Semester	T.E. Semester V – Information Technology
Subject	Advance DevOps Lab
Subject Professor In-charge	Prof. Indu Anoop
Laboratory	(Leave blank for now)

Student Name	Rahul Chougule	
Roll Number	20101A0055	
Grade and Subject Teacher's Signature		

Experiment	6	
Problem Statement	To build, change, destroy AWS/GCP/ Microsoft Azure/DigitalOcean infrastructure using terraform	
Resources / Apparatus Required	Hardware: Computer System	Software: Web Browser
Details	**Terraform** Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc. Terraform can manage both existing service providers and custom in-house solutions. **Key Features** **Infrastructure as Code**: You describe your infrastructure using Terraform's high-level configuration language in human-readable, declarative configuration files. This allows you to create a blueprint that you can version, share, and reuse.	

Resource Graph

Terraform builds a resource graph and creates or modifies non-dependent resources in parallel. This allows Terraform to build resources as efficiently as possible and gives you greater insight into your infrastructure.

Change Automation

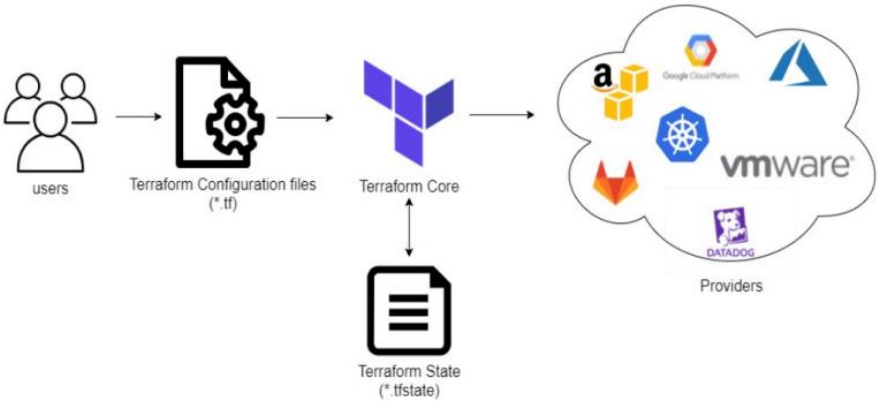
Terraform can apply complex changesets to your infrastructure with minimal human interaction. When you update configuration files, Terraform determines what changed and creates incremental execution plans that respect dependencies.

How does Terraform work?

Terraform works with two major components:

one is the **terraform core**: it takes the terraform configuration which is being provided by the user and then takes the terraform state which is managed by terraform itself. As such, this gets fed into the core that is responsible for figuring out what is that graph of our different resources for example how these different pieces relate to each other or what needs to be created/updated/destroyed, it does all the essential lifecycle management.

On the backside, terraform supports many different **providers**, such as: cloud providers (AWS,GCP,AZURE) and they also could be on-premise infrastructure (VMware, OpenStack.) But this support is not restricted or limited only to Infrastructure As A Service , terraform can also manage higher level like Platform As A Service(Kubernetes, Lambdas..)or even Software As A Service (DataDog, GitHub..)



All of these are important pieces of the infrastructure, they are all part of the logical end-to-end delivery.

	Terraform has over a hundred providers for different technologies, and each provider gives terraform users access to their resources. It also gives you the ability to create infrastructure at different levels. **Terraform Workflow:** These are the list of steps we are going to perform **1.** Create a file and save it as **main.tf** 2. Execute the command **terraform init** to initialize 3. Execute the command **terraform plan** to check what change would be made. (Should always do it) 4. If you are happy with the changes it is claiming to make, then execute **terraform apply** to commit and start the build

Code	**Pre-requisite:** An AMI of ubuntu 20 system with terraform installed.
	
	Steps to build, change, destroy AWS infrastructure using Terraform
	
	Step: 1 : To BUILD an AWS infrastructure
	
	1.1 Write your **main.tf** file
	
	Use command to create a file and edit it
	touch main.tf
	nano main.tf
	Edit to following contents
	`provider "aws"{`
	`  access_key = "AKIAQFSXWMIZOST6QORJ"`
	`  secret_key = "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"`
	`  region = "us-east-1"`
	`}`
	
	`resource "aws_instance" "terraform-VIT"{`
	`  ami = "ami-0149b2da6ceec4bb0"`
	`  instance_type = "t2.micro"`
	`}`
	Replace the access key and secret key values of the new IAM user which needs to be created in the region mentioned. Also replace the ami value to the virtual system's ami value [From launch instances portal]
	
	1.2 Initialize the terraform
	
	Write the command
	terraform init

```
root@ip-172-31-28-218:/opt# terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.32.0...
- Installed hashicorp/aws v4.32.0 (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!
```

1.3 Execute plan phase to understand what changes to be done.

terraform plan -lock=false

```
root@ip-172-31-28-218:/opt# terraform plan -lock=false

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.terraform-VIT will be created
  + resource "aws_instance" "terraform-VIT" {
      + ami                                  = "ami-0149b2da6ceec4bb0"
Plan: 1 to add, 0 to change, 0 to destroy.
```

1.4 Apply the actions which were planned in apply phase

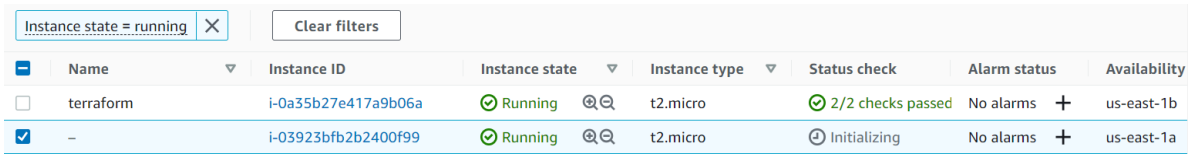
terraform apply -lock=false

```
root@ip-172-31-28-218:/opt# terraform apply -lock=false

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.terraform-VIT will be created
  + resource "aws_instance" "terraform-VIT" {
      + ami                                  = "ami-0149b2da6ceec4bb0"
Plan: 1 to add, 0 to change, 0 to destroy.

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

  Enter a value: yes

aws_instance.terraform-VIT: Creating...
aws_instance.terraform-VIT: Still creating... [10s elapsed]
aws_instance.terraform-VIT: Still creating... [20s elapsed]
aws_instance.terraform-VIT: Still creating... [30s elapsed]
aws_instance.terraform-VIT: Creation complete after 32s [id=i-03923bfb2b2400f99]

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

Type ‘yes’ to confirm to apply.

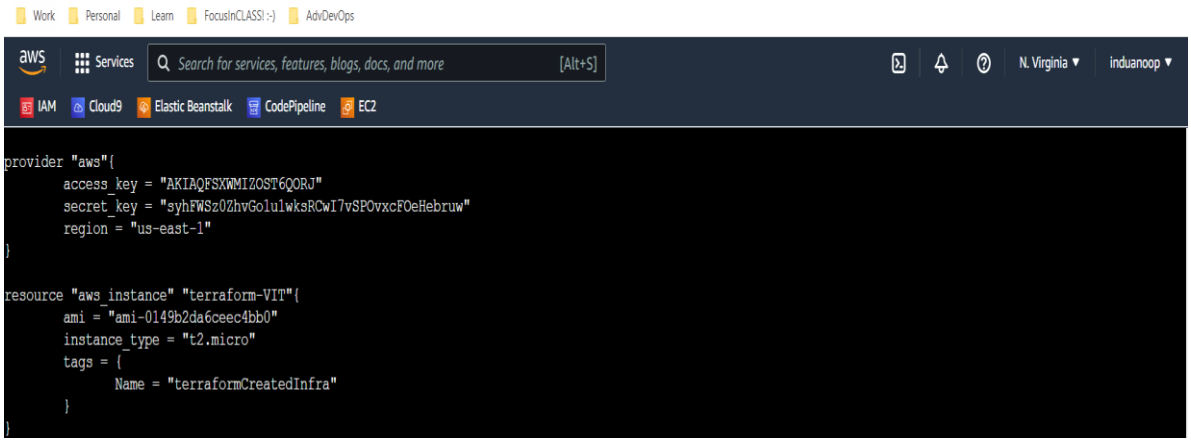
Step 2: Confirm the infrastructure created

Go to EC2 console to check if a new instance is created as per the code written in main.tf file.



Step 3: CHANGE the infrastructure created using terraform

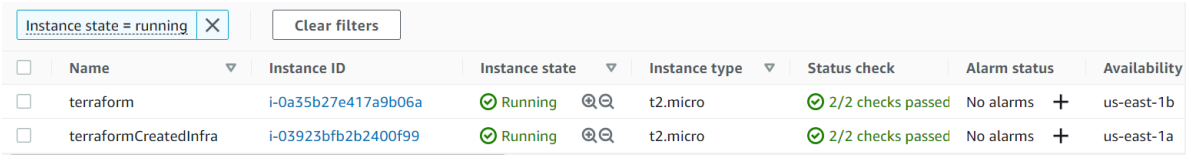
Modify main.tf to include instance name.



Repeat steps from 1.2.

```
Terraform will perform the following actions:

  # aws_instance.terraform-VIT will be updated in-place
  ~ resource "aws_instance" "terraform-VIT" {
        id                                   = "i-03923bfb2b2400f99"
      ~ tags                                 = {
          + "Name" = "terraformCreatedInfra"
        }
      ~ tags_all                             = {
          + "Name" = "terraformCreatedInfra"
        }
        # (29 unchanged attributes hidden)

        # (7 unchanged blocks hidden)
    }

Plan: 0 to add, 1 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.terraform-VIT: Modifying... [id=i-03923bfb2b2400f99]
aws_instance.terraform-VIT: Modifications complete after 1s [id=i-03923bfb2b2400f99]

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

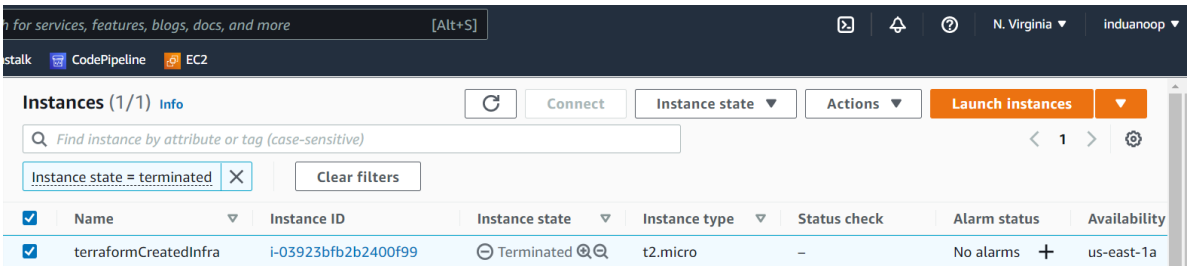


Resource successfully changed to include instance name.

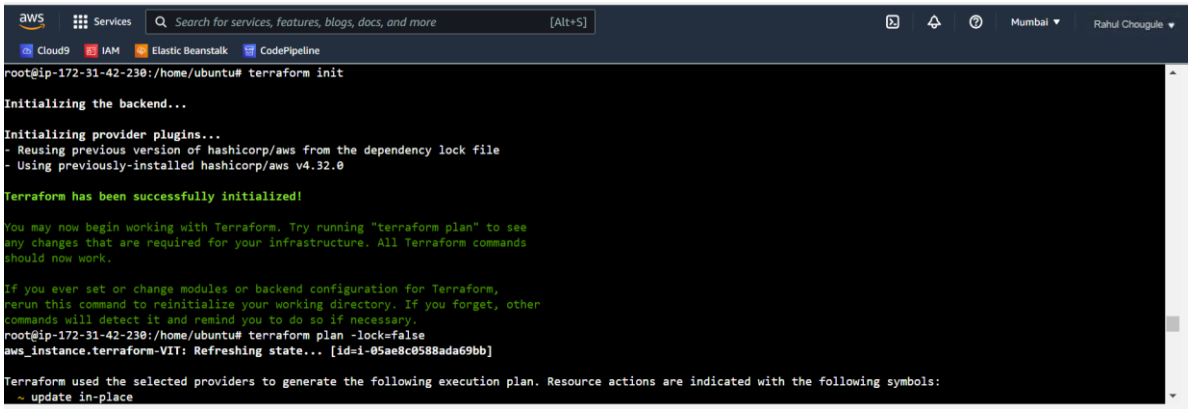
Step 4: DESTROY the built infrastructure

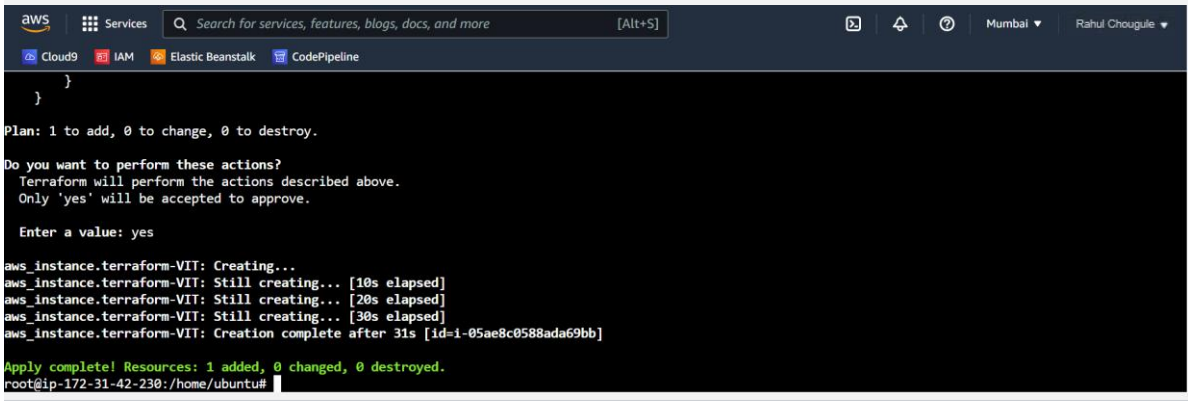
terraform destroy

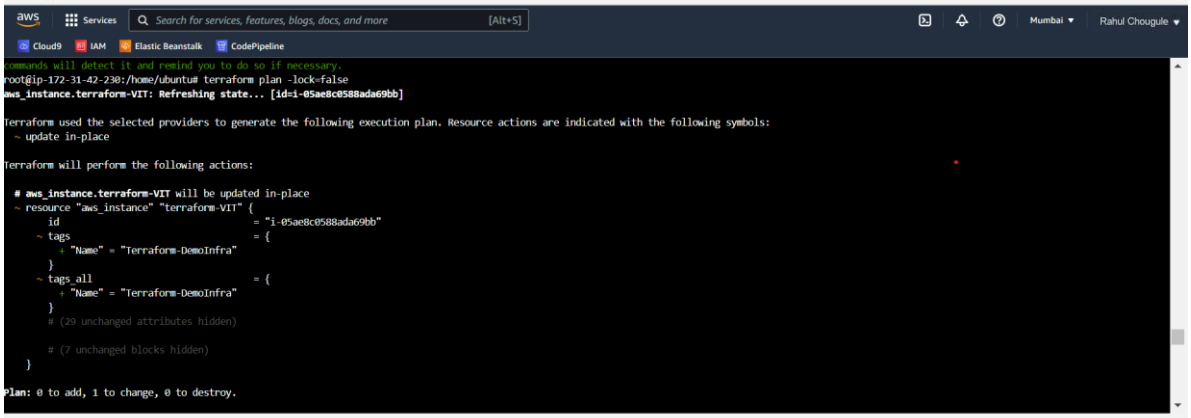
```
Plan: 0 to add, 0 to change, 1 to destroy.

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

aws_instance.terraform-VIT: Destroying... [id=i-03923bfb2b2400f99]
aws_instance.terraform-VIT: Still destroying... [id=i-03923bfb2b2400f99, 10s elapsed]
aws_instance.terraform-VIT: Still destroying... [id=i-03923bfb2b2400f99, 20s elapsed]
aws_instance.terraform-VIT: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
```

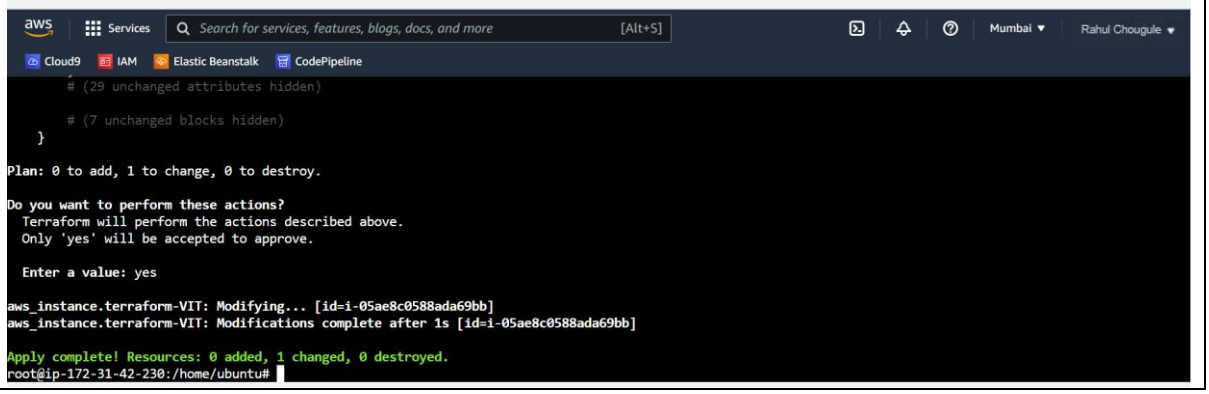


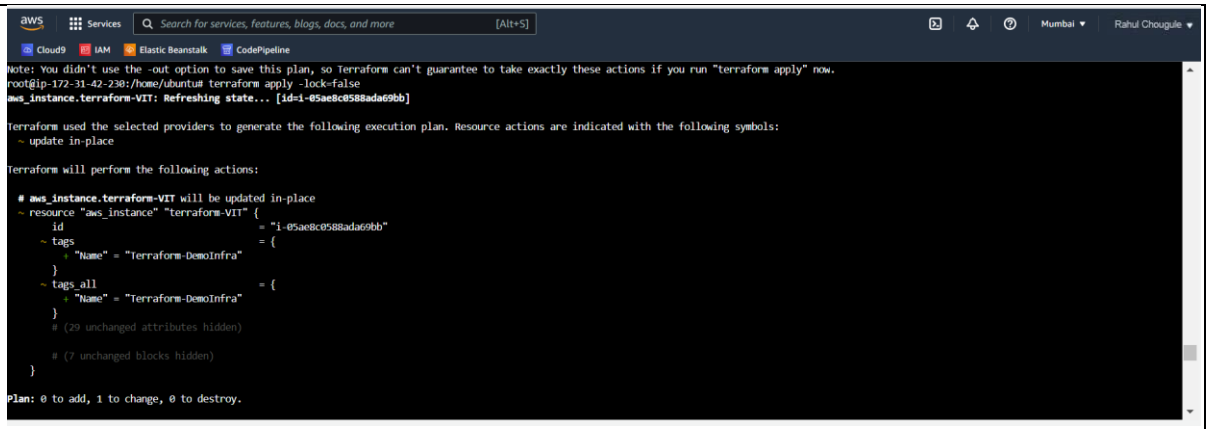
Outputs:-


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Mumbai
Rahul Chougule
Cloud9
IAM
Elastic Beanstalk
CodePipeline
root@ip-172-31-42-230:/home/ubuntu# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.32.0
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.
root@ip-172-31-42-230:/home/ubuntu# terraform plan -lock=false
aws_instance.terraform-VIT: Refreshing state... [id=i-05ae8c0588ada69bb]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
~ update in-place

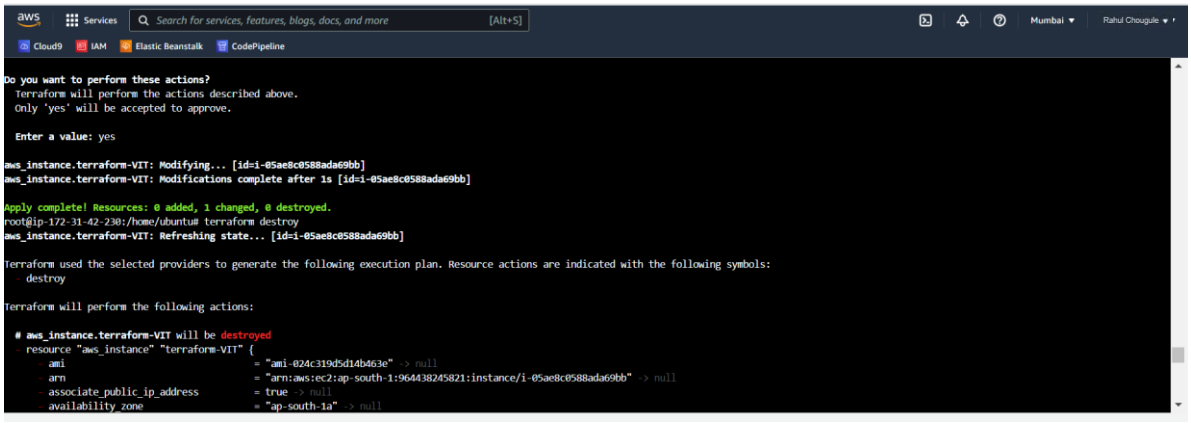

aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Mumbai
Rahul Chougule
Cloud9
IAM
Elastic Beanstalk
CodePipeline
}
}
Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes
aws_instance.terraform-VIT: Creating...
aws_instance.terraform-VIT: Still creating... [10s elapsed]
aws_instance.terraform-VIT: Still creating... [20s elapsed]
aws_instance.terraform-VIT: Still creating... [30s elapsed]
aws_instance.terraform-VIT: Creation complete after 31s [id=i-05ae8c0588ada69bb]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
root@ip-172-31-42-230:/home/ubuntu#


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Mumbai
Rahul Chougule
Cloud9
IAM
Elastic Beanstalk
CodePipeline
commands will detect it and remind you to do so if necessary.
root@ip-172-31-42-230:/home/ubuntu# terraform plan -lock=false
aws_instance.terraform-VIT: Refreshing state... [id=i-05ae8c0588ada69bb]
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
~ update in-place
Terraform will perform the following actions:
# aws_instance.terraform-VIT will be updated in-place
~ resource "aws_instance" "terraform-VIT" {
id = "i-05ae8c0588ada69bb"
~ tags = {
+ "Name" = "Terraform-DemoInfra"
}
~ tags_all = {
+ "Name" = "Terraform-DemoInfra"
}
# (29 unchanged attributes hidden)
# (7 unchanged blocks hidden)
}
Plan: 0 to add, 1 to change, 0 to destroy.

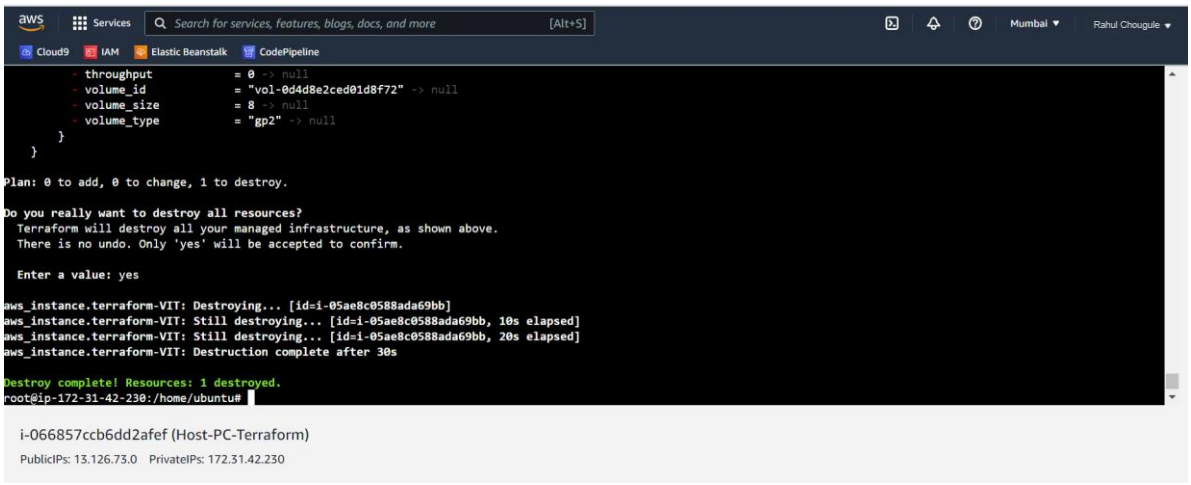

aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Mumbai
Rahul Chougule
Cloud9
IAM
Elastic Beanstalk
CodePipeline
# (29 unchanged attributes hidden)
# (7 unchanged blocks hidden)
}
Plan: 0 to add, 1 to change, 0 to destroy.
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes
aws_instance.terraform-VIT: Modifying... [id=i-05ae8c0588ada69bb]
aws_instance.terraform-VIT: Modifications complete after 1s [id=i-05ae8c0588ada69bb]
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
root@ip-172-31-42-230:/home/ubuntu#



```
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
root@ip-172-31-42-230:/home/ubuntu# terraform apply -lock=false
aws_instance.terraform-VIT: Refreshing state... [id=i-05ae8c0588ada69bb]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_instance.terraform-VIT will be updated in-place
  ~ resource "aws_instance" "terraform-VIT" {
        id                                   = "i-05ae8c0588ada69bb"
      ~ tags                                 = {
          + "Name" = "Terraform-DemoInfra"
        }
      ~ tags_all                             = {
          + "Name" = "Terraform-DemoInfra"
        }
        # (29 unchanged attributes hidden)

        # (7 unchanged blocks hidden)
    }

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



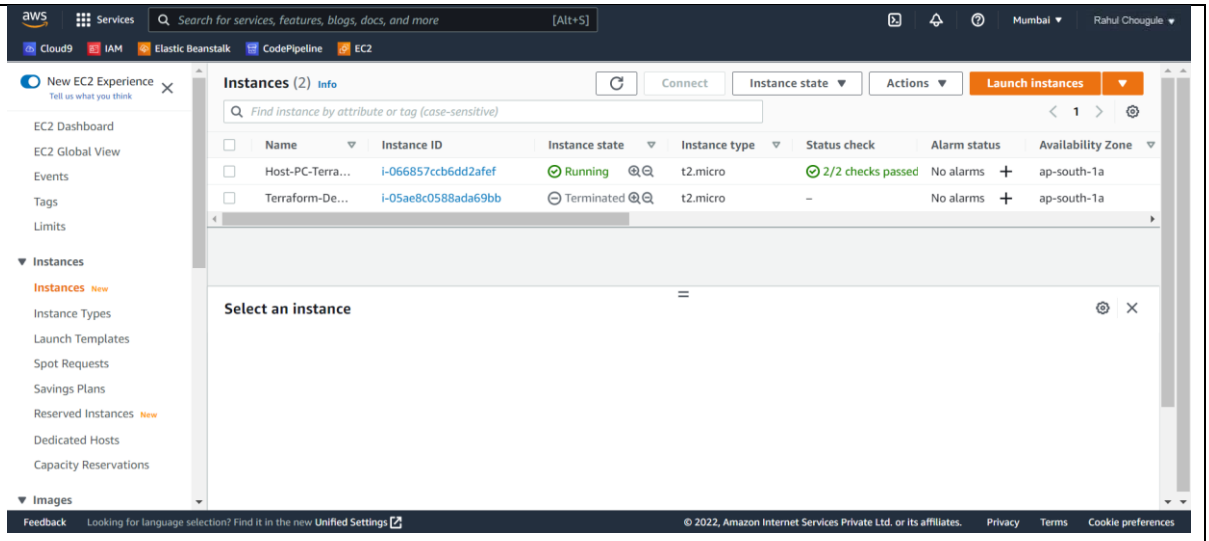
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.terraform-VIT: Modifying... [id=i-05ae8c0588ada69bb]
aws_instance.terraform-VIT: Modifications complete after 1s [id=i-05ae8c0588ada69bb]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
root@ip-172-31-42-230:/home/ubuntu# terraform destroy
aws_instance.terraform-VIT: Refreshing state... [id=i-05ae8c0588ada69bb]

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.terraform-VIT will be destroyed
  - resource "aws_instance" "terraform-VIT" {
      - ami                                  = "ami-024c319d5d14b463e" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:964438245821:instance/i-05ae8c0588ada69bb" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
```



```
      - throughput           = 0 -> null
      - volume_id            = "vol-0d4d8e2ced01d8f72" -> null
      - volume_size          = 8 -> null
      - volume_type          = "gp2" -> null
    }
  }

Plan: 0 to add, 0 to change, 1 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

aws_instance.terraform-VIT: Destroying... [id=i-05ae8c0588ada69bb]
aws_instance.terraform-VIT: Still destroying... [id=i-05ae8c0588ada69bb, 10s elapsed]
aws_instance.terraform-VIT: Still destroying... [id=i-05ae8c0588ada69bb, 20s elapsed]
aws_instance.terraform-VIT: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
root@ip-172-31-42-230:/home/ubuntu#
```

i-066857ccb6dd2afef (Host-PC-Terraform)

PublicIPs: 13.126.73.0 PrivateIPs: 172.31.42.230

	
Conclusion	Successfully implemented through code how to build, change, destroy AWS infrastructure using terraform