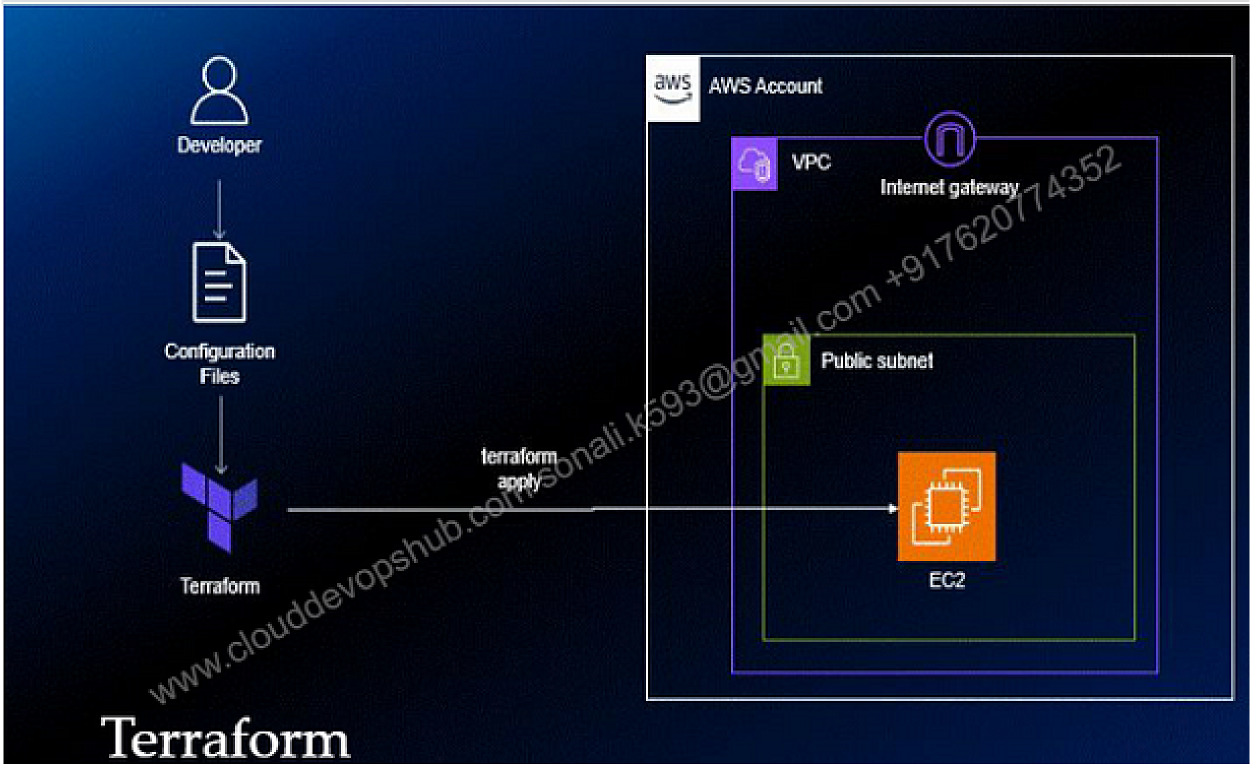
Terraform



Infrastructure as Code (IaC) means to manage your IT infrastructure using configuration file (code).

Benefits of IaC

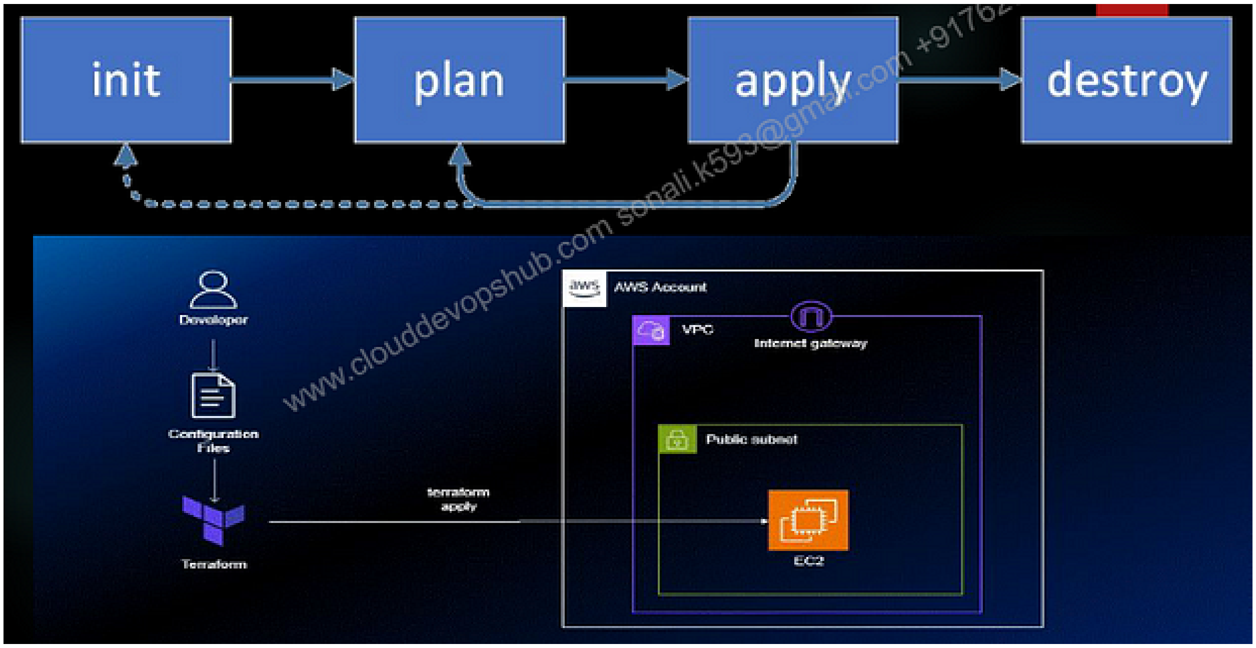
- **Scalability**: Easily scalable across environments and systems.
- **Speed**: Significantly reduces time for infrastructure provisioning.
- **Reduced Configuration Drift**: Keeps environments uniform and prevents issues due to "drift."
- **Disaster Recovery**: IaC templates can recreate environments quickly in the event of failures or disasters.

Popular IaC Tools

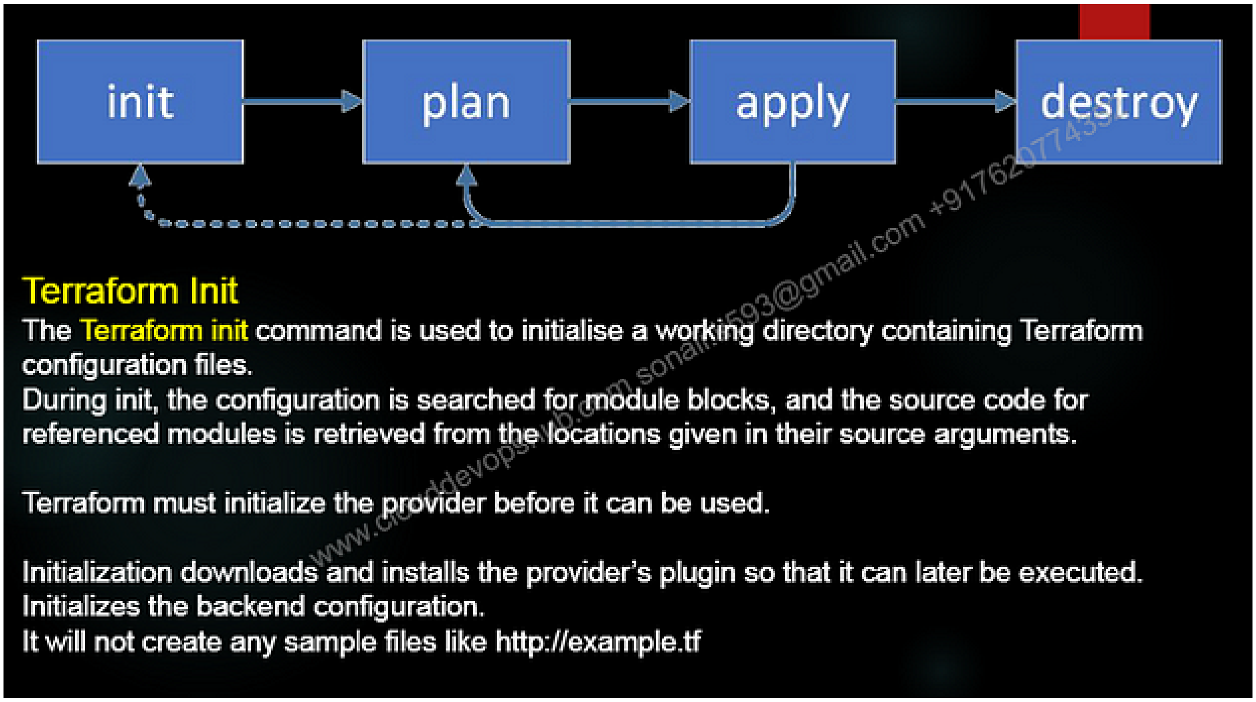
- **Terraform**: A declarative tool supporting multiple cloud providers.
- **AWS CloudFormation**: Specific to AWS, allowing management of AWS resources.
- **Ansible**: Primarily used for configuration management and automation but can handle infrastructure.

Terraform

Terraform is an Infrastructure as Code (IaC) tool developed by HashiCorp, widely used to define, provision, and manage infrastructure across multiple cloud providers and on-premises environments. Terraform uses a declarative configuration language (HashiCorp Configuration Language or HCL) to define infrastructure, enabling users to create infrastructure setups with reusable and versioned code.



Terraform Init — The init utility is used for initialization purpose



Terraform Init

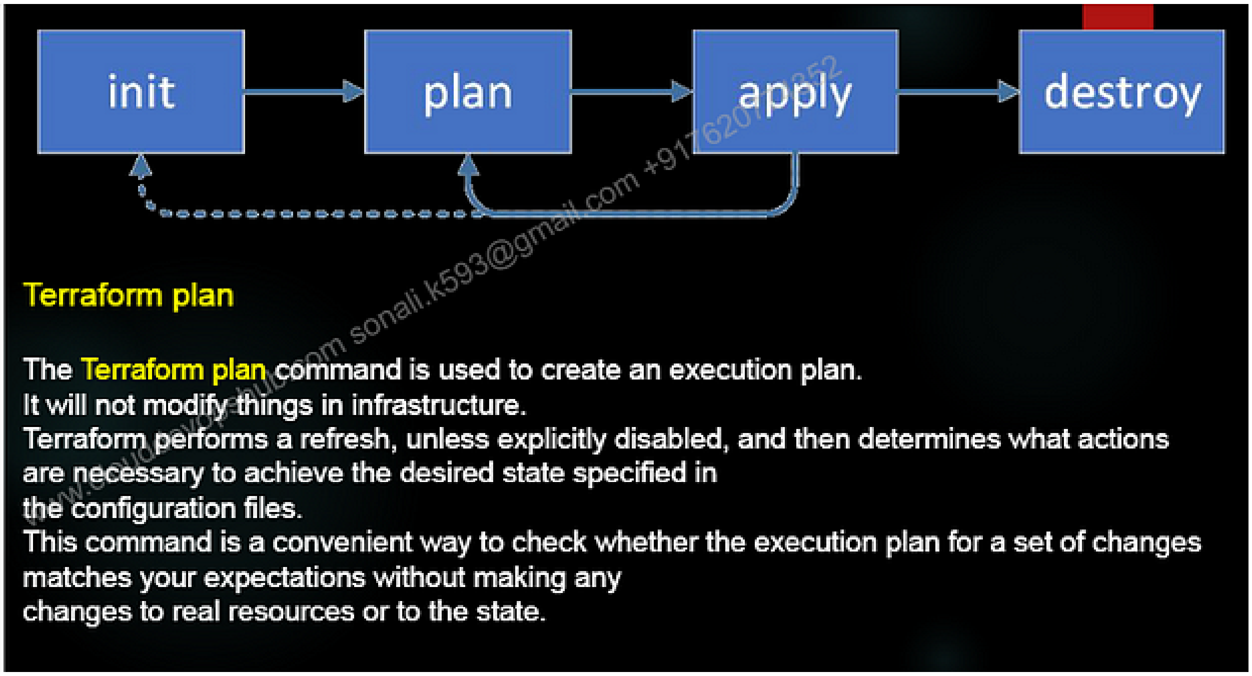
The Terraform init command is used to initialise a working directory containing Terraform configuration files.
During init, the configuration is searched for module blocks, and the source code for referenced modules is retrieved from the locations given in their source arguments.

Terraform must initialize the provider before it can be used.

Initialization downloads and installs the provider's plugin so that it can later be executed.
Initializes the backend configuration.
It will not create any sample files like http://example.tf

Terraform plan -

Creates an execution plan to show what changes Terraform will make to achieve the desired state. This is a crucial step to validate changes before applying them.

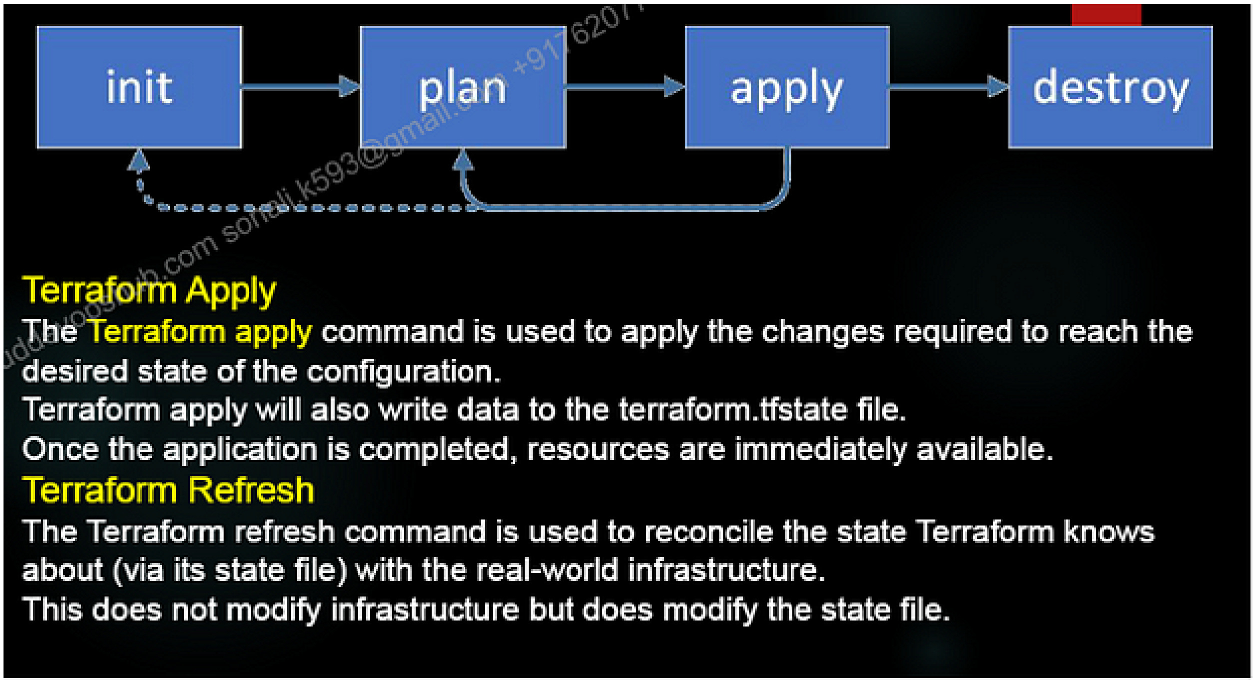


Terraform plan

The Terraform plan command is used to create an execution plan.
It will not modify things in infrastructure.
Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in
the configuration files.
This command is a convenient way to check whether the execution plan for a set of changes matches your expectations without making any
changes to real resources or to the state.

Terraform Apply -

Applies the changes required to reach the desired state. It provisions or modifies infrastructure resources as needed.



Terraform Apply

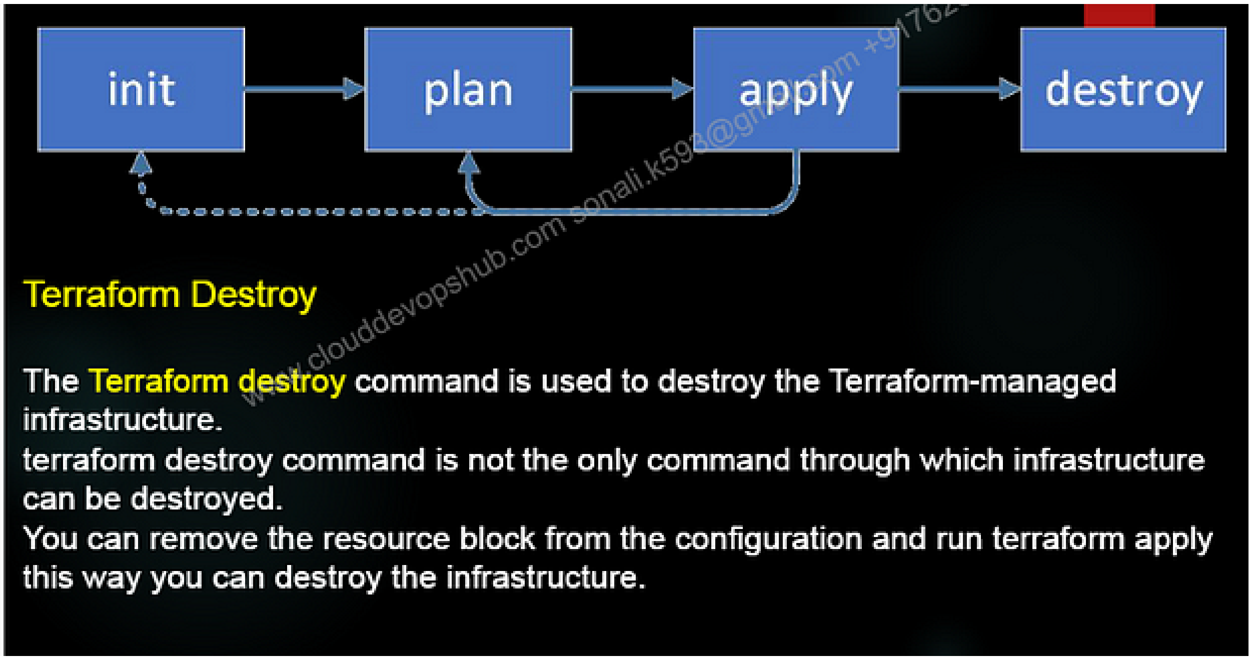
The Terraform apply command is used to apply the changes required to reach the desired state of the configuration.
Terraform apply will also write data to the terraform.tfstate file.
Once the application is completed, resources are immediately available.

Terraform Refresh

The Terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure.
This does not modify infrastructure but does modify the state file.

Terraform Destroy-

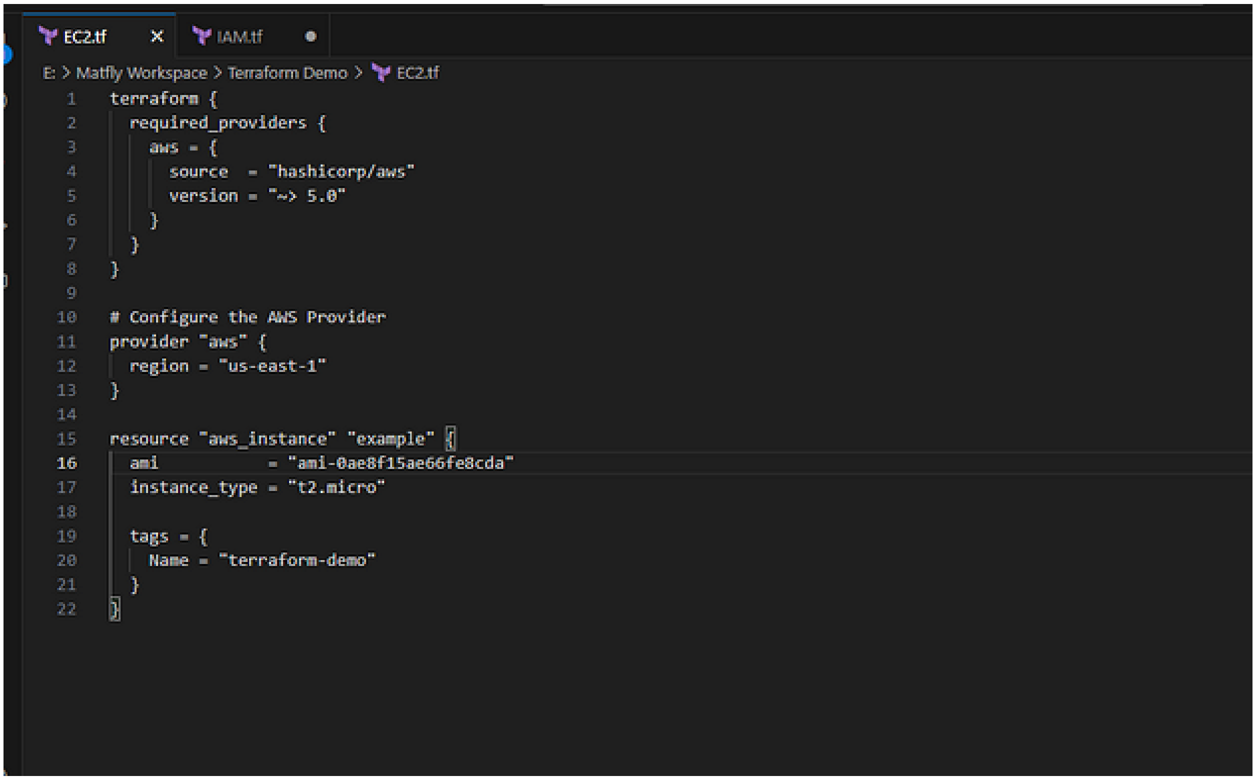
Removes all resources defined in the configuration, essentially tearing down the infrastructure.



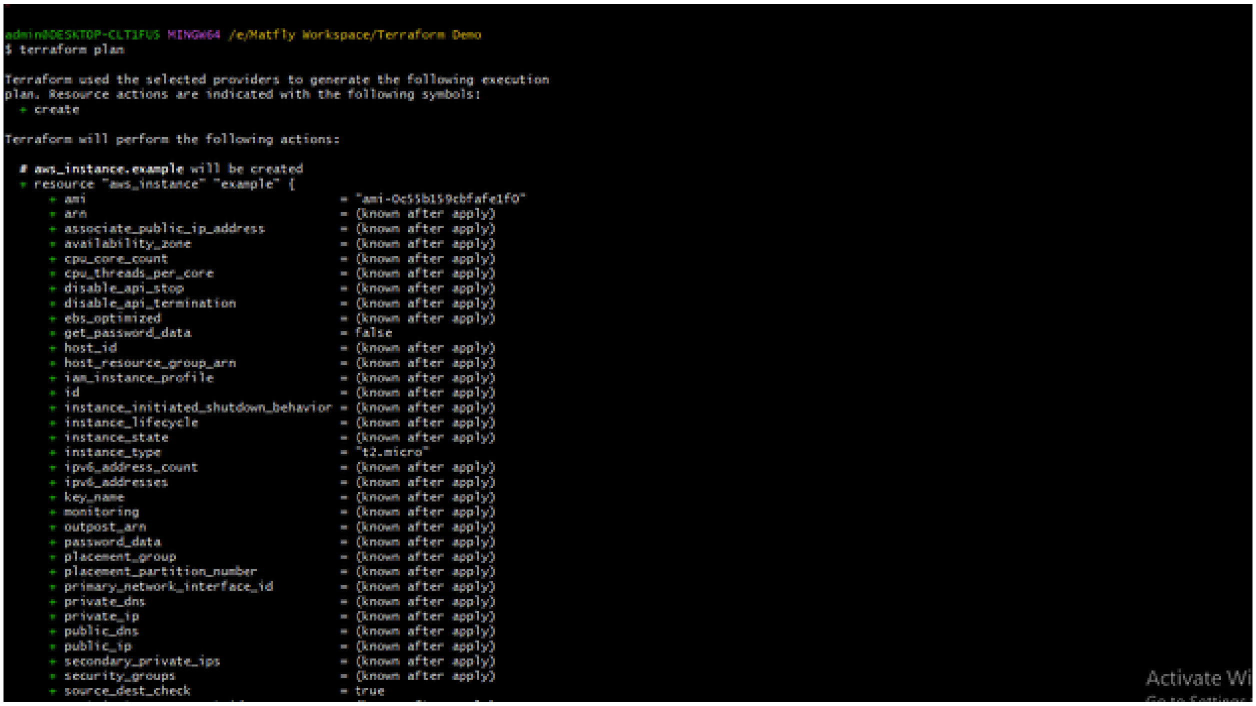
Terraform Destroy

The Terraform destroy command is used to destroy the Terraform-managed infrastructure.
terraform destroy command is not the only command through which infrastructure can be destroyed.
You can remove the resource block from the configuration and run terraform apply this way you can destroy the infrastructure.

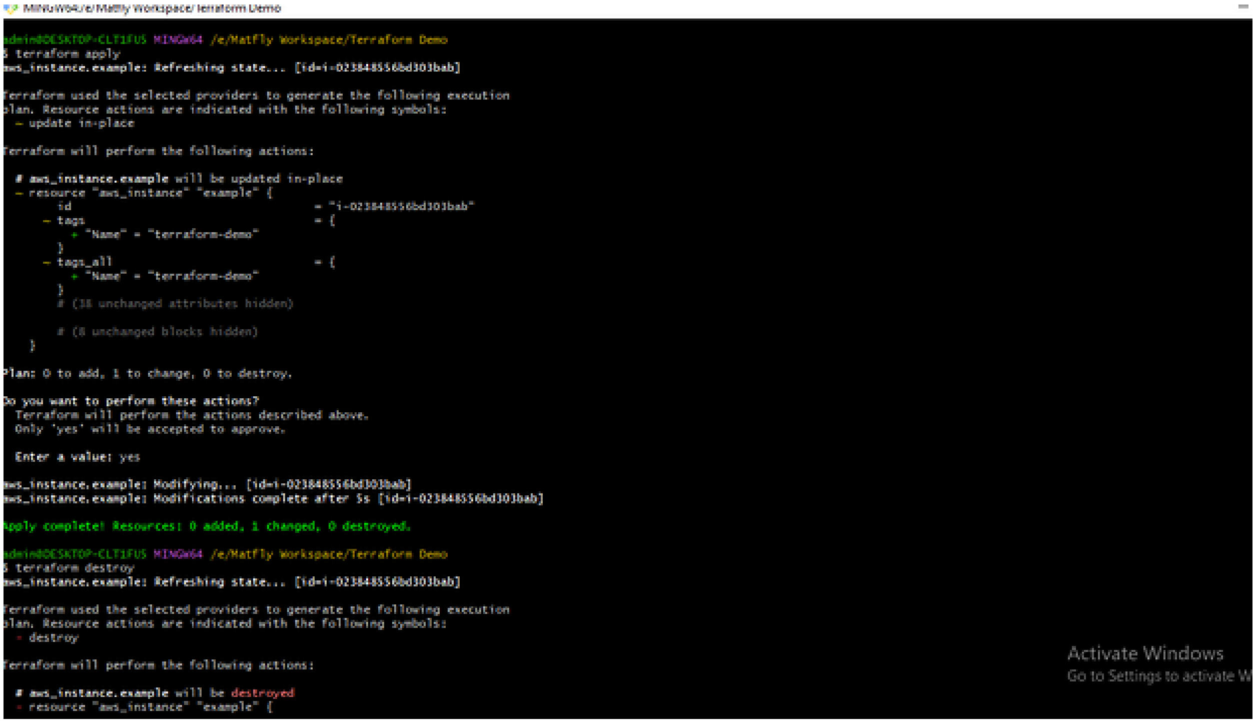
EC2 Instance Creation -



```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "example" {
  ami           = "ami-0ae8f15ae66fe8cda"
  instance_type = "t2.micro"

  tags = {
    Name = "terraform-demo"
  }
}
```

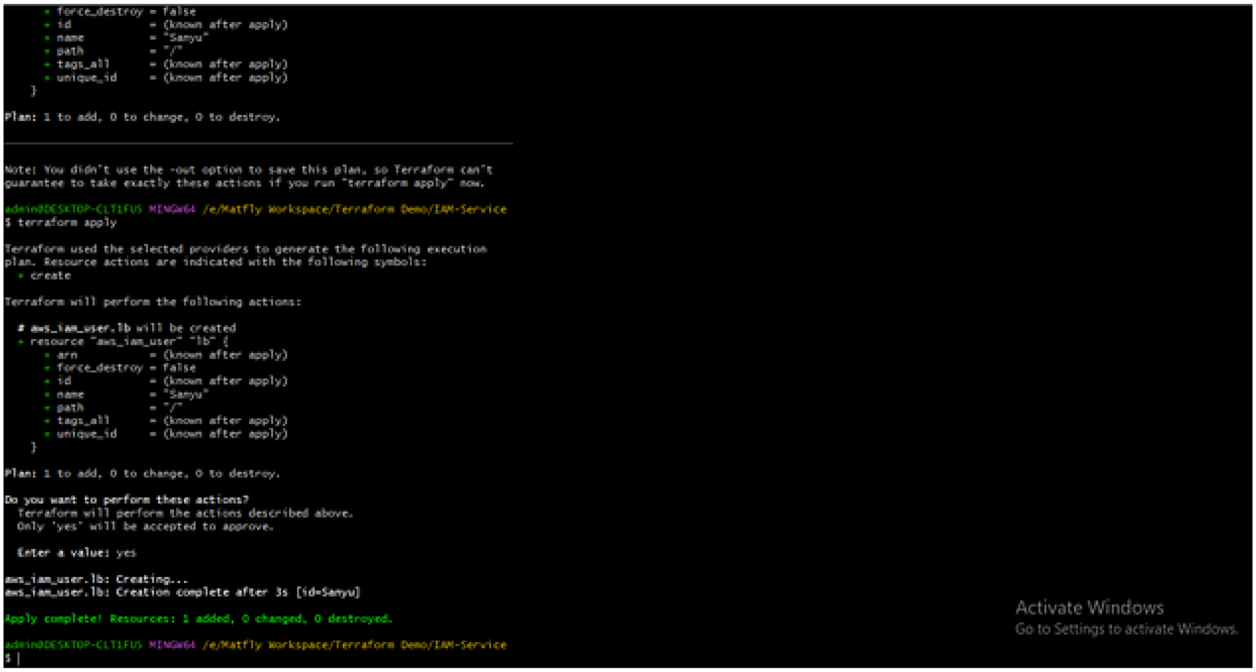


```
admin@DESKTOP-CLT1FUS MINGW64 /e/Matfly Workspace/Terraform Demo
$ terraform plan

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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-0c55b159cbfafe1f0"
      + 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_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (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)
      + placement_partition_number           = (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
```



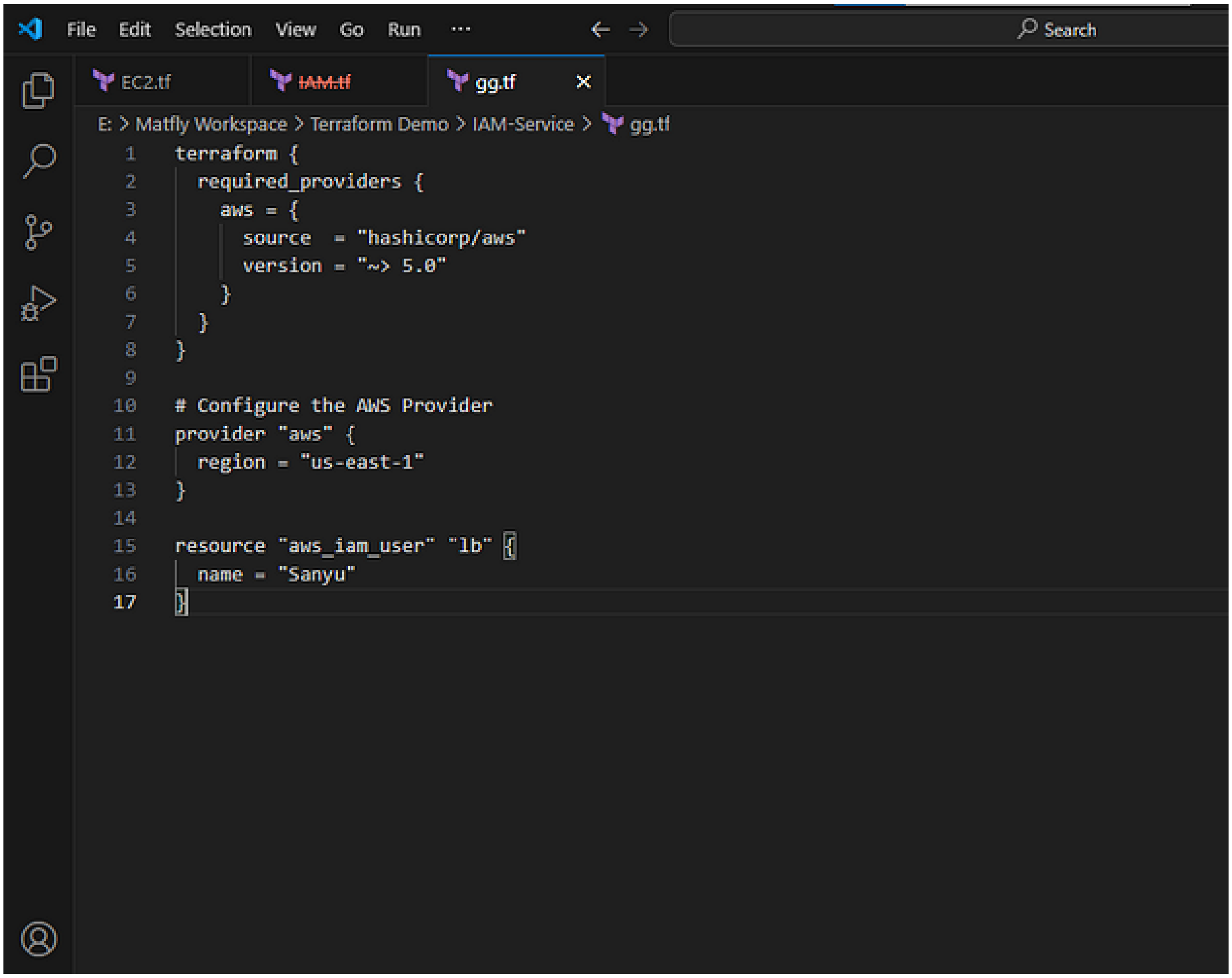
MINGW64:/e/Matfly Workspace/Terraform Demo

```
admin@DESKTOP-CLT1FUS MINGW64 /e/Matfly Workspace/Terraform Demo
$ terraform apply
aws_instance.example: Refreshing state... [id=i-023848556bd303bab]

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.example will be updated in-place
  ~ resource "aws_instance" "example" {
        id                                   = "i-023848556bd303bab"
      ~ tags                                 = {
          + "Name" = "terraform-demo"
        }
      ~ tags_all                             = {
          + "Name" = "terraform-demo"
        }
        # (38 unchanged attributes hidden)

        # (8 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.example: Modifying... [id=i-023848556bd303bab]
aws_instance.example: Modifications complete after 5s [id=i-023848556bd303bab]

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

admin@DESKTOP-CLT1FUS MINGW64 /e/Matfly Workspace/Terraform Demo
$ terraform destroy
aws_instance.example: Refreshing state... [id=i-023848556bd303bab]

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.example will be destroyed
  - resource "aws_instance" "example" {
```

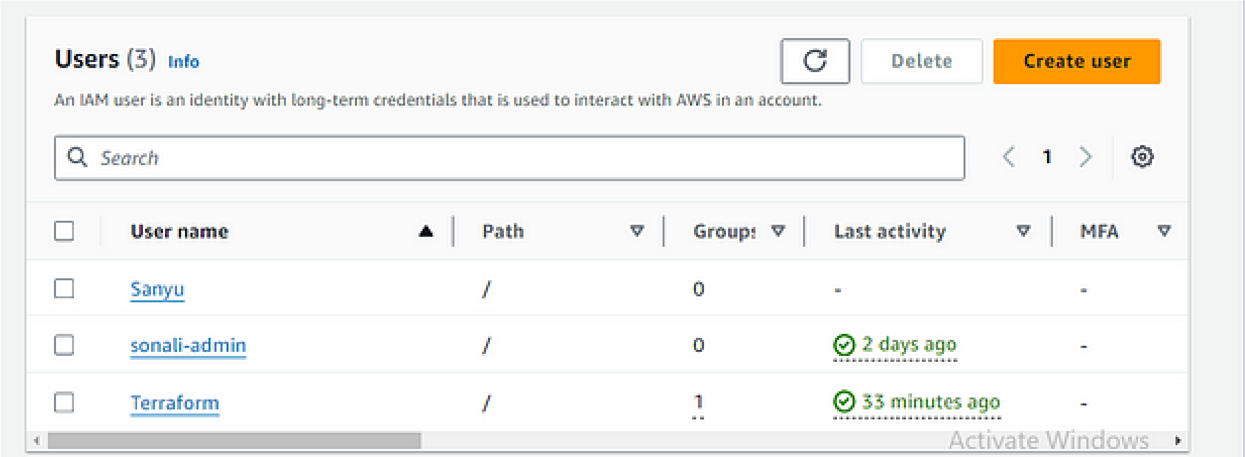


```
      + force_destroy = false
      + id            = (known after apply)
      + name          = "Sanyu"
      + path          = "/"
      + tags_all      = (known after apply)
      + unique_id     = (known after apply)
    }

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

─────────────────────────────────────────────────────────────────────────────

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.

admin@DESKTOP-CLT1FUS MINGW64 /e/Matfly Workspace/Terraform Demo/IAM-Service
$ terraform apply

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_iam_user.lb will be created
  + resource "aws_iam_user" "lb" {
      + arn           = (known after apply)
      + force_destroy = false
      + id            = (known after apply)
      + name          = "Sanyu"
      + path          = "/"
      + tags_all      = (known after apply)
      + unique_id     = (known after apply)
    }

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

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

  Enter a value: yes

aws_iam_user.lb: Creating...
aws_iam_user.lb: Creation complete after 3s [id=Sanyu]

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

admin@DESKTOP-CLT1FUS MINGW64 /e/Matfly Workspace/Terraform Demo/IAM-Service
$ |
```

IAM User Creation using Terraform code

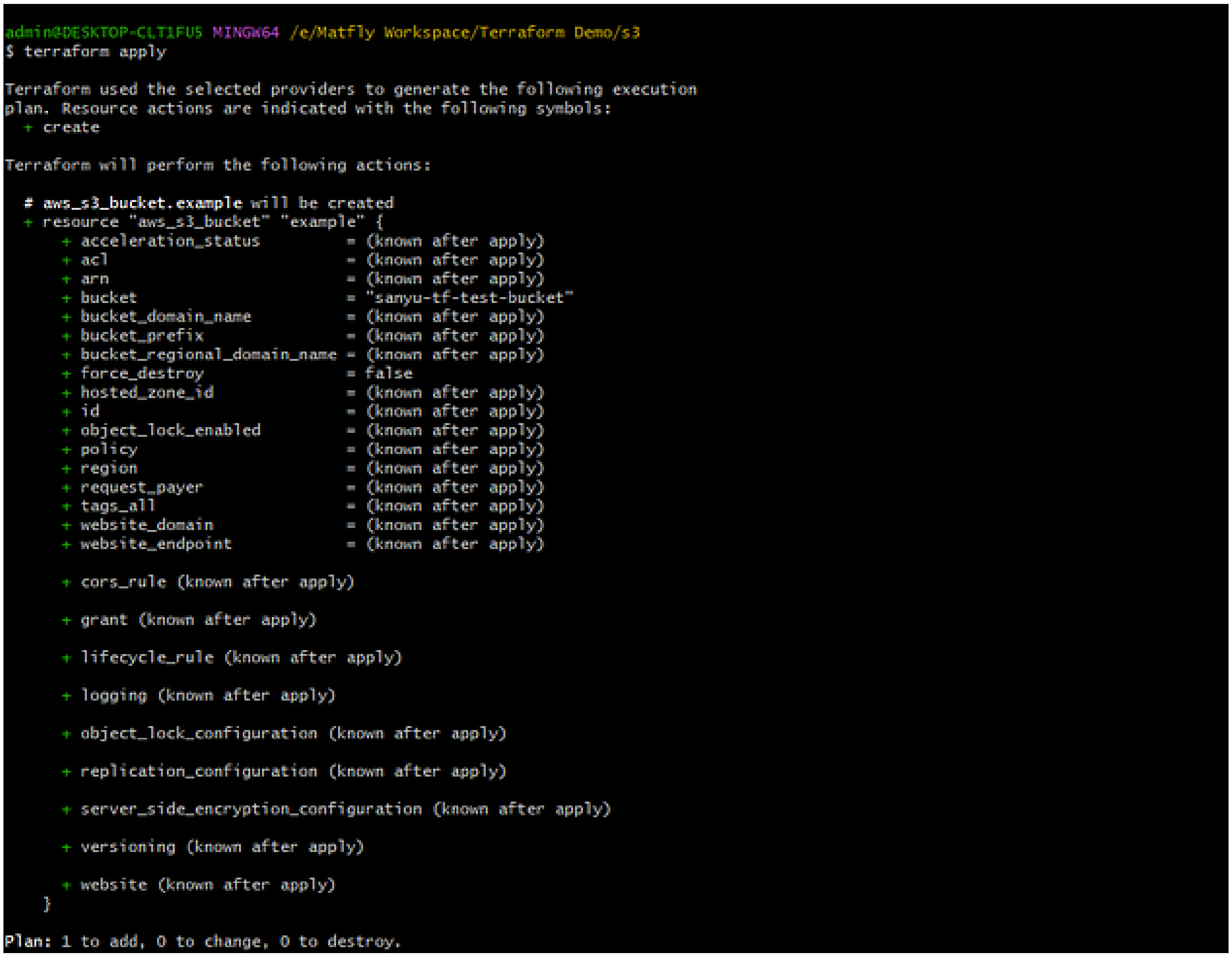

File
Edit
Selection
View
Go
Run
Search
EC2.tf
~~IAM.tf~~
gg.tf
E: > Matfly Workspace > Terraform Demo > IAM-Service > gg.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

resource "aws_iam_user" "lb" {
  name = "Sanyu"
}
```



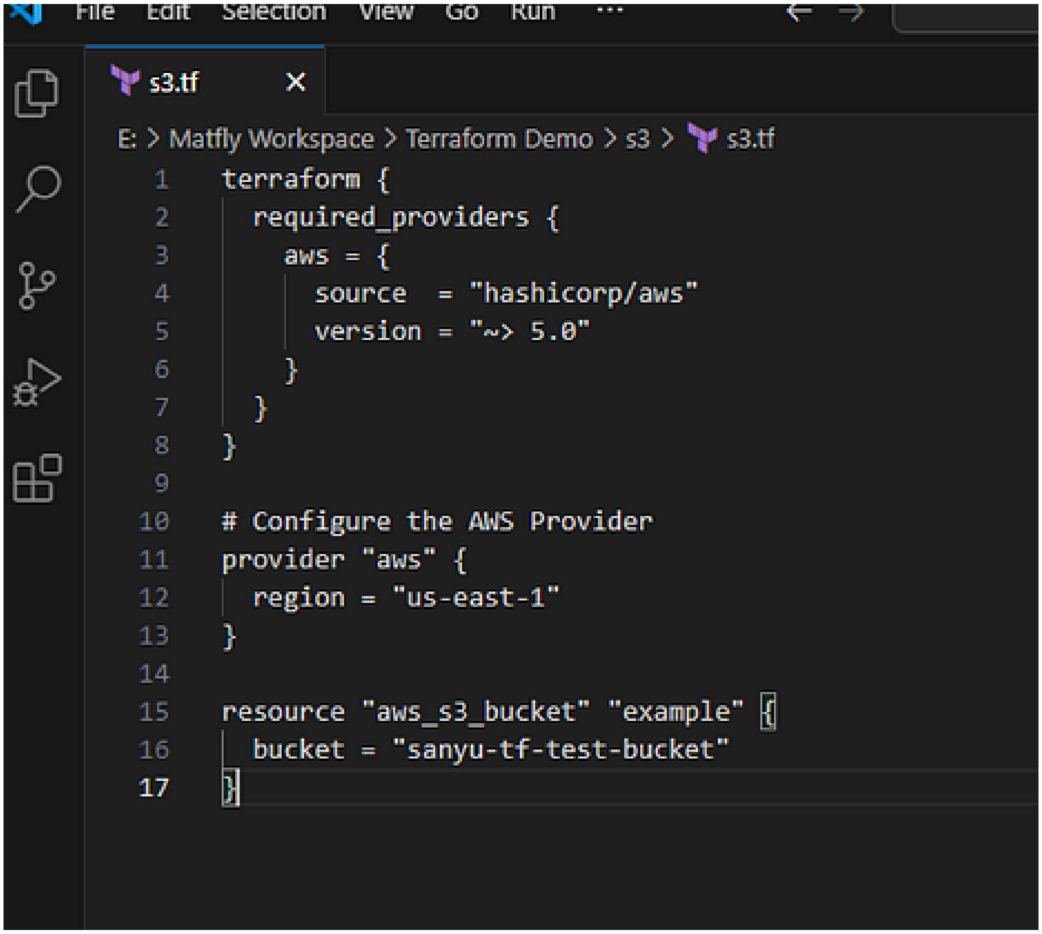
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

□	User name	Path	Groups	Last activity	MFA
□	Sanyu	/	0	-	-
□	sonali-admin	/	0	✓ 2 days ago	-
□	Terraform	/	1	✓ 33 minutes ago	-

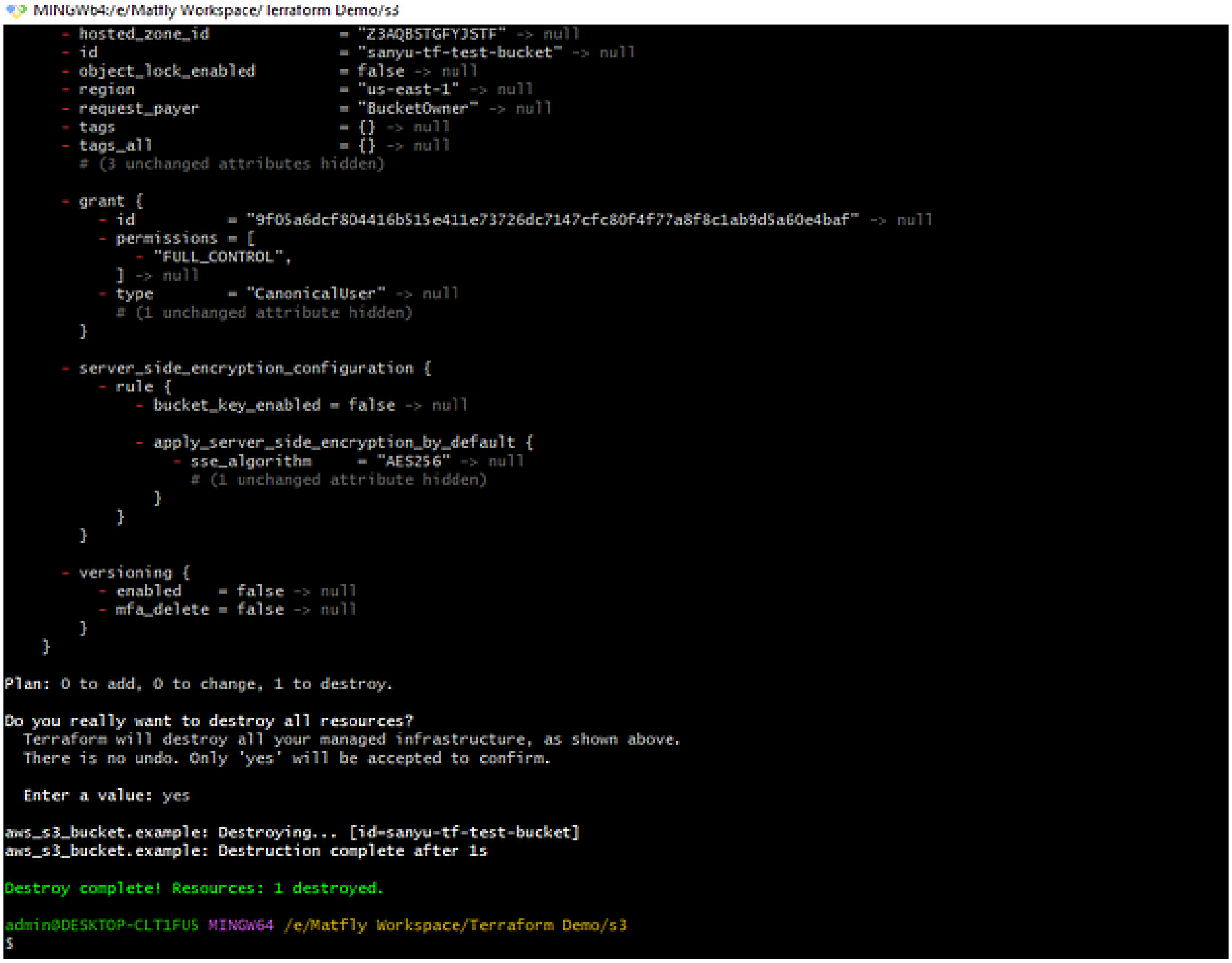


```
admin@DESKTOP-CLT1FU5 MINGW64 /e/Matfly Workspace/Terraform Demo/s3
$ terraform apply

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_s3_bucket.example will be created
  + resource "aws_s3_bucket" "example" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "sanyu-tf-test-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)

      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

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

S3 Bucket Created-

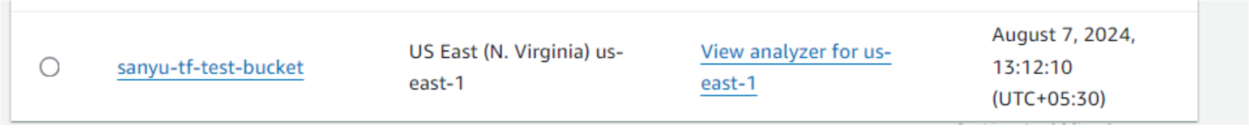


```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

resource "aws_s3_bucket" "example" {
  bucket = "sanyu-tf-test-bucket"
}
```



MINGW64:/e/Matfly Workspace/Terraform Demo/s3

```
      - hosted_zone_id              = "Z3AQBSTGFYJSTF" -> null
      - id                          = "sanyu-tf-test-bucket" -> null
      - object_lock_enabled         = false -> null
      - region                      = "us-east-1" -> null
      - request_payer               = "BucketOwner" -> null
      - tags                        = {} -> null
      - tags_all                    = {} -> null
        # (3 unchanged attributes hidden)

      - grant {
          - id          = "9f05a6dcf804416b515e411e73726dc7147cfc80f4f77a8f8c1ab9d5a60e4baf" -> null
          - permissions = [
              - "FULL_CONTROL",
            ] -> null
          - type        = "CanonicalUser" -> null
            # (1 unchanged attribute hidden)
        }

      - server_side_encryption_configuration {
          - rule {
              - bucket_key_enabled = false -> null

              - apply_server_side_encryption_by_default {
                  - sse_algorithm     = "AES256" -> null
                    # (1 unchanged attribute hidden)
                }
            }
        }

      - versioning {
          - enabled    = false -> null
          - mfa_delete = false -> 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_s3_bucket.example: Destroying... [id=sanyu-tf-test-bucket]
aws_s3_bucket.example: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.

admin@DESKTOP-CLT1FUS MINGW64 /e/Matfly Workspace/Terraform Demo/s3
$
```



○	sanyu-tf-test-bucket	US East (N. Virginia) us-east-1	View analyzer for us-east-1	August 7, 2024, 13:12:10 (UTC+05:30)