



EXPERIMENT NO. 01

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

Steps:

1. Login with your AWS account.
2. Navigate to Cloud 9 service from Developer tools section as below:
3. Click on Create Environment :



4. Provide name for the Environment (WebAppIDE) and click on next.
5. Keep all the Default settings as shown in below:



AWS Cloud9

Your environments

Shared with you

Account environments

How to guide

AWS Cloud9 > Environments > Create environment

Step 1: Create environment

Step 2: Configure settings

Step 3: Review

Configure settings

Environment settings

Environment type: **Info**
When you create an environment, you select an instance type. AWS Cloud9 environments are created on Amazon EC2 instances. You can create an environment on a new EC2 instance or on an existing EC2 instance. You can also create an environment on a new EC2 instance or on an existing EC2 instance.

☒ Create a new EC2 instance for environment (default, recommended)
Launches a new EC2 instance in the region that you select when you create the environment.

☐ Create a new EC2 instance for environment (advanced)
Launches a new EC2 instance in the region that you select when you create the environment.

☐ Create and run in private subnet (SSH connection)
Participates in the private subnet that you select when you create the environment.

Instance type

☒ t2.micro (1 GB RAM + 1 vCPU)
Best for development and testing.

☐ t2.xlarge (4 GB RAM + 2 vCPU)
Best for production and development.

☐ t2.2xlarge (8 GB RAM + 2 vCPU)
Best for production and development.

☐ Other instance type
Select an instance type.

t2.micro

Platform

☒ Amazon Linux 2 (recommended)

☐ Amazon Linux AMI

☐ Ubuntu Server 18.04 LTS

Cost-saving settings

Choose a cost-saving option for your environment. You can choose to stop your environment when you are not using it. This helps you save money. You can also choose to stop your environment when you are not using it. This helps you save money.

After 30 minutes (default)

IAM role

AWS Cloud9 uses an IAM role to access AWS services on your behalf. This role is called **AWSCloud9Role**. You can also create your own role. You can also create your own role. You can also create your own role.

AWSCloud9Role

Network settings (advanced)

No tags associated with this resource

Add new tag

You selected 0 tags.

Cancel Previous step Next step

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

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



7. Till that time open IAM Identity and Access Management in order to Add user In other tab.

Review

Environment name and settings

Name: WebAppIDE

Description: We description provided

Instance type: EC2

Instance type: t2.micro

Subnet:

Platform: Amazon Linux 2 (recommended)

Provisioning setting: After 30 minutes (default)

IAM role: AWSServiceRoleForAWSCloud9 (predefined)

We recommend the following best practices for using your AWS Cloud9 environment:

- Use secure 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

Identity and Access Management (IAM)

Dashboard

Access management

User groups

Users

Roles

Policies

Identity providers

Account settings

Access reports

Access analyzer

Archive rules

Analyzers

Settings

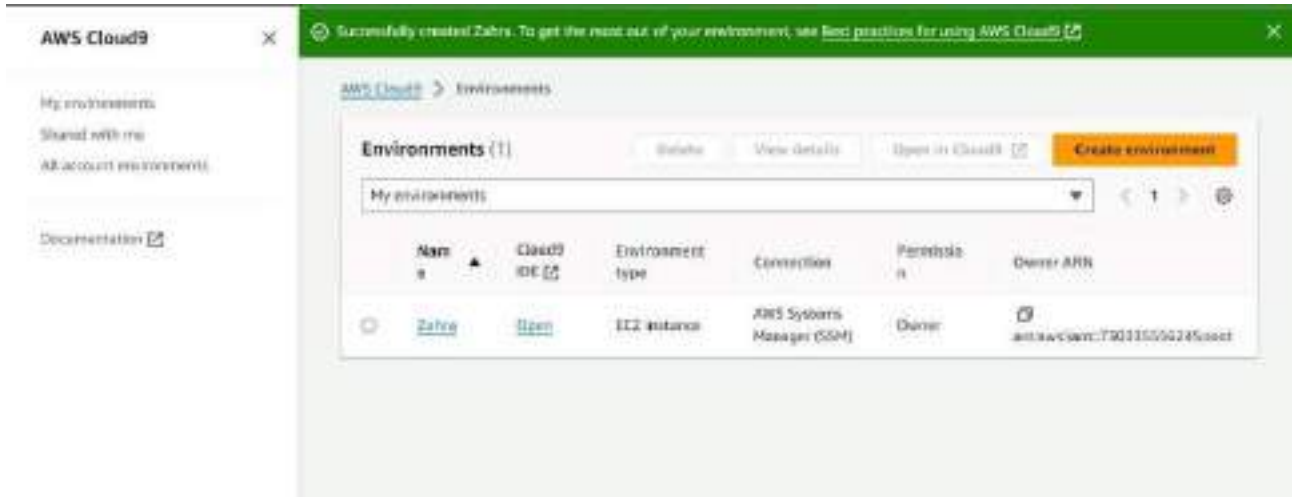
Add user Delete user

Find users by username or access key

User name	Groups
-----------	--------



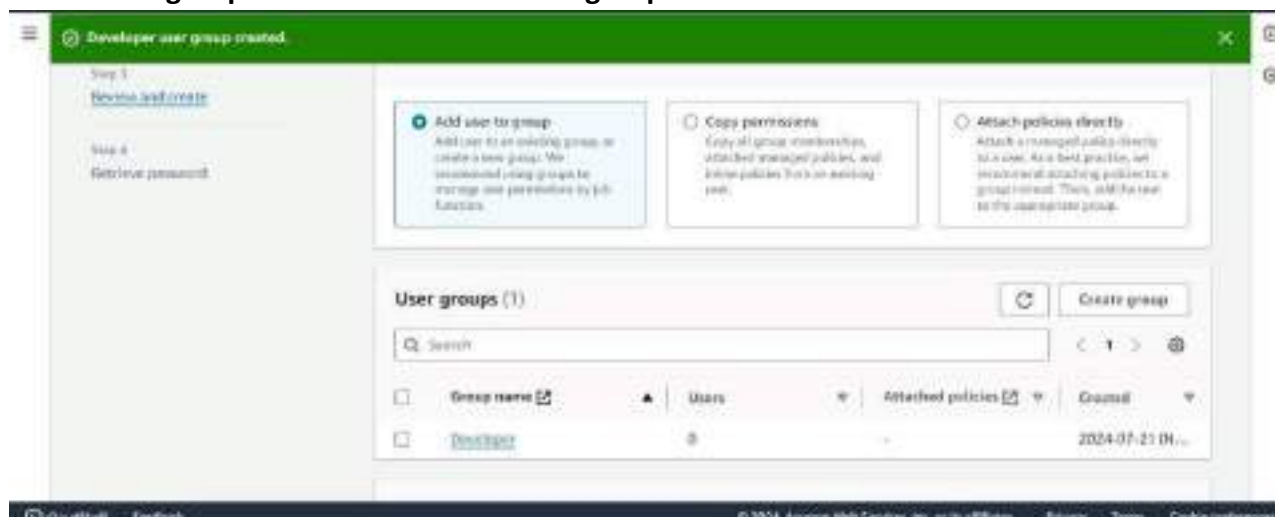
8. Add user provide manual password if you want and click on Next permission tab.



9.
Click on Create group

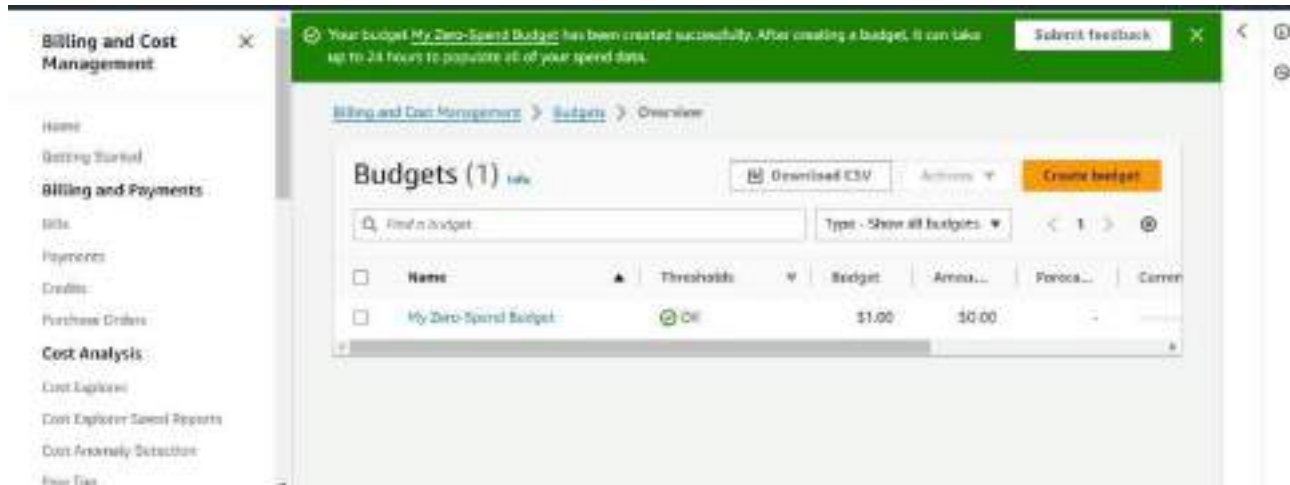


10. Provide group name and click on create group.

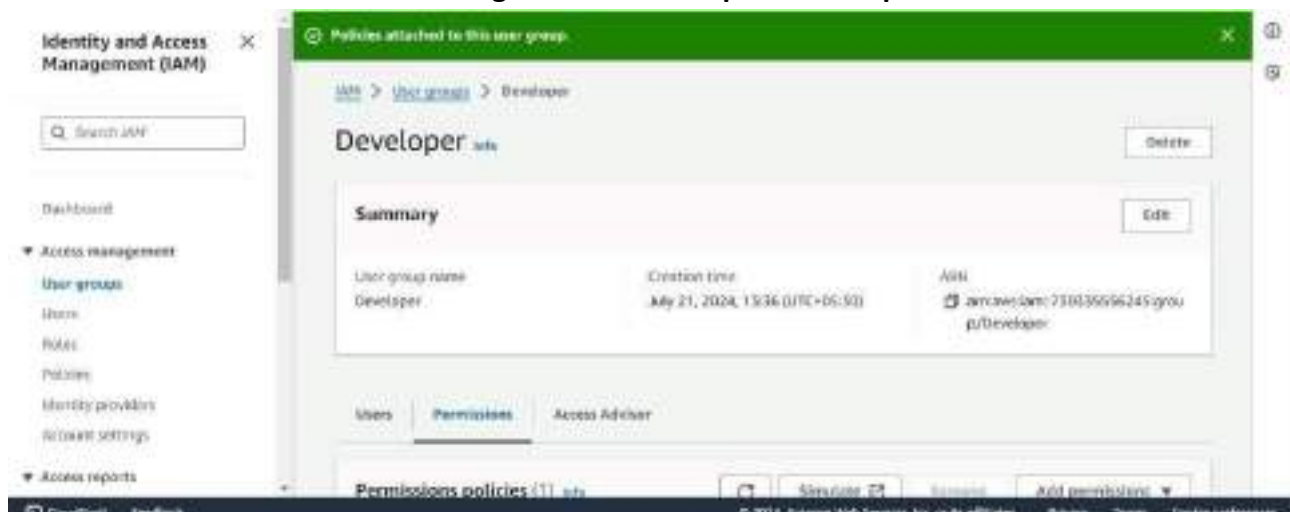




11. After that group is created click on next if u want to provide tag else click on Review for user settings and click on create user as shown in fig.



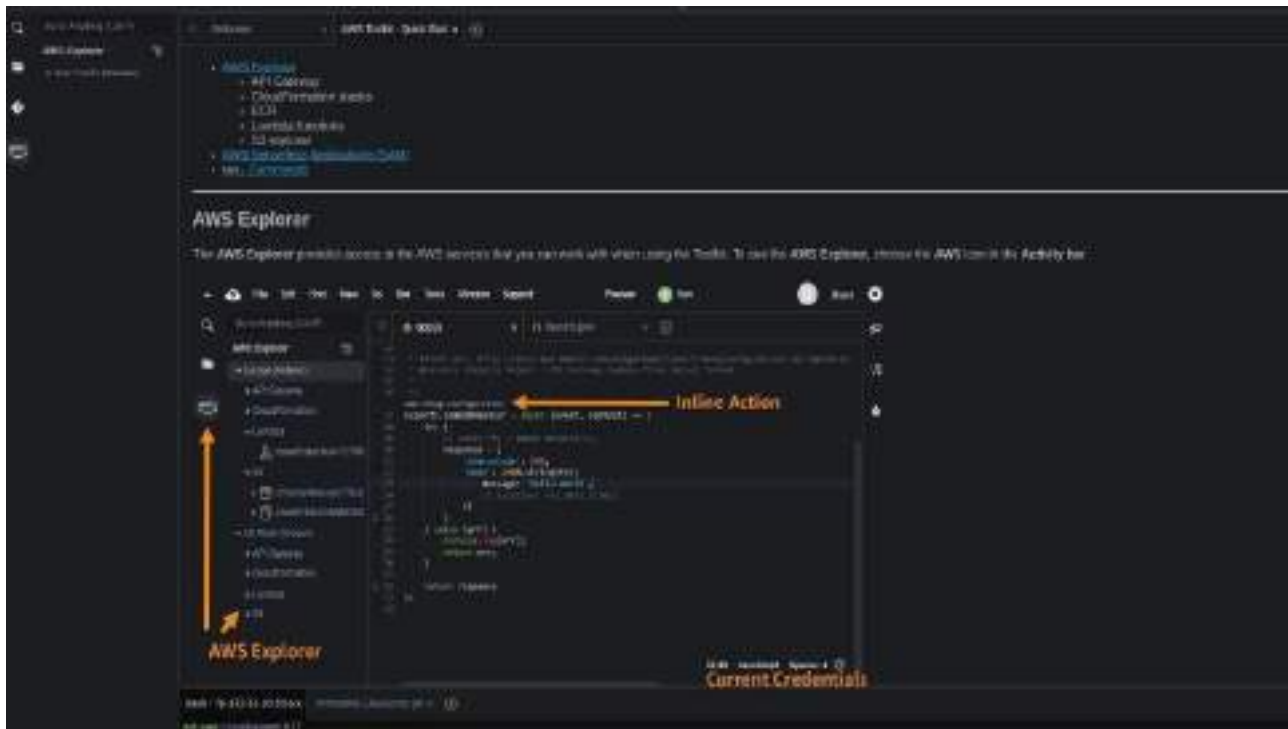
12. Now close that window and Navigate to user Groups from left pane in IAM.



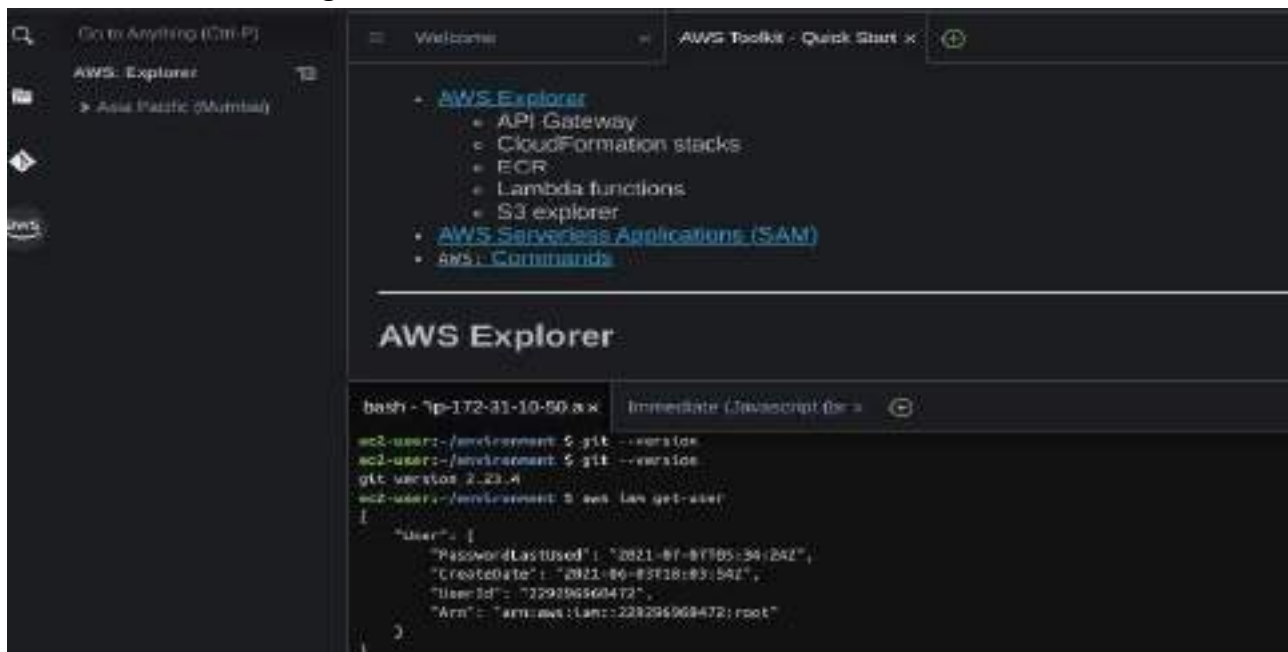
13. click on your group name which you have created and nevgate to permission tab as shown:

14. Now click on Add permission and select Attach Policy after that search for Cloud9 related policy and select Awscloud9EnviornmentMember policy and add it.

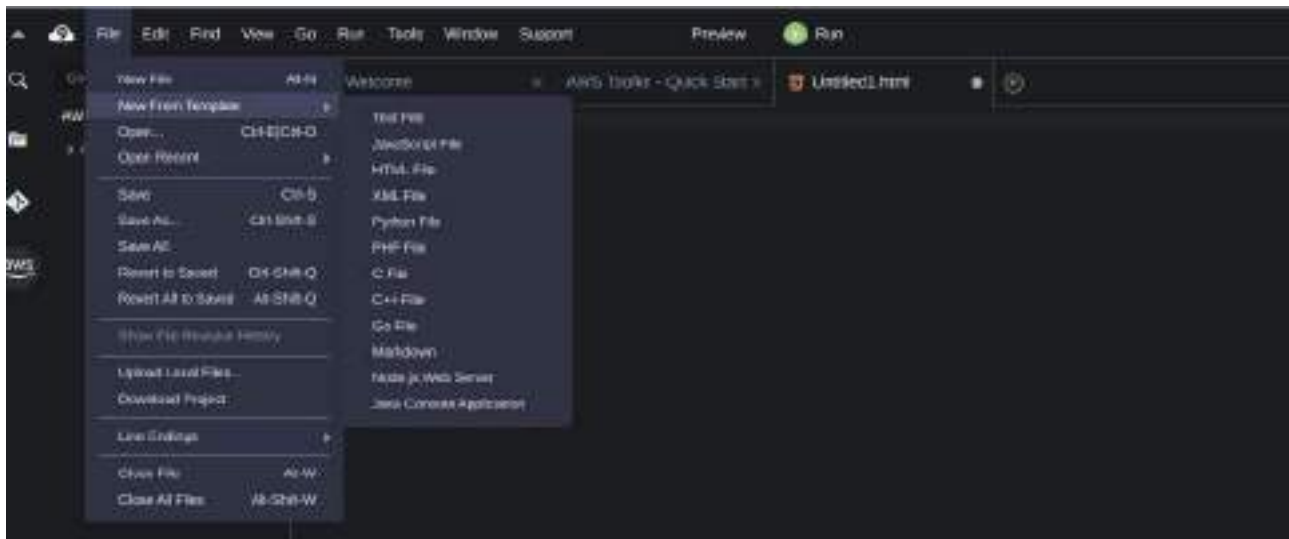
15. now we move towards our cloud9 IDE Enviornment tab it shows as shown :



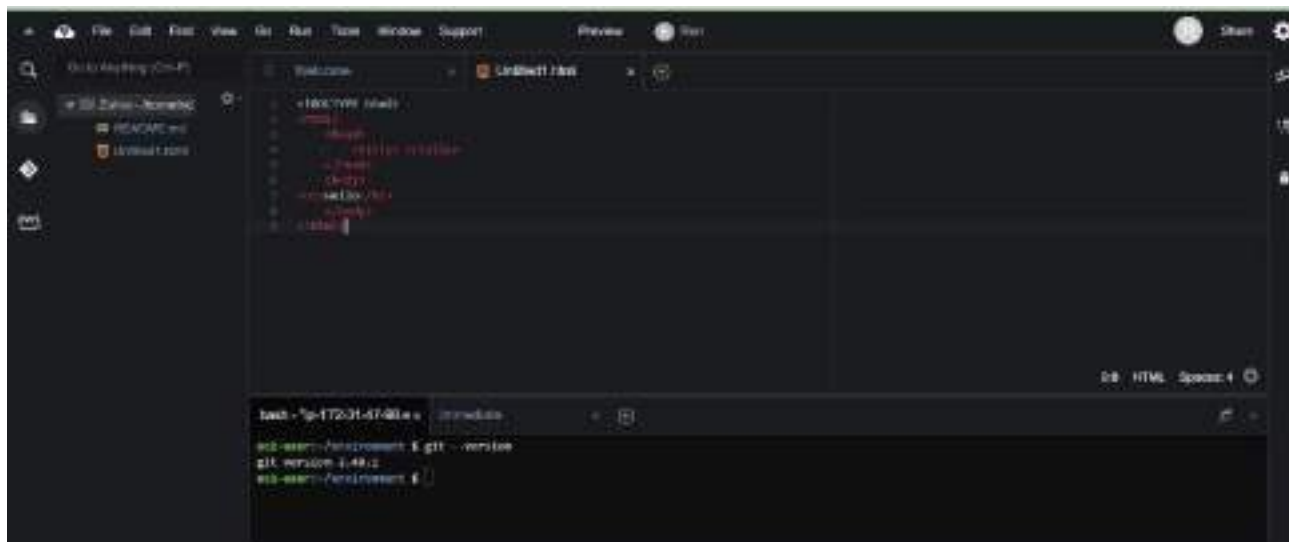
16. If you check at bottom side Cloud9 IDE also giving you and aws CLI for command operations: as we here checked git version, iam user details and so on...



17. 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.



18. Edit html file and save it



19. 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.



Sign in



Root user

Account owner that performs tasks requiring unrestricted access. [Learn more](#)



IAM user

User within an account that performs daily tasks. [Learn more](#)

Account ID (12 digits) or account alias

229296960472

☐ Remember this account

Next

By continuing, you agree to the [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

New to AWS?

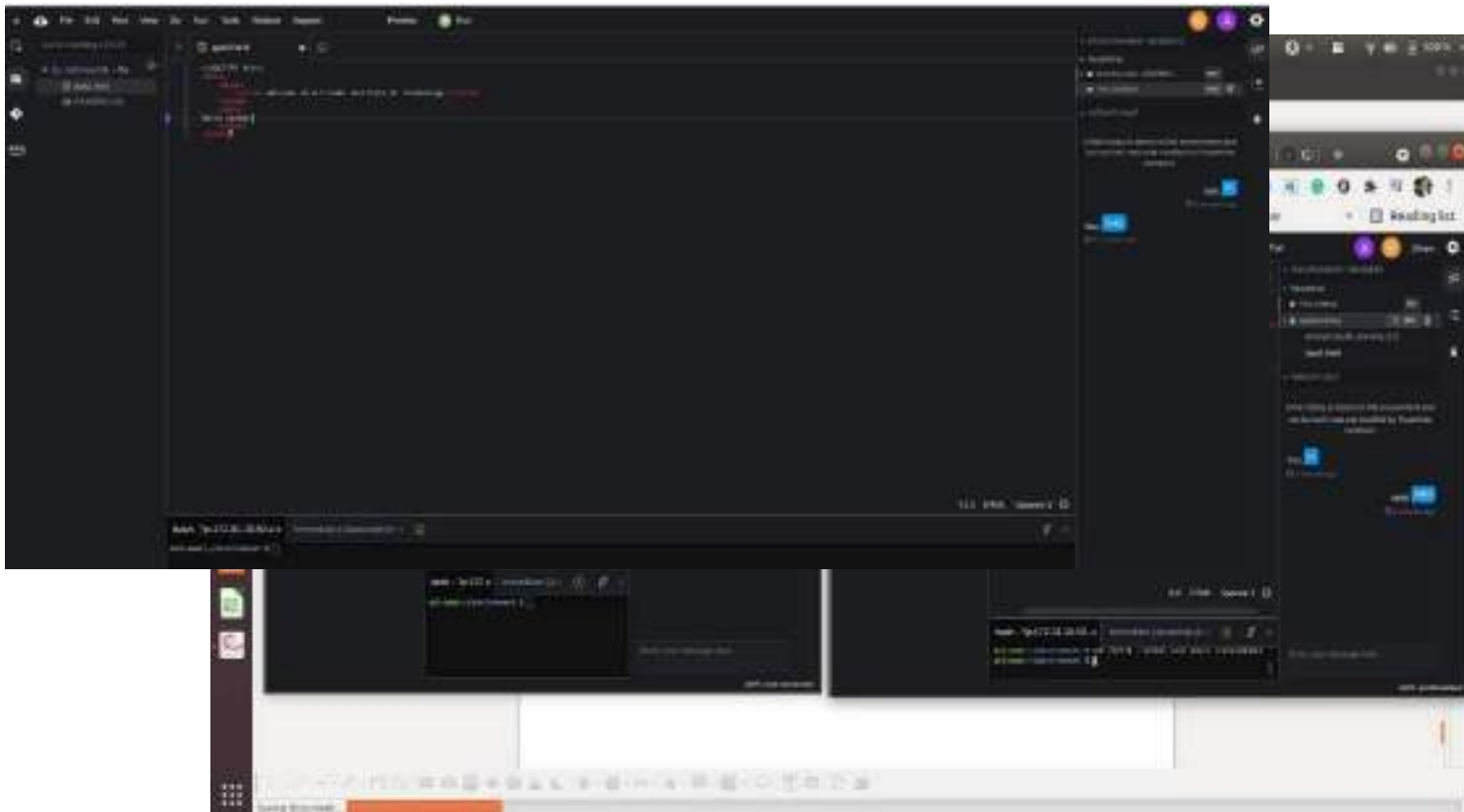
Create a new AWS account

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

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



PARSHVANATH CHARITABLE TRUST'S
A. P. SHAH INSTITUTE OF TECHNOLOGY
Department of Information Technology
(NBA Accredited)



24. you can also explore settings where you can update permissions of your teammates as from RW to R only or you can remove user too.

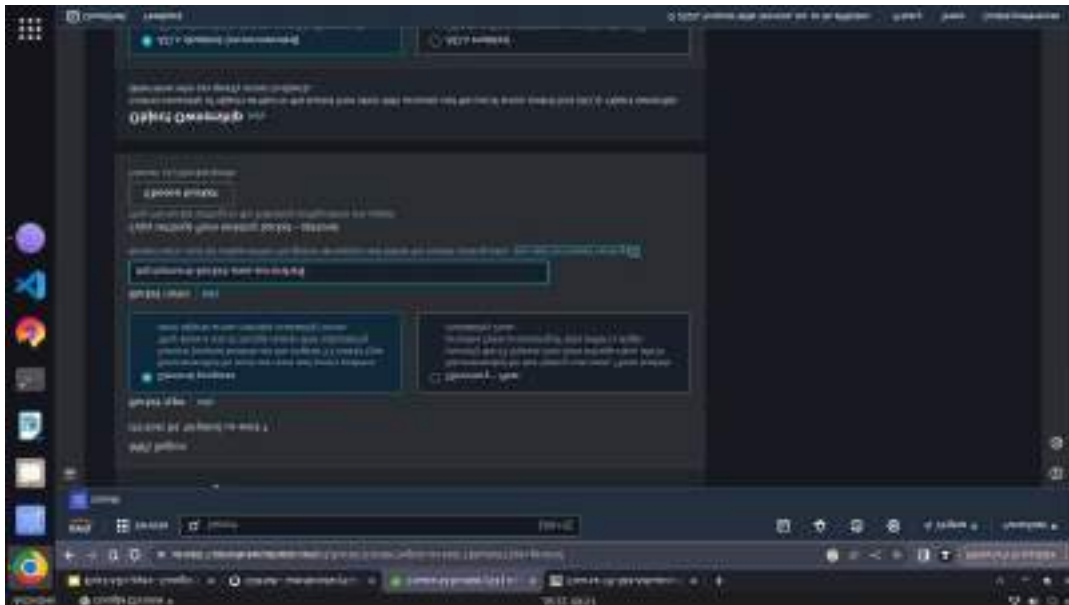


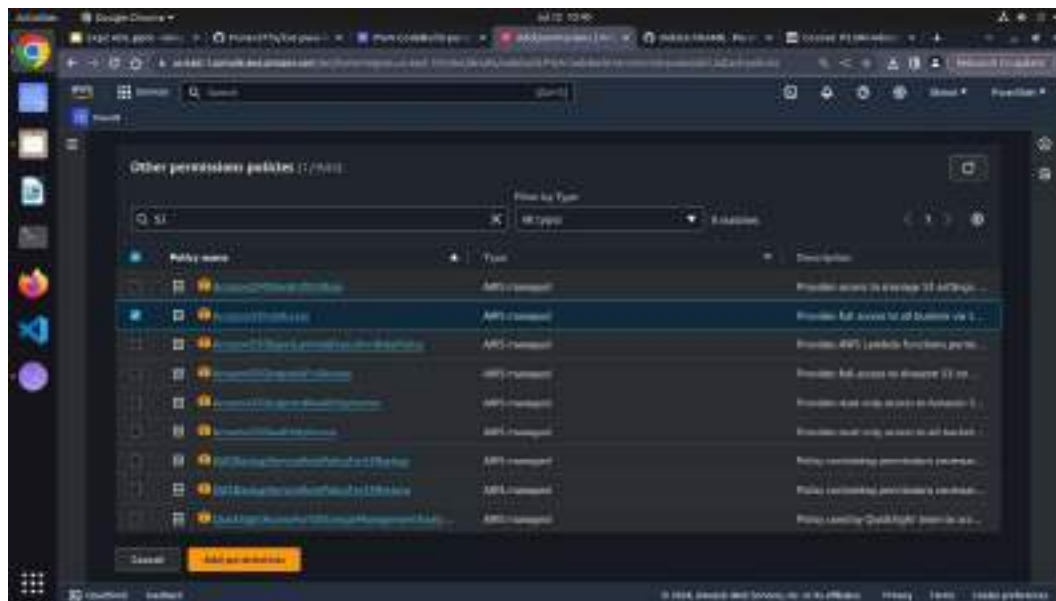
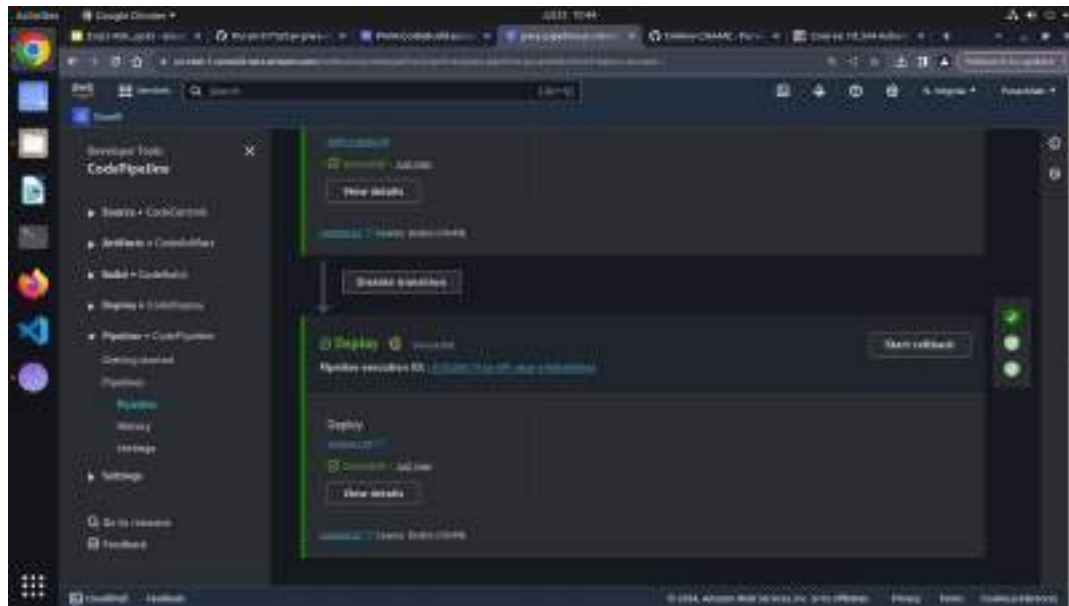
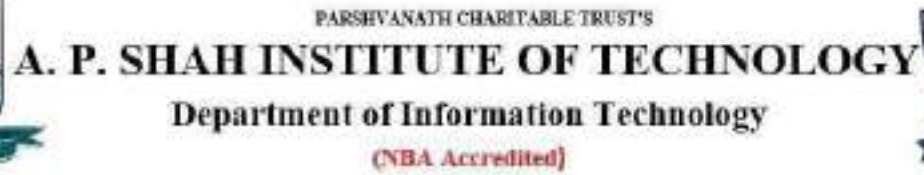


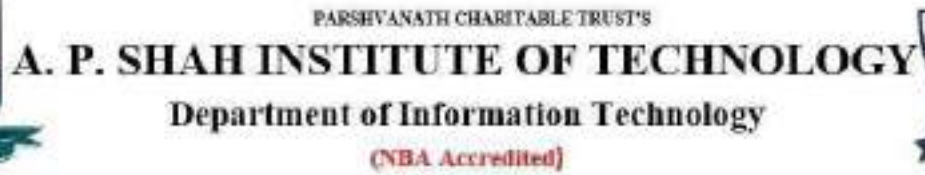
EXPERIMENT NO. 02

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.

Step1: Create a deployment environment









a. If you plan to use Amazon S3 as your source, you will retrieve the sample code from the AWS GitHub repository, save it to your computer, and upload it to an Amazon S3 bucket.

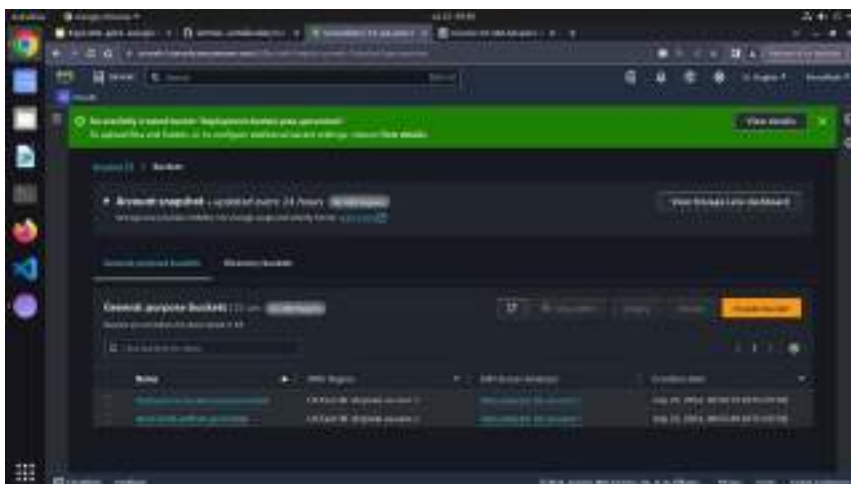
- Visit our GitHub repository containing the sample code at
<https://github.com/imoisharma/aws-codepipeline-s3-codedeploy-linux-2.0>
- Click the dist folder.

b. Save the source files to your computer:

- Click the file named aws-codepipeline-s3-aws-codedeploy_linux.zip
- Click View Raw.
- Save the sample file to your local computer.

c. open the Amazon S3 console and create your Amazon S3 bucket:

- Click Create Bucket





- **Bucket Name:** type a unique name for your bucket, such as awscodepipeline-demobucket-variables. All bucket names in Amazon S3 must be unique, so use one of your own, not one with the name shown in the example.
- **Region:** In the drop-down, select the region where you will create your pipeline, such as ap-South-1
- Click Create.

d. The console displays the newly created bucket, which is empty.

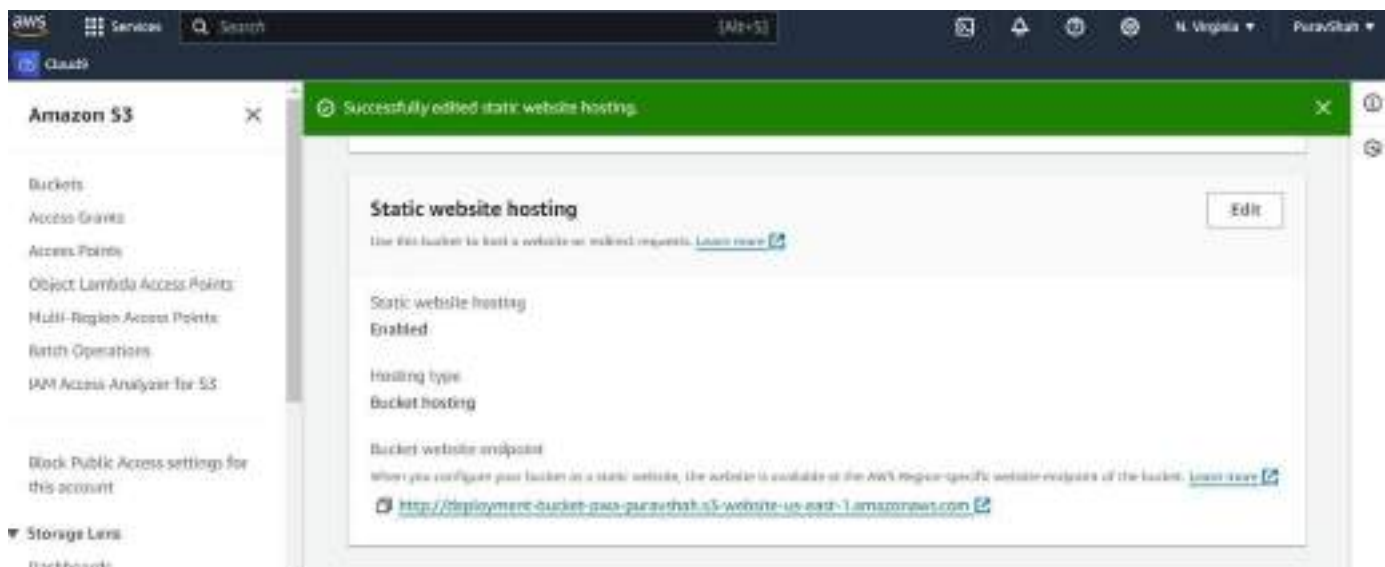
- Click Properties.
- Expand Versioning and select Enable Versioning. When versioning is enabled, Amazon S3 saves every version of every object in the bucket.

e. You will now upload the sample code to the Amazon S3 bucket:

- Click Upload.
- Follow the on-screen directions to upload the .zip file containing the sample code you downloaded from GitHub.



PARSHVANATH CHARITABLE TRUST'S
A. P. SHAH INSTITUTE OF TECHNOLOGY
Department of Information Technology
(NBA Accredited)





Step3: Create your Pipeline

In this step, you will create and configure a simple pipeline with two actions: source and deploy. You will provide CodePipeline with the locations of your source repository and deployment environment.

A true continuous deployment pipeline requires a build stage, where code is compiled and unit tested. CodePipeline lets you plug your preferred build provider into your pipeline. However, in this we will skip the build stage.

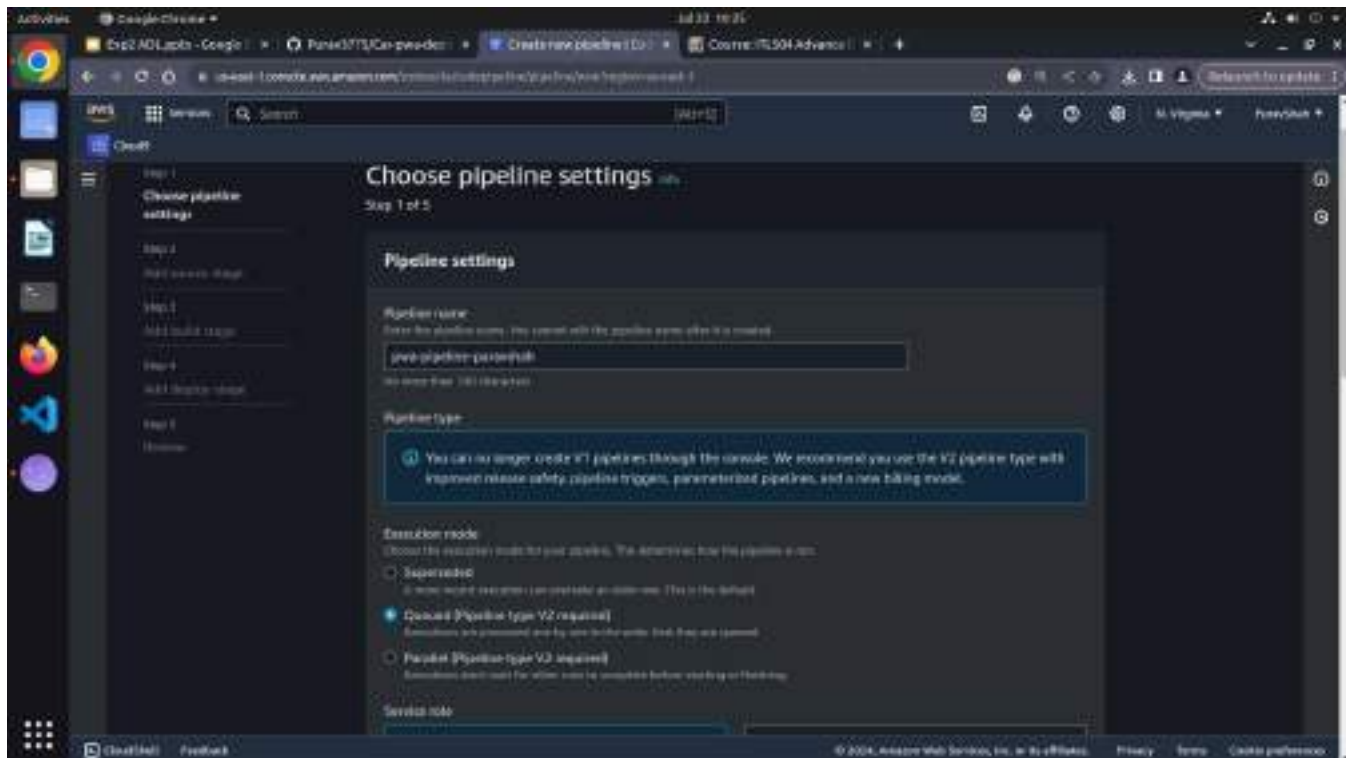
Goto Pipeline again and create it

In above you can give zip file name in S3 object Key and choose bucket name which you created

In Step 4: Deploy Stage:



- Deployment provider: Click AWS Elastic Beanstalk.
- Application name: MYEBS.
- Environment name: Click Myebs-env.
- Click Next step.





After your pipeline is created, the pipeline status page appears and the pipeline automatically starts to run. You can view progress as well as success and failure messages as the pipeline perform each action.

To verify your pipeline ran successfully, monitor the progress of the pipeline as it moves through each stage. The status of each stage will change from No executions yet to In Progress, and then to either Succeeded or Failed. The pipeline should complete the first run within a few minutes.

Now go to your EBS environment and click on the URL to view the sample website you deployed.

AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in using AWS Identity and Access Management](#). Here are some sample policies.

Step 1: Select Policy Type

A policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an S3 Topic Policy, an S3 Queue Policy, and an S3 Queue Policy.

Select Type of Policy

S3 Bucket Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See a description of elements that you can use in a statement.

Principal	Effect	Action
	<input type="radio"/> Allow <input type="radio"/> Deny	

Use a comma to separate multiple values.

AWS Services

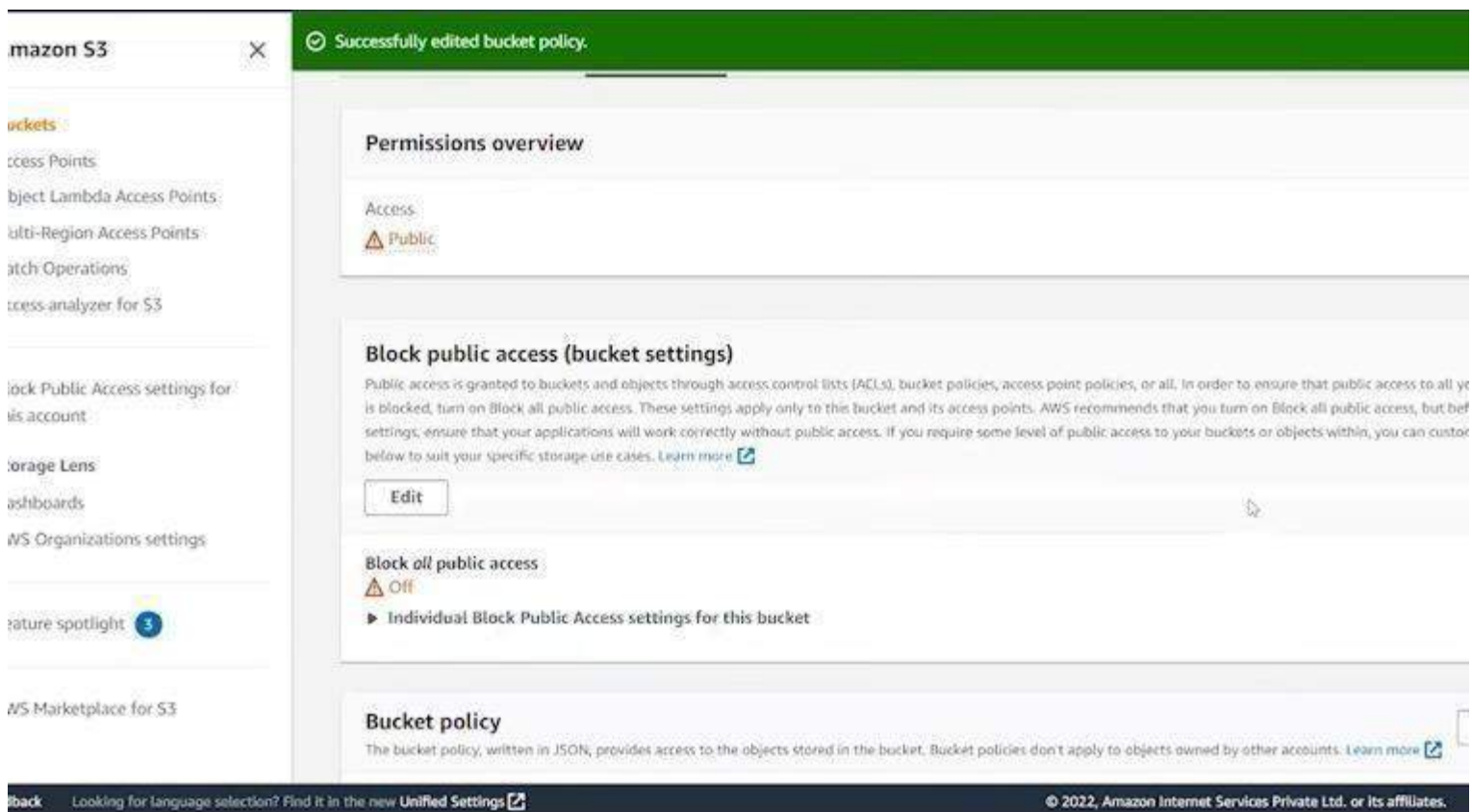


You have successfully created an automated software release pipeline using AWS CodePipeline!

Using CodePipeline, you created a pipeline that uses GitHub, Amazon S3, or AWS CodeCommit as the source location for application code and then deploys the code to an Amazon EC2 instance managed by AWS Elastic Beanstalk.

Step 5: Commit a change and then update your app Step

6: Clean up your resources





To avoid future charges, you will delete all the resources you launched throughout this tutorial, which includes the pipeline, the Elastic Beanstalk application, and the source you set up to host the code.

- a. First, you will delete your pipeline:
- b. Second, delete your Elastic Beanstalk application:

Conclusion: Write your own findings.



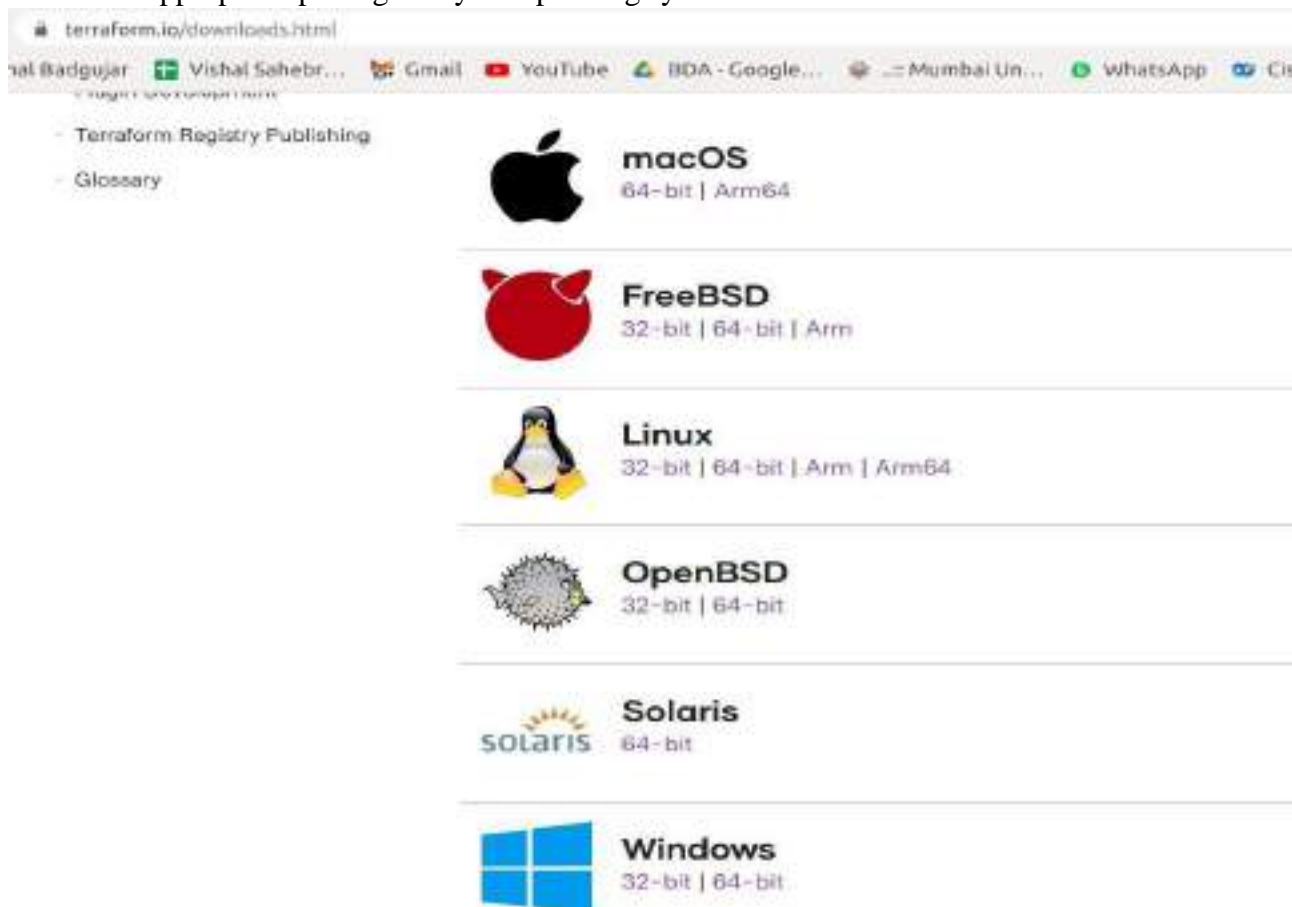
EXPERIMENT NO. 05

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

Terraform Installation Steps on Ubuntu18.04

Step: 1 Terraform uses HashiCorp Configuration Language (HCL) to manage environments of Operators and Infrastructure teams. To download go to site
<https://www.terraform.io/downloads.html>

Select the appropriate package for your operating system and architecture.





Step:2 unzip the archive by using below command

```
vishal@master:~$ unzip terraform_1.0.3_linux_amd64.zip
```

The archive will extract a single binary called **terraform**.

Step 3: Change the directory to unzipped folder

```
vishal@master:~$ cd terraform_1.0.3_linux_amd64/
```

and Move the terraform binary to a directory included in your system's PATH in my case *usr/local/bin/*

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Desktop/terraform_1.9.3_linux_amd64$ sudo mv terraform /usr/local/bin/  
[sudo] password for apsiti:  
Sorry, try again.  
[sudo] password for apsiti:
```

Step 4: To check whether Terraform is installed, run:

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Desktop/terraform_1.9.3_linux_amd64$ terraform -v  
Terraform v1.9.3  
on linux_amd64  
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Desktop/terraform_1.9.3_linux_amd64$
```




EXPERIMENT NO. 06

Aim: To Build, change, and destroy AWS infrastructure Using Terraform.

Pre-requisites:

1. Install the AWS CLI version 2 on Linux

Follow these steps from the command line to install the AWS CLI on Linux.

Install curl on linux

```
vishal@apsit:~$ sudo apt-get install curl
```

```
vishal@apsit:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
vishal@apsit:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left  Speed
100 41.8M  100 41.8M    0     0  2529k      0  0:00:16  0:00:16 --:--:-- 2555k
```

```
vishal@apsit:~$ sudo apt install unzip
```

```
vishal@apsit:~$ sudo apt install unzip
```

```
vishal@apsit:~$ sudo unzip awscliv2.zip
```

```
vishal@apsit:~$ sudo unzip awscliv2.zip
```

```
vishal@apsit:~$ sudo ./aws/install
```

```
vishal@apsit:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
```



vishal@apsit:~\$ aws --version

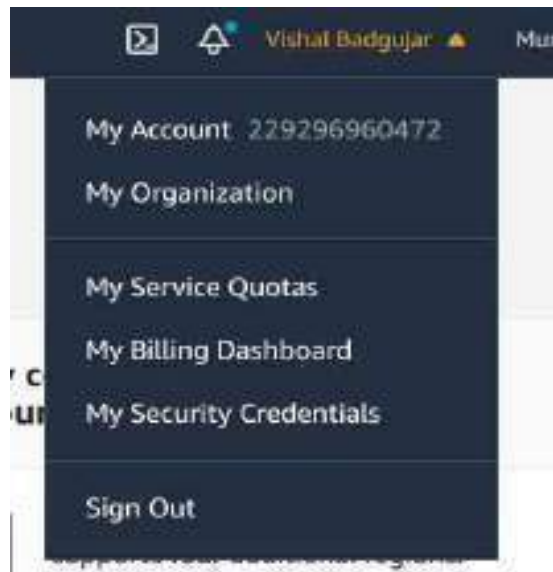
it should display the below output.

aws-cli/2.1.29 Python/3.8.8 Linux/5.4.0-1038-aws exe/x86_64.ubuntu.18 prompt/off

```
vishal@apsit:~$ aws --version
aws-cli/2.2.25 Python/3.8.8 Linux/5.4.0-80-generic exe/x86_64.ubuntu.18 prompt/off
```

2. Create a new access key if you don't have one. Make sure you download the keys in your local machine.

Login to AWS console, click on username and go to My security credentials.





PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity, see [AWS Identity Credentials](#). To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#).

▲ Password

▲ Multi-factor authentication (MFA)

▼ Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, and other AWS tools.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend that you rotate your secret keys regularly. If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key.

Created	Access Key ID	Last Used
---------	---------------	-----------

Continue on security credentials, click on access keys

Perform below commands in Linux where you have installed Terraform

First setup your access keys, secret keys and region code locally.

vishal@apsit:~\$aws configure

Created	Access Key ID	Secret Access Key
Jul 4th 2021	AKIAIYKZ	SECRETKEY

You can check region as

Vishal Badgujar Mumbai

US East (N. Virginia) us-east-1

US East (Ohio) us-east-2

US West (N. California) us-west-1

US West (Oregon) us-west-2

Africa (Cape Town) af-south-1

Asia Pacific (Hong Kong) ap-east-1

Asia Pacific (Mumbai) **ap-south-1**

Asia Pacific (Osaka) ap-northeast-3

Asia Pacific (Seoul) ap-northeast-2

Asia Pacific (Singapore) ap-southeast-1

Asia Pacific (Sydney) ap-southeast-2

Asia Pacific (Tokyo) ap-northeast-1

Canada (Central) ca-central-1

Europe (Frankfurt) eu-central-1

Europe (Ireland) eu-west-1

Last Used Region	Last Used Service	Status
us-east-1	S3	Active

shown in below image :



```
vishal@apsit:~$ aws configure
AWS Access Key ID [None]: AKIATKYZJ6PMFLTCGGPV
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]:
vishal@apsit:~$
```

Create one Directory for Terraform project in which all files of terraform we can save

```
vishal@apsit:~$ cd ~
vishal@apsit:~$ mkdir project-terraform
vishal@apsit:~$ cd project-terraform
```

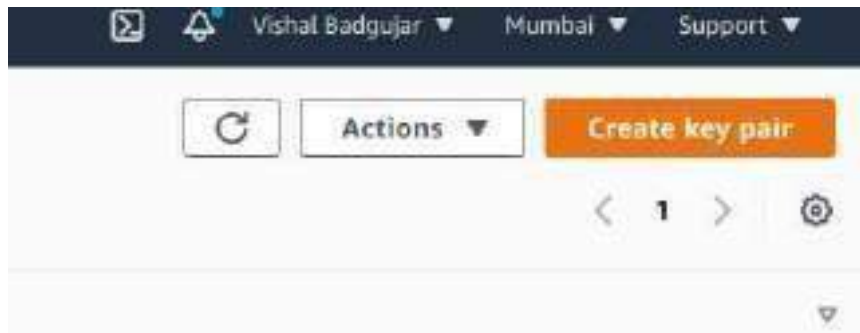
```
vishal@apsit:~$ mkdir project-terraform
vishal@apsit:~$ cd project-terraform/
vishal@apsit:~/project-terraform$
```

Create Terraform Files

```
vishal@apsit:~$ sudo nano variables.tf
```



In order to provide key name in variables first create key pair as shown:



Give name to key pair file as **terraform**

Create key pair

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name
terraform
The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Private key file format
☒ .pem
For use with OpenSSH.
☐ .ppk
For use with PuTTY.

Tags (Optional)
No tags associated with the resource.
Add tag
You can add 50 more tags.

Cancel Create key pair

Key pair is generated

Use your Region and Key name in variable.tf as shown and provide instance type which you want to create.

```
File Edit View Search Terminal Help
GNU nano 2.9.3 variables.tf Modified

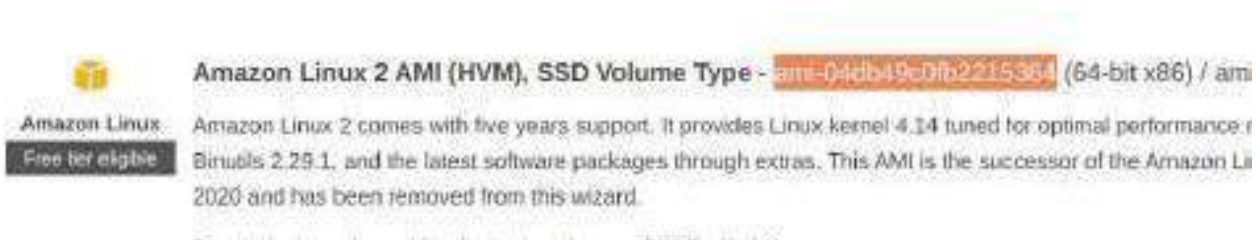
variable "aws_region" {
  description = "The AWS region to create things in."
  default     = "ap-south-1"
}

variable "key_name" {
  description = "SSH keys to connect to ec2 instance"
  default     = "terraform"
}

variable "instance_type" {
  description = "instance type for ec2"
  default     = "t2.micro"
}

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    ^U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line  ^E Redo
```

After creating variable terraform file note down the AMI ID of instance which u want to create which we will use to configure our instance in main.tf file.





Now create main.tf file:

```
vishal@apsit:~/project-terraform$ sudo nano main.tf
```

```
provider "aws" {  
    region = var.aws_region  
}  
  
#Create security group with firewall rules  
resource "aws_security_group" "security_jenkins_port" {  
    name      = "security_jenkins_port"  
    description = "security group for jenkins"  
  
    ingress {  
        from_port = 8080  
        to_port   = 8080  
        protocol  = "tcp"  
        cidr_blocks = ["0.0.0.0/0"]  
    }  
  
    ingress {  
        from_port = 22
```



```
to_port    = 22
protocol   = "tcp"
cidr_blocks = ["0.0.0.0/0"]
}

# outbound from Jenkins server
egress {
  from_port = 0
  to_port   = 65535
  protocol  = "tcp"
  cidr_blocks = ["0.0.0.0/0"]
}

tags = {
  Name = "security_jenkins_port"
}
}

resource "aws_instance" "myFirstInstance" {
  ami          = "ami-0b9064170e32bde34"
  key_name     = var.key_name
  instance_type = var.instance_type
  security_groups = [ "security_jenkins_port" ]
  tags = {
    Name = "jenkins_instance"
  }
}
```



Create Elastic IP address

```
resource "aws_eip" "myFirstInstance" {  
    vpc      = true  
    instance = aws_instance.myFirstInstance.id  
    tags = {  
        Name = "jenkins_elstic_ip"  
    }  
}
```

Put AMI-ID in above highlighted space and Now execute the below command:

```
vishal@apsit:~/project-terraform$ terraform init
```

you should see like below screenshot.

```
vishal@apsit:~/project-terraform$ terraform init  
Initializing the backend...  
  
Initializing provider plugins...  
- Finding latest version of hashicorp/aws...  
- Installing hashicorp/aws v3.52.0...  
- Installed hashicorp/aws v3.52.0 (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.
```

Execute the below command



```
vishal@apsit:~/project-terraform$ terraform plan
```

the above command will show how many resources will be added.
Plan: 3 to add, 0 to change, 0 to destroy.

```
vishal@apsit:~/project-terraform$ terraform plan
Plan: 3 to add, 0 to change, 0 to destroy.

# aws_elb.myFirstInstance will be created
+ resource "aws_elb" "myFirstInstance" {
  + allocation_id      = (known after apply)
  + association_id     = (known after apply)
  + carrier_ip         = (known after apply)
```

Execute the below command

```
vishal@apsit:~/project-terraform$ terraform apply
```

Provide the value as Yes for applying terraform

```
Plan: 3 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
```

Plan: 3 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

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.



Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

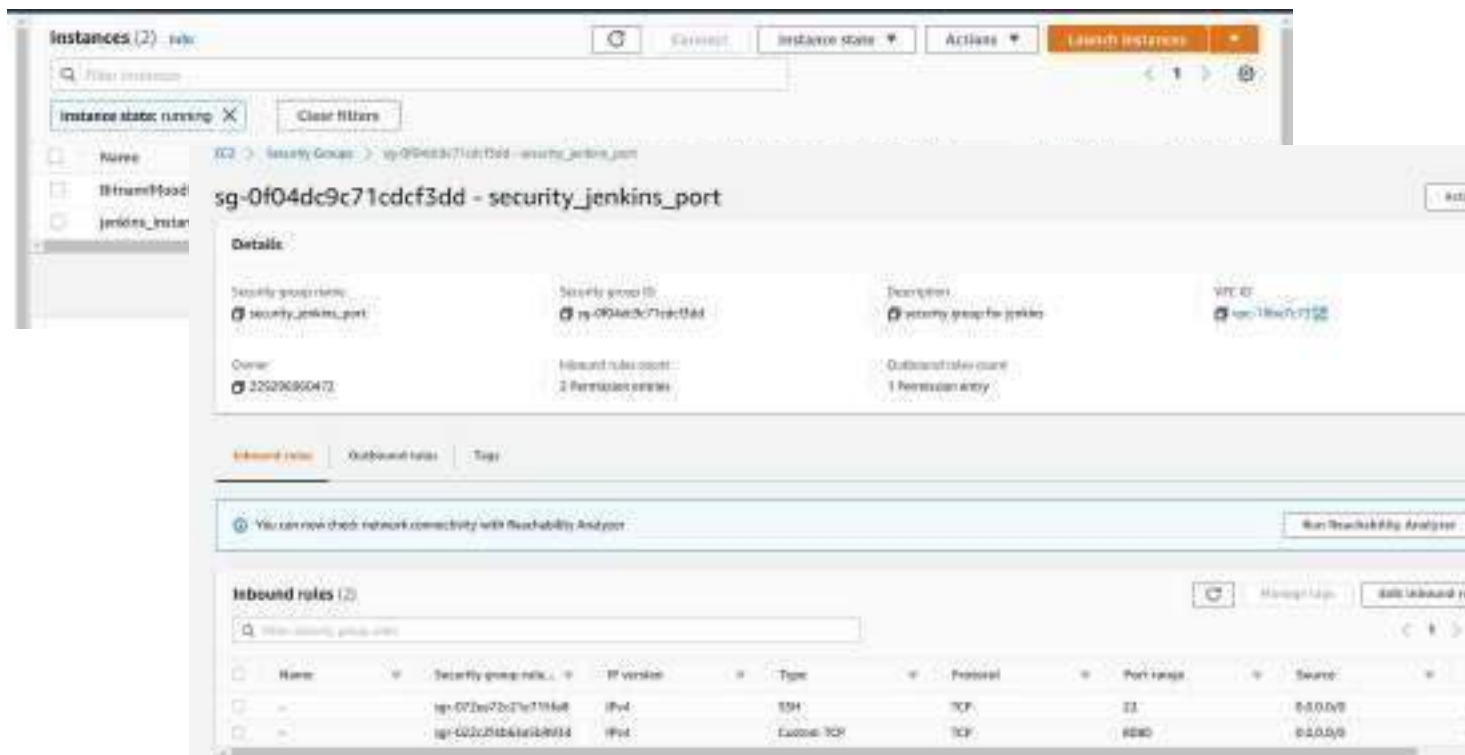
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

```
aws_instance.myFirstInstance: Creating...
aws_instance.myFirstInstance: Still creating... [10s elapsed]
aws_instance.myFirstInstance: Still creating... [20s elapsed]
aws_instance.myFirstInstance: Still creating... [30s elapsed]
aws_instance.myFirstInstance: Creation complete after 32s [id=i-0a4a0fb7e55252d0f]
aws_eip.myFirstInstance: Creating...
aws_eip.myFirstInstance: Creation complete after 1s [id=eipalloc-0fd8f60524b10fc93]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Now login to EC2 console, to see the new instances up and running, you can see Jenkins_instance is up and running which we deploy from terraform.



You can also check the security group resource details which you created from terraform :



PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



Terraform destroy

you can also destroy or delete your instance by using terraform destroy command :

```
vishal@apsit:~/project-terraform$ terraform destroy
```

```
Plan: 0 to add, 0 to change, 3 to destroy.
```

```
Do you really want to destroy all resources?
```

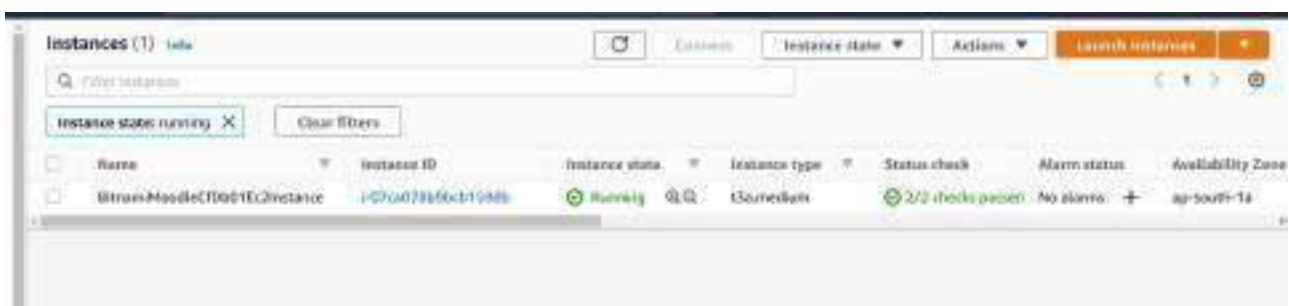
```
Terraform will destroy all your managed infrastructure, as shown above.  
There is no undo. Only 'yes' will be accepted to confirm.
```

```
Enter a value: yes
```

```
Enter a value: yes
```

```
aws_eip.myFirstInstance: Destroying... [id=eipalloc-0fd8f60524b10fc93]  
aws_security_group.security_jenkins_port: Destroying... [id=sg-0f04dc9c71cdf3dd]  
aws_eip.myFirstInstance: Destruction complete after 2s  
aws_instance.myFirstInstance: Destroying... [id=i-0a4a0fb7e55252d0f]  
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdf3dd, 10s elapsed]  
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 10s elapsed]  
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdf3dd, 20s elapsed]  
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 20s elapsed]  
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0f04dc9c71cdf3dd, 30s elapsed]  
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 30s elapsed]  
aws_security_group.security_jenkins_port: Destruction complete after 38s  
aws_instance.myFirstInstance: Still destroying... [id=i-0a4a0fb7e55252d0f, 40s elapsed]  
aws_instance.myFirstInstance: Destruction complete after 40s  
  
Destroy complete! Resources: 3 destroyed.
```

Now you can see instance which you created by using terraform is deleted successfully from aws console also you can check it will removed successfully:



All the Resources including Security groups, EC2 instances using terraform will be deleted. In this way we can automate infrastructure set up using terraform in aws cloud.



PARSHVANATH CHARITABLE TRUST'S
A. P. SHAH INSTITUTE OF TECHNOLOGY
Department of Information Technology
(NBA Accredited)



OUTPUT SCREENSHOTS :

Account details

Edit account name, email, and password

Account name
AmitRajgore

Email address
amitrajgore20@apsit.edu.in

AWS account ID
905418268590

Canonical user ID
95e922ef4162df7012245c7039ccbae579fa85496a84f0818979676-def91cc48

Multi-factor authentication (MFA) (1)

Remove

Resync

Assign MFA device

Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Type	Identifier	Certifications	Created on
<input type="radio"/> Virtual	arn:aws:iam::905418268590:mfa/Amit	Not Applicable	Thu Jul 11 2024

Access keys (0)

Create access key

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
<input type="radio"/> AKIA5FTZCMAJHESMIN72	Now	None	N/A	N/A	<input checked="" type="checkbox"/> Active

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ curl -s https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip -o "awscli.zip"
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 56.9M 100 56.9M 0 0 42.2M 0 0:00:01 0:00:01 -- -- 42.2M
Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access
keys (active or inactive) at a time. Learn more
```

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
<input type="radio"/> AKIA5FTZCMAJHESMIN72	Now	None	N/A	N/A	<input checked="" type="checkbox"/> Active

```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ aws --version
aws-cli/2.17.20 Python/3.11.9 Linux/5.4.0-150-generic exe/x86_64.ubuntu.18
```



PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



```
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ aws configure
AWS Access Key ID [*****4PM7]: AKIA5FTZCMAJ02R14PM7
AWS Secret Access Key [*****8lfl]: rjw1W4vRNjkGx8chYGgc7tUhljTbyry9GUV8lfl
Default region name [Global]: ap-south-1
Default output format [None]:
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ cd -
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ cd Amit
bash: cd: Amit: No such file or directory
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ mkdir Amit
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~$ cd Amit
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Amit$ pwd
/home/apsit/Amit
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Amit$ sudo nano variables.tf
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Amit$ sudo nano main.tf
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Amit$ sudo nano variables.tf
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Amit$ sudo nano main.tf
apsit@apsit-HP-280-Pro-G6-Microtower-PC:~/Amit$
```



```
File Edit View Search Terminal Help
awscli@apsit-HP-280-Pro-GS-Microtower-PC:~/AWS$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.60.0...
- Installed hashicorp/aws v5.60.0 (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.
awscli@apsit-HP-280-Pro-GS-Microtower-PC:~/AWS$
```

```
awscli@apsit-HP-280-Pro-GS-Microtower-PC:~/AWS$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.60.0...
- Installed hashicorp/aws v5.60.0 (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.
awscli@apsit-HP-280-Pro-GS-Microtower-PC:~/AWS$
```




PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



```
Warning: Argument is deprecated
  with aws_elb.myFirstInstance,
  on main.tf line 42, in resource "aws_elb" "myFirstInstance":
  42: vpc = true

use domain attribute instead

(and one more similar warning elsewhere)

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

Enter a value: yes

aws_security_group.security_jenkins_port: Creating...
aws_instance.myFirstInstance: Creating...
aws_security_group.security_jenkins_port: Creation complete after 2s [id=sg-0fc1761930fa1c626]
aws_instance.myFirstInstance: Still creating... [10s elapsed]
aws_instance.myFirstInstance: Still creating... [20s elapsed]
aws_instance.myFirstInstance: Still creating... [30s elapsed]
aws_instance.myFirstInstance: Creation complete after 31s [id=t-0879b4fb40869846c]
aws_elb.myFirstInstance: Creating...
aws_elb.myFirstInstance: Creation complete after 2s [id=elballoc-0b6b9dd05add9e]

Warning: Argument is deprecated
  with aws_elb.myFirstInstance,
  on main.tf line 42, in resource "aws_elb" "myFirstInstance":
  42: vpc = true

use domain attribute instead

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
apaligapall-HP-Z80-Pro-G8-Microtower-PC:~/AHLIS
```

The screenshot shows the AWS Management Console 'Instances' page. It displays a table with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm states, Availability Zone, and Public IP+DNS. One instance is listed with the name 'jenkins_instance', ID 'i-1879a4fb40869846c', state 'Running', type 't2.micro', and is located in 'ap-south-1a'.

Name	Instance ID	Instance state	Instance type	Status check	Alarm states	Availability Zone	Public IP+DNS
jenkins_instance	i-1879a4fb40869846c	Running	t2.micro	2/2 (Checks passing)	view alarms	ap-south-1a	ec2-13-232-123



PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



```
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_elb.myFirstInstance: Destroying... [id=elballoc-0b4080956a51add94]
aws_security_group.security_jenkins_port: Destroying... [id=sg-0fc1761930fa1c626]
aws_elb.myFirstInstance: Destruction complete after 1s
aws_instance.myFirstInstance: Destroying... [id=i-007904fb40069046c]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0fc1761930fa1c626, 10s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-007904fb40069046c, 10s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0fc1761930fa1c626, 20s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-007904fb40069046c, 20s elapsed]
aws_security_group.security_jenkins_port: Still destroying... [id=sg-0fc1761930fa1c626, 30s elapsed]
aws_instance.myFirstInstance: Still destroying... [id=i-007904fb40069046c, 30s elapsed]
aws_security_group.security_jenkins_port: Destruction complete after 37s
aws_instance.myFirstInstance: Still destroying... [id=i-007904fb40069046c, 40s elapsed]
aws_instance.myFirstInstance: Destruction complete after 40s

Destroy complete! Resources: 3 destroyed.
apsit@apsit-HP-200-Pro-G6-Microtower-PC:~/AHITS
```

Conclusion: The knowledge of applying IaC (Infrastructure as Code) was quite amusing to learn. It is very well explained in a simple proficient way . It broadens our knowledge domain of Devops and its practical application.