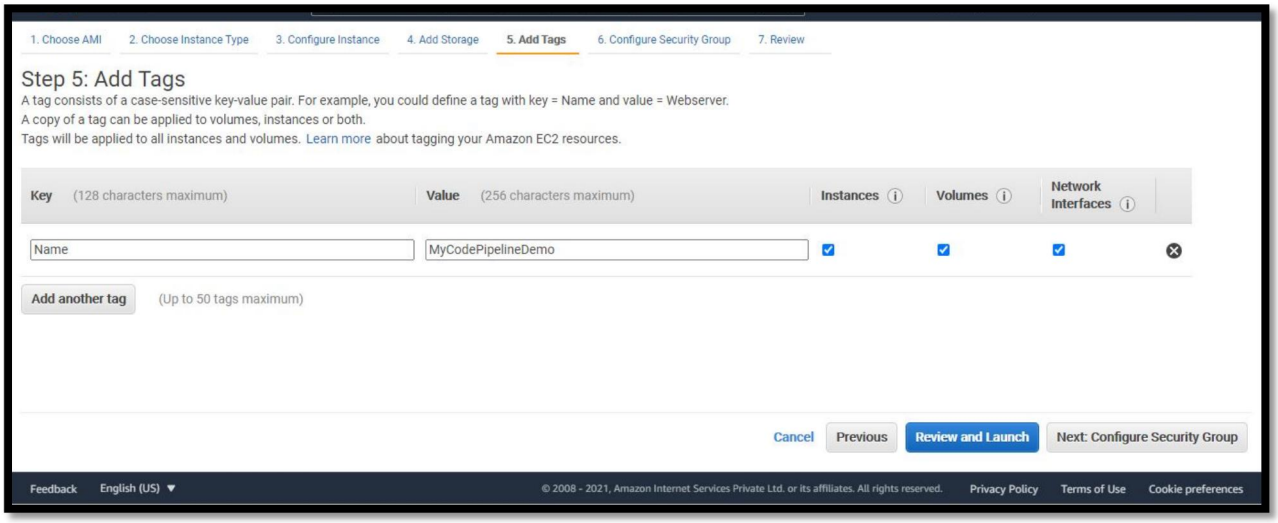




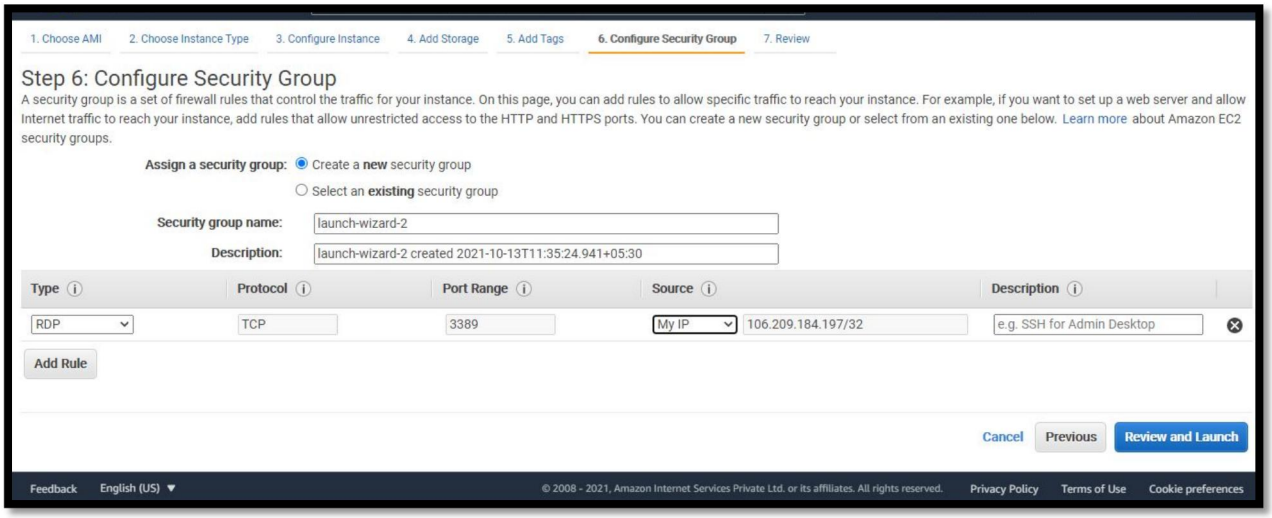
6. Leave the Step 4: Add Storage page unchanged, and then choose Next: Add Tags.



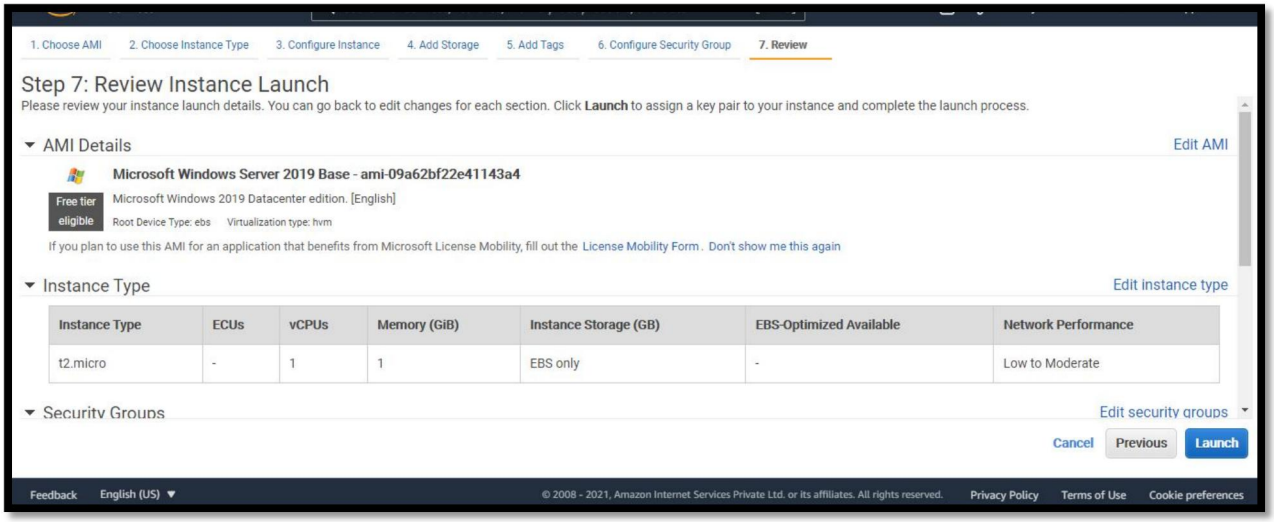
7. On the Add Tags page, choose Add Tag. Enter Name in the Key field, enter MyCodePipelineDemo in the Value field, and then choose Next: Configure Security Group.



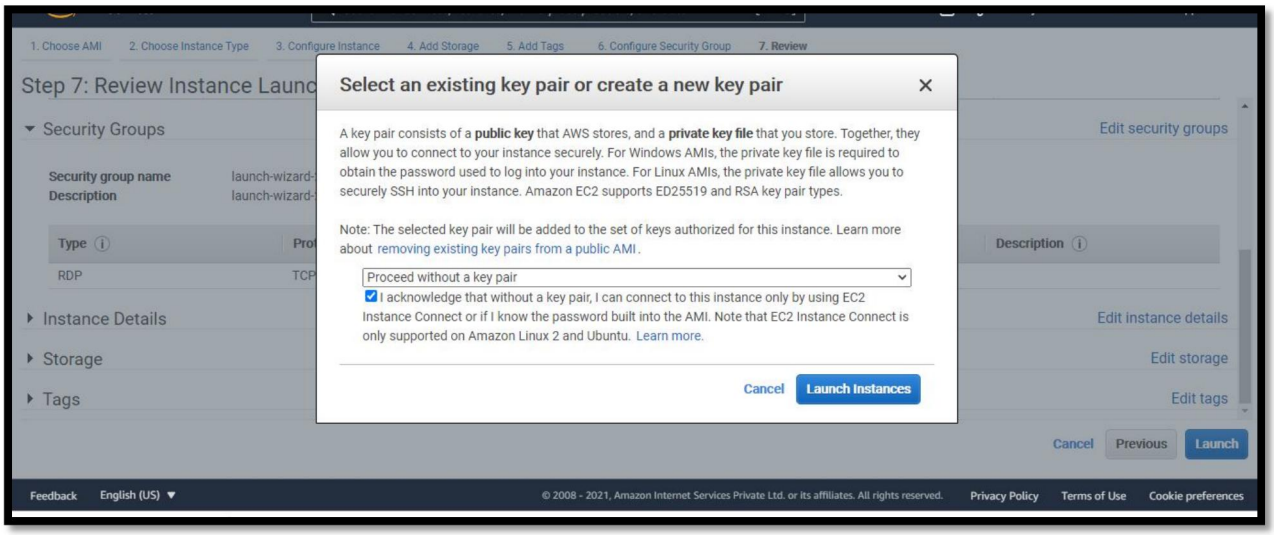
8. On the Configure Security Group page, allow port 80 communication so you can access the public instance endpoint.

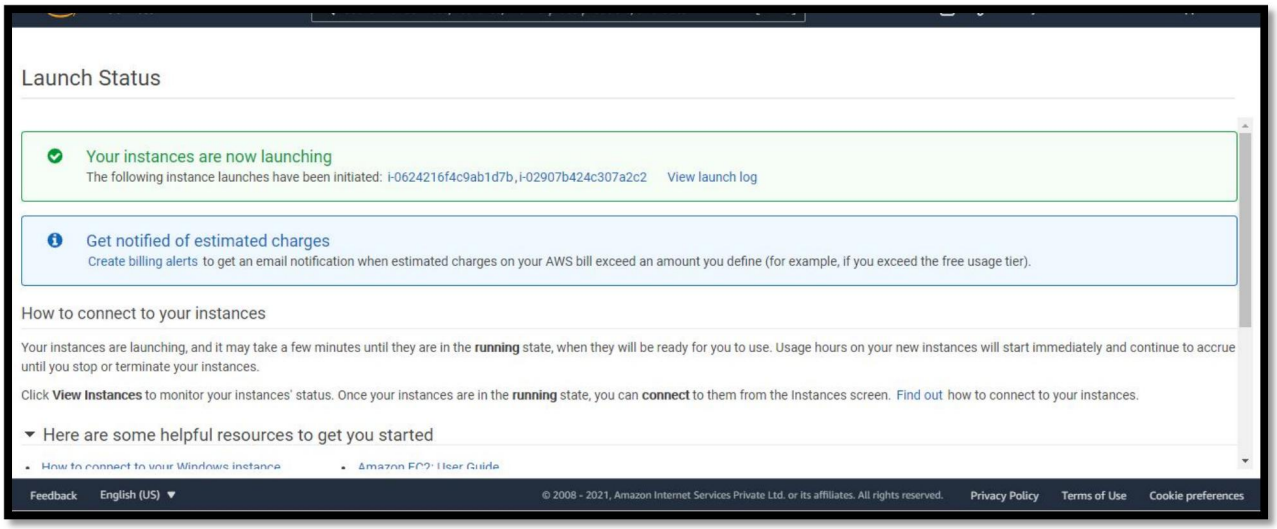


9. Choose Review and Launch.

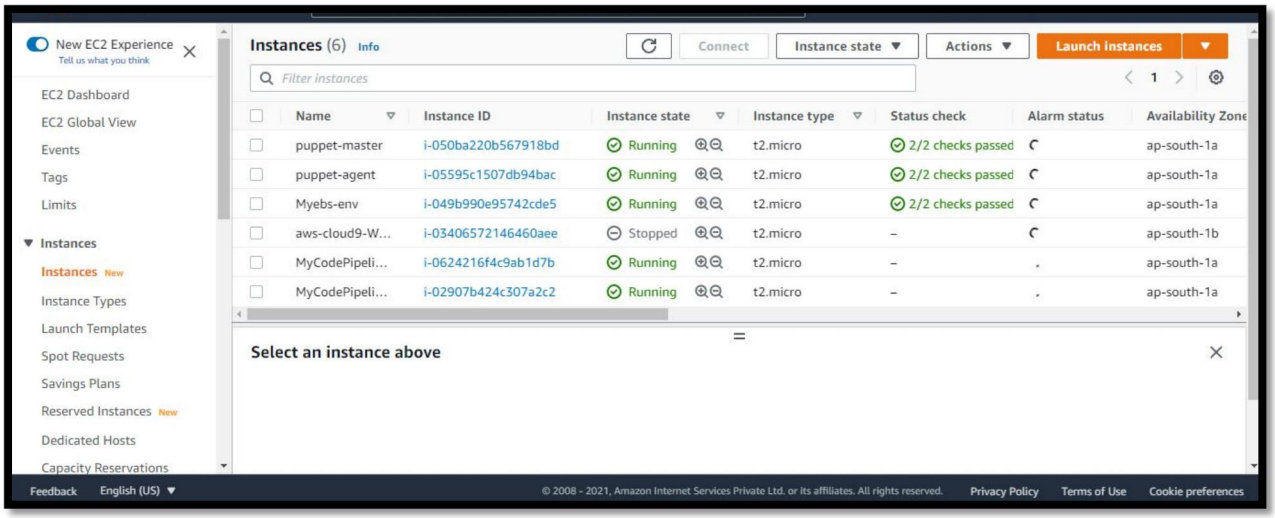


10.On the Review Instance Launch page, choose Launch. When prompted for a key pair, choose Proceed without a key pair.
When you are ready, select the acknowledgment check box, and then choose Launch Instances.

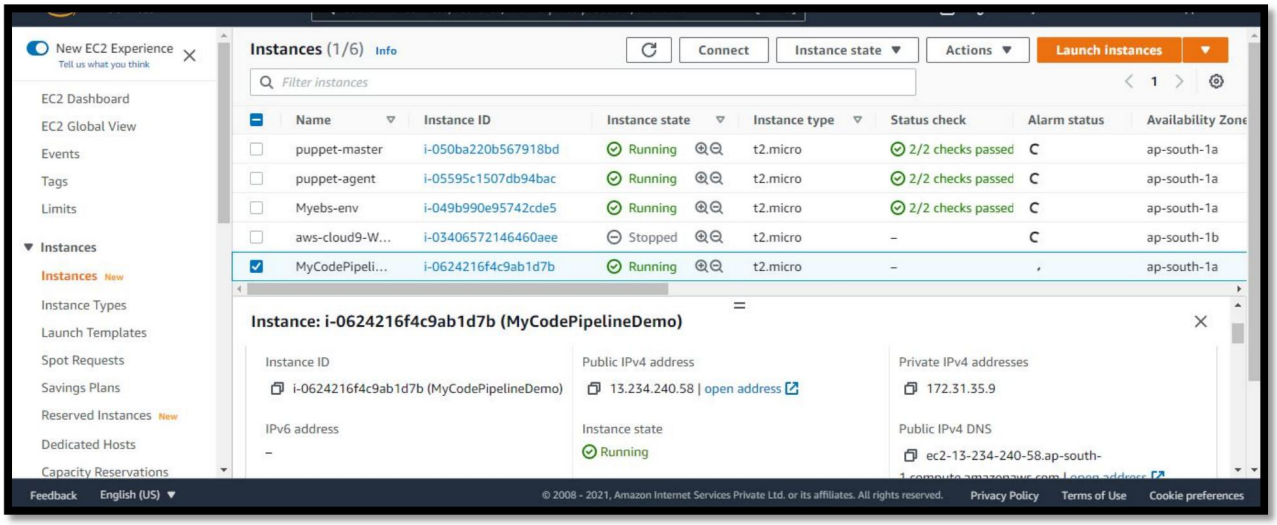


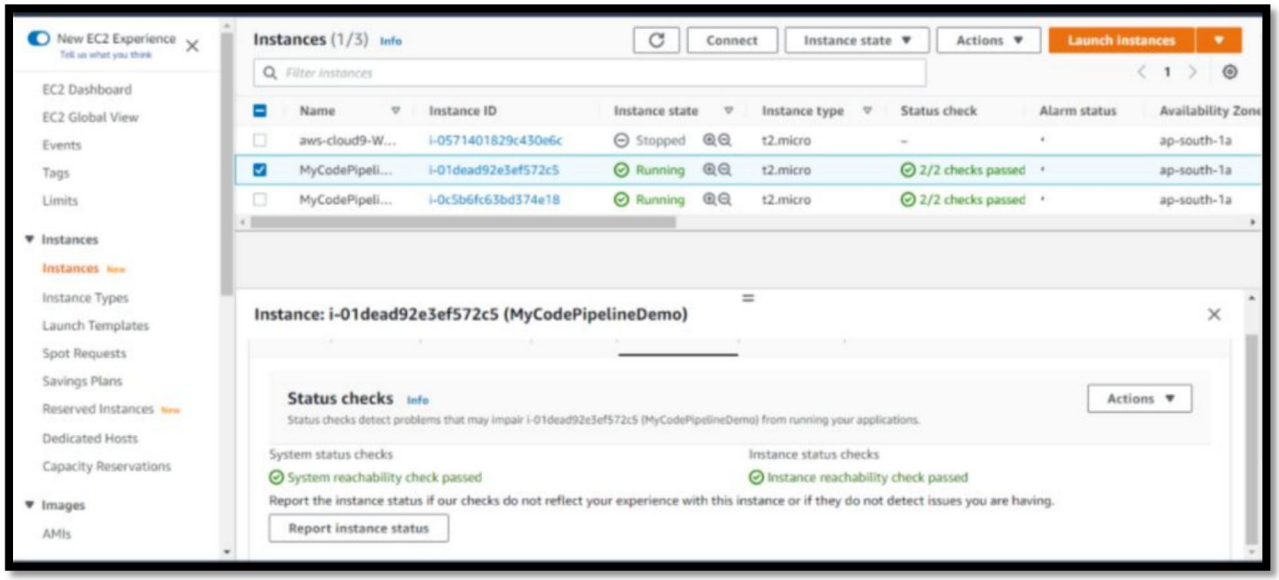


11. Choose View Instances to close the confirmation page and return to the console.
12. You can view the status of the launch on the Instances page. When you launch an instance, its initial state is pending. After the instance starts, its state changes to running, and it receives a public DNS name. (If the Public DNS column is not displayed, choose the Show/Hide icon, and then select Public DNS.)



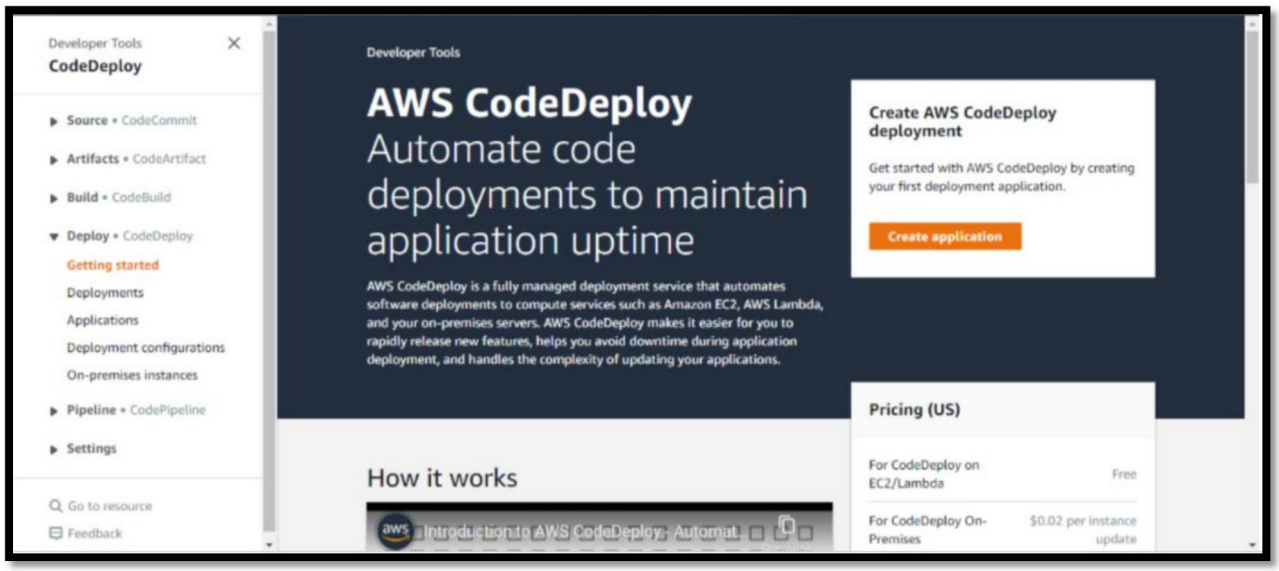
13. It can take a few minutes for the instance to be ready for you to connect to it. Check that your instance has passed its status checks. You can view this information in the Status Checks column.



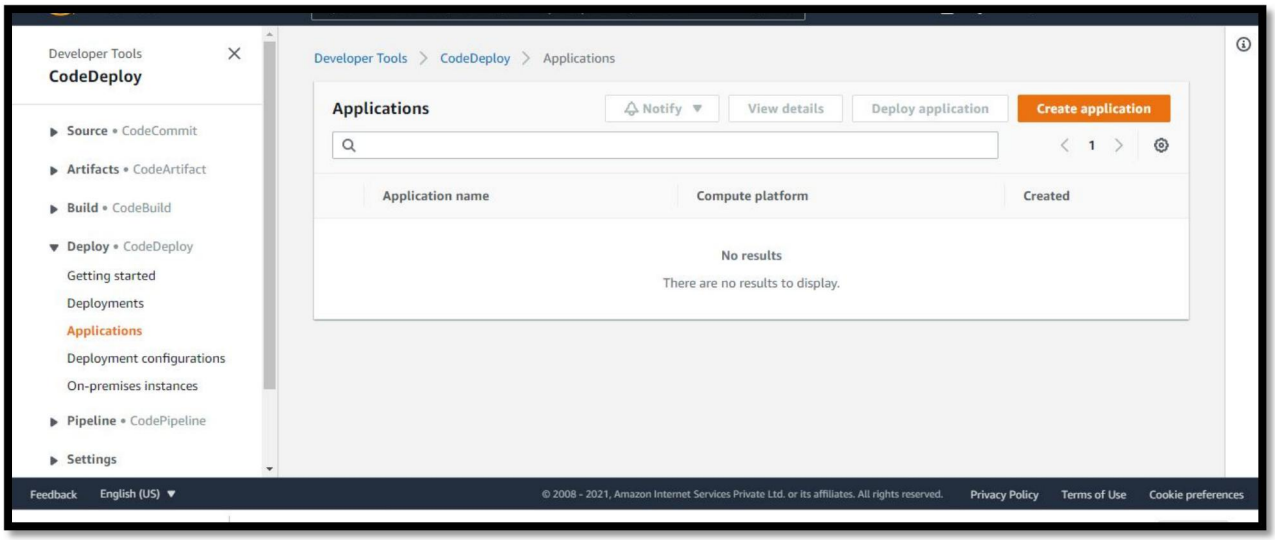


To create an application in CodeDeploy

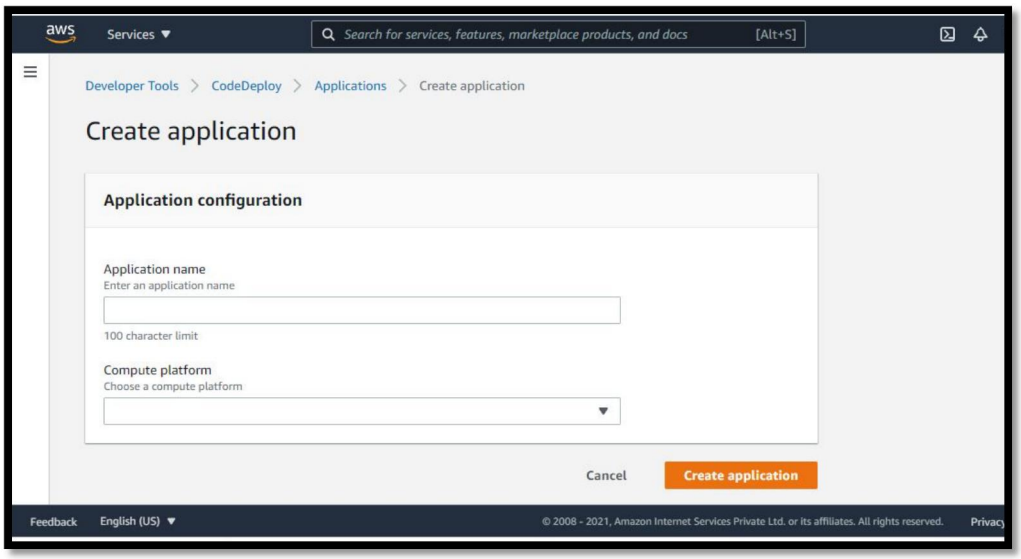
1. Open the CodeDeploy console at https://console.aws.amazon.com/codedeploy.



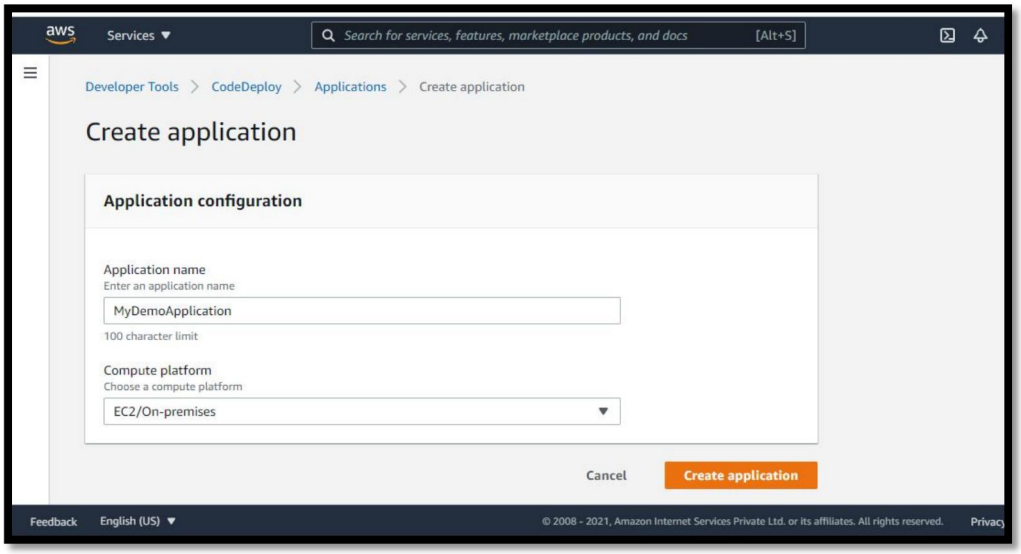
2. If the Applications page does not appear, on the AWS CodeDeploy menu, choose Applications.



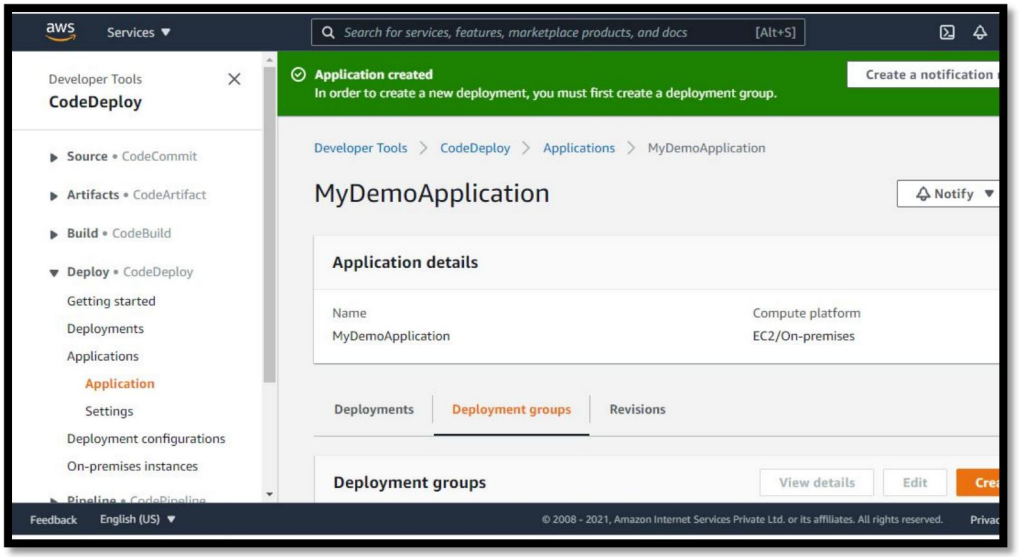
3. Choose Create application.



4. In Application name, enter MyDemoApplication.
5. In Compute Platform, choose EC2/On-premises.

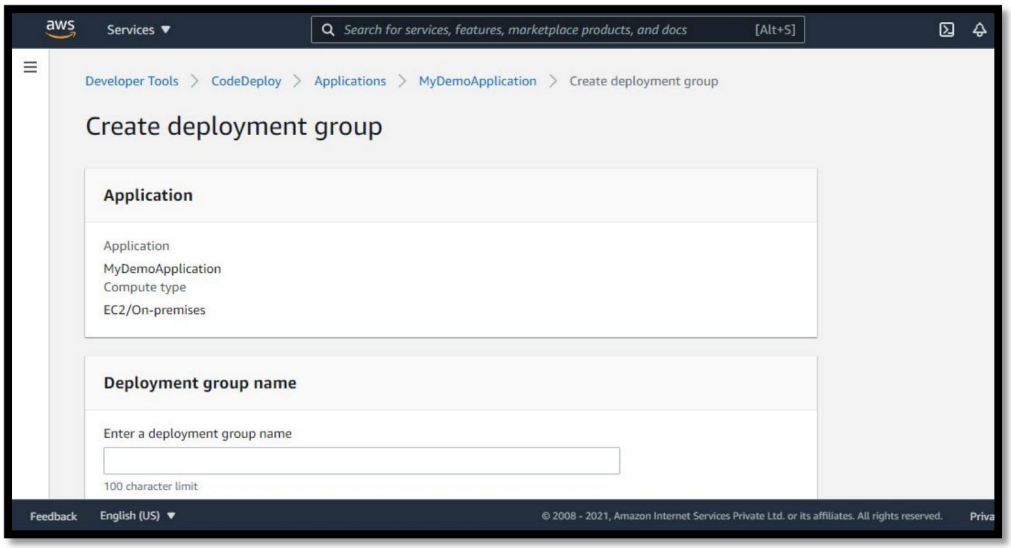


6. Choose Create application.

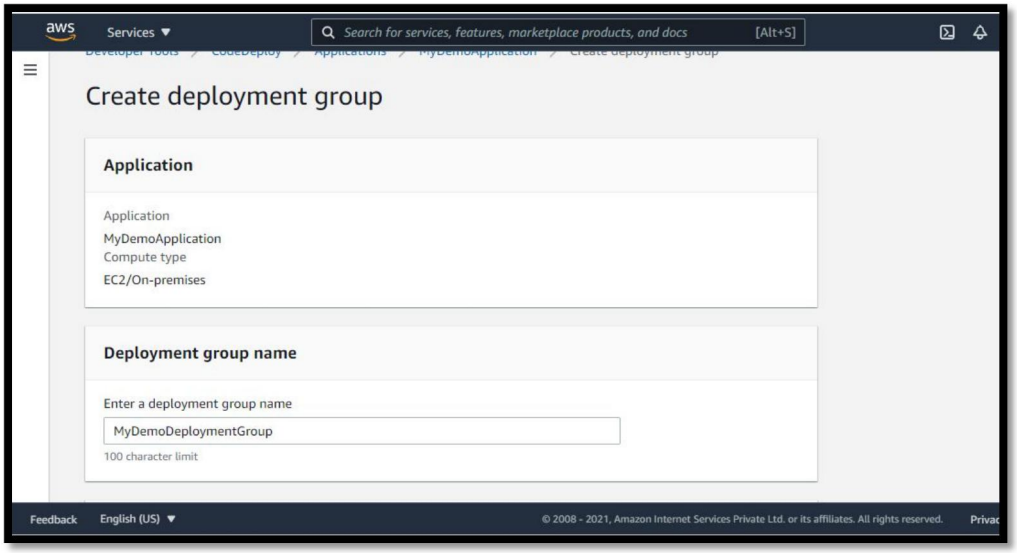


To create a deployment group in CodeDeploy

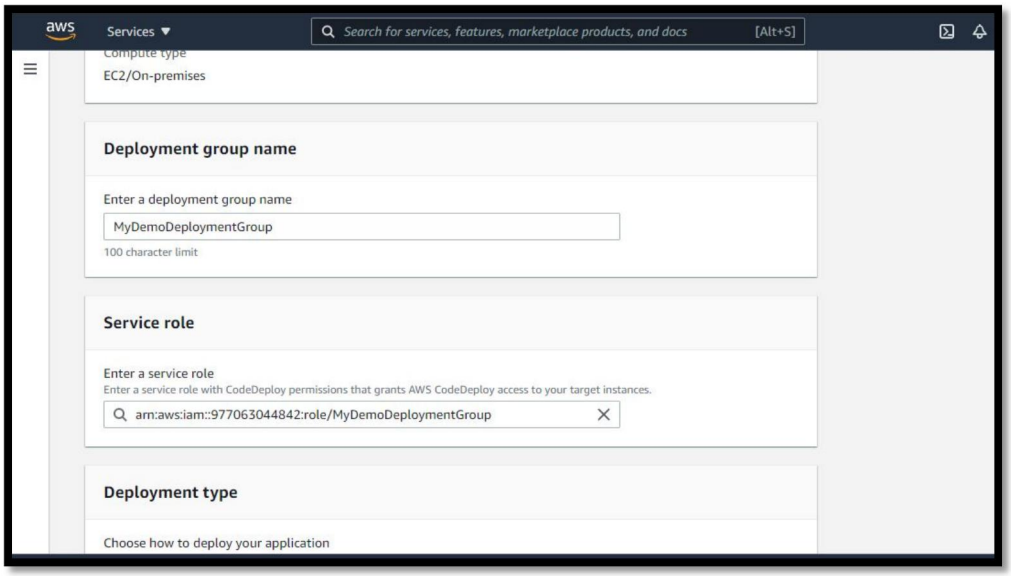
1. On the page that displays your application, choose Create deployment group.



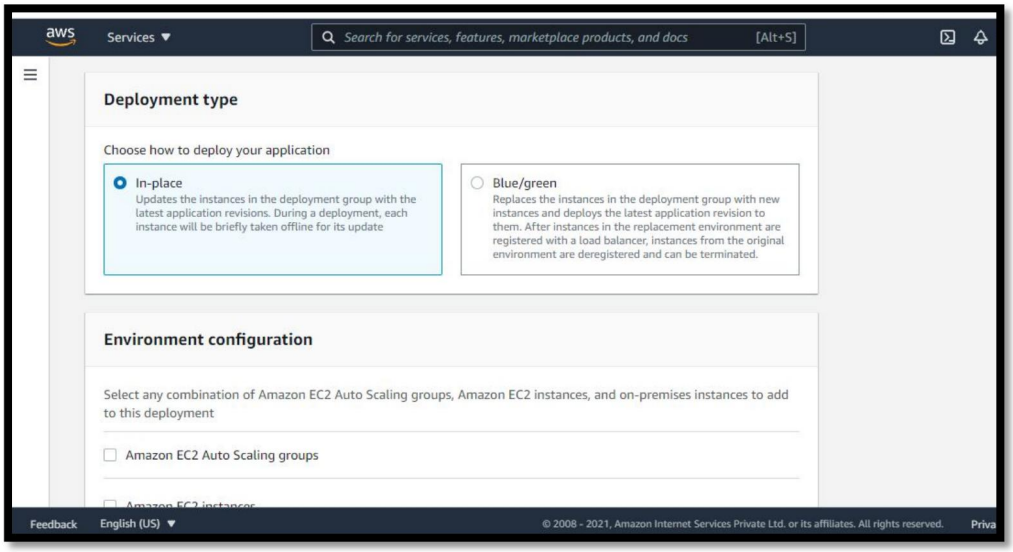
2. In Deployment group name, enter MyDemoDeploymentGroup.



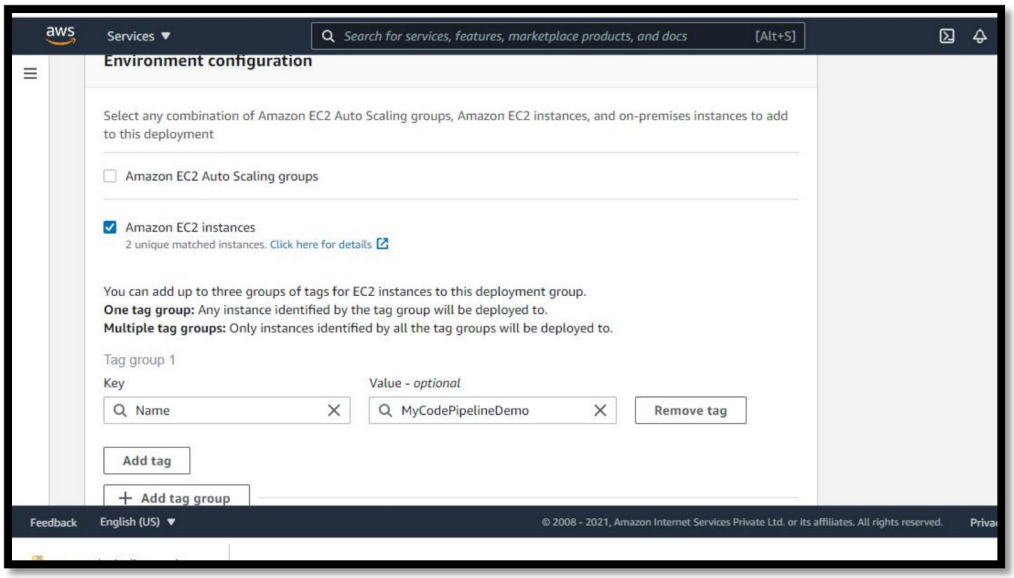
3. In Service Role, choose a service role that trusts AWS CodeDeploy with, at minimum, the trust and permissions described in Create a Service Role for CodeDeploy. To get the service role ARN, see Get the Service Role ARN (Console).



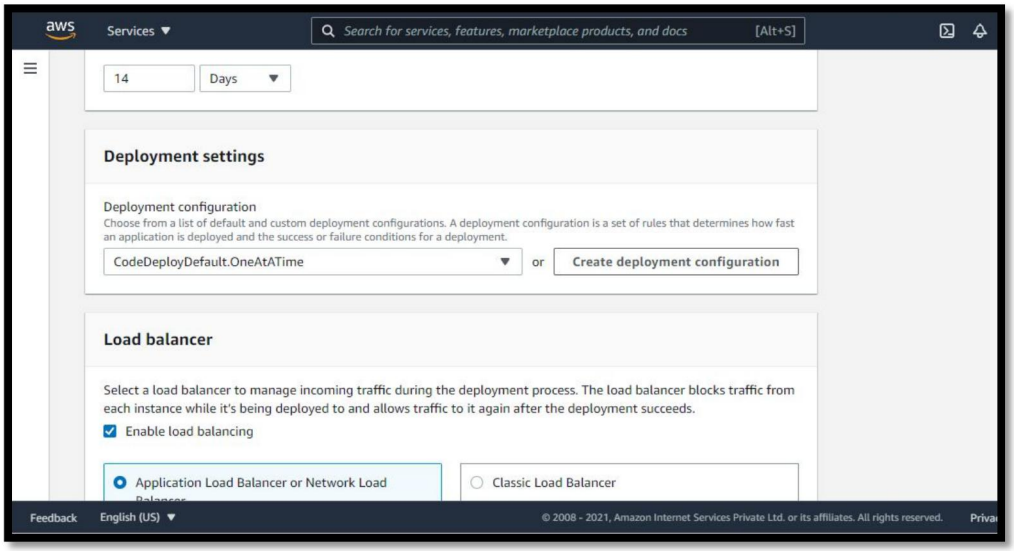
4. Under Deployment type, choose In-place.



5. Under Environment configuration, choose Amazon EC2 Instances. Choose Name in the Key field, and in the Value field, enter MyCodePipelineDemo.

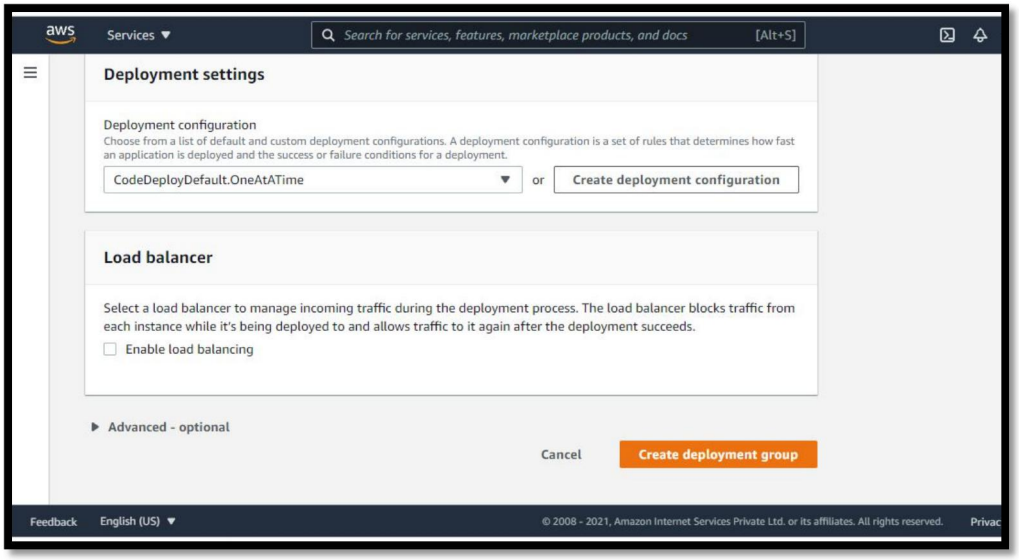


6. Under Deployment settings, choose CodeDeployDefault.OneAtaTime.

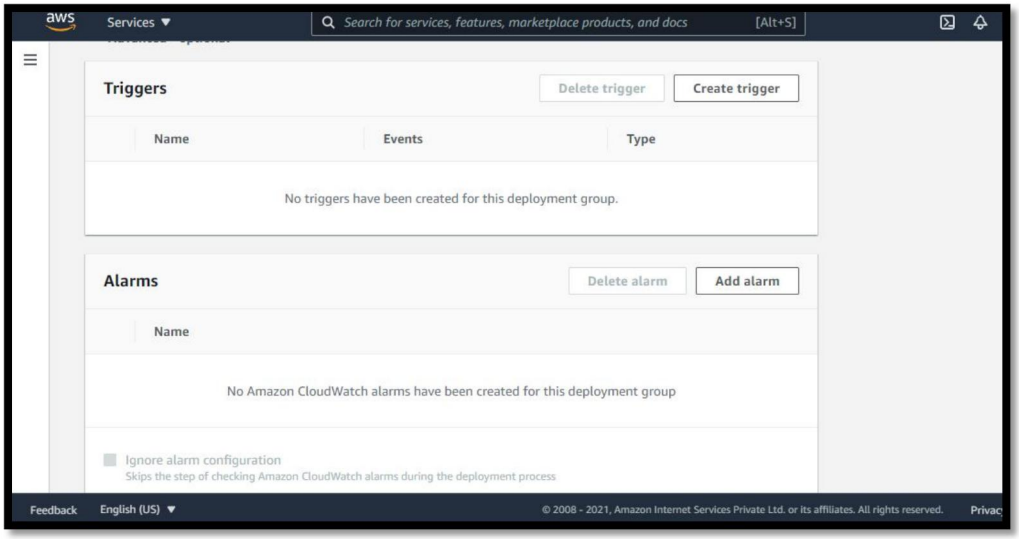


7. Under Load Balancer, make sure the Enable load balancing box is not selected. You do not need to set up a load balancer or choose a target group

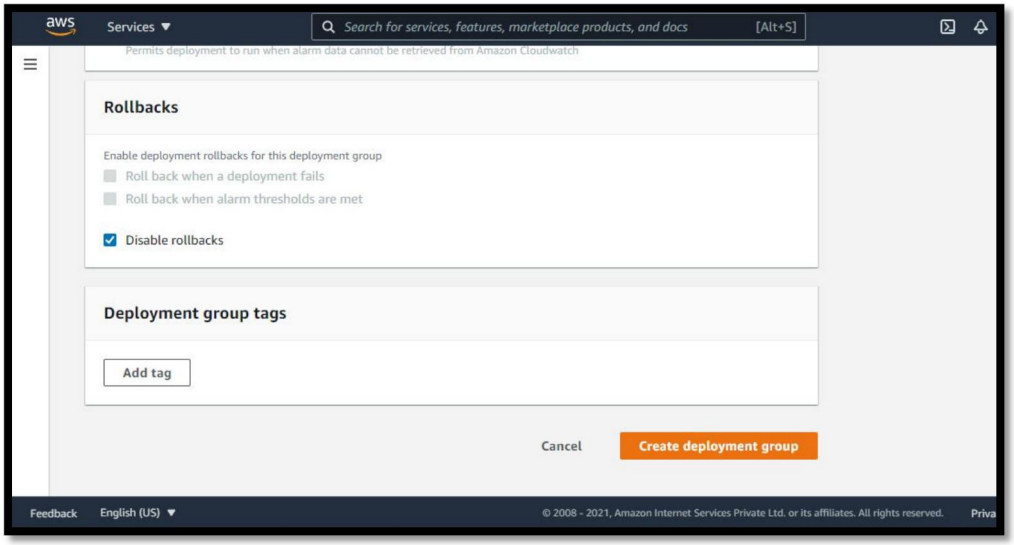
for this example. After you deselect the checkbox, the load balancer options do not display.

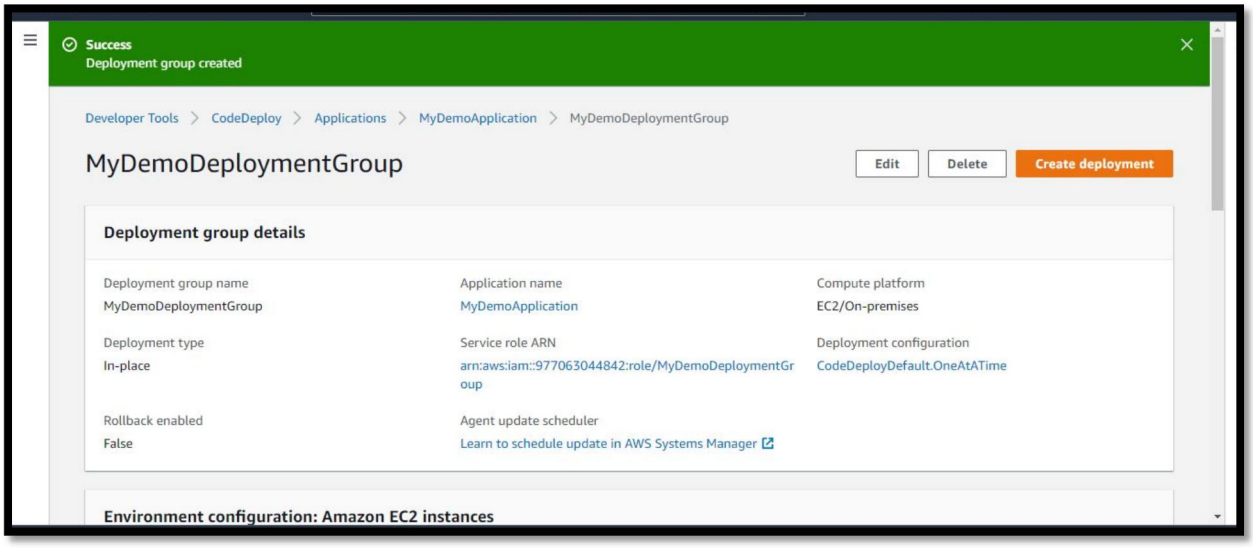


8. In the Advanced section, leave the defaults.



9. Choose Create deployment group.





Conclusion :-

We have successfully Build an Application using AWS CodeBuild and Deploy on S3 / SEBS using AWS CodePipeline, deploy Sample Application on EC2 instance using AWS CodeDeploy.

Experiment 03

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

Theory:

A Kubernetes Cluster is a set of nodes that run containerized applications. Containerizing applications packages an app with its dependencies and some necessary services.

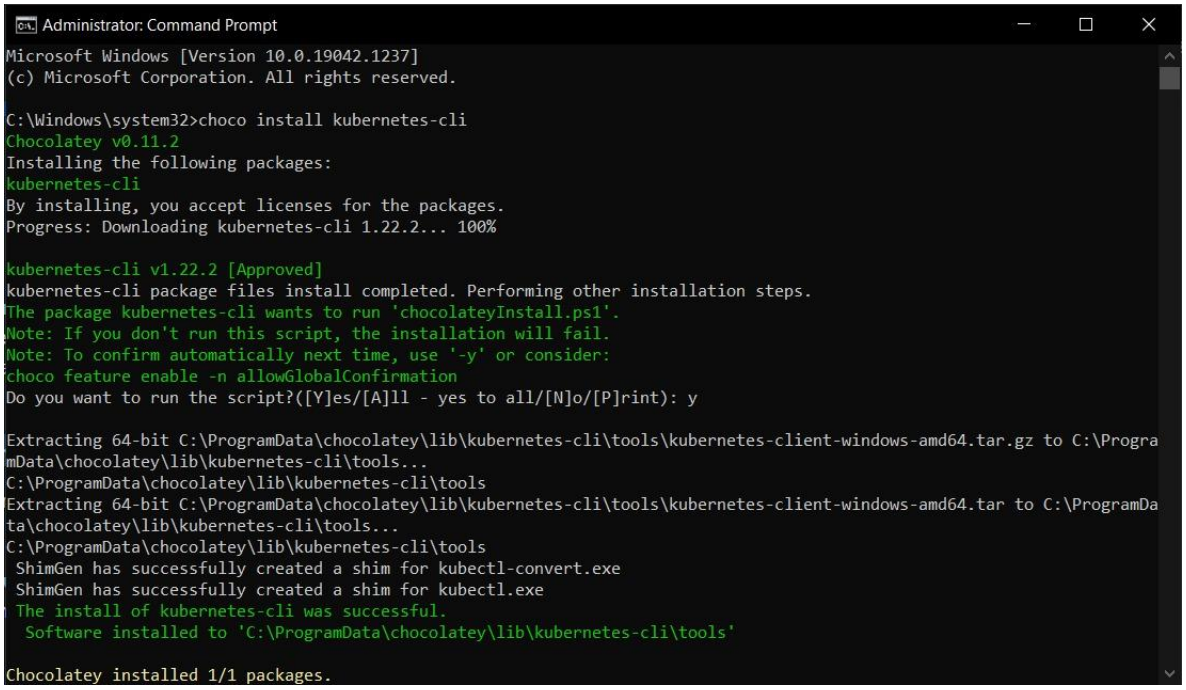
They are more lightweight and flexible that virtual machines. IN this way, Kubernetes Cluster allow for applications to be more easily developed, moved and managed.

Kubernetes Cluster allow containers to run across multiple machines and environments: virtual, physical, clou-based and on-premises. Kubernetes containers are not restricted to a specific operating system unlike virtual machines. Instead, they are able to share operating systems and run anywhere.

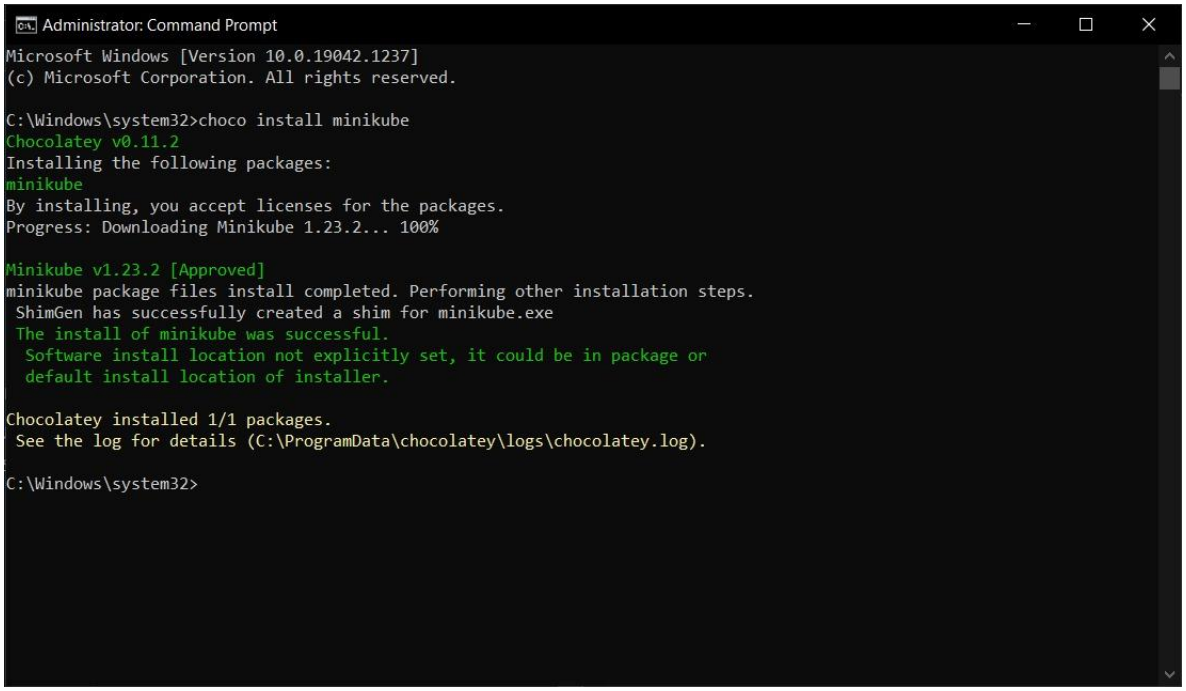
Kubernetes clusters are comprised of one master node and a number of worker nodes. These nodes can either be physical computers or virtual machines depending on the cluster.

Outputs:

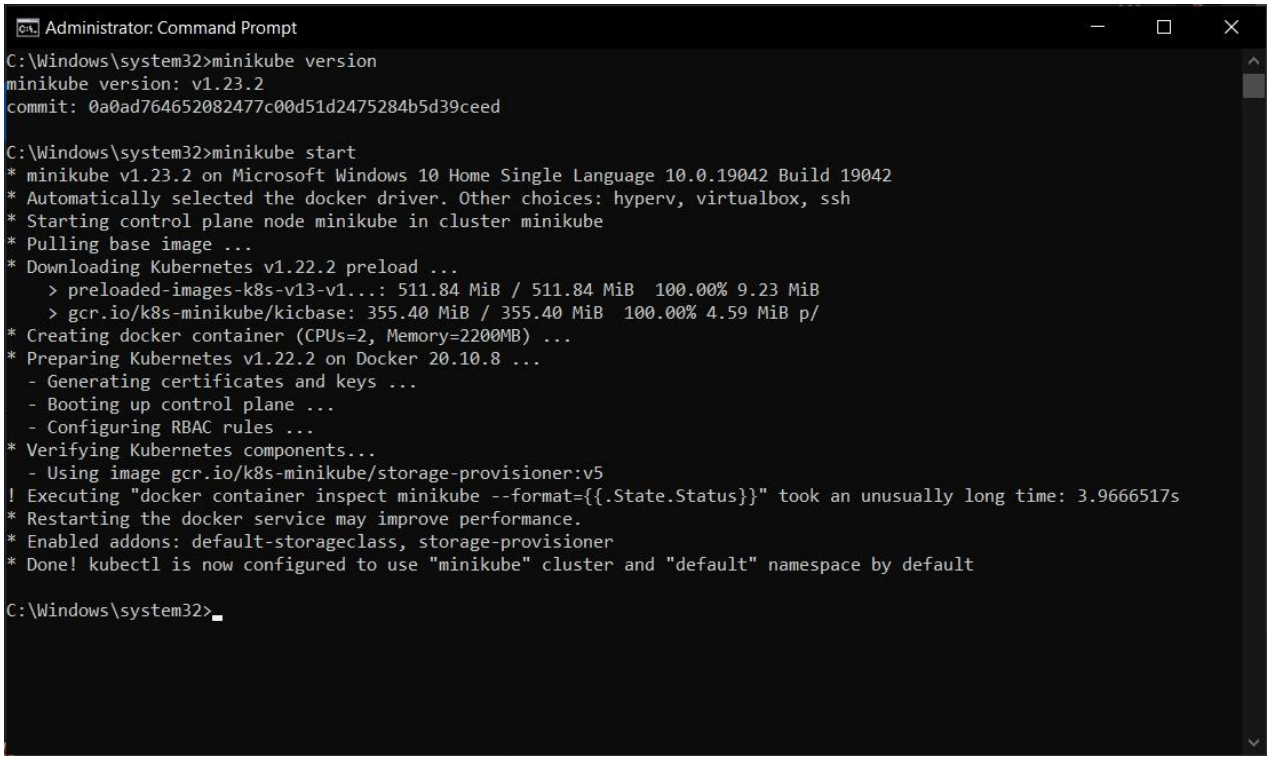
Installing the Kubernetes-cli using chocolatey.

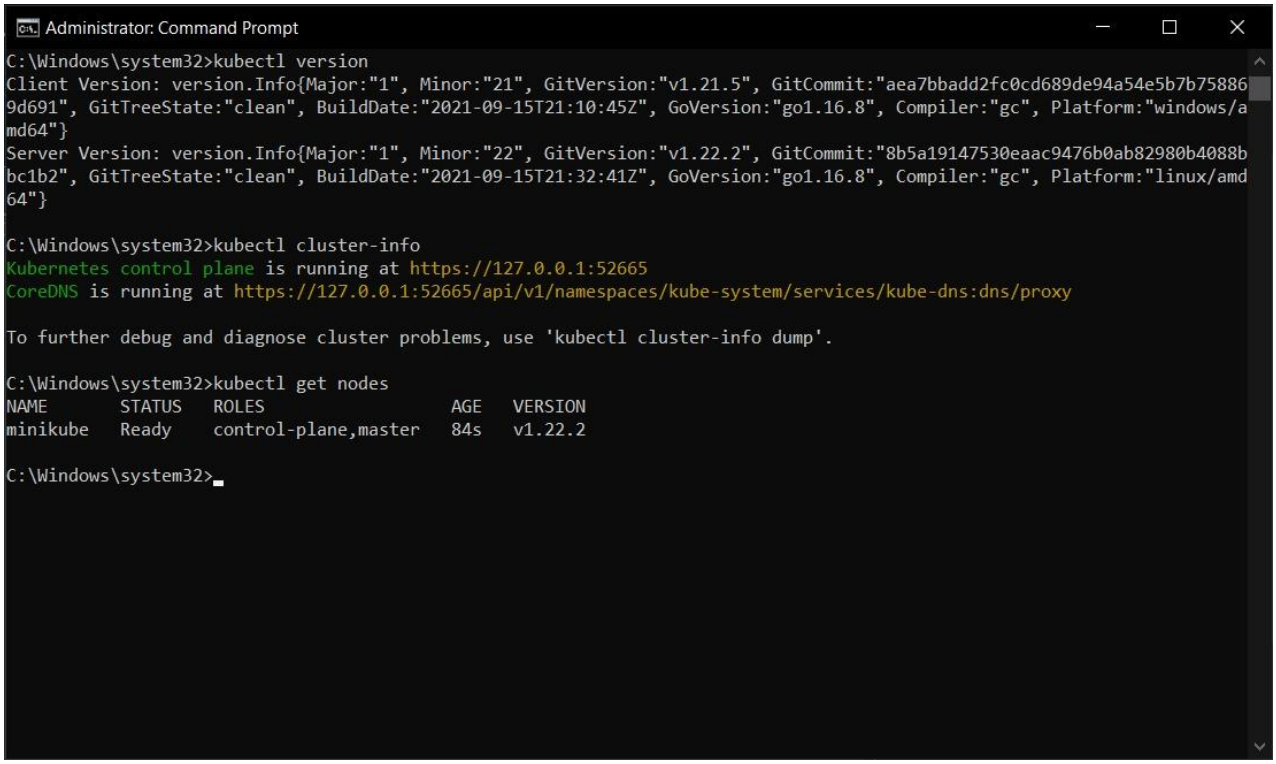


Installing minikube:



Spinning up Kubernetes cluster:





Conclusion: Kubernetes architecture was studied in detail and a Kubernetes cluster was spun up successfully.

Experiment 04

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

Theory:

Kubectl:

The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

Deploying applications using kubectl:

Once you have a running Kubernetes cluster, you can deploy your containerized applications on top of it. To do so, you create a Kubernetes Deployment configuration. The Deployment instructs Kubernetes how to create and update instances of your application. Once you've created a Deployment, the Kubernetes control plane schedules the application instances included in that Deployment to run on individual Nodes in the cluster.

Once the application instances are created, a Kubernetes Deployment Controller continuously monitors those instances. If the Node hosting an instance goes down or is deleted, the Deployment controller replaces the instance with an instance on another Node in the cluster. This provides a self-healing mechanism to address machine failure or maintenance.

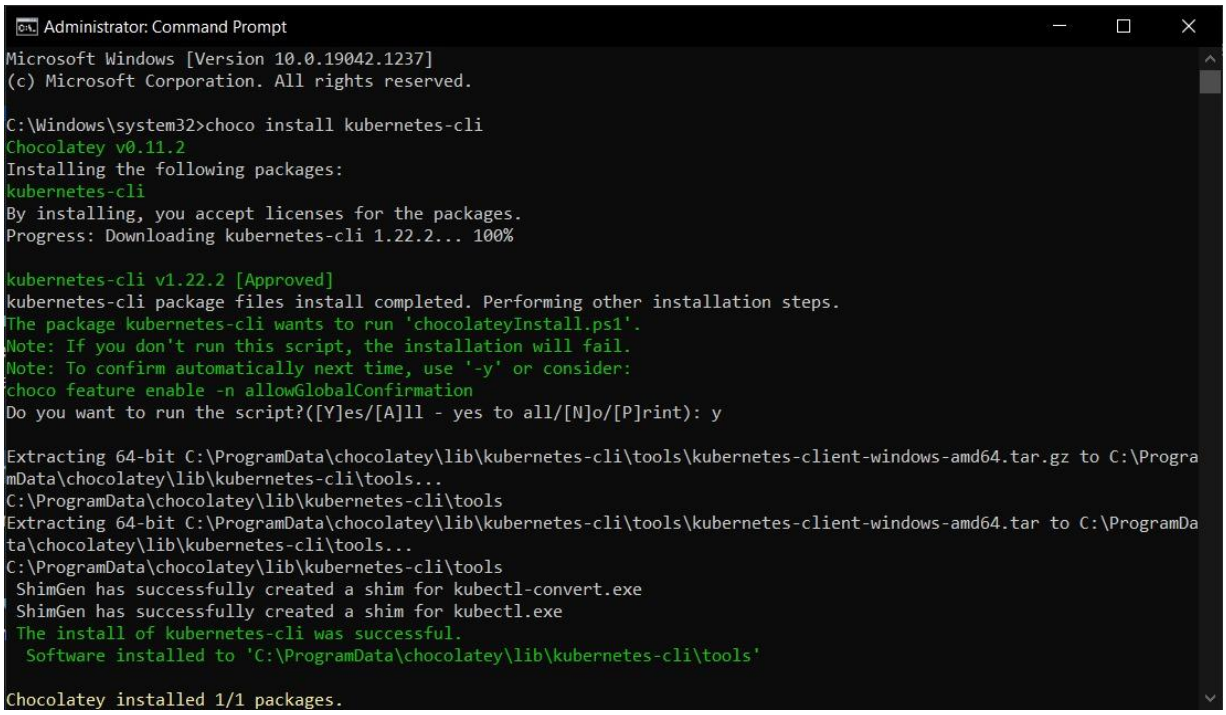
In a pre-orchestration world, installation scripts would often be used to start applications, but they did not allow recovery from machine failure. By both creating your application instances and keeping them running across Nodes, Kubernetes Deployments provide a fundamentally different approach to application management.

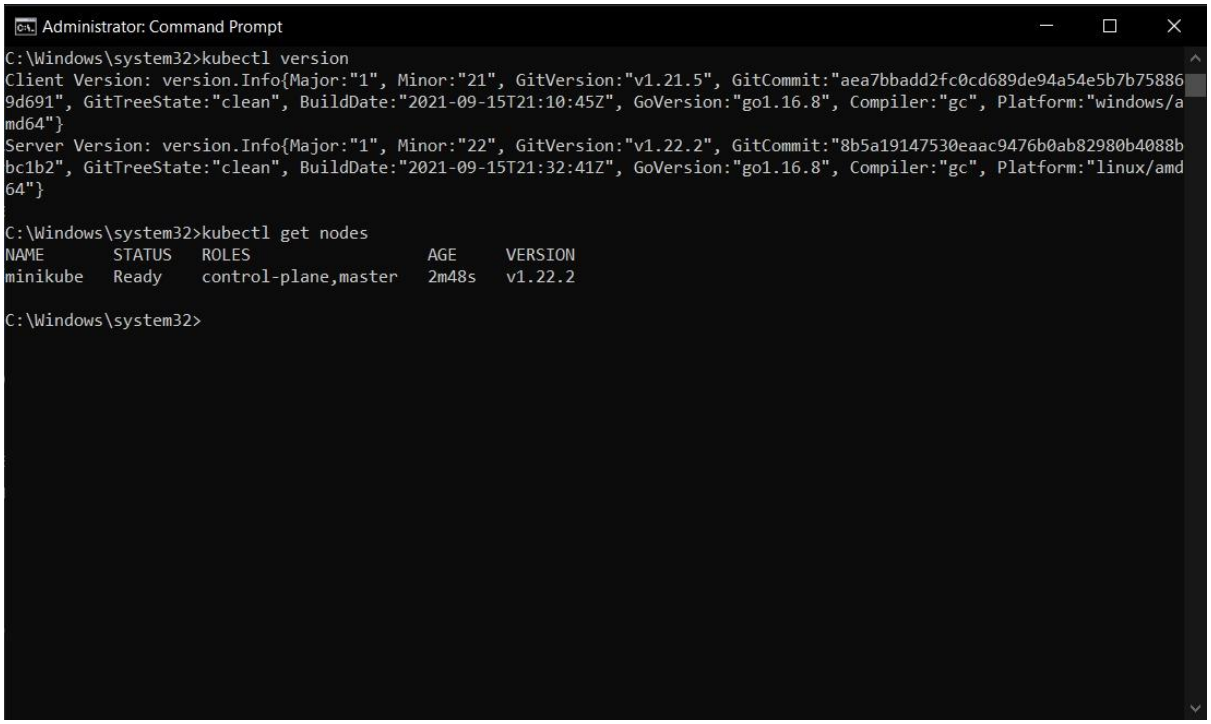
You can create and manage a Deployment by using the Kubernetes command line interface, Kubectl. Kubectl uses the Kubernetes API to interact with the cluster. In this module, you'll learn the most common Kubectl commands needed to create Deployments that run your applications on a Kubernetes cluster.

When you create a Deployment, you'll need to specify the container image for your application and the number of replicas that you want to run.

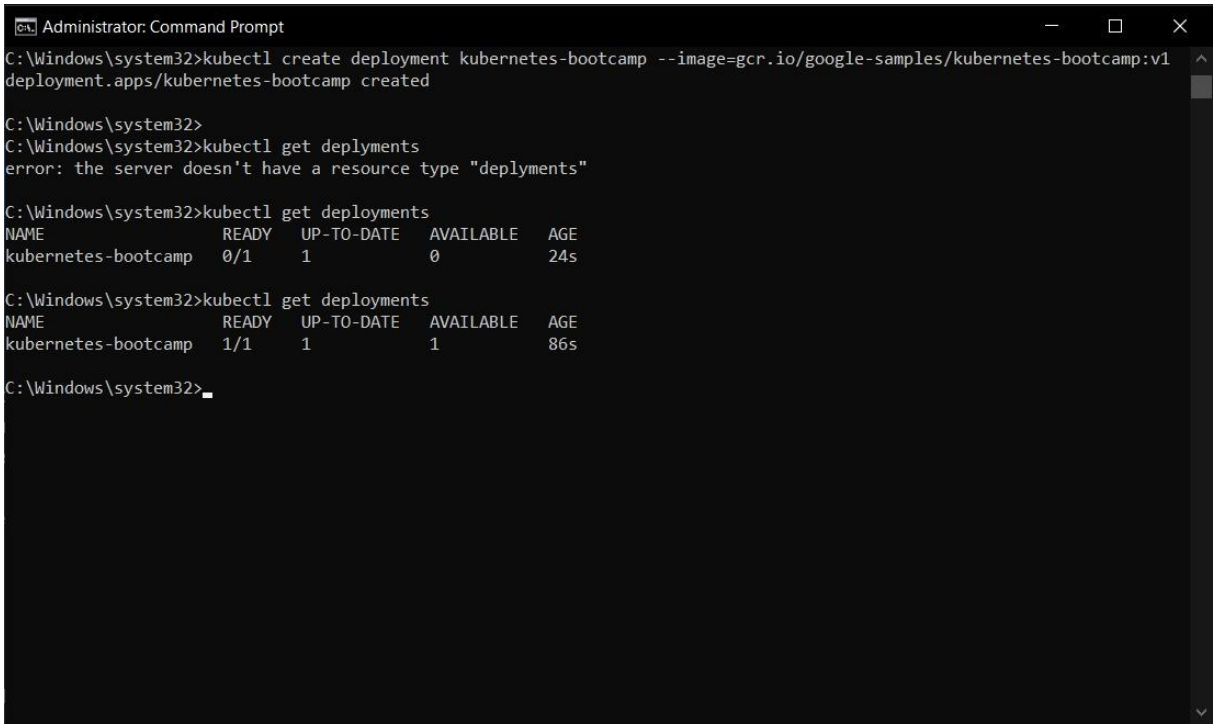
Outputs:

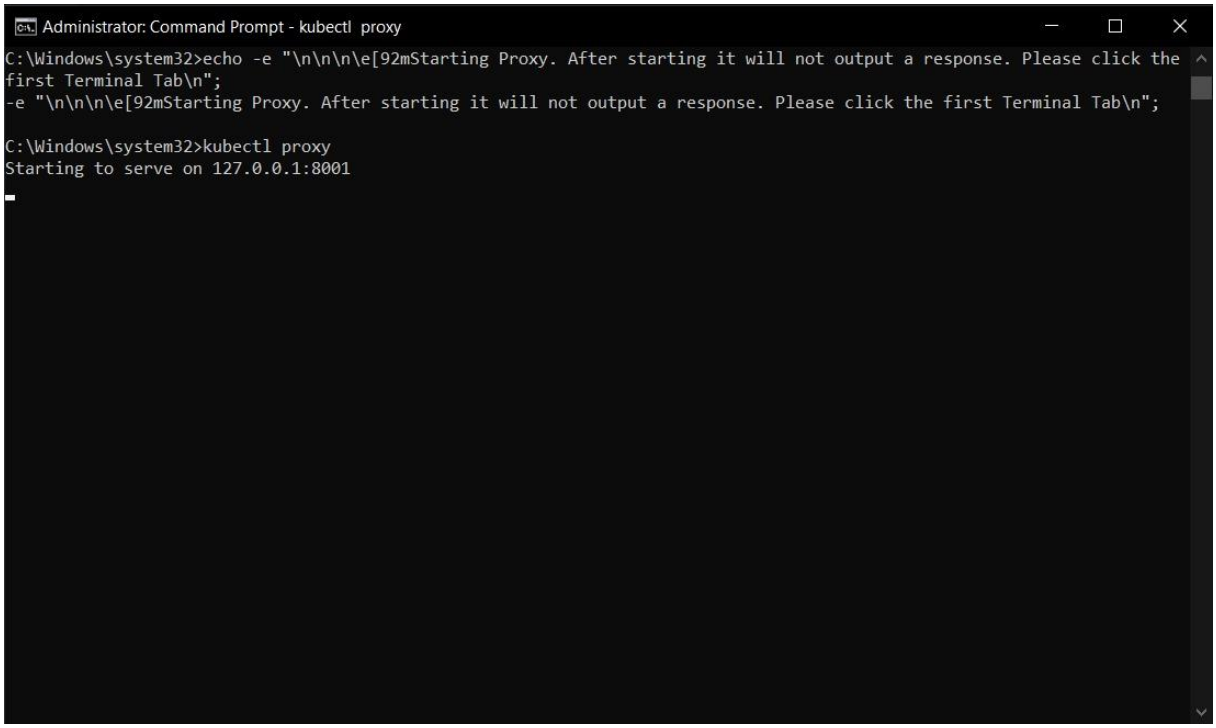
Installing kubectl using chocolatey:



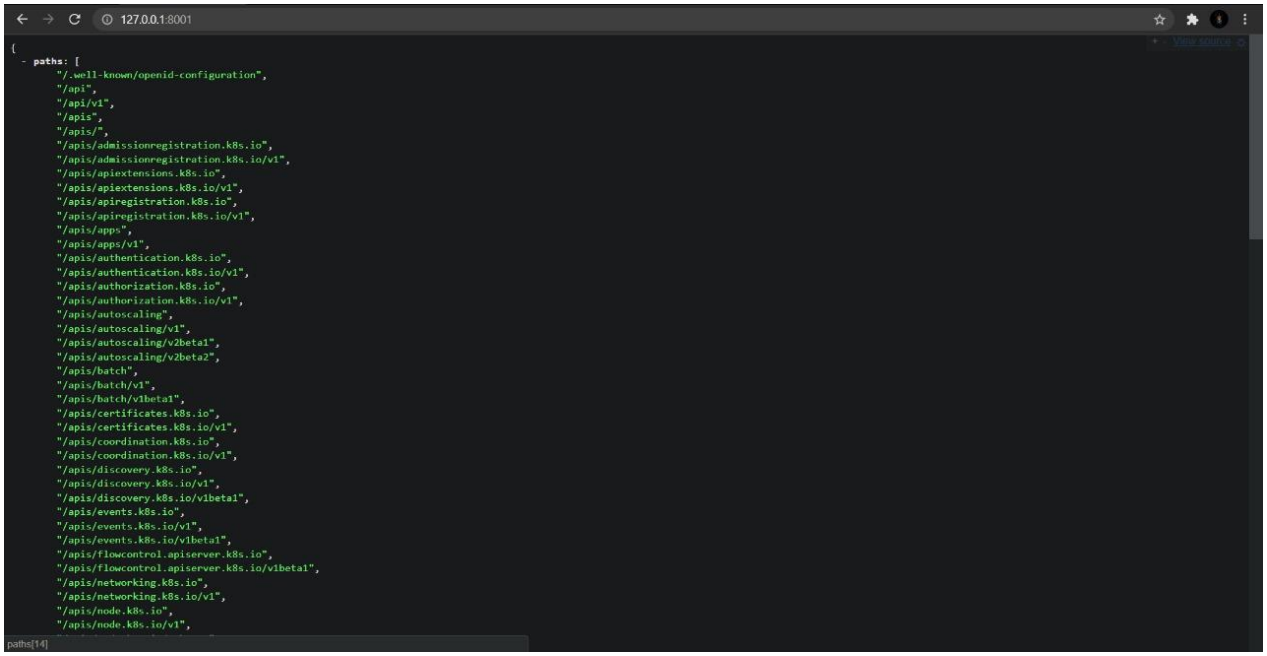


Deploying first application using Kubernetes:





Checking the deployed application:



Conclusion: Kubectl was successfully installed and an application was successfully deployed using Kubernetes.

EXPERIMENT NO : 05

AIM : To understand terraform lifecycle, core concepts/terminologies and install it on a Linux Machine

THEORY :

Terraform is an open-source infrastructure as code software tools created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL) or optionally JSON.

Terraform supports a number of cloud infrastructure providers such Amazon Web Services, Microsoft Azure, IBM Cloud, Google Cloud Platform, DigitalOcean, Oracle Cloud Infrastructure, Yandex.Cloud, vMware vSphere and OpenStack.

Terraform has four major commands

$ terraform init

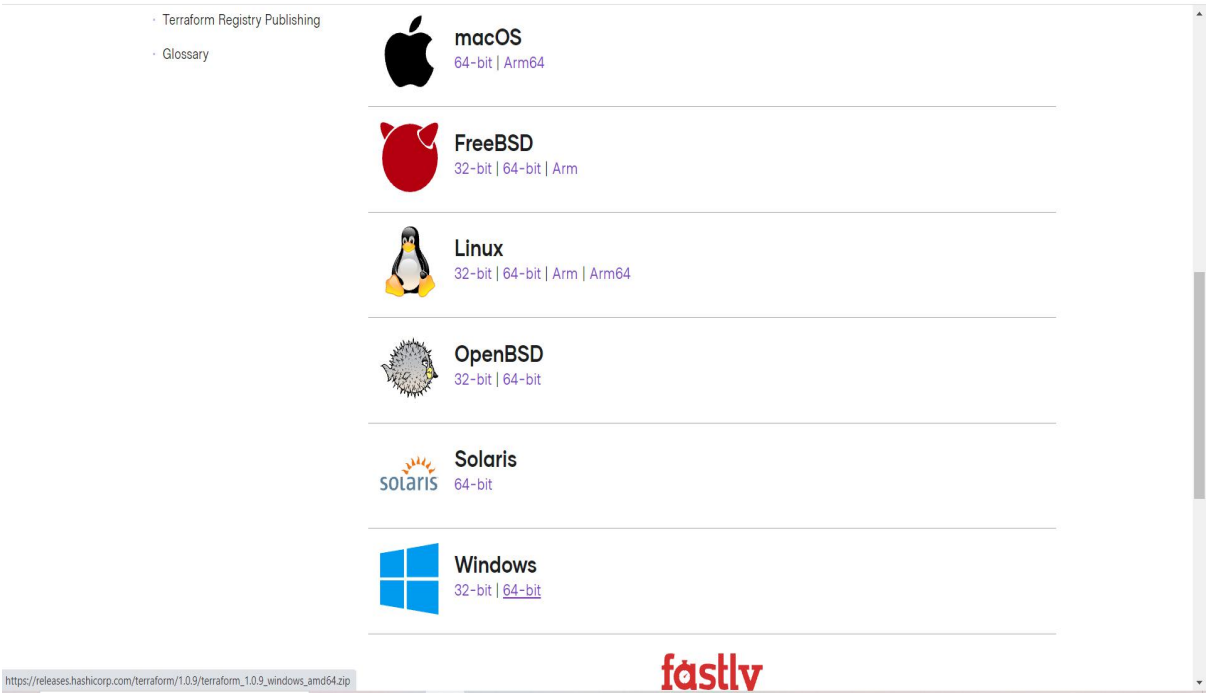
$ terraform plan

$ terraform apply

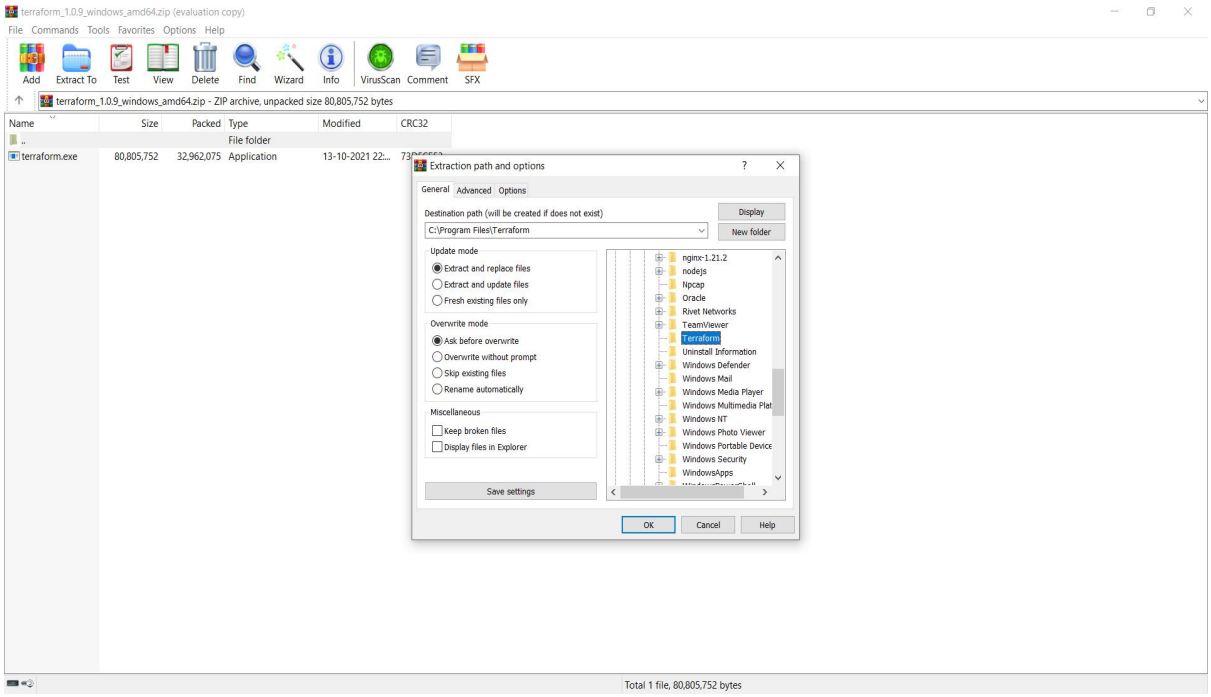
$ terraform destroy

INSTALLATION OF TERRAFORM

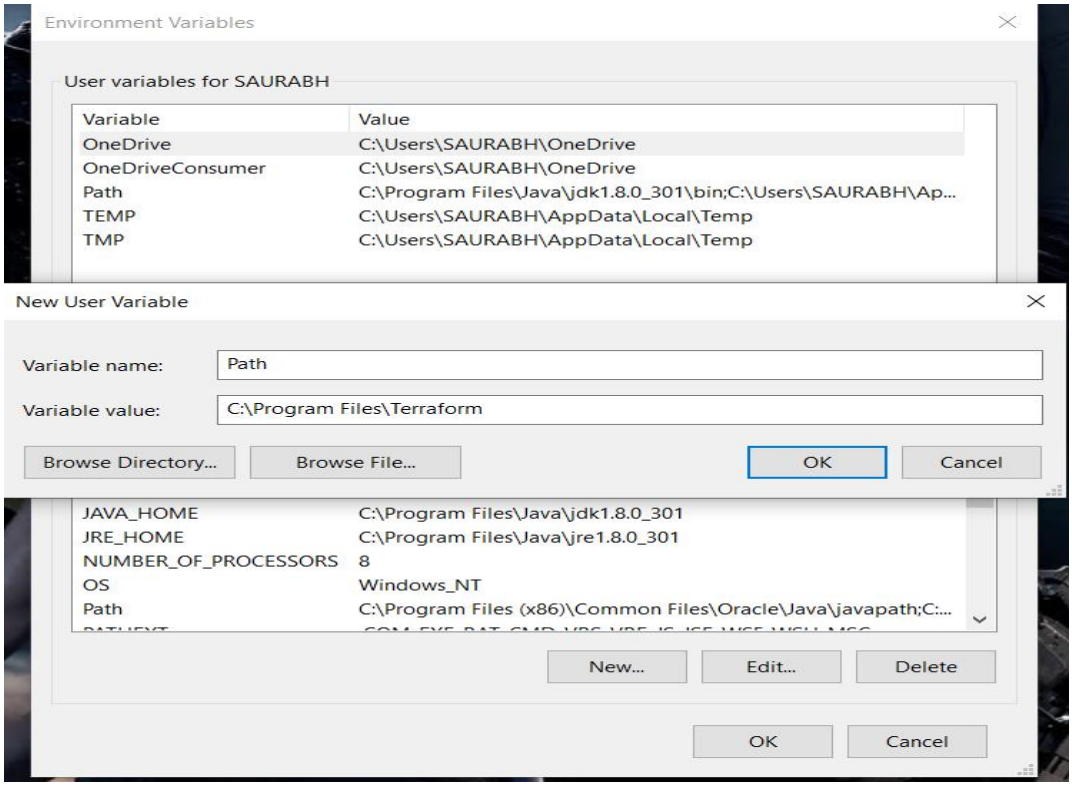
Step 1: Download the Terraform Cli utility



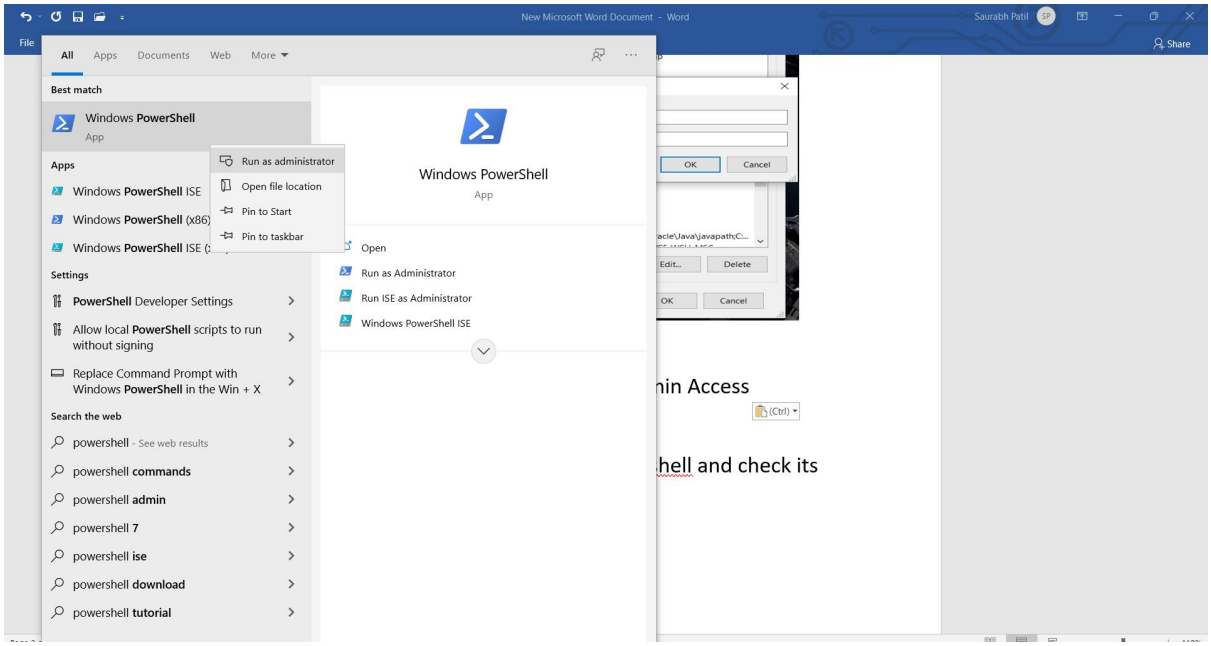
Step 2: Extract the downloaded setup file Terraform.exe in C:\Terraform directory



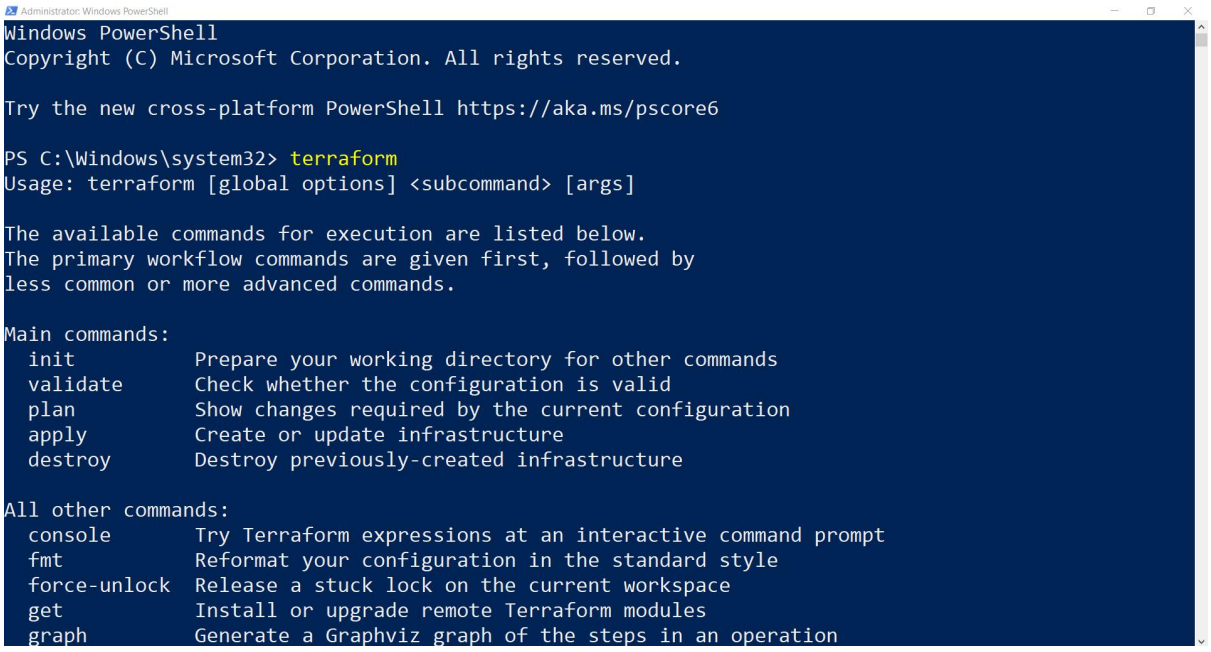
Step 3: Set the System path for Terraform in Environment variables

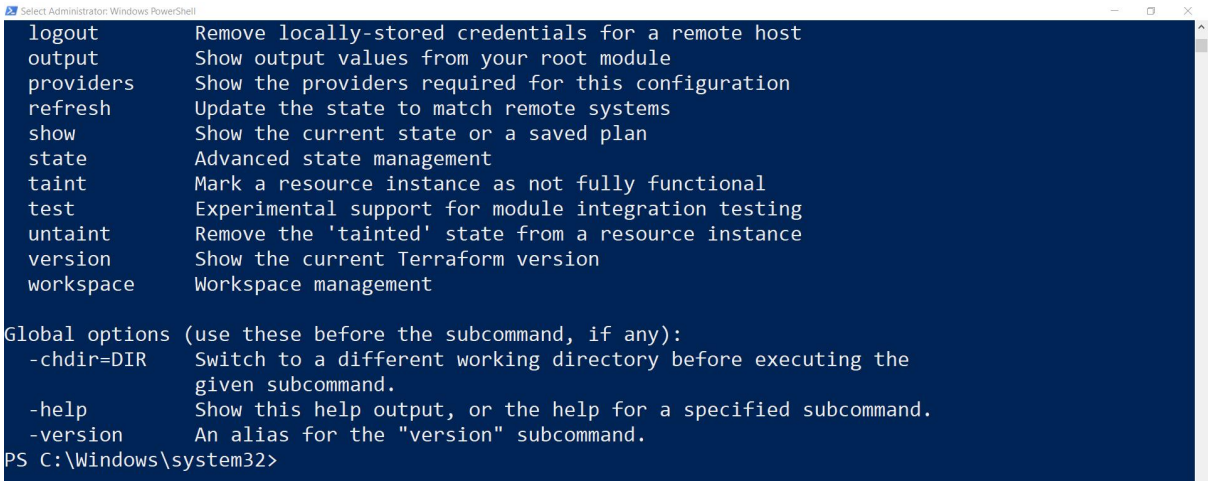


Step 4: Open Powershell with Admin Access



Step 5: Open Terraform in Powershell and check its functionality





CONCLUSION : Hence we can conclude that we have learned and implemented terraform lifecycle, core concepts/terminologies and install it on a Linux Machine.

EXPERIMENT NO : 06

AIM : To Build, change and destroy AWS/GCP/Microsoft Azure/DigitalOcean infrastructure using Terraform

THEORY :

Terraform is an open-source infrastructure as code software tools created by HashiCorp. Users define and provide data center infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL) or optionally JSON.

Terraform supports a number of cloud infrastructure providers such Amazon Web Services, Microsoft Azure, IBM Cloud, Google Cloud Platform, DigitalOcean, Oracle Cloud Infrastructure, Yandex.Cloud, vMware vSphere and OpenStack.

Terraform has four major commands

$ terraform init

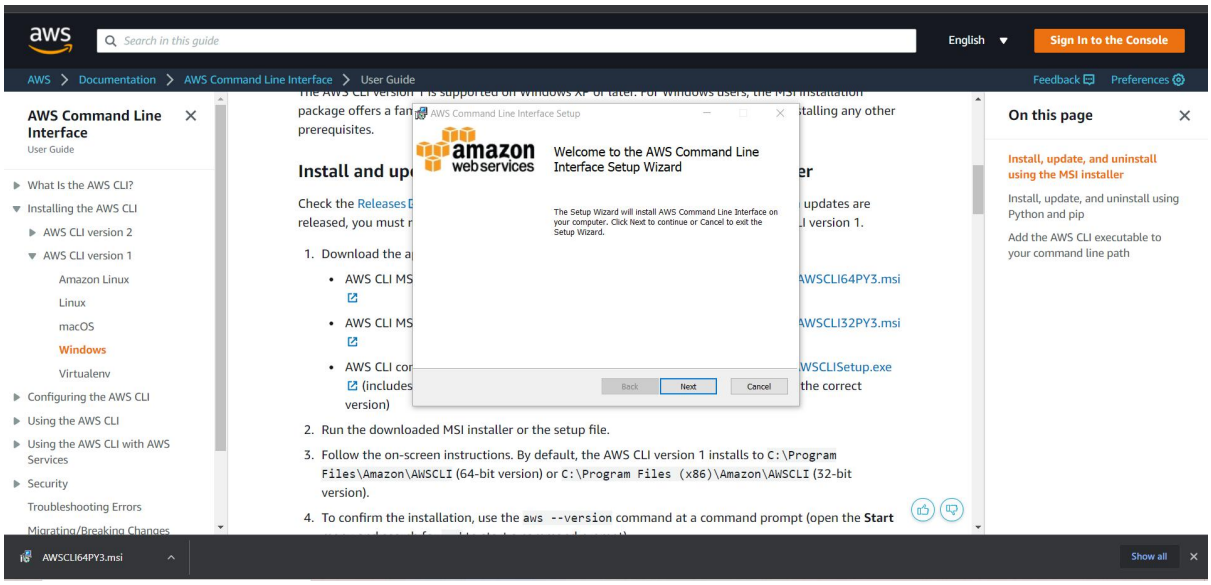
$ terraform plan

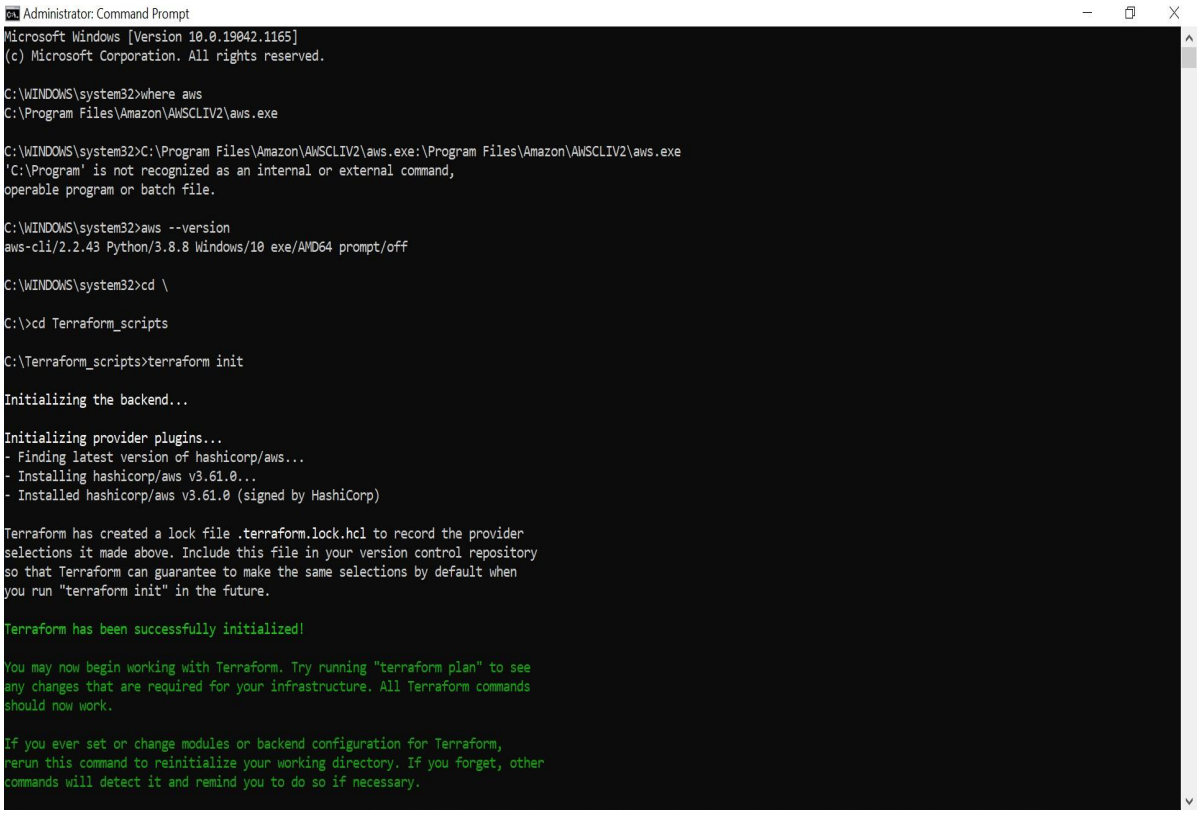
$ terraform apply

$ terraform destroy

DESTROY AWS INFRASTRUCTURE USING TERRAFORM

Step 1: Download AWS Cli and set environment variable




Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.
C:\WINDOWS\system32>where aws
C:\Program Files\Amazon\AWSCLIV2\aws.exe
C:\WINDOWS\system32>C:\Program Files\Amazon\AWSCLIV2\aws.exe:\Program Files\Amazon\AWSCLIV2\aws.exe
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
C:\WINDOWS\system32>aws --version
aws-cli/2.2.43 Python/3.8.8 Windows/10 exe/AMD64 prompt/off
C:\WINDOWS\system32>cd \
C:\>cd Terraform_scripts
C:\Terraform_scripts>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.61.0...
- Installed hashicorp/aws v3.61.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

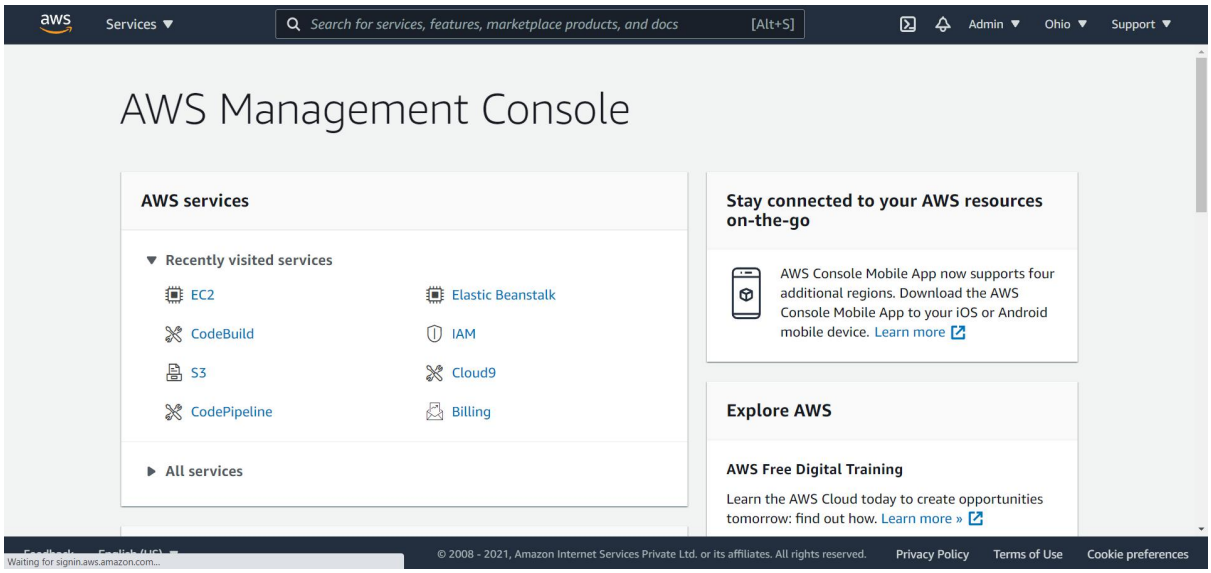
Administrator: Command Prompt

```
commands will detect it and remind you to do so if necessary.

C:\Terraform_scripts>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.ubuntu will be created
  + resource "aws_instance" "ubuntu" {
      + ami                                  = "ami-0747bdcabd34c712a"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
```

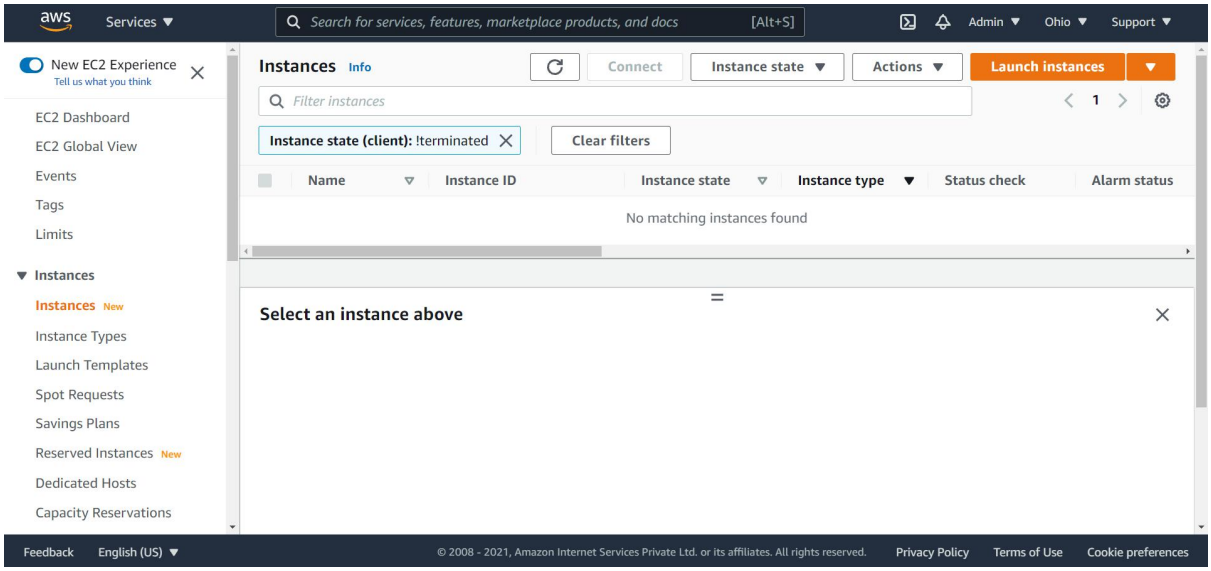
Administrator: Command Prompt

```
      + enclave_options {
          + enabled = (known after apply)
        }

      + ephemeral_block_device {
          + device_name  = (known after apply)
          + no_device    = (known after apply)
          + virtual_name = (known after apply)
        }

      + metadata_options {
          + http_endpoint               = (known after apply)
          + http_put_response_hop_limit = (known after apply)
          + http_tokens                 = (known after apply)
        }

      + network_interface {
          + delete_on_termination = (known after apply)
          + device_index          = (known after apply)
          + network_interface_id  = (known after apply)
        }

      + root_block_device {
          + delete_on_termination = (known after apply)
          + device_name           = (known after apply)
          + encrypted             = (known after apply)
          + iops                  = (known after apply)
          + 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.

_____________________________________________________________________________
```

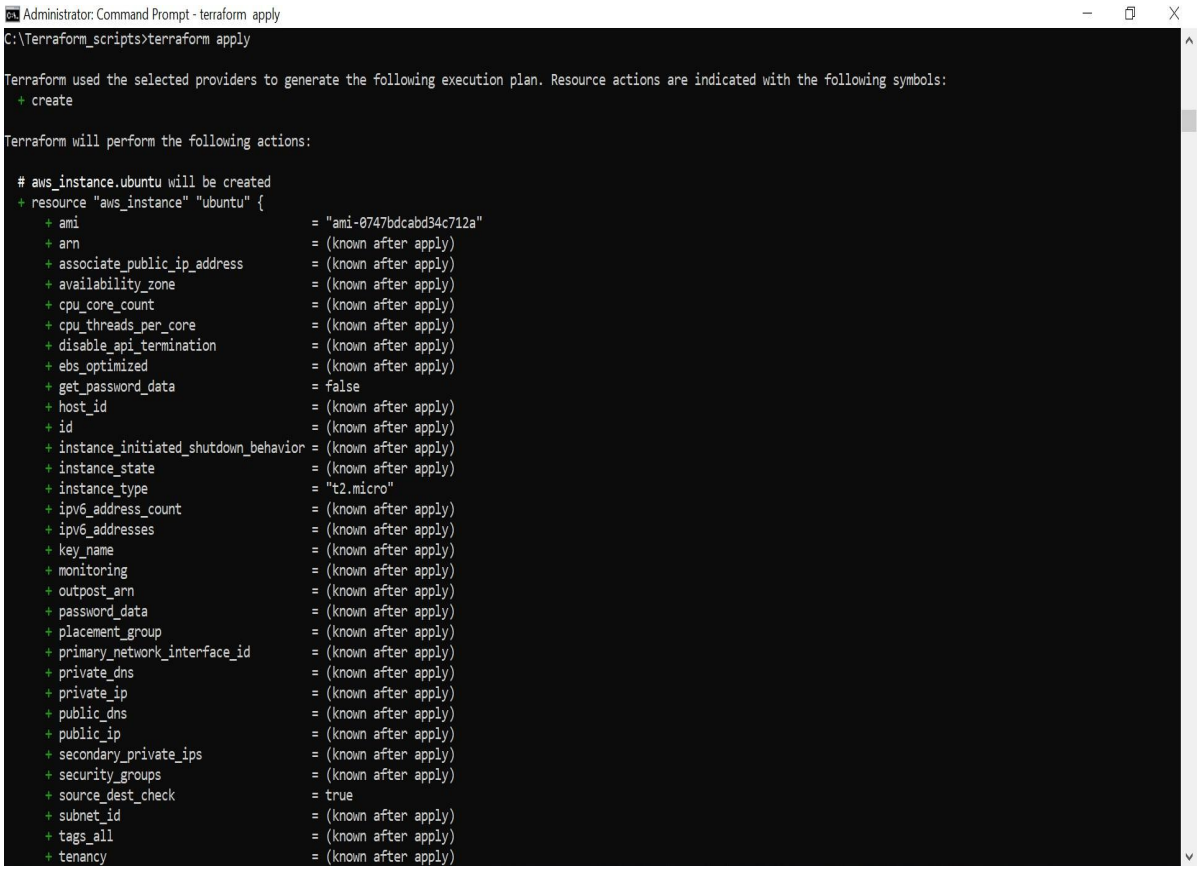
Step 2: Log into your AWS account

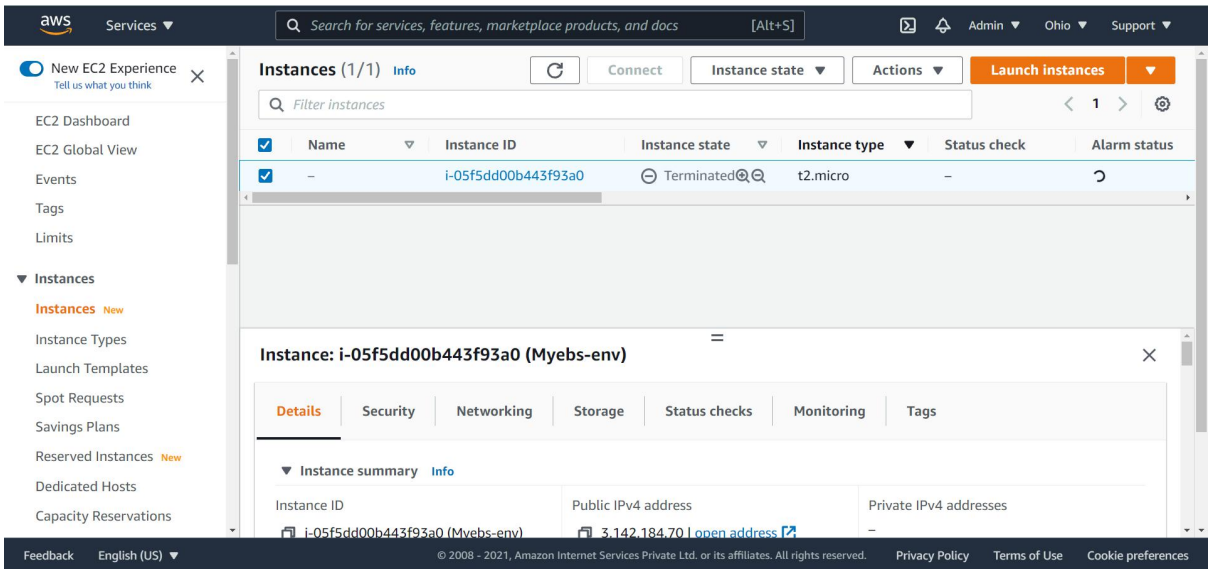


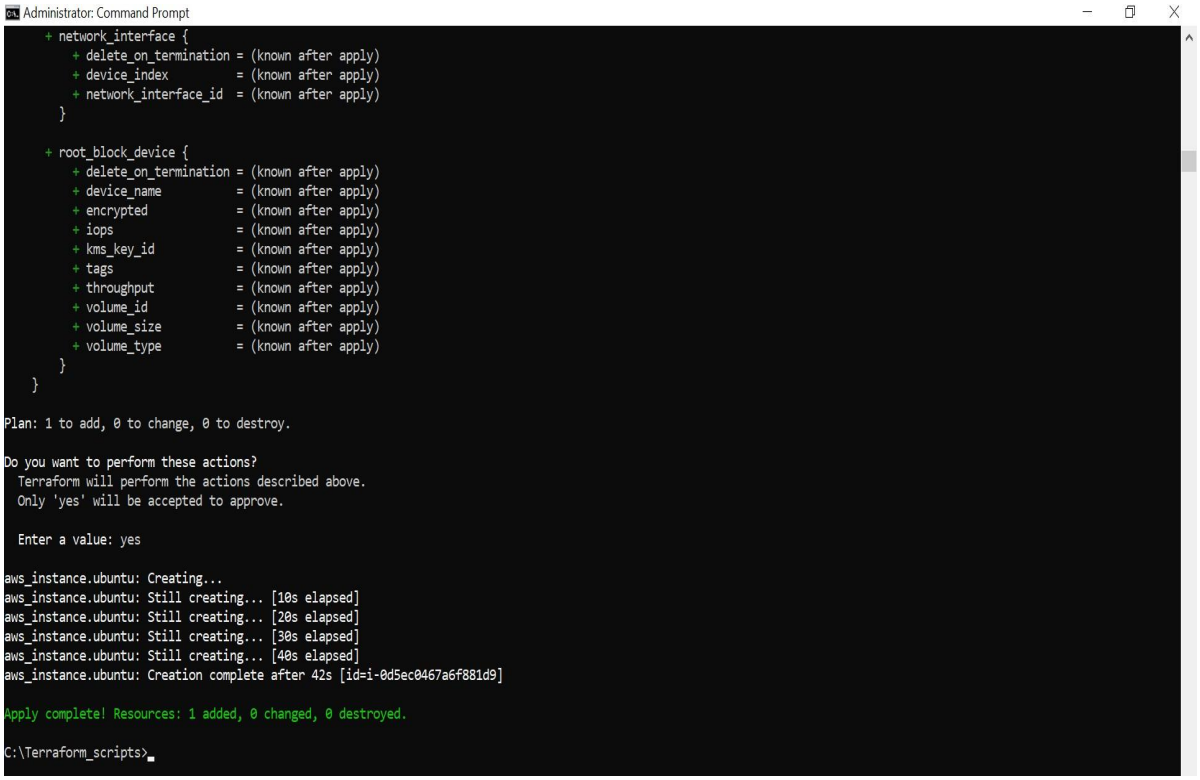
Step 3: Open the instance section in EC2 services



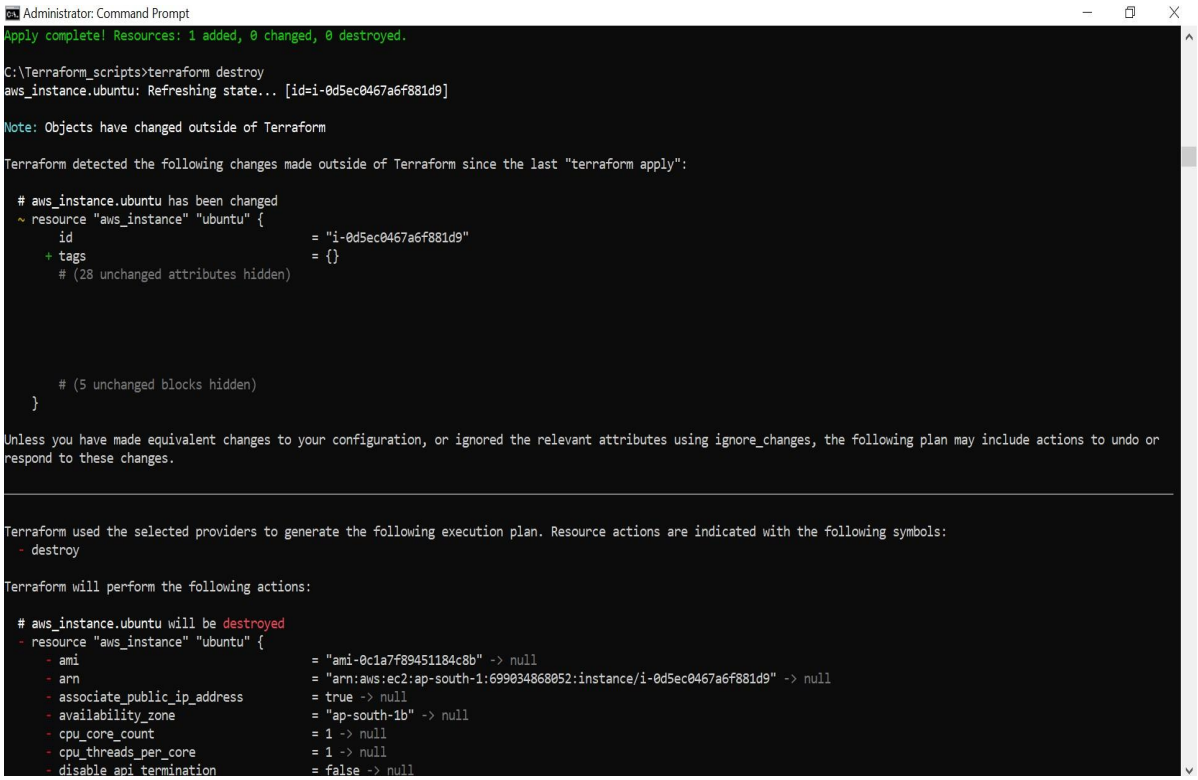
Step 4: Write command terraform apply to create a new instance



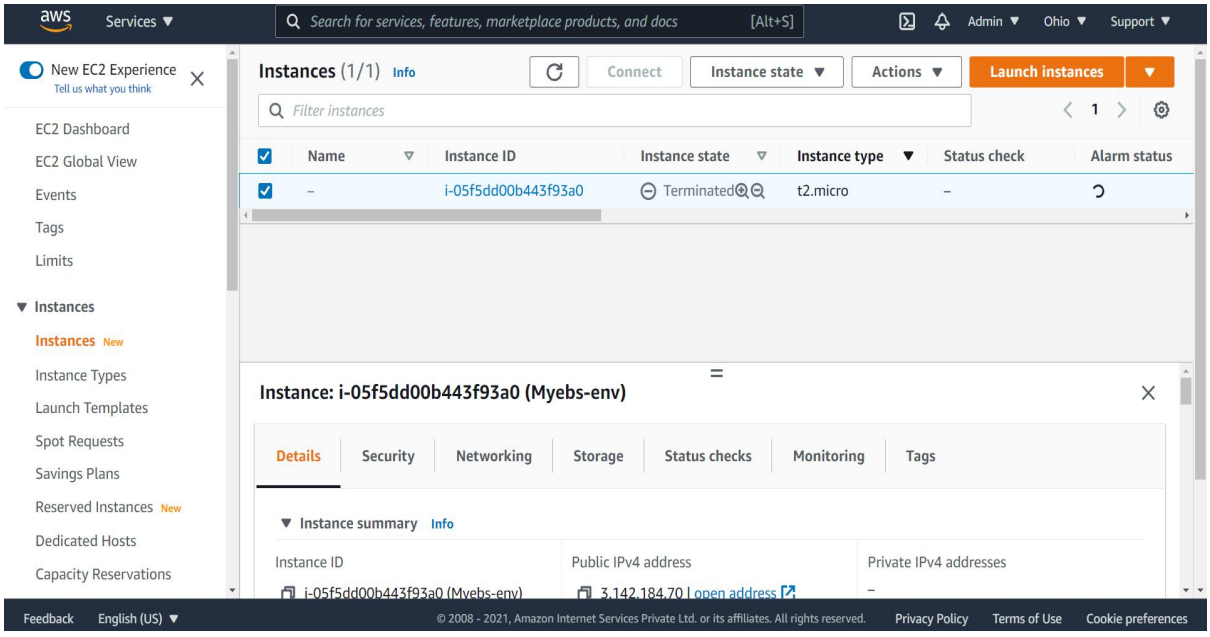




Step 5: Type the command terraform destroy to delete



the instance created



CONCLUSION : Hence we can conclude that we have learned and implemented To Build, change and destroy AWS/GCP/Microsoft Azure/DigitalOcean infrastructure using Terraform.

EXPERIEMENT NO : 07

AIM : To understand Static Analysis SAST process and learn tointegrate Jenkins SAST to SonarQube/GitLab.

THEORY :

SonarQube is a Code Quality Assurance tool that collects and analyzes source code, and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time. Everything from minor styling choices, to design errors are inspected and evaluated by SonarQube. This provides users with a rich searchable history of thecode to analyze where the code is messing up and determine whether or not it is styling issues, code defeats, code duplication, lack of test coverage, or excessivelycomplex code. The software will analyze source code from different aspects and drills down the code layer by layer, moving module level down to the class level, with each level producing metric values and statistics

that should reveal problematic areas in the source code that needs improvement.

Sonarqube also ensures code reliability, Application security, and reduces technical debt by making your code base clean and maintainable. Sonarqubealso provides support for 27 different languages, including C, C++, Java, Javascript, PHP, GO, Python, and much more.SonarQube also provides Ci/CD integration, and gives feedback during code review with branch analysis and pull request decoration.

SonarQube Benefits:

- CI tools do not have a plugin which would make all of these toolswork easily together
- CI tools do not have plugins to provide nice drill-down featuresthat SonarQube has
- CI Plugins does not talk about overall compliance value
- CI plugins do not provide managerial perspective
- There is no CI plugin for Design or Architectural issues
- CI plugins do not provide a dashboard for overall

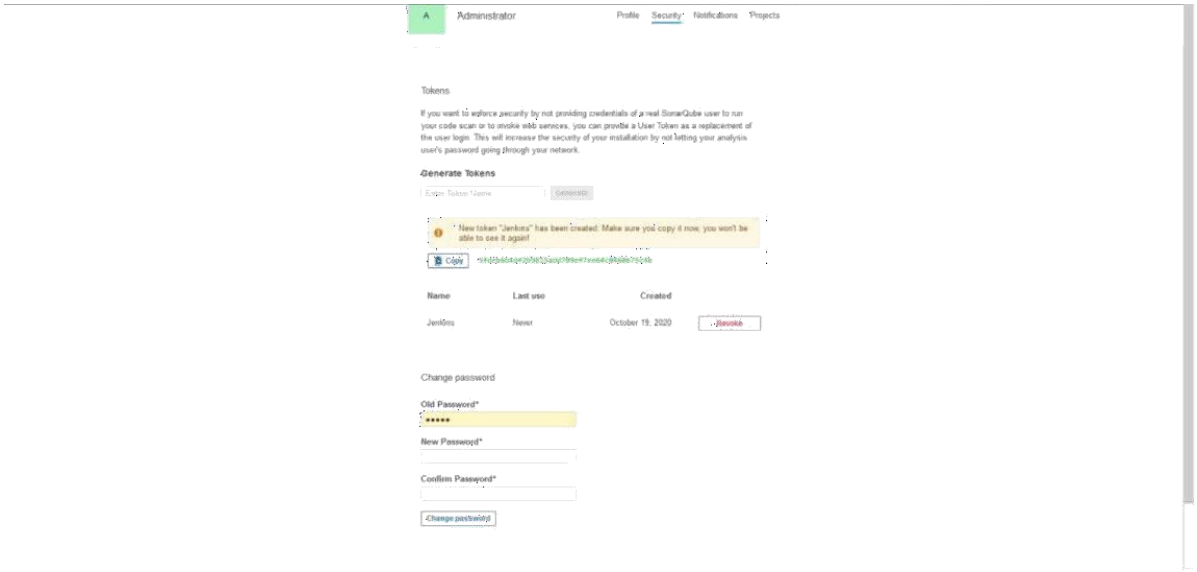
project quality

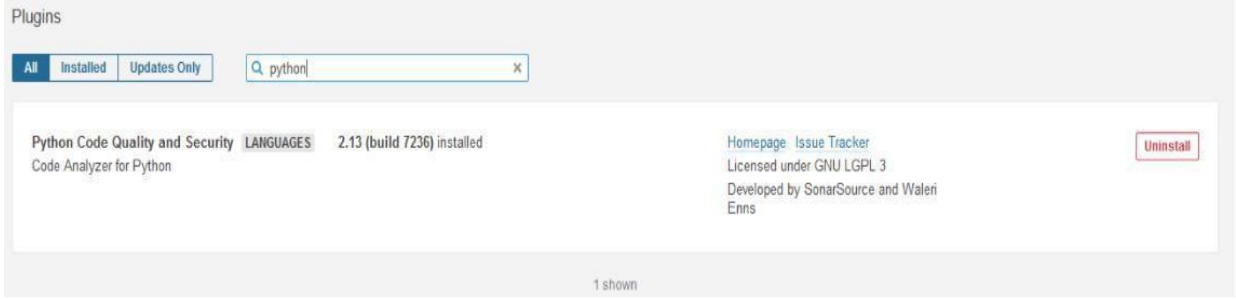
SonarQube Setup

Step 1: SonarQube Instance

```
$ docker run -d -p 9000:9000 sonarqube
```

Step 2: Generate user token

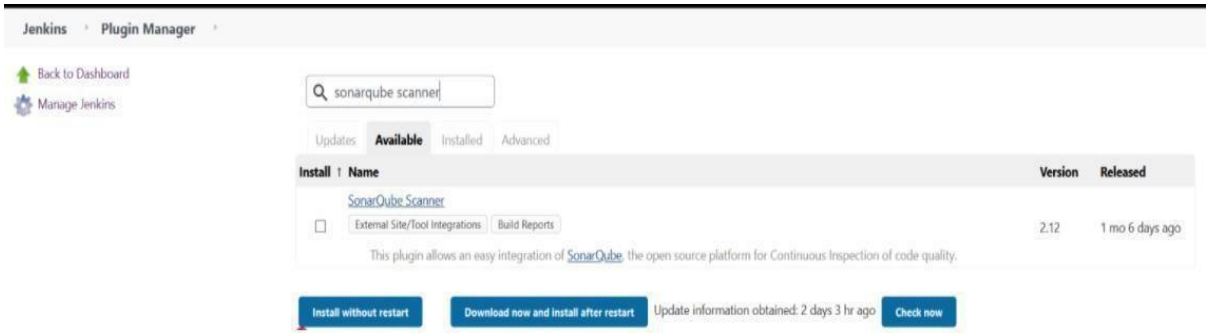




Step 3: Add necessary plugin

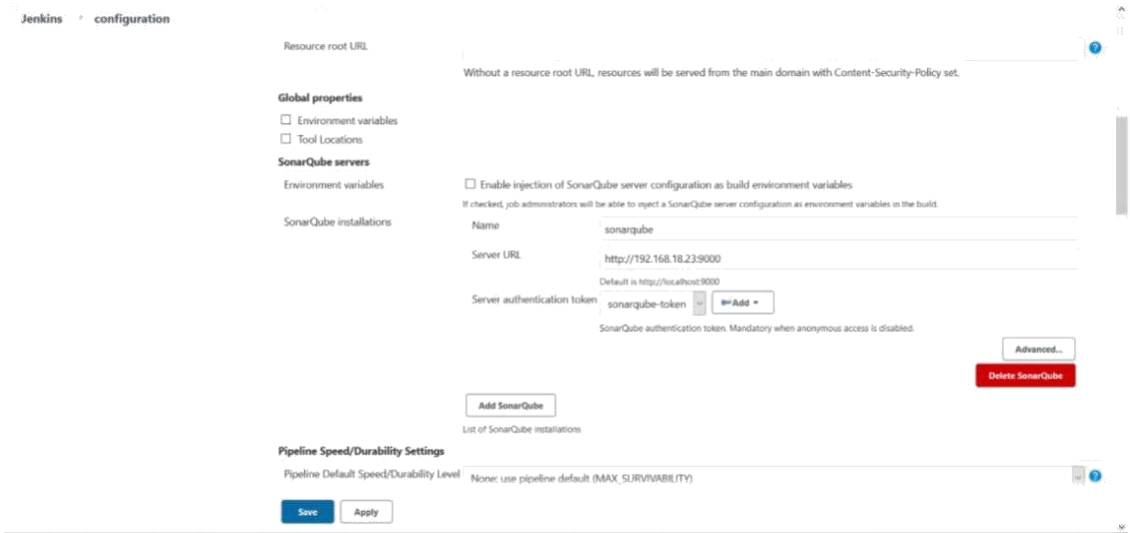
```
sonar.projectKey=movie-crud-flask
sonar.projectName=movie-crud-flask
sonar.projectVersion=1.0
sonar.projectBaseDir=.
sonar.python.bandit.reportPaths=/report/banditResult.json
```

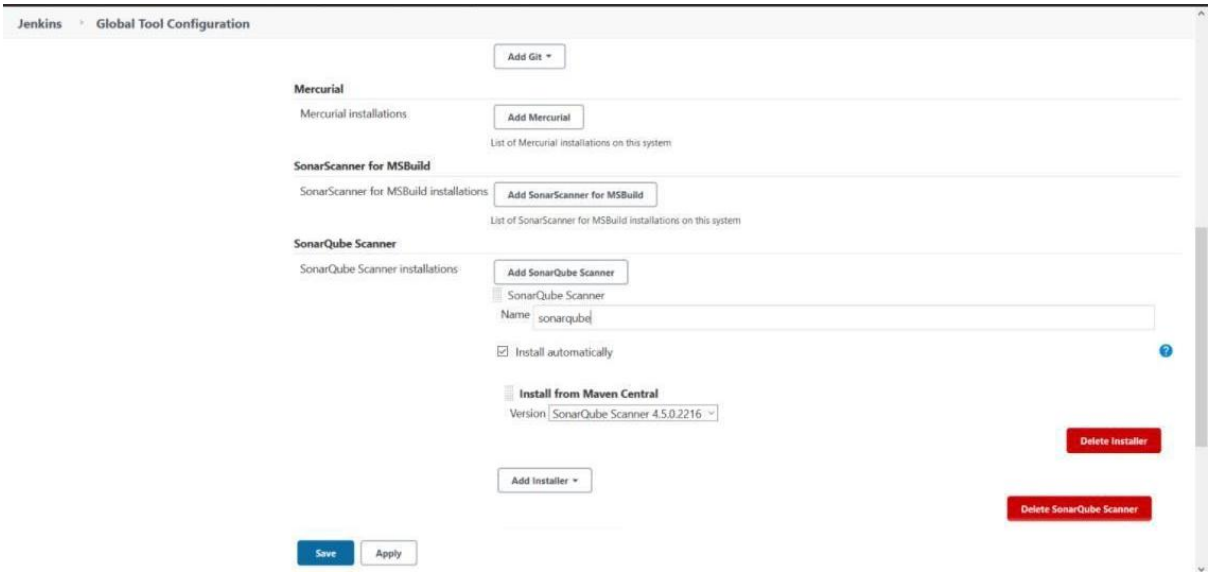
Step 4: Configuring SonarQube in Codebase



Step 5: Jenkins Setup for SonarQube

Step 6: Tool configuration with SonarQube Scanner



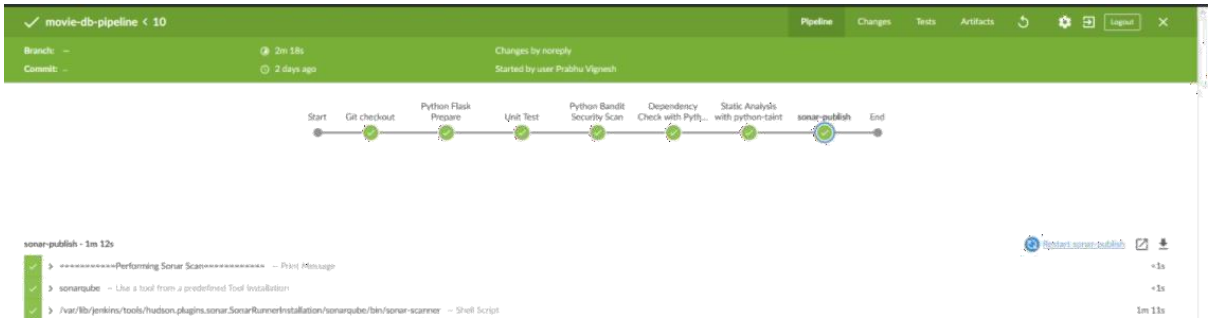


Step 7: SonarQube Scanner in Jenkins Pipeline

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

```
pipeline {
    agent any
    stages {
        stage ("Git checkout"){
            steps {
                git branch: "master",
                    url: "https://github.com/PrabhuVignesh/movie-crud-flask.git"
                sh "ls"
            }
        }
        stage ("Python Flask Prepare"){
            steps {
                sh "pip3 install -r requirements.txt"
            }

        }
        stage ("Unit Test"){
            steps{
                sh "python3 test_basic.py"
            }
        }
        stage ("Python Bandit Security Scan"){
            steps{
                sh "cat report/banditResult.json"
                sh "sh run_bandit.sh || true"
                sh "ls"
            }
        }
        stage ("Dependency Check with Python Safety"){
            steps{
                sh "docker run --rm --volume \$(pwd) pyupio/safety:latest safety check"
                sh "docker run --rm --volume \$(pwd) pyupio/safety:latest safety check --json > report.json"
            }
        }
        stage ("Static Analysis with python-taint"){
            steps{
                sh "docker run --rm --volume \$(pwd) vickyrajagopal/python-taint-docker pyt ."
            }
        }
        stage ("sonar-publish"){
            steps {
                echo "==========Performing Sonar Scan=========="
                sh "${tool("sonarqube")}/bin/sonar-scanner"
            }
        }

    }
}
```

Step 8: Check the output



CONCLUSION : Hence, we Successfully understood Static Analysis SAST process and learnt to integrate Jenkins SAST to SonarQube/GitLab.

EXPERIMENT NO . 8

AIM :- 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.

THEORY :-

SonarQube is a universal tool for static code analysis that has become more or less the industry standard. Keeping code clean, simple, and easy to read is also a lot easier with SonarQube. SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.

Benefits of SonarQube

1. Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.
2. Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
3. Quality code - Code quality control is an inseparable part of the process of software development.
4. Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
5. Increase consistency - Determines where the code criteria are breached and enhances the quality

Business scaling - No restriction on the number of projects to be evaluated.

6. Enhance developer skills - Regular feedback on quality problems helps developers to improve their coding skills.

SonarQube Setup

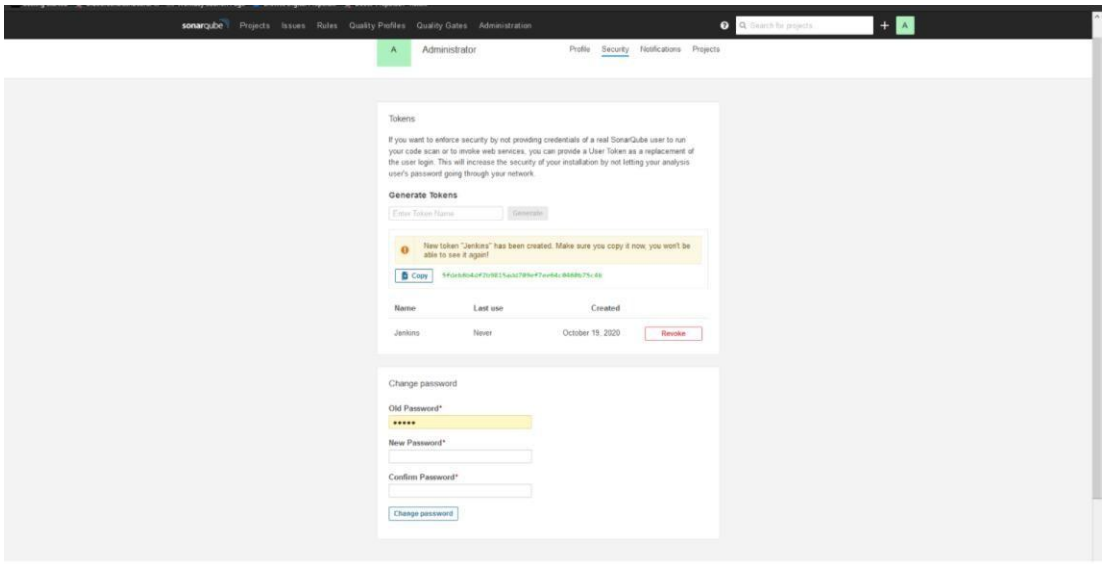
SonarQube Instance

Before proceeding with the integration, we will setup SonarQube Instance. Choice of the platform is yours.

```
$ docker run -d -p 9000:9000 sonarqube
```

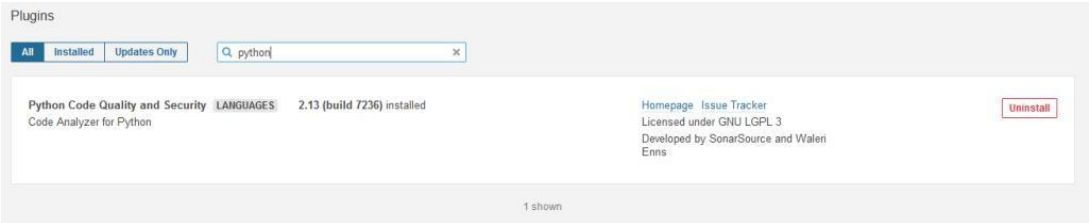
Generate User Token

Now, we need to get the SonarQube user token to make connection between Jenkins and SonarQube. For the same, go to User > My Account > Security and then, from the bottom of the page you can create new tokens by clicking the Generate Button. Copy the Token and keep it safe.

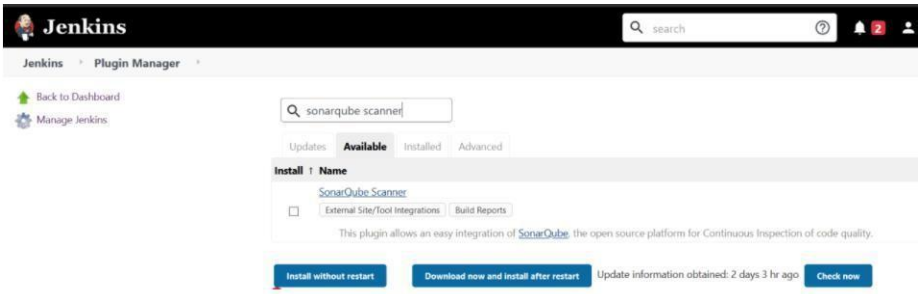


Add necessary plugin

In this Tutorial, we are following a Python-based application. So, we need to add a python plugin in the SonarQube so that it will collect the Bugs and Static code analysis from Jenkins. For the same, go to Administration > Marketplace > Plugins. Then in the search box, search for Python. Then, you will see Python Code Quality and Security (Code Analyzer for Python). Just install. That's all from the SonarQube side.

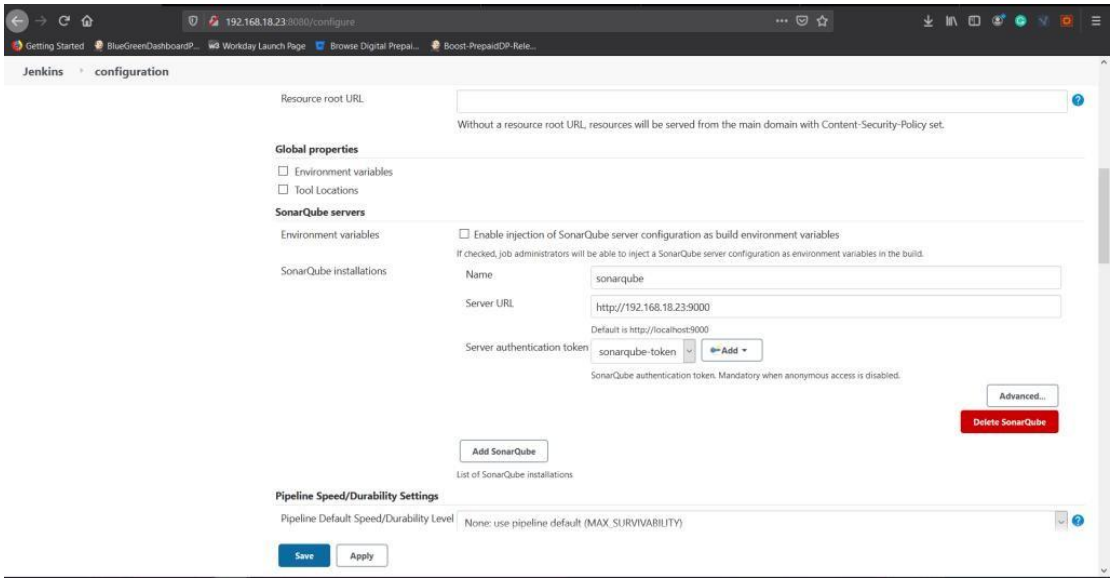


Jenkins Setup for SonarQube
SonarQube plugin in Jenkins.
Before all, we need to install the SonarQube Scanner plugin in Jenkins. For the same, go to Manage Jenkins > Plugin Manager > Available. From here, type SonarQube Scanner then select and install.

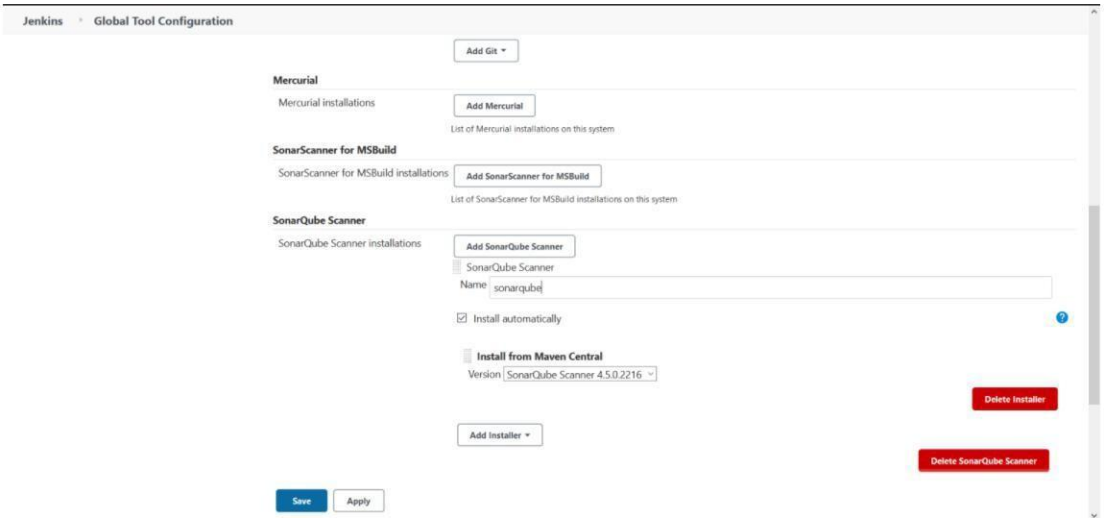


SonarQube Scanner Plugin for Jenkins

Tool Configuration SonarQube Scanner
Now, we need to configure the Jenkins plugin for SonarQube Scanner to make a connection with the SonarQube Instance. For that, got to Manage Jenkins > Configure System > SonarQube Server. Then, Add SonarQube. In this, give the Installation Name, Server URL then Add the Authentication token in the Jenkins Credential Manager and select the same in the configuration.



Then, we need to set-up the SonarQube Scanner to scan the source code in the various stage. For the same, go to Manage Jenkins > Global Tool Configuration > SonarQube Scanner. Then, Click Add SonarQube Scanner Button. From there, give some name of the scanner type and Add Installer of your choice. In this case, I have selected SonarQube Scanner from Maven Central.



SonarQube Scanner in Jenkins Pipeline

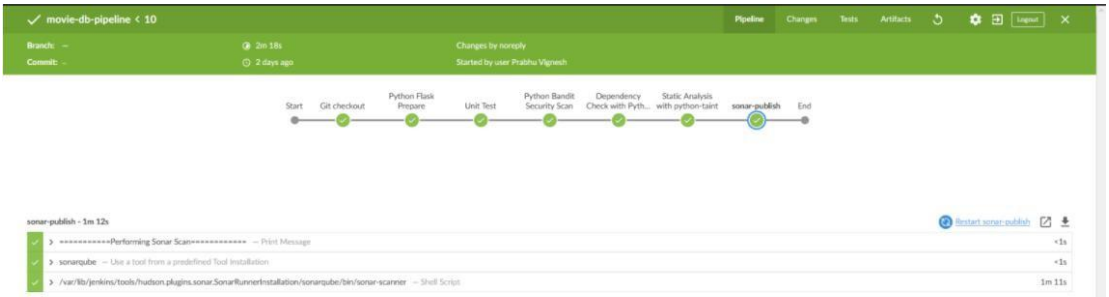
Now, It's time to integrate the SonarQube Scanner in the Jenkins Pipeline. For the same, we are going to add one more stage in the Jenkinsfile called sonar-publish.

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

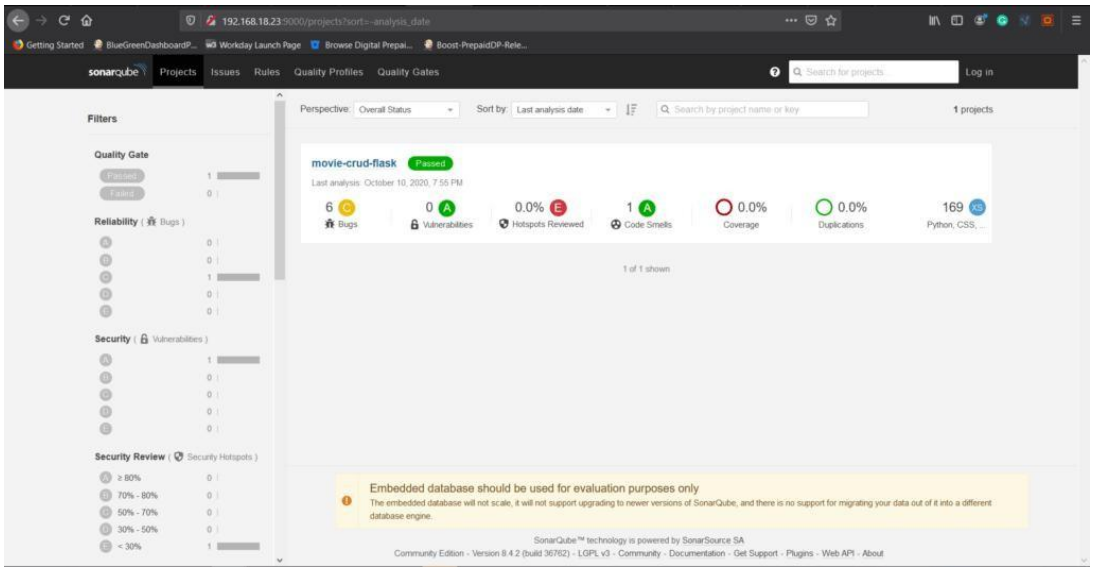
Were this will collect the SonarQube Server information from the sonar-project.properties file and publish the collected information to the SonarQube Server. So, the overall code will look like the below snippet.

```
pipeline {
        agent any
        stages {
                stage ("Git
                        checkout"){ step
                        s {
                                git branch: "master",
                                        url:
"https://github.com/PrabhuVignesh/movie-crud-flask.git"
                                sh "ls"
                        }
                }
                stage ("Python Flask
                        Prepare"){ steps {
                                sh "pip3 install -r requirements.txt"
                        }

                }
                stage ("Unit
                        Test"){ steps{
                                sh "python3 test_basic.py"
                        }
                }
                stage ("Python Bandit Security Scan"){
```

```
                steps{
                        sh "cat report/banditResult.json"
                        sh "sh run_bandit.sh || true"
                        sh "ls"
                }
        }
        stage ("Dependency Check with Python
                Safety"){ steps{
                        sh "docker run --rm --volume \$(pwd)
pyupio/safety:latest safety check"
                        sh "docker run --rm --volume \$(pwd)
pyupio/safety:latest safety check --json > report.json"
                }
        }
        stage ("Static Analysis with python-
                taint"){ steps{
                        sh "docker run --rm --volume \$(pwd)
vickyrajagopal/python-taint-docker pyt ."
                }
        }
        stage ("sonar-
                publish"){ steps {
                        echo "===========Performing Sonar
Scan============"
                        sh "${tool("sonarqube")}/bin/sonar-
scanner"
                }
        }

    }
}
```



Where it will just execute the SonarQube Scanner and collect the SAST information and Python bandit report in the format of JSON. Then, it will publish the same in the SonarQube Server. If you login to the SonarQube and visit the Dashboard, you will see the Analysis of the project there.



CONCLUSION :- Hence , Creating 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 is successfully implemented.

EXPERIMENT NO. : 09

AIM : To understand Continuous monitoring and Installation and configuration of Nagios Core, Nagios Plugins and NRPE on Linux machine

THEORY :

Nagios is a popular and open-source application that is used for continuous monitoring of systems, networks, services and applications. It constantly monitors the status of machines and various services. In case of any issue, it provides early warning so that administrator can take required actions. Nagios performs all checks on local and host machines using the external-programs known as plugins. It also provides you with a web interface that allows viewing the status of hosts and services, history, logs and generating reports.

Nagios was originally designed to run under Linux, but it also runs on other Unix variants. It is free software licensed under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

Installation and configuration of Nagios Core

Installing Nagios Core involves downloading the Nagios Core source code; then, configuring, making and installing it on thehost that will run Nagios Core instance.

Installing Nagios Prerequisites

```
[user@nagios]# yum install -y httpd php php-cli gcc glibc glibc-common gd gd-devel net-snm
```

Open port 80 for httpd

```
[user@nagios]# firewall-cmd --zone=public --add-port=80/tcp
[user@nagios]# firewall-cmd --zone=public --add-port=80/tcp --permanent
```

Creating a Nagios User and Group

Create a user and group for Nagios Core.

```
[user@nagios]# useradd nagios
[user@nagios]# passwd nagios
[user@nagios]# groupadd nagcmd
[user@nagios]# usermod -a -G nagcmd nagios
```

Then, execute the following:

```
[user@nagios]# usermod -a -G nagcmd apache
```

Download Nagios Source Code and Plug-Ins

Download the latest version of Nagios Core and Plugins

```
[user@nagios]# wget --inet4-only https://assets.nagios.com/downloads/nagioscore/releases/
[user@nagios]# wget --inet4-only http://www.nagios-plugins.org/download/nagios-plugins-2.
[user@nagios]# tar zxf nagios-4.3.1.tar.gz
[user@nagios]# tar zxf nagios-plugins-2.2.1.tar.gz
[user@nagios]# cd nagios-4.3.1
```

Make and Install Nagios Core

To make and install Nagios Core, first run ./configure.

```
[user@nagios]# ./configure --with-command-group=nagcmd
```

After running `./configure`, compile the Nagios Core source code.

```
[user@nagios]# make all
```

After making Nagios Core, install it.

```
[user@nagios]# make install
[user@nagios]# make install-init
[user@nagios]# make install-config
[user@nagios]# make install-commandmode
[user@nagios]# make install-webconf
```

Copy the event handlers and change their ownership.

```
[user@nagios]# cp -R contrib/eventhandlers/ /usr/local/nagios/libexec/
[user@nagios]# chown -R nagios:nagios /usr/local/nagios/libexec/eventhandlers
```

Finally, run the pre-flight check.

```
[user@nagios]# /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

Installation and configuration of **Nagios Plug-Ins**

Download the latest version of the Nagios plug-ins. Then, makeand install them.

```
[user@mon]# wget http://www.nagios-plugins.org/download/nagios-plugins-2.2.1.tar.gz
[user@mon]# tar zxf nagios-plugins-2.2.1.tar.gz
[user@mon]# cd nagios-plugins-2.2.1
[user@mon]# ./configure
[user@mon]# make
[user@mon]# make install
```

Installation and configuration of NRPE (Nagios RemotePlugin Executor)

```
[user@mon]# cd ~
[user@mon]# wget https://github.com/NagiosEnterprises/nrpe/releases/download/nrpe-3.1.0/n
[user@mon]# tar xvfz nrpe-3.1.0.tar.gz
[user@mon]# cd nrpe-3.1.0
[user@mon]# ./configure
[user@mon]# make all
[user@mon]# make install-groups-users
[user@mon]# make install
[user@mon]# make install-config
[user@mon]# make install-init
```

Enable and Start NRPE

```
[user@mon]# systemctl enable nrpe
[user@mon]# systemctl start nrpe
```

Open Port 5666

Open port 5666 to allow communication with NRPE.

```
[user@mon]# firewall-cmd --zone=public --add-port=5666/tcp
[user@mon]# firewall-cmd --zone=public --add-port=5666/tcp --permanent
```

Edit the NRPE configuration with the Nagios server's IP address.

```
[user@mon]# vim /usr/local/nagios/etc/nrpe.cfg
```

```
allowed_hosts=127.0.0.1,<ip-address-of-nagios-core>
```

Add the IP address of the Nagios Core server tothe allowed_hosts setting. Then, restart nrpe

```
[user@mon]# systemctl restart nrpe
```

Test the Installation

Ensure that the make and install procedures worked.

CONCLUSION :- Hence , we successfully understood Continuousmonitoring and Installation and configuration of Nagios
Core, Nagios Plugins and NRPE (Nagios Remote Plugin Executor)on Linux Machine is implemented.

Experiment no . 10

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

Theory :

Linux – Network Monitoring Tools
Network monitoring is using a system (hardware or software) that continuously observes your network and the data flows through it, depending on how the monitoring solution actually functions and informs the network administrator. We can keep a check on all the activities of our network easily. While Network management we need network Monitoring.

To monitor Windows Machines you will need to follow several steps and they are:

Install NSClient++ addon on the Windows Machine.
Configure Nagios Server for monitoring Windows Machine.
Add new host and service definitions for Windows machine monitoring.
Restart the Nagios Service.
To make this guide simple and easier, a few of configuration already done for you in the Nagios installation.

A check_nt command definition already added to the command.cfg file. This definition command is used by check_nt plugin to monitor Windows services.
A windows-server host template already created in the templates.cfg file. This template allows you to add new Windows host definitions.

Check Nagios Configuration path
Login to Nagios Server. Use the following command to check the Nagios configuration path.
$ ps -ef | grep nagios

```
[root@devopsmyway ec2-user]# ps -ef | grep nagios
nagios     2694     1  0 12:50 ?        00:00:00 /usr/sbin/nagios -d /etc/nagios/nagios.cfg
nagios     2697  2694  0 12:50 ?        00:00:00 /usr/sbin/nagios --worker /var/spool/nagios/cmd/nagios.qh
nagios     2698  2694  0 12:50 ?        00:00:00 /usr/sbin/nagios --worker /var/spool/nagios/cmd/nagios.qh
nagios     2699  2694  0 12:50 ?        00:00:00 /usr/sbin/nagios --worker /var/spool/nagios/cmd/nagios.qh
nagios     2700  2694  0 12:50 ?        00:00:00 /usr/sbin/nagios --worker /var/spool/nagios/cmd/nagios.qh
nagios     2702  2694  0 12:50 ?        00:00:00 /usr/sbin/nagios -d /etc/nagios/nagios.cfg
root       3103  2904  0 12:55 pts/0    00:00:00 grep --color=auto nagios
[root@devopsmyway ec2-user]#
```

Create config files for Windows and Linux host
Create a directory, say montitorhosts in /etc/nagios/objects/
$ mkdir /etc/nagios/objects/monitorhosts

```
[root@ip-172-31-25-189 ec2-user]# mkdir /etc/nagios/objects/monitorhosts
```

Create two directories, say linuxhosts and windowshosts
in/etc/nagios/objects/monitorhosts/
$ mkdir /etc/nagios/objects/monitorhosts/windowshosts

```
[root@ip-172-31-25-189 ec2-user]# mkdir /etc/nagios/objects/monitorhosts/windowshosts
```

$ mkdir /etc/nagios/objects/monitorhosts/linuxhosts

```
[root@ip-172-31-25-189 ec2-user]# mkdir /etc/nagios/objects/monitorhosts/linuxhosts
```

$ cp /etc/nagios/objects/windows.cfg
/etc/nagios/objects/monitorhosts/windowshosts/windowsserver.cfg

```
[root@devopsmyway objects]# cp /etc/nagios/objects/windows.cfg /etc/nagios/objects/monitorhosts/windowshost
s/windowsserver.cfg
[root@devopsmyway objects]#
```

$ cp /etc/nagios/objects/localhost.cfg
/etc/nagios/objects/monitorhosts/linuxhosts/linuxserver.cfg

```
[root@devopsmyway objects]# cp /etc/nagios/objects/localhost.cfg /etc/nagios/objects/monitorhosts/linuxhost
s/linuxserver.cfg
[root@devopsmyway objects]#
```

$ nano /etc/nagios/objects/monitorhosts/windowshosts/windowsserver.cfg

```
define host {

    use                     windows-server          ; Inherit default values from a template
    host_name               winserver               ; The name we're giving to this host
    alias                   My Windows Server       ; A longer name associated with the host
    address                 172.31.28.185            ; IP address of the host
}
```

```
define service {

    use                     generic-service
    host_name               winserver
    service_description     W3SVC
    check_command           check_nt!SERVICESTATE!-d SHOWALL -l W3SVC
}

# Create a service for monitoring the Explorer.exe process
# Change the host_name to match the name of the host you defined above

define service {

    use                     generic-service
    host_name               winserver
    service_description     Explorer
    check_command           check_nt!PROCSTATE!-d SHOWALL -l Explorer.exe
}
```

$ nano /etc/nagios/objects/monitorhosts/linuxhosts/linuxserver.cfg

```
define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.
    host_name               linuxserver
    alias                   linuxserver
    address                 172.31.25.189
}
```

```
define service {

    use                     local-service           ; Name of service template to use
    host_name               linuxserver
    service_description     Total Processes
    check_command           check_local_procs!250!400!RSZDT
}



# Define a service to check the load on the local machine.

define service {

    use                     local-service           ; Name of service template to use
    host_name               linuxserver
    service_description     Current Load
    check_command           check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
}
```

```
# Define an optional hostgroup for Linux machines

define hostgroup {

    hostgroup_name          linux-servers1          ; The name of the hostgroup
    alias                   Linux Servers          ; Long name of the group
    members                 linuxserver             ; Comma separated list of hosts that belong to this group
}
```

$ nano /etc/nagios/nagios.cfg

cfg_dir=/etc/nagios/objects/monitorhosts

```
# directive as shown below:

#cfg_dir=/etc/nagios/servers
#cfg_dir=/etc/nagios/printers
#cfg_dir=/etc/nagios/switches
#cfg_dir=/etc/nagios/routers

cfg_dir=/etc/nagios/objects/monitorhosts
```

Check the Nagios Configuration
$ /usr/sbin/nagios -v /etc/nagios/nagios.cfg

```
[root@devopsmyway ec2-user]# /usr/sbin/nagios -v /etc/nagios/nagios.cfg
```

```
Running pre-flight check on configuration data...

Checking objects...
        Checked 23 services.
        Checked 3 hosts.
        Checked 3 host groups.
        Checked 0 service groups.
        Checked 1 contacts.
        Checked 1 contact groups.
        Checked 24 commands.
        Checked 5 time periods.
        Checked 0 host escalations.
        Checked 0 service escalations.
Checking for circular paths...
        Checked 3 hosts
        Checked 0 service dependencies
        Checked 0 host dependencies
        Checked 5 timeperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0
```

Restart Nagios Service
$ service nagios restart

```
[root@devopsmyway ec2-user]# service nagios restart
Redirecting to /bin/systemctl restart nagios.service
[root@devopsmyway ec2-user]#
```

Configuration in Linux host
Login to Linux Server and Install nrpe plugin.

$ sudo yum install nrpe -y

```
[root@linuxserver ec2-user]# sudo yum install nrpe -y
```

Open nrpe config file

$ nano /etc/nagios/nrpe.cfg

```
[root@linuxserver ec2-user]# nano /etc/nagios/nrpe.cfg
```

Put the IP address of Nagios Server in allowed_hosts in nrpe.cfg.

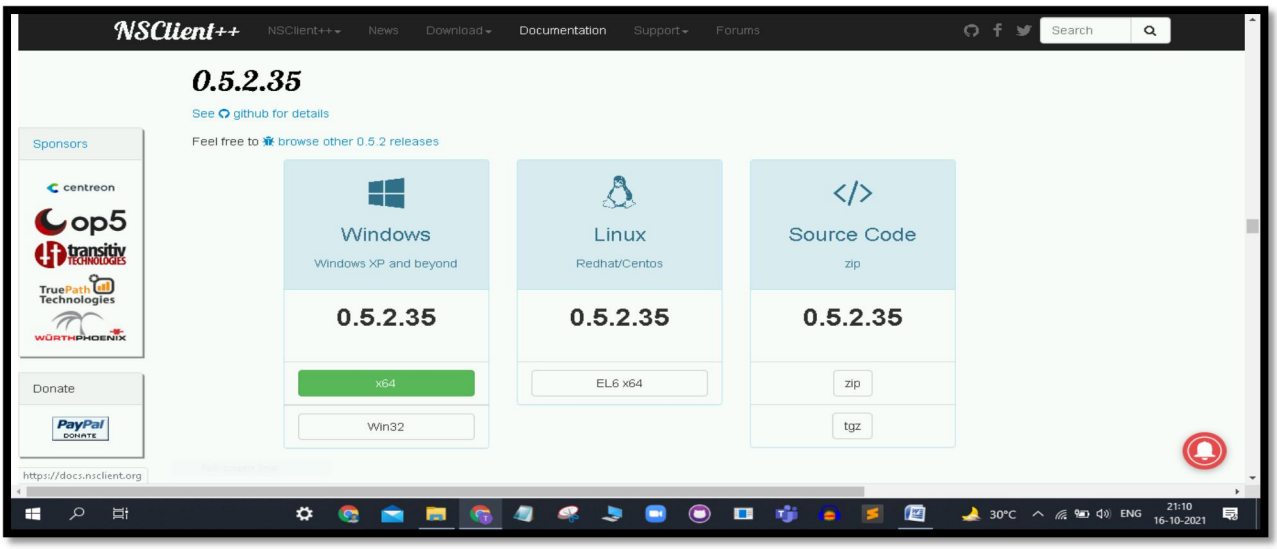
```
allowed_hosts=127.0.0.1,172.31.22.60
```

Restart nrpe service

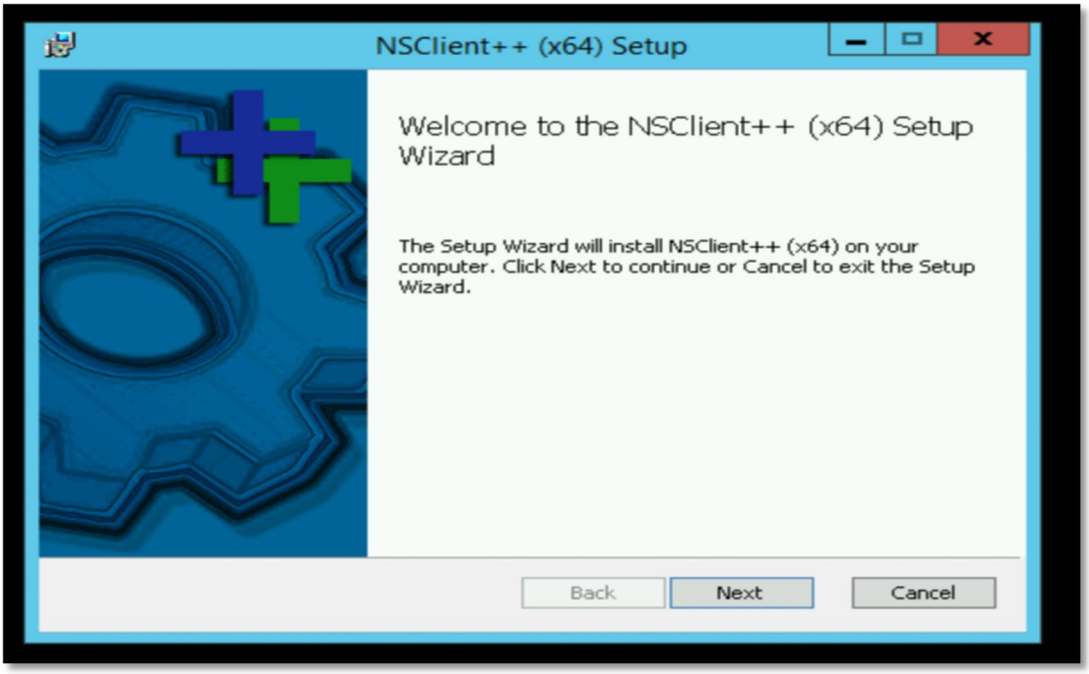
```
[root@devopsmyway ec2-user]# service nrpe restart
Redirecting to /bin/systemctl restart nrpe.service
[root@devopsmyway ec2-user]#
```

Configuration in Windows host
Log in to your Windows Server and download nsclient++ and install it. You can use the following link to download the nsclient++ for windows.

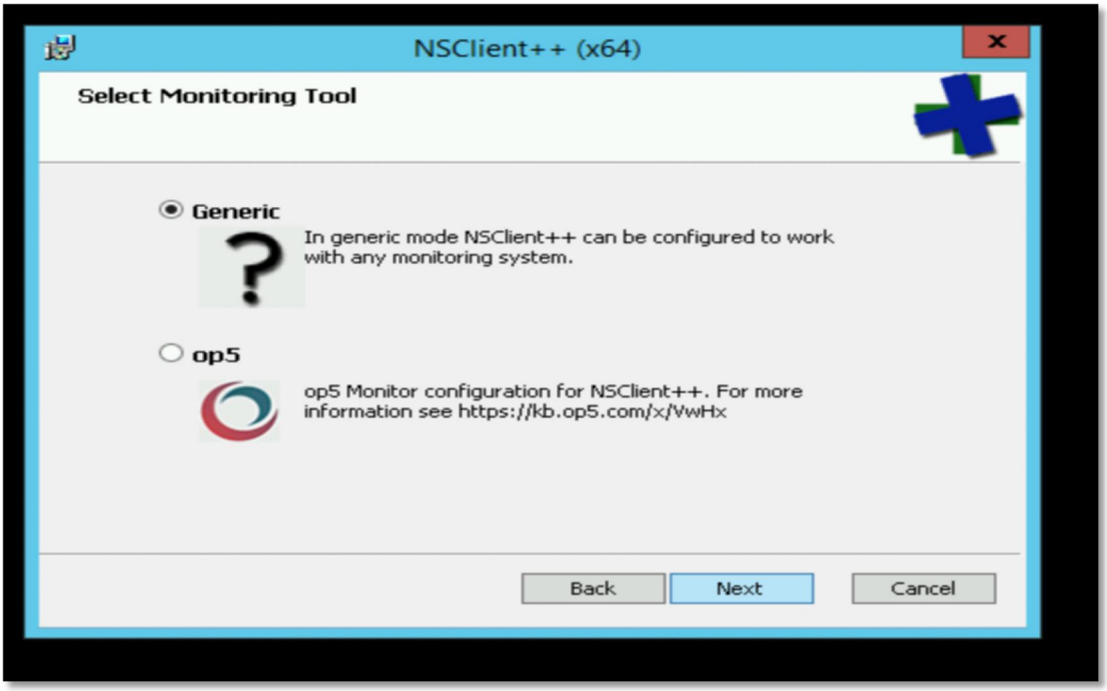
http://nsclient.org/download/



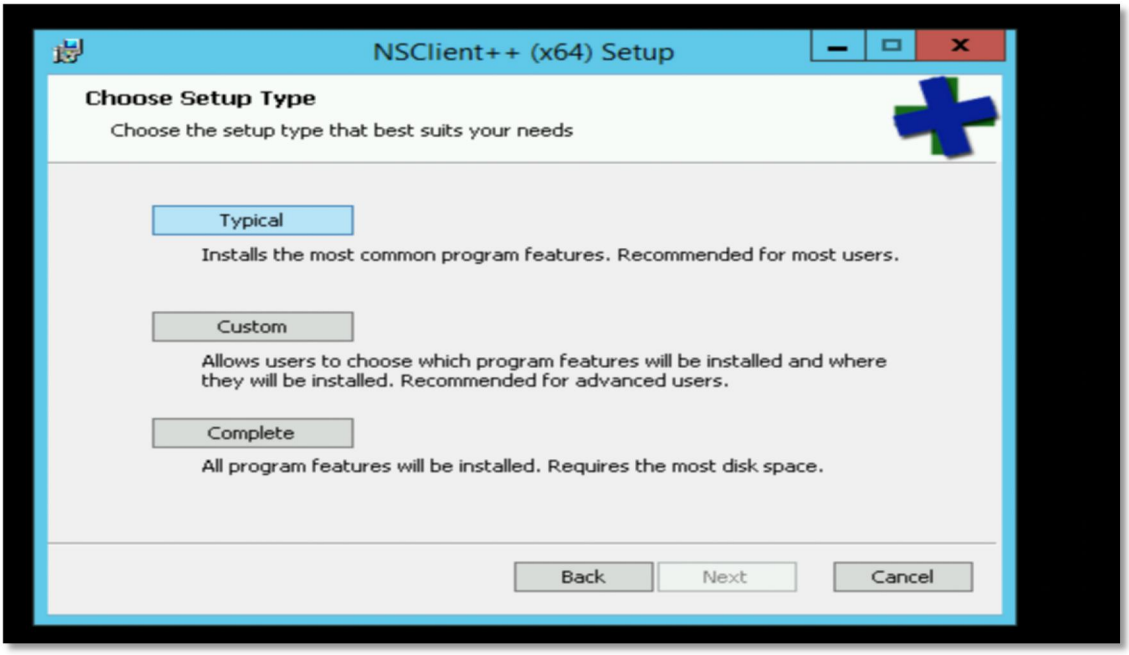
Install Nsclient++ in your Windows Server.



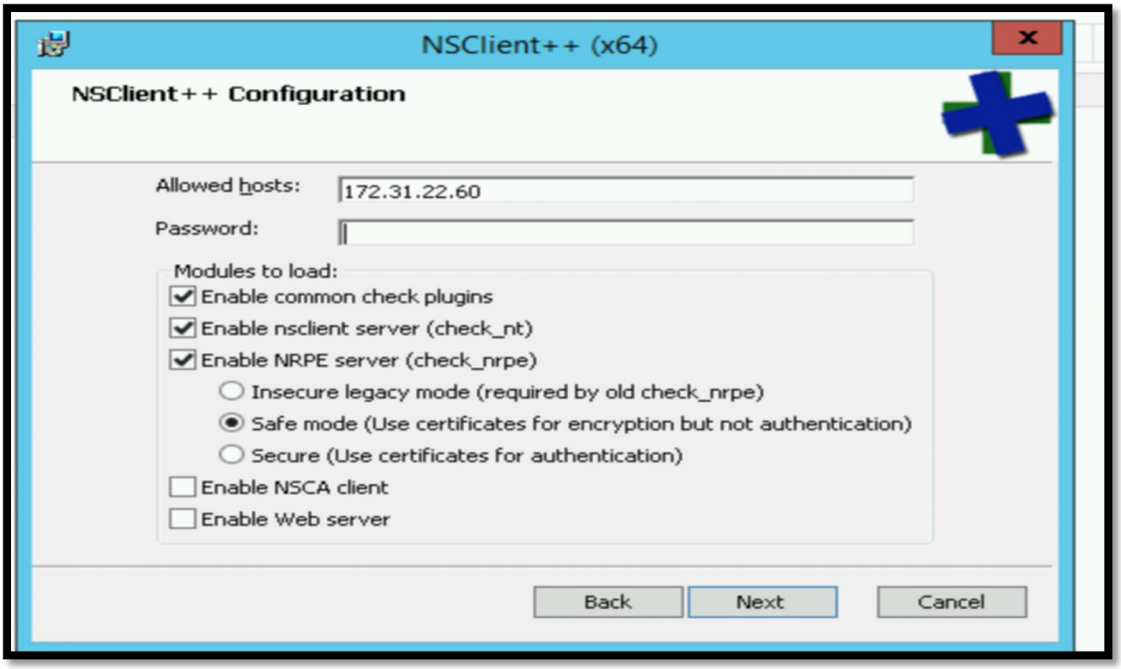
Select Generic



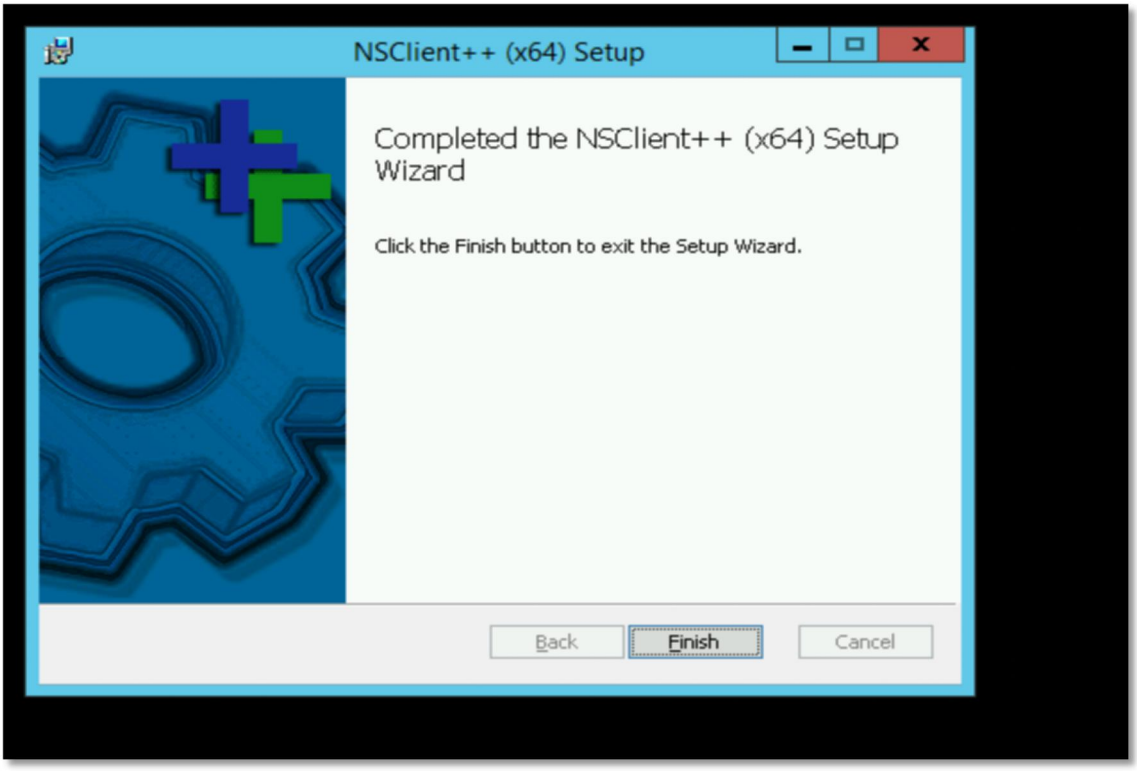
Select Typical



Enter Nagios Server IP Address in Allowed Hosts and tick mark the modules as mentioned in the below screenshot.



Click on Finish



nsclient.ini settings

Now open the following file as run as administrator in your Windows Server

C:\Program Files\NSClient++\nsclient.ini

CheckExternalScripts = enabled

CheckHelpers = enabled

CheckEventLog = enabled

CheckNSCP = enabled

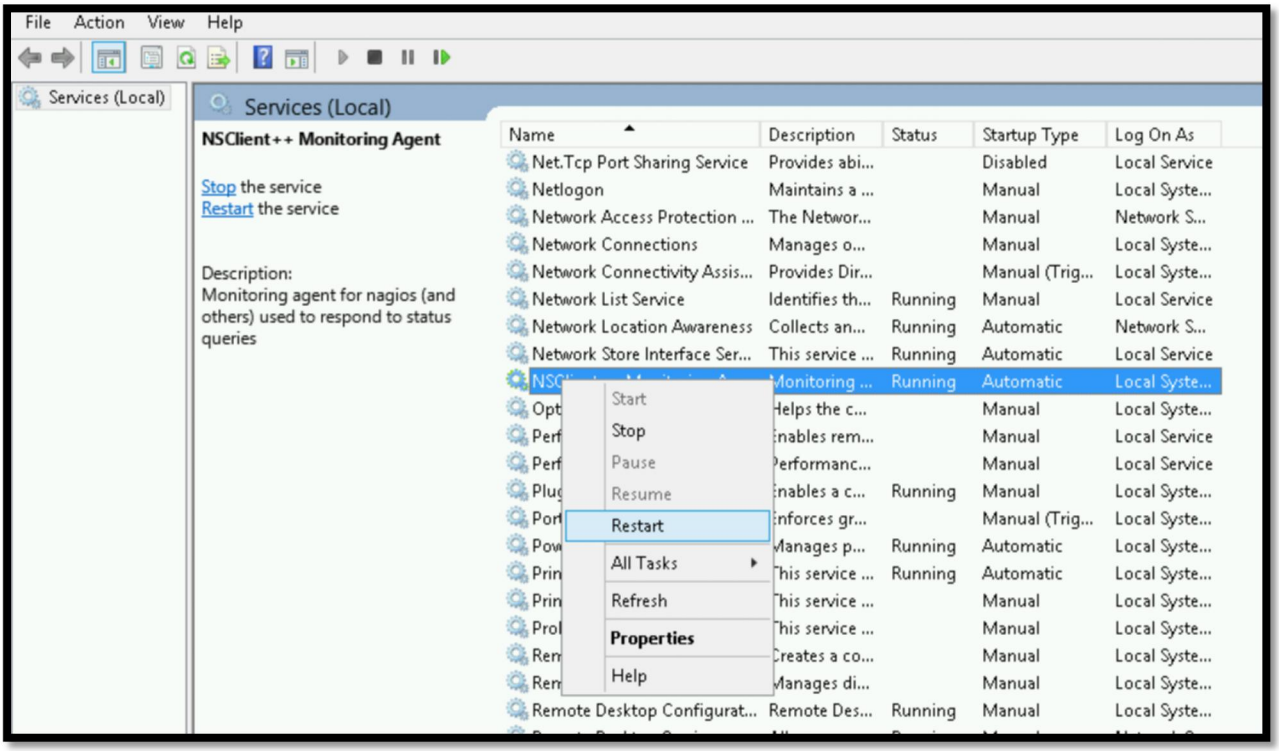
CheckDisk = enabled

CheckSystem = enabled

NSClientServer = enabled

NRPEServer = enabled

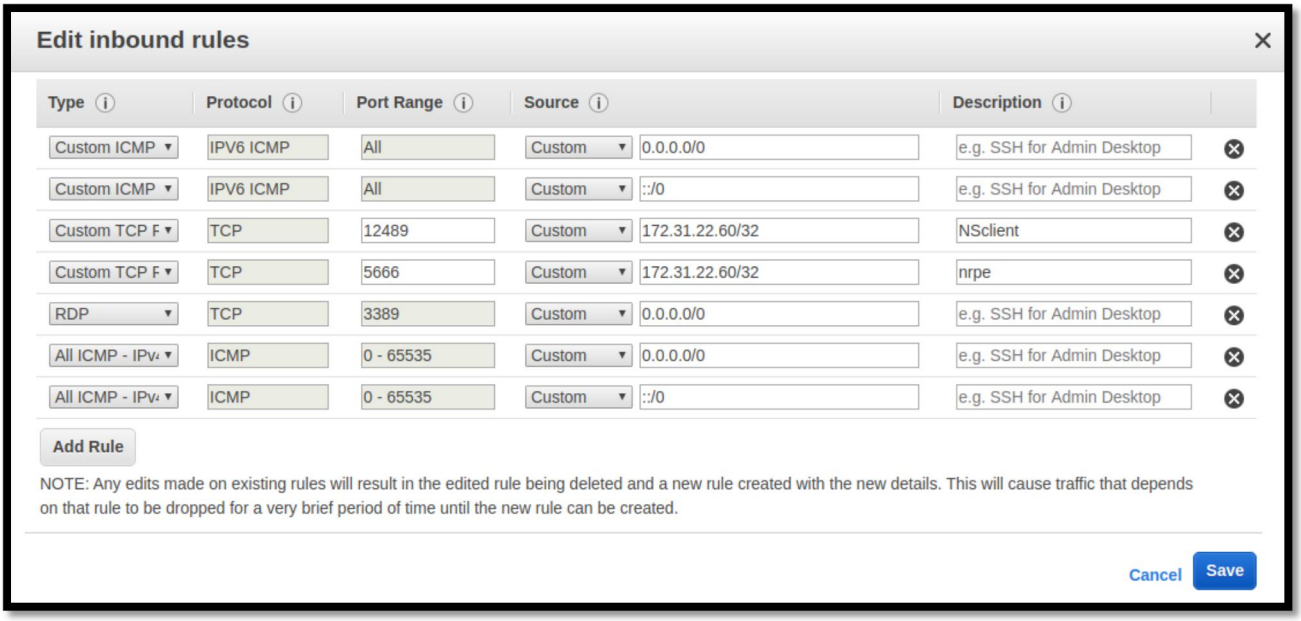
After changes restart the nsclient++ service in services.



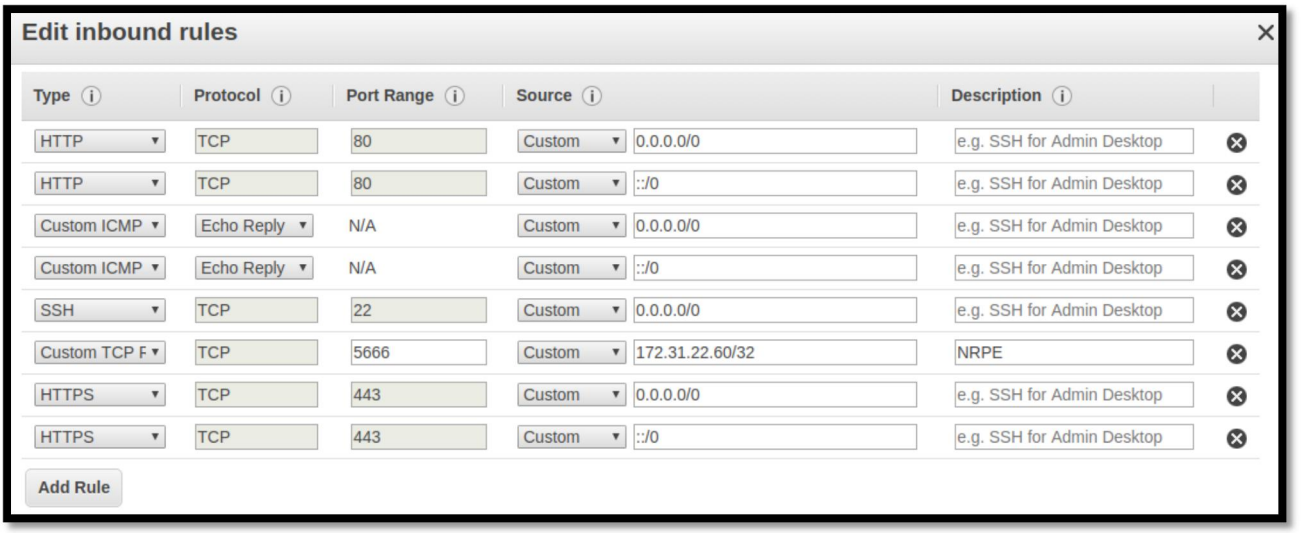
We are now all done in our Windows Server.

AWS Security Group Configuration for Windows and Linux Server

Open Security Group for Windows Server and allow port 5666 and 12489 and ICMP for Nagios Server IP.



Open Security Group for Linux Server and allow port 5666 and ICMP port for Nagios Server IP.



Note: If your servers are not in the AWS environment, you can allow these ports in the local firewall of both the servers.

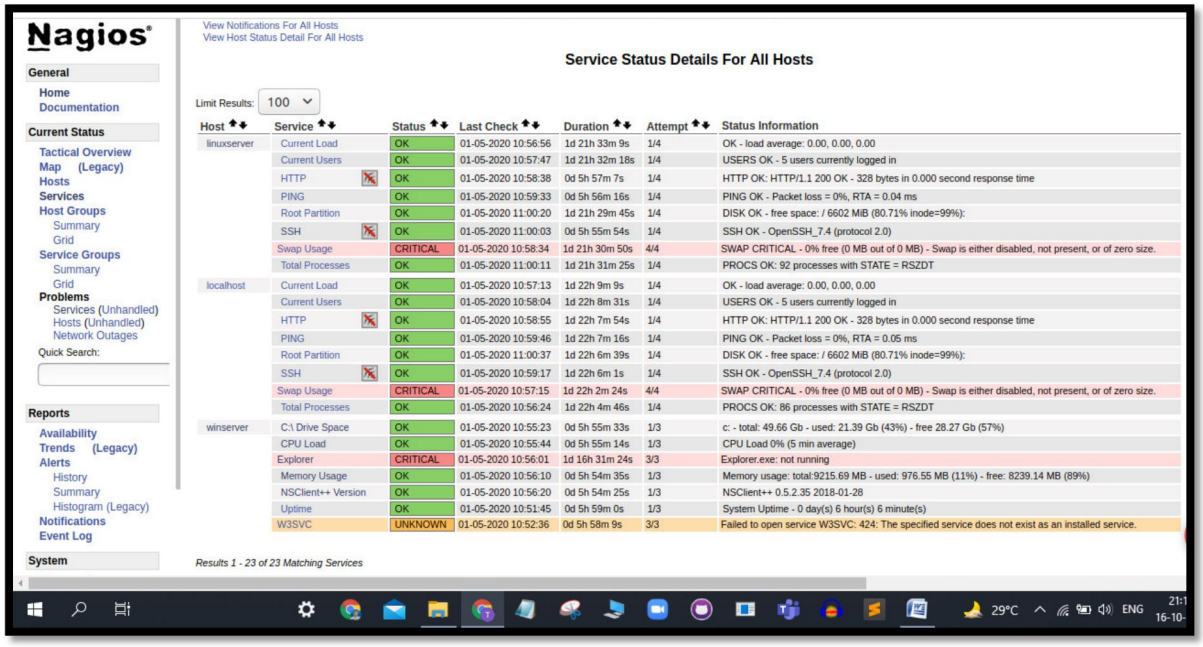
Monitor Windows and Linux Host

Now your both Linux and Windows Servers are ready to Monitor. You can monitor your servers using the following URL.

http://NagiosServerPublicIP/nagios

Default Username: nagiosadmin

Default Password : nagiosadmin



Conclusion:

Hence, We successfully performed Port, Service monitoring, Windows/Linux server monitoring using Nagios.

Assignment no. 1

AWS CodeStar:

AWS CodeStar enables you to quickly develop, build, and deploy applications on AWS. AWS CodeStar provides a unified user interface, enabling you to easily manage your software development activities in one place. With AWS CodeStar, you can set up your entire continuous delivery toolchain in minutes, allowing you to start releasing code faster. AWS CodeStar makes it easy for your whole team to work together securely, allowing you to easily manage access and add owners, contributors, and viewers to your projects. Each AWS CodeStar project comes with a project management dashboard, including an integrated issue tracking capability powered by Atlassian JIRA Software. With the AWS CodeStar project dashboard, you can easily track progress across your entire software development process, from your backlog of work items to teams' recent code deployments. Visit here to learn more.

There is no additional charge for using AWS CodeStar. You only pay for the AWSresources that you provision for developing and running your application (for example, Amazon EC2 instances).

AWS CodeStar feature

- Project templates

AWS CodeStar provides a number of project templates to help you quickly start developing applications for deployment on Amazon EC2, AWS Lambda, andAWS Elastic Beanstalk with support for many popular programming languages including Java, JavaScript, Python, Ruby, and PHP. With AWS CodeStar, you can use a code editor of your choice such as Visual Studio, Eclipse, or the o Team access management

AWS CodeStar uses AWS Identity and Access Management (IAM) to manage developer identities and provides built-in, role-based security policies that allow you to easily secure access to your team. AWS CodeStar allows you to share your projects using three levels of access:

i. Owners
ii. Contributors
iii. Viewers

- Hosted Git repository

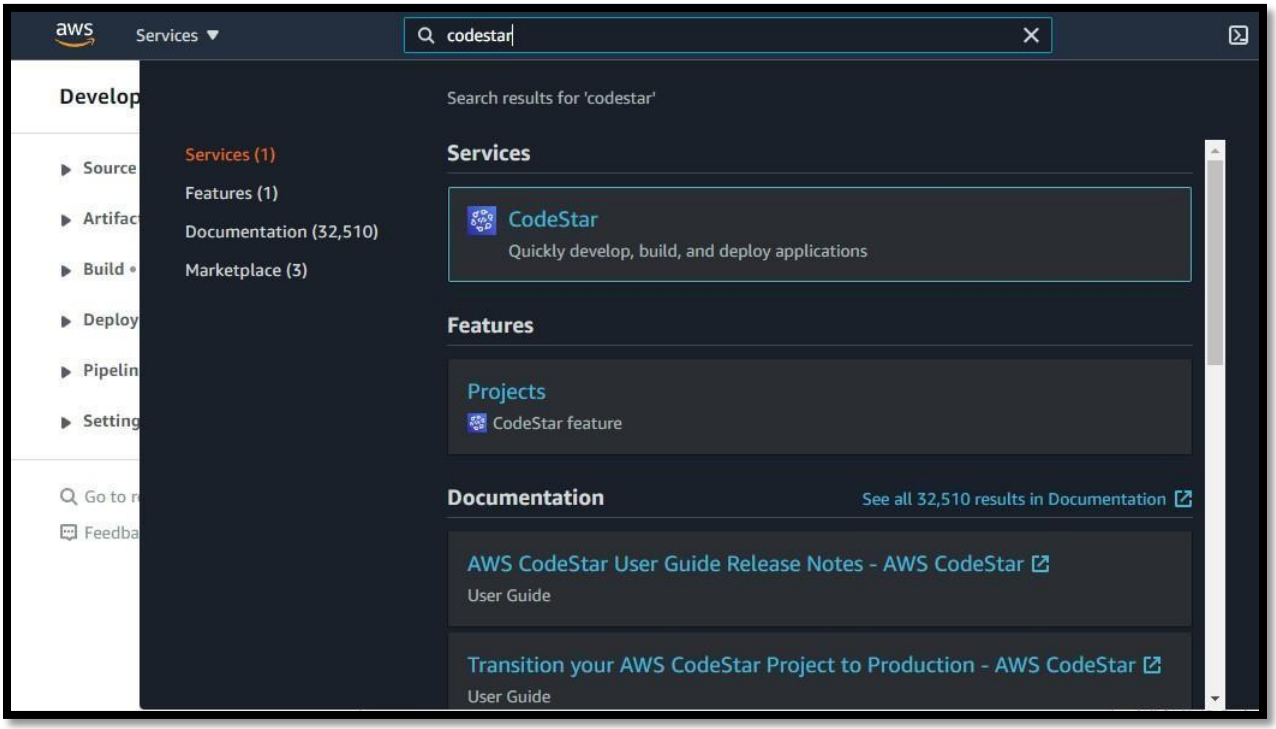
AWS CodeStar stores your application code securely on AWS CodeCommit, a fullymanaged source control service that eliminates the need to manage your own infrastructure to host Git repositories. You can also choose to have your project source code stored in a GitHub repository in your own GitHub account. o Fully managed build service

AWS CodeStar compiles and packages your source code with AWS CodeBuild, a fullymanaged build service that makes it possible for you to build, test, and integrate code more frequently.

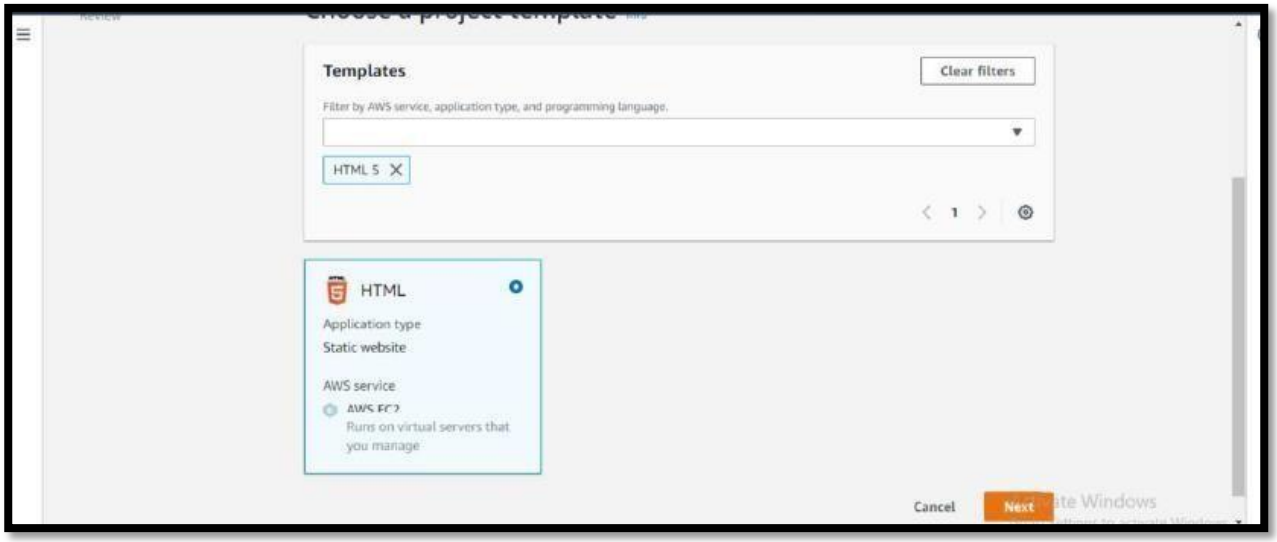
- Automated continuous delivery pipeline

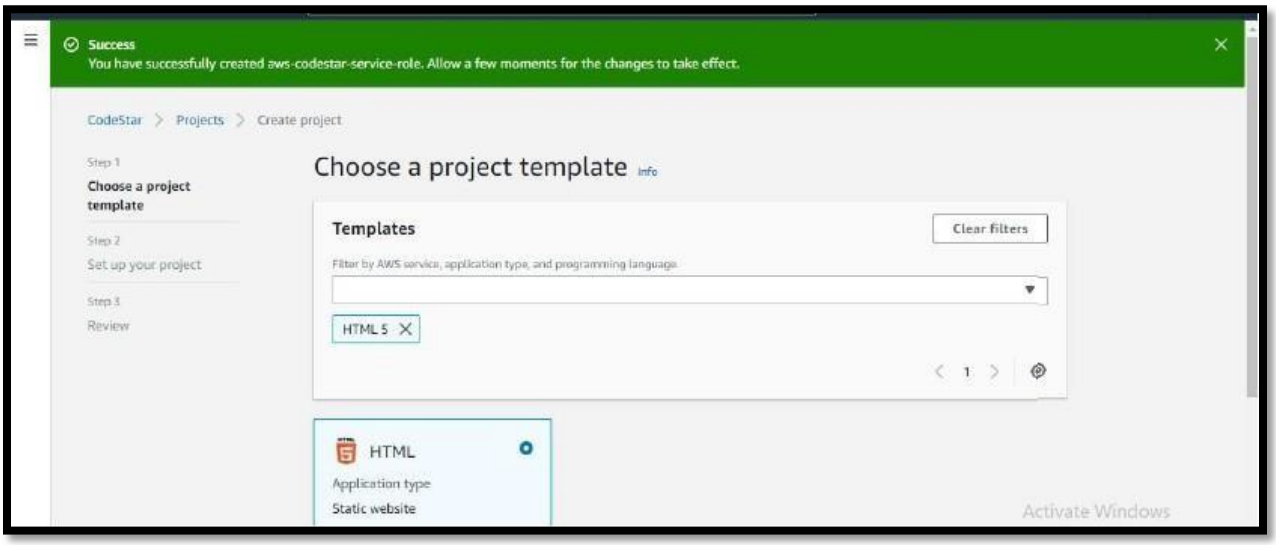
AWS CodeStar accelerates software release with the help of AWS CodePipeline, a continuous integration and continuous delivery (CI/CD) service. Each project comes preconfigured with an automated pipeline that continuously builds, tests, and deploys your code with each commit.

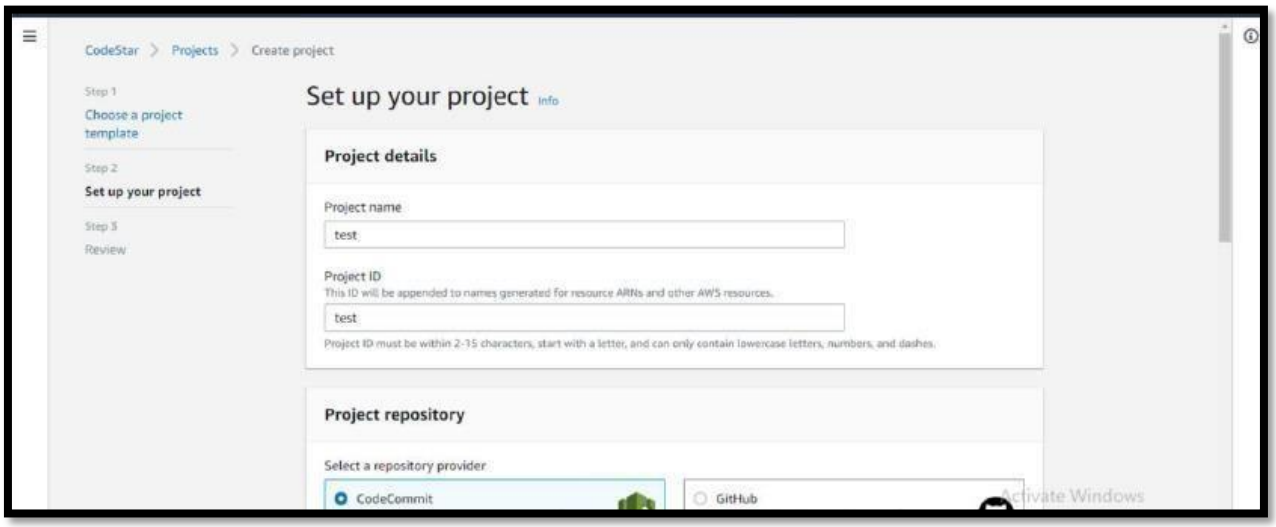
Sample Application:

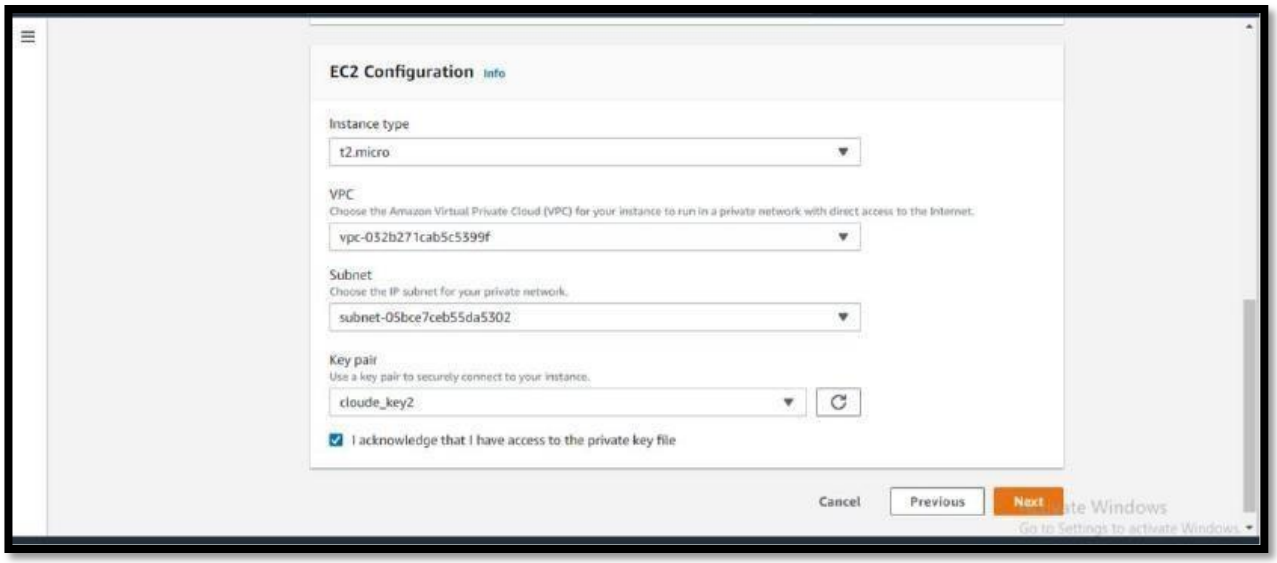


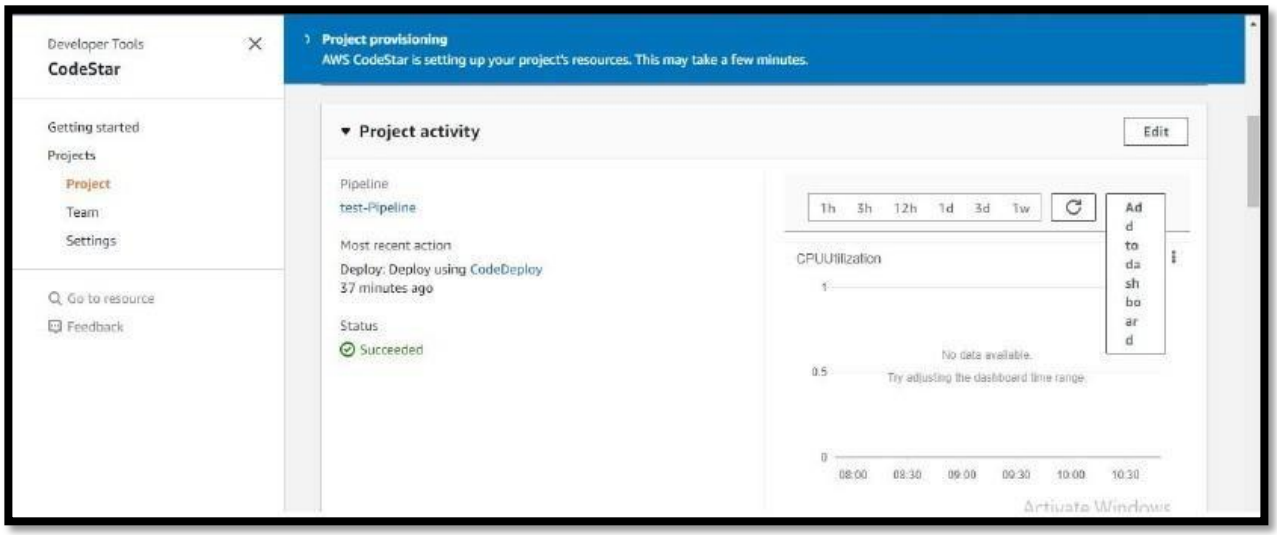
Select HTML5 AND EC2:

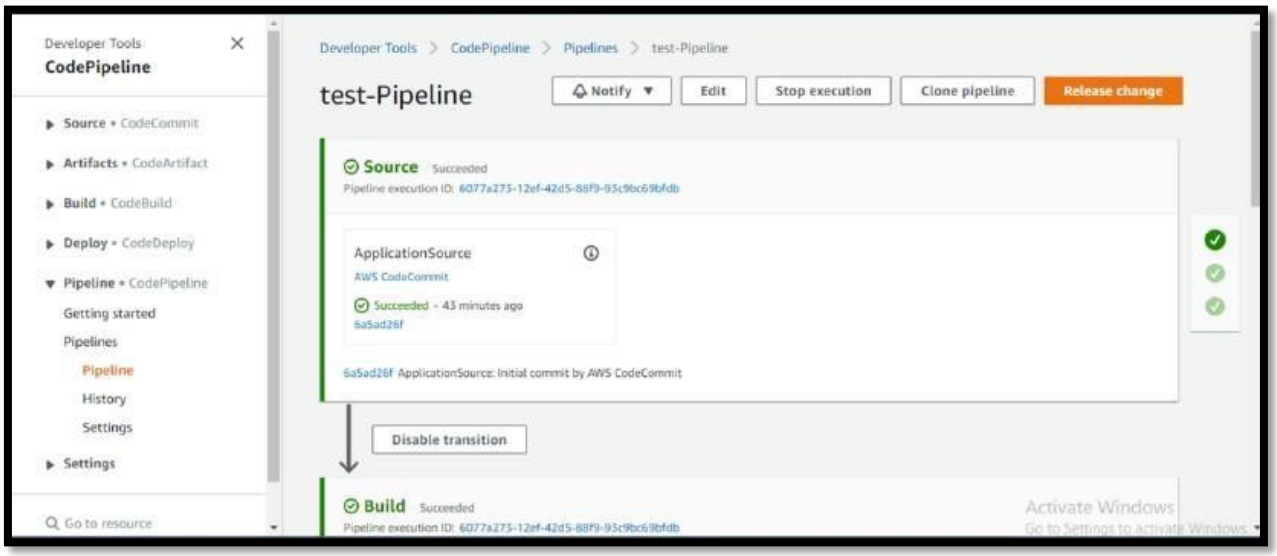


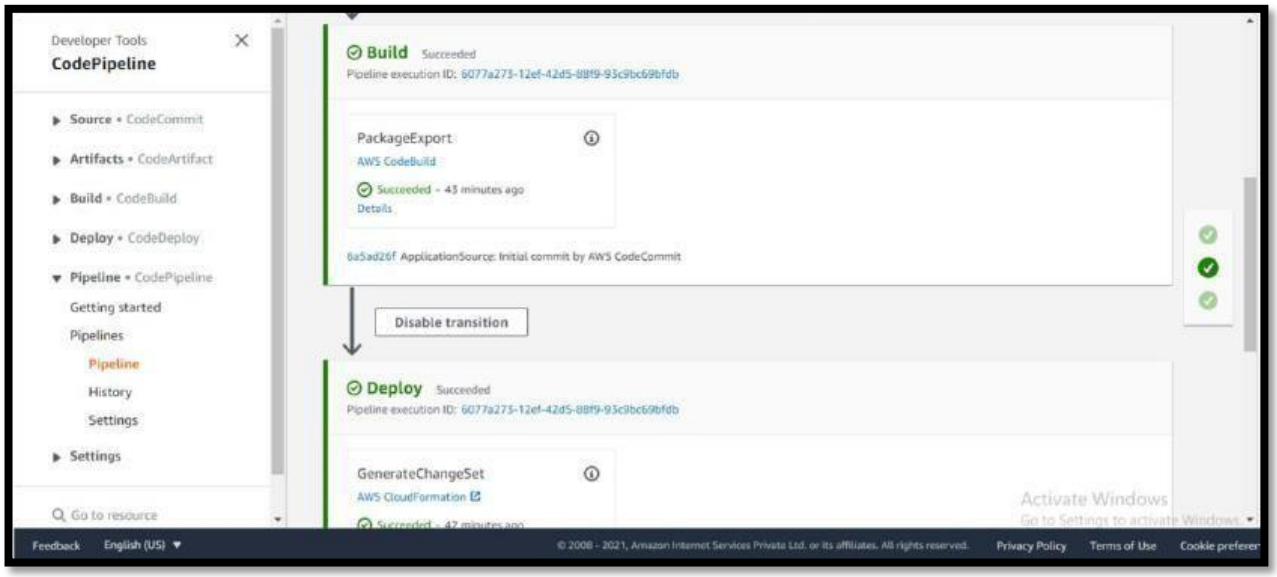

Success
You have successfully created aws-codestar-service-role. Allow a few moments for the changes to take effect.
CodeStar > Projects > Create project
Step 1
Choose a project template
Step 2
Set up your project
Step 3
Review
Choose a project template Info
Templates
Clear filters
Filter by AWS service, application type, and programming language
HTML 5
1
HTML
Application type
Static website
Activate Windows

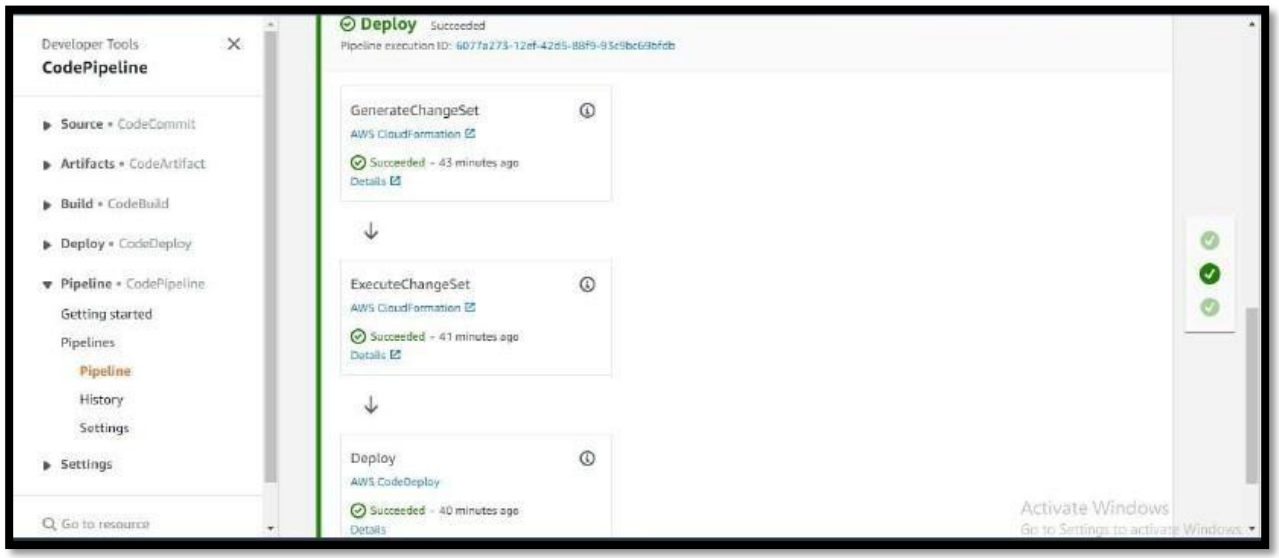

CodeStar > Projects > Create project
Step 1
Choose a project template
Step 2
Set up your project
Step 3
Review
Set up your project Info
Project details
Project name
test
Project ID
This ID will be appended to names generated for resource ARNs and other AWS resources.
test
Project ID must be within 2-15 characters, start with a letter, and can only contain lowercase letters, numbers, and dashes.
Project repository
Select a repository provider
CodeCommit
GitHub
Activate Windows

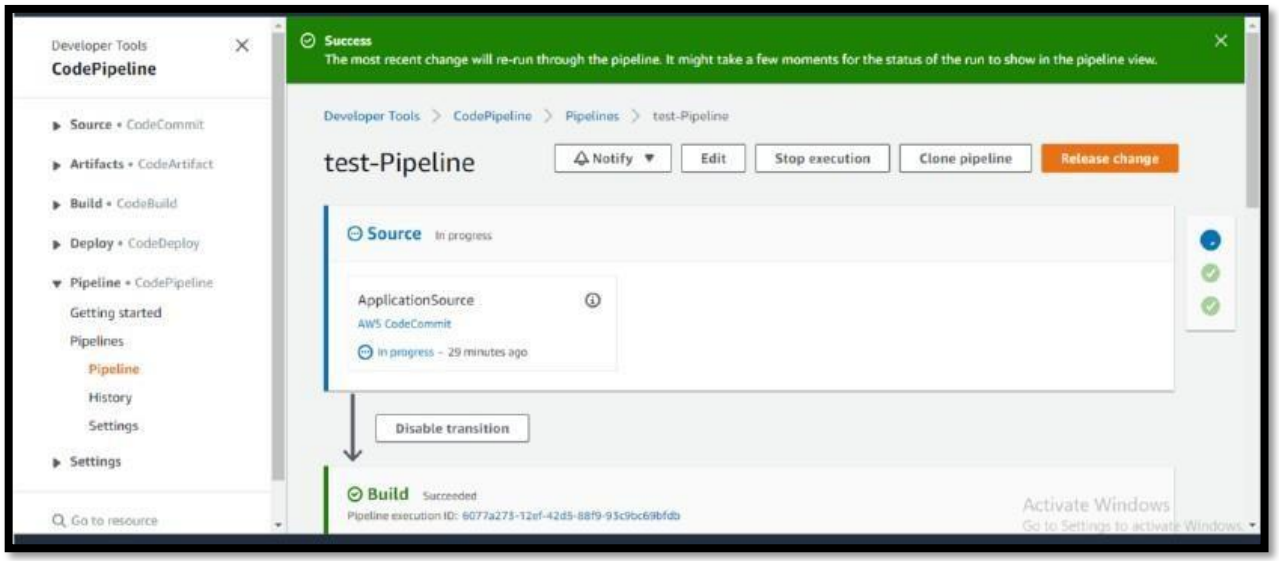

EC2 Configuration Info
Instance type
t2.micro
VPC
Choose the Amazon Virtual Private Cloud (VPC) for your instance to run in a private network with direct access to the Internet.
vpc-032b271cab5c5399f
Subnet
Choose the IP subnet for your private network.
subnet-05bce7ceb55da5302
Key pair
Use a key pair to securely connect to your instance.
cloude_key2
I acknowledge that I have access to the private key file
Cancel
Previous
Next
Activate Windows
Go to Settings to activate Windows.

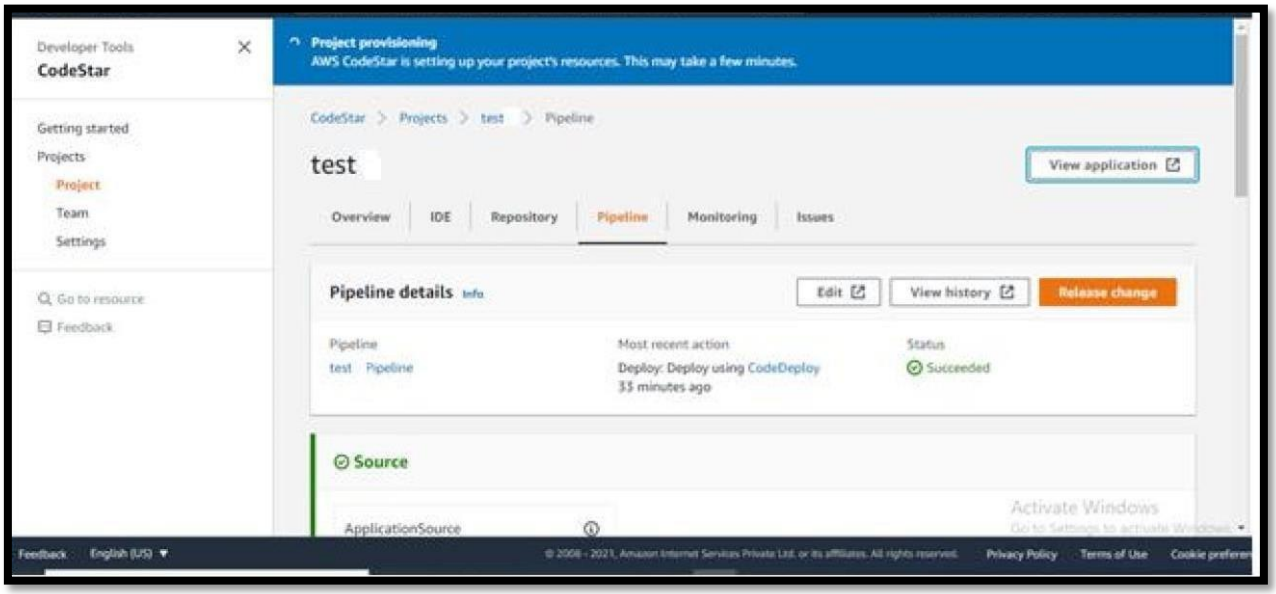

Developer Tools
CodeStar
Getting started
Projects
Project
Team
Settings
Go to resource
Feedback
Project provisioning
AWS CodeStar is setting up your project's resources. This may take a few minutes.
Project activity
Edit
Pipeline
test-Pipeline
Most recent action
Deploy: Deploy using CodeDeploy
37 minutes ago
Status
Succeeded
1h 3h 12h 1d 3d 1w
Add to dashboard
CPUUtilization
No data available.
Try adjusting the dashboard time range.
08:00 08:30 09:00 09:30 10:00 10:30

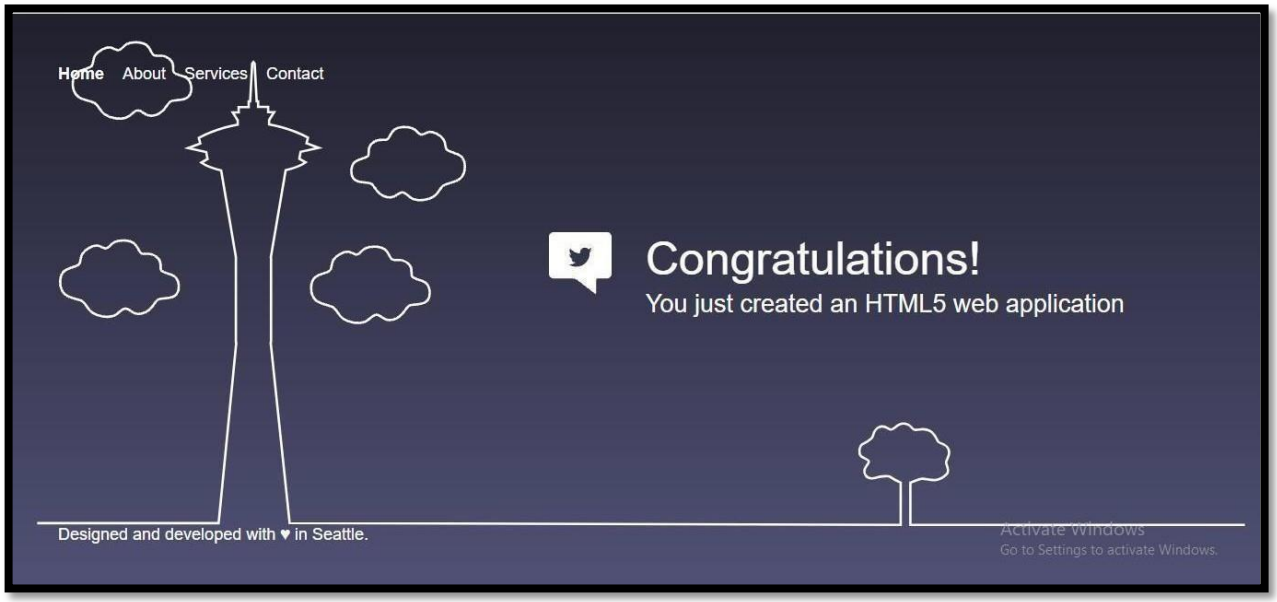

Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Developer Tools > CodePipeline > Pipelines > test-Pipeline
test-Pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Source Succeeded
Pipeline execution ID: 6077a273-12ef-42d5-88f9-93c9bc69bfdb
ApplicationSource
AWS CodeCommit
Succeeded - 43 minutes ago
6a5ad26f
6a5ad26f ApplicationSource: Initial commit by AWS CodeCommit
Disable transition
Build Succeeded
Pipeline execution ID: 6077a273-12ef-42d5-88f9-93c9bc69bfdb


Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Build Succeeded
Pipeline execution ID: 6077a273-12ef-42d5-88f9-93c9bc69bfdb
PackageExport
AWS CodeBuild
Succeeded - 43 minutes ago
Details
6a5ad26f ApplicationSource: Initial commit by AWS CodeCommit
Disable transition
Deploy Succeeded
Pipeline execution ID: 6077a273-12ef-42d5-88f9-93c9bc69bfdb
GenerateChangeSet
AWS CloudFormation
Feedback
English (US)
© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences


Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Deploy Succeeded
Pipeline execution ID: 6077a273-12ef-42d5-88f9-93c9bc69bfdb
GenerateChangeSet
AWS CloudFormation
Succeeded - 43 minutes ago
Details
ExecuteChangeSet
AWS CloudFormation
Succeeded - 41 minutes ago
Details
Deploy
AWS CodeDeploy
Succeeded - 40 minutes ago
Details
Activate Windows


Developer Tools
CodePipeline
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Deploy • CodeDeploy
Pipeline • CodePipeline
Getting started
Pipelines
Pipeline
History
Settings
Settings
Go to resource
Success
The most recent change will re-run through the pipeline. It might take a few moments for the status of the run to show in the pipeline view.
Developer Tools > CodePipeline > Pipelines > test-Pipeline
test-Pipeline
Notify
Edit
Stop execution
Clone pipeline
Release change
Source In progress
ApplicationSource
AWS CodeCommit
In progress - 29 minutes ago
Disable transition
Build Succeeded
Pipeline execution ID: 6077a273-12ef-42d5-88f9-93c9bc69bfdb
Activate Windows


Developer Tools
CodeStar
Getting started
Projects
Project
Team
Settings
Go to resource
Feedback
Project provisioning
AWS CodeStar is setting up your project's resources. This may take a few minutes.
CodeStar > Projects > test > Pipeline
test
View application
Overview
IDE
Repository
Pipeline
Monitoring
Issues
Pipeline details Info
Edit
View history
Release change
Pipeline
test Pipeline
Most recent action
Deploy: Deploy using CodeDeploy
35 minutes ago
Status
Succeeded
Source
ApplicationSource
Activate Windows
Feedback
English (US)
Privacy Policy
Terms of Use


Home
About
Services
Contact
Congratulations!
You just created an HTML5 web application
Designed and developed with ♥ in Seattle.
Activate Windows
Go to Settings to activate Windows.

Assignment no. 02

Q. Define input variable, query data with output and store remote state of terraform.

Terraform supports a few different variable formats. Depending on the usage, the variables are generally divided into inputs and outputs.

- The input variables are used to define values that configure your infrastructure. These values can be used again and again without having to remember their every occurrence in the event it needs to be updated.

- Output variables, in contrast, are used to get information about the infrastructure after deployment. These can be useful for passing on information such as IP addresses for connecting to the server.

Input variables
Input variables are usually defined by stating a name, type and a default value. However, the type and default values are not strictly necessary. Terraform can deduct the type of the variable from the default or input value.

Variables can be predetermined in a file or included in the command-line options. As such, the simplest variable is just a name while the type and value are selected based on the input.
The input variables,use a couple of different types: strings, lists, maps, and boolean.

Output variables
Output variables provide a convenient way to get useful information about your infrastructure. As you might have noticed, much of the server details are calculated at deployment and only become available afterwards. Using output variables you can extract any server-specific values including the calculated details.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }

  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
```

```
}

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

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

```
variable "instance_name" {
  description = "Value of the Name tag for the EC2 instance"
  type        = string
  default     = "ExampleAppServerInstance"
}
```

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

   tags = {
-    Name = "ExampleAppServerInstance"
+    Name = var.instance_name
   }
 }
```

Apply your configuration

```
$ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
```

```
  # aws_instance.app_server will be created
  + resource "aws_instance" "app_server" {
      + ami                          = "ami-08d70e59c07c61a3a"
      + arn                          = (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
```

```
  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: Creation complete after 50s [id=i-0bf954919ed765de1]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Query Data

Cloud infrastructure, applications, and services emit data, which Terraform can query and act on using data sources. Terraform uses data sources to fetch information from cloud provider APIs, such as disk image IDs, or information about the rest of your infrastructure through the outputs of other Terraform configurations.

Data sources allow you to load data from APIs or other Terraform workspaces. You can use this data to make your project's configuration more flexible, and to

connect workspaces that manage different parts of your infrastructure. You can also use data sources to connect and share data between workspaces in Terraform Cloud and Terraform Enterprise.

Initial configuration

```
# main.tf

terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }

  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
}

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

  tags = {
    Name = var.instance_name
  }
}

# variables.tf

variable "instance_name" {
```

```
# variables.tf

variable "instance_name" {
  description = "Value of the Name tag for the EC2 instance"
  type        = string
  default     = "ExampleAppServerInstance"
}
```

```
$ terraform apply
aws_instance.app_server: Refreshing state... [id=i-0bf954919ed765de1]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:

Terraform will perform the following actions:

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

Changes to Outputs:
  + instance_id        = "i-0bf954919ed765de1"
  + instance_public_ip = "54.186.202.254"
```

```
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


Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

instance_id = "i-0bf954919ed765de1"
instance_public_ip = "54.186.202.254"
```

Destroy infrastructure

```
$ terraform destroy

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.app_server will be destroyed
  - resource "aws_instance" "app_server" {
      - ami                          = "ami-08d70e59c07c61a3a" -> null
      - arn                          = "arn:aws:ec2:us-west-2:561656980159:instance/i-0bf95
##...
```

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

Changes to Outputs:
  - instance_id        = "i-0bf954919ed765de1" -> null
  - instance_public_ip = "54.186.202.254" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.app_server: Destroying... [id=i-0bf954919ed765de1]
aws_instance.app_server: Still destroying... [id=i-0bf954919ed765de1, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-0bf954919ed765de1, 20s elapsed]
```

```
  Enter a value: yes

aws_instance.app_server: Destroying... [id=i-0bf954919ed765de1]
aws_instance.app_server: Still destroying... [id=i-0bf954919ed765de1, 10s elapsed]
aws_instance.app_server: Still destroying... [id=i-0bf954919ed765de1, 20s elapsed]
aws_instance.app_server: Still destroying... [id=i-0bf954919ed765de1, 30s elapsed]
aws_instance.app_server: Destruction complete after 31s

Destroy complete! Resources: 1 destroyed.
```

Store Remote State

Remote backends allow Terraform to use a shared storage space for state data. The Terraform Cloud remote backend also allows teams to easily version, audit, and collaborate on infrastructure changes. Terraform Cloud also securely stores variables, including API tokens and access keys. It provides a safe, stable environment for long-running Terraform processes.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }

  required_version = ">= 0.14.9"
}

provider "aws" {
  profile = "default"
  region  = "us-west-2"
}
```

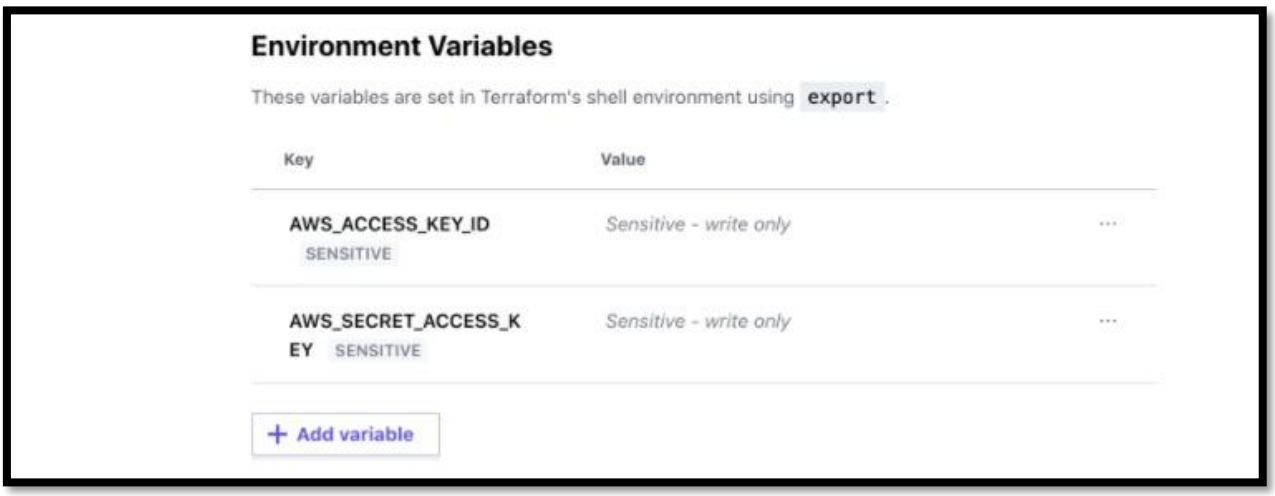
```
terraform {
+ backend "remote" {
+   organization = "<ORG_NAME>"
+   workspaces {
+     name = "Example-Workspace"
+   }
+ }
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.27"
    }
  }
}
```

Login to terraform

```
$ terraform login
Terraform will request an API token for app.terraform.io using your browser.

If login is successful, Terraform will store the token in plain text in
the following file for use by subsequent commands:
    /Users/<USER>/.terraform.d/credentials.tfrc.json

Do you want to proceed?
  Only 'yes' will be accepted to confirm.

  Enter a value:
```

Initialize Terraform

```
$ terraform init

Initializing the backend...
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "local" backend to the
  newly configured "remote" backend. No existing state was found in the newly
  configured "remote" backend. Do you want to copy this state to the new "remote"
  backend? Enter "yes" to copy and "no" to start with an empty state.

  Enter a value: yes

Releasing state lock. This may take a few moments...

Successfully configured the backend "remote"! Terraform will automatically
use this backend unless the backend configuration changes.
```




```
$ terraform apply

## ...
No changes. Infrastructure is up-to-date.
```

```
$ terraform destroy
Running apply in the remote backend. Output will stream here. Pressing Ctrl-C
will cancel the remote apply if it's still pending. If the apply started it
will stop streaming the logs, but will not stop the apply running remotely.

Preparing the remote apply...

To view this run in a browser, visit:
https://app.terraform.io/app/hashicorp-learn/rita-test/runs/run-AcG4nmdSaFDTUfQN

Waiting for the plan to start...

Terraform v0.15.3
on linux_amd64
Configuring remote state backend...
```

```
Configuring remote state backend...
Initializing Terraform configuration...
aws_instance.app_server: Refreshing state... [id=i-039d6d420ad46aff4]

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # aws_instance.app_server will be destroyed
```

Conclusion :
Hence, We Successfully Defined input variable, query data with output and store remote state of terraform.