**Terraform**

Terraform is an **open-source** (developed by **Hashi Corp**) **infrastructure as code** (IaC) software tool which can be used to **provision/creation the infrastructure** of a cloud platform. **HCL**(HashiCorp Configuration Language) used to create or provision infrastructure which can be human readable **configuration files** that can be versioned, shared and reused. Terraform can manage low-level components like **compute, storage, and networking resources**, as well as high-level components like DNS entries and SaaS features.

Configuration file extentaion**.tf**

Syntax: **<block> <parameters> <name-optional>{**

**Arguments(**Configurations**)**

**}**

**Block** means what type of thing we are creating, it can be resource, output, variable, data

**Parameter** can be resource type, resource name

Ex. block- resource, then parameter – aws\_ec2, aws\_s3\_bucket

Format in configuration file- resource aws\_s3\_bucket

**Name** this name is name which is recognized to terraform

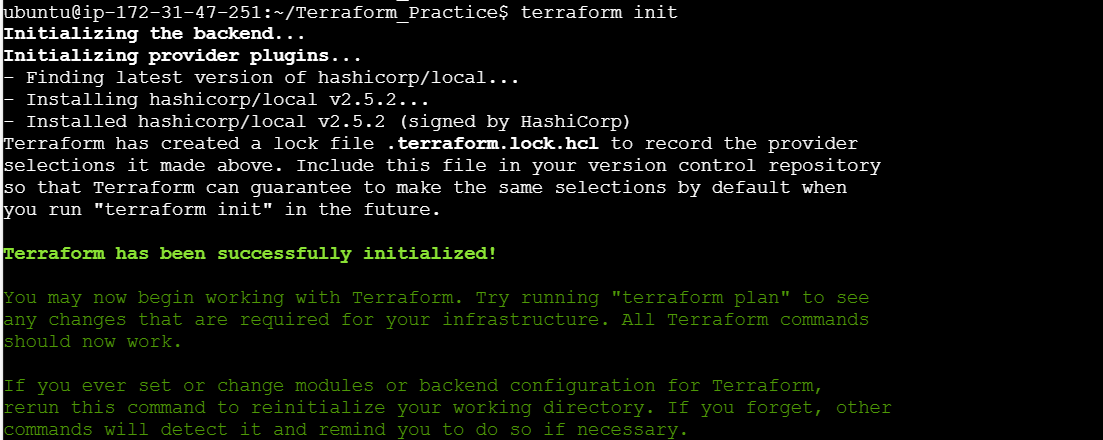
**Arguments** means like name ex. for EC2 instance we have provided instance name that name is given as arguments

Ex. resource local\_file my\_local\_file {

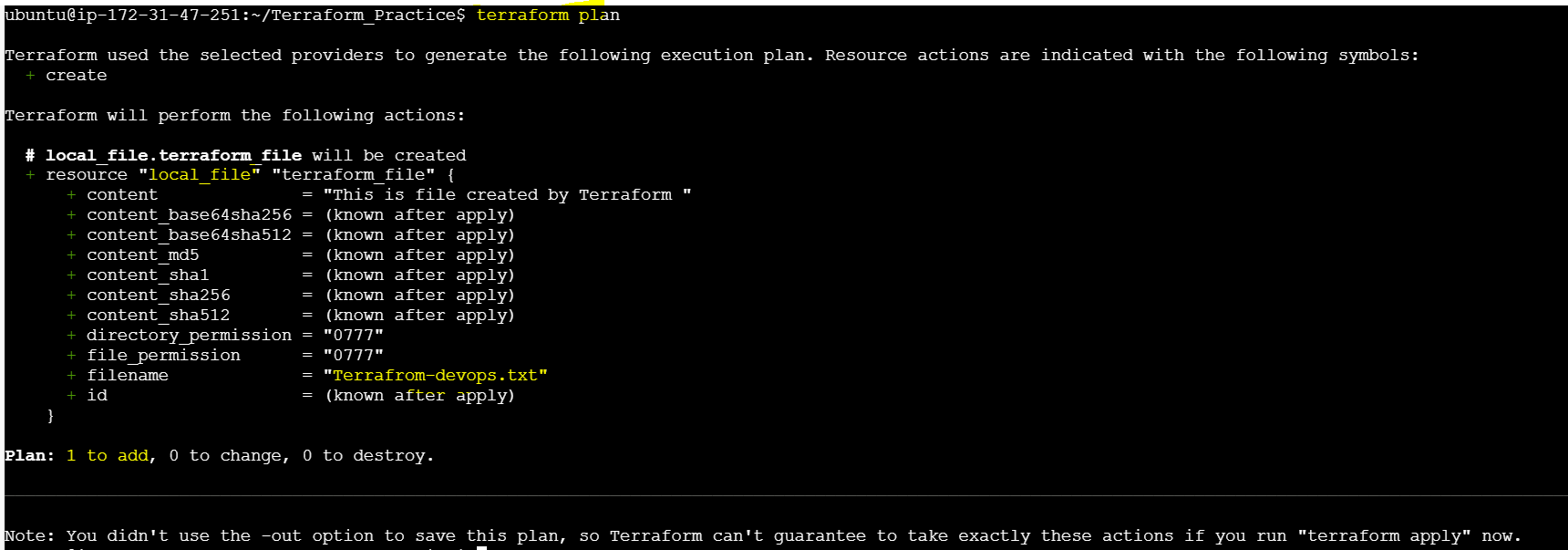
filename =”devops.txt”

content =”This is file which created using terraform”

} my\_local\_file -> resource name, devops.txt 🡪actual file name which should create once this tf file executed



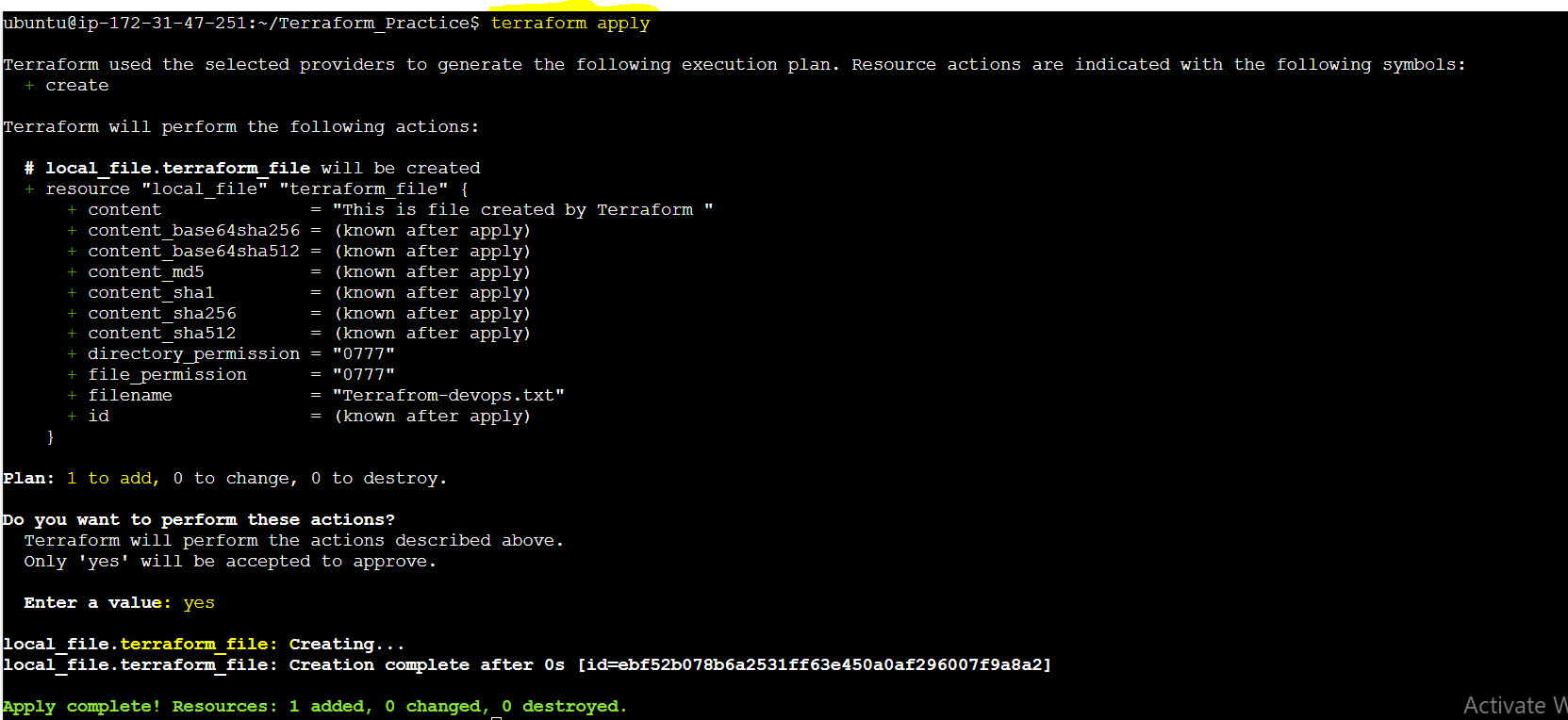
**Terraform Init-**



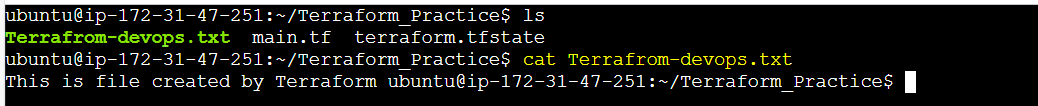
**Terraform apply-**

Before apply-





After apply



terraform state list - shows terraform file name as per above ex. it shows my\_local\_file and in file system file name will be devops.txt

ex. it shows my\_local\_file and in file system file name will be devops.txt

Rough Output

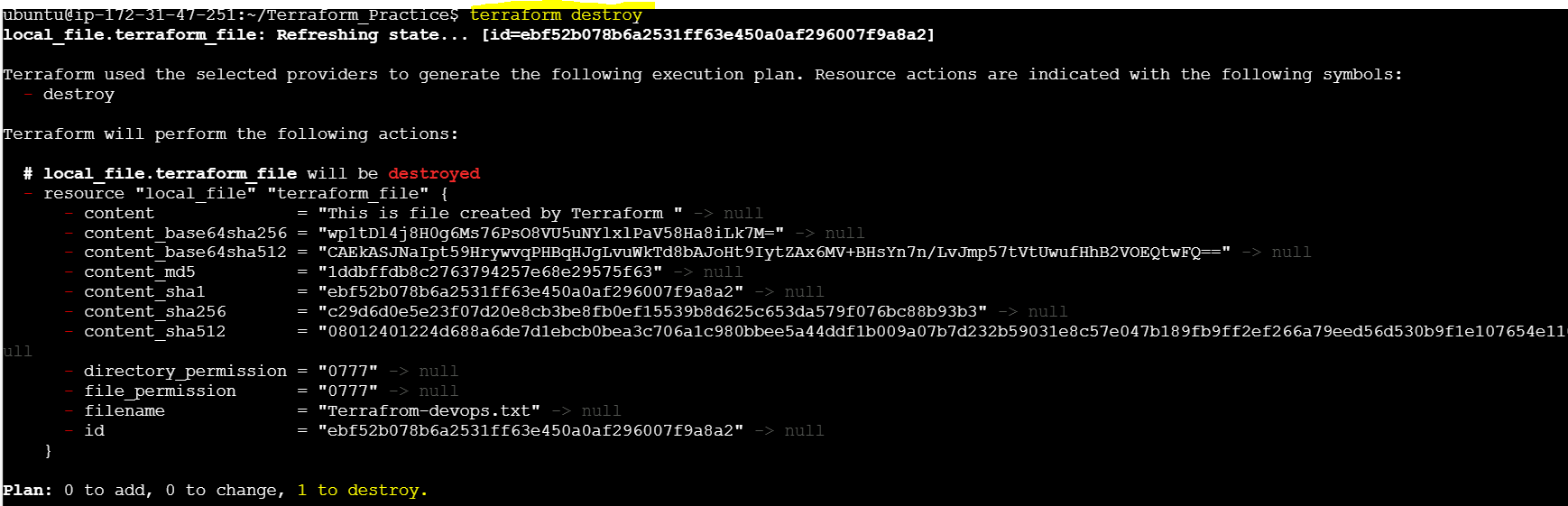
Final Output

Object

.tf

init plan apply

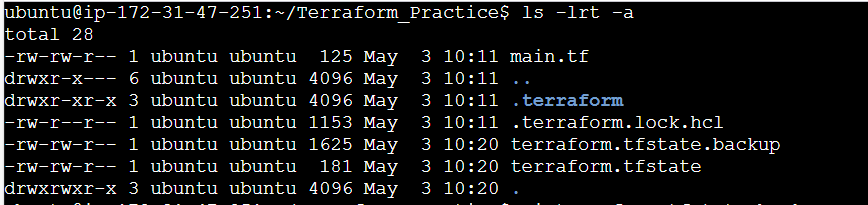
we can delete the final out put using cmd- **terraform destroy** it will ask conformation and after that we cannot able to see that devops.txt file



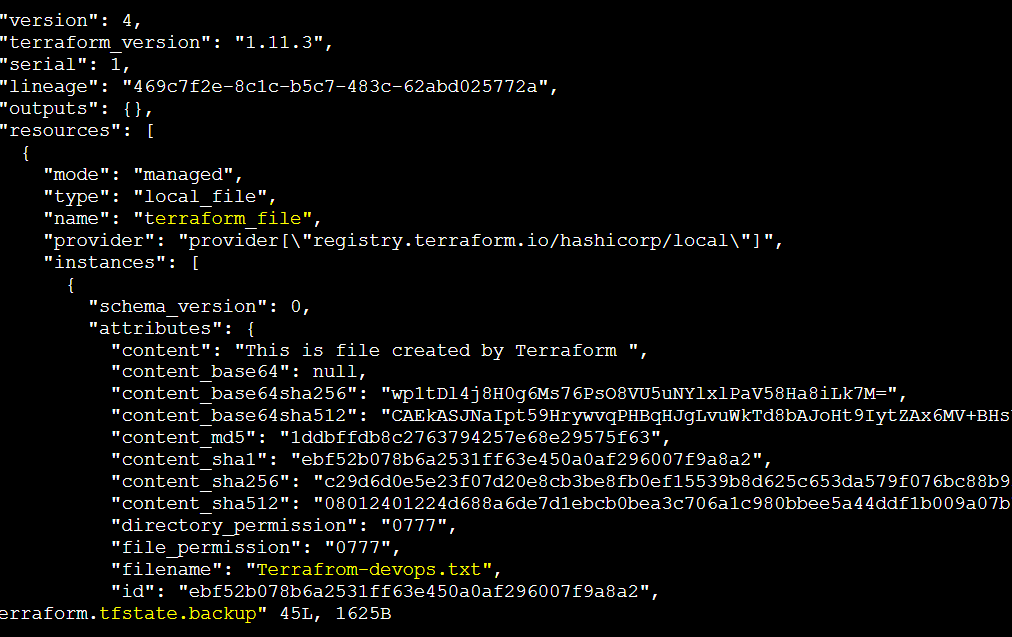
After delete-



Once apply cmd done- terraform.tfstate file created. And once destroy cmd executed .backup file craeted



Terraform.tfstate.backup file contains following -



**S3 bucket**(Simple Storage Service) is aws service which is simple storage which will store

Terraform code-

resource aws\_s3\_bucket my\_s3\_bucket{

name: “my\_data\_bucket”,

}

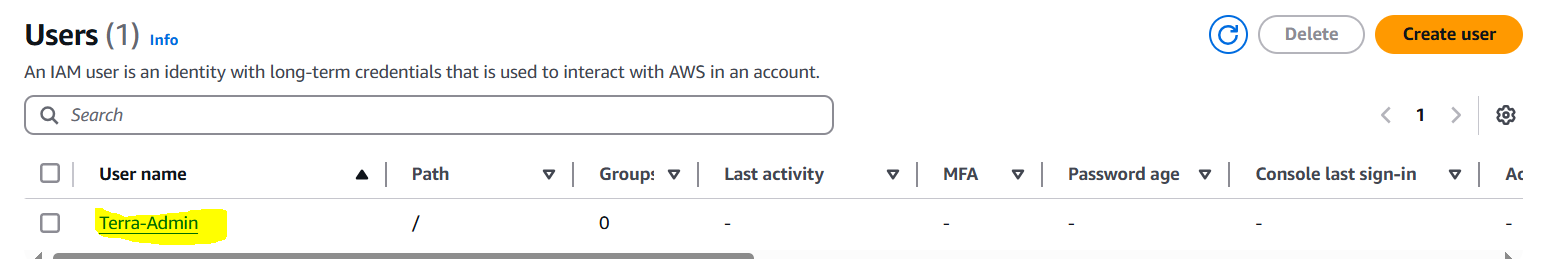
**To create AWS services we have to setup following-**

How Terraform works-

Using Terraform we have to create any aws resource for that below set up required which will provide ubuntu user to create aws resources.( means terraform installed in one server that server does not have direct access to create another EC2 or staoge so we have to set up that for that we can to do below activities)

1. **AWS IAM** (Identity and access manger) user- terraform user—from which identity we will create the resource on aws.

Here policies present we have add that policies while creating IAM user that policies will decide what all activities user(where terraform will perform or executed their task) can perform. Ex. user created- terra-admin



1. **AWS cli-** to setup the IAM user into terfrm server required cli. aws -- version cmd shows cli version

Aws cli installation required

**Aws configure –**cmd bind the aws iam user with ec2 teramform server, this cmd required access key, on aws IAM user have that option present

Now we able able to see all service and their resources present on aws account

Required access for **aws configure** cmd-



Checking the s3 is visible or not, but in my Aws account no s3 bucket available-



1. **Provider—**aws terraform plugin need to install. And to add that we have to create file terraform.tf and add provider code their.

**Ex.** Block- provider, aws- parameter, region- agrumnets

provider “aws”{

region=”eu-west-1”

}

Terraform init—once this cmd executed one hidden folder created .terraform in that hasicorp folder two folder created---aws and local

1. **Terraform .tf file**

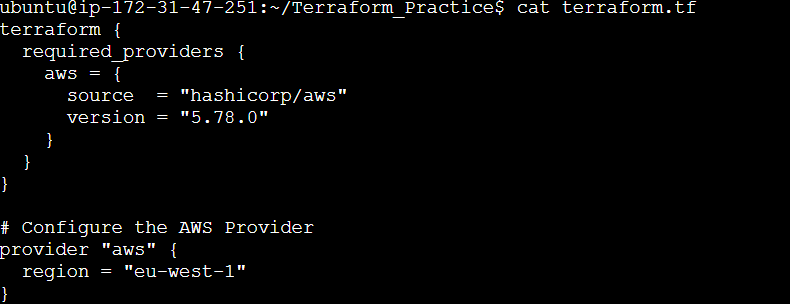
AMI(Amazon Machine Instance)

Vpc (Virtual Private Cloud)

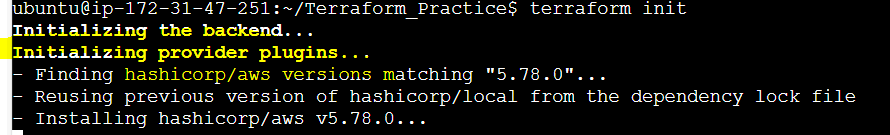
Before adding providers-



After adding providers-



After adding provider in terraform file, terraform init required

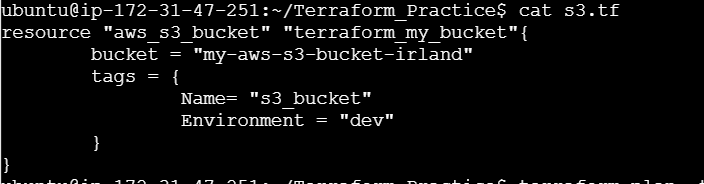


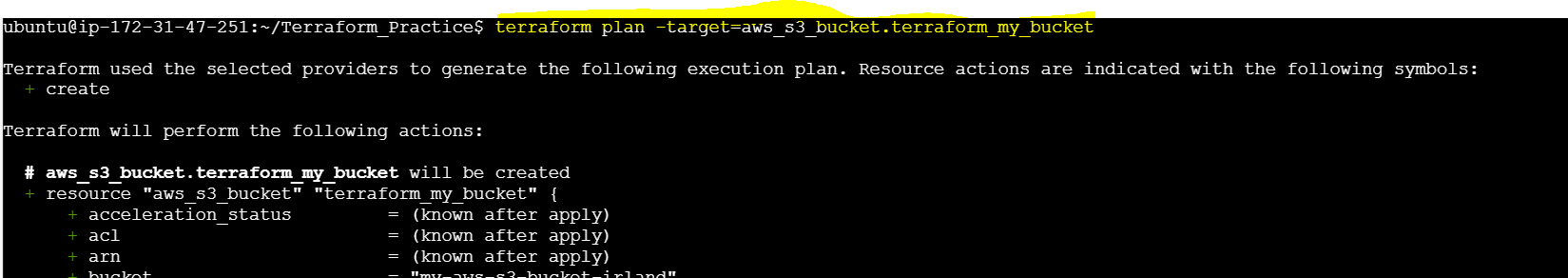
* Earlier only local folder present

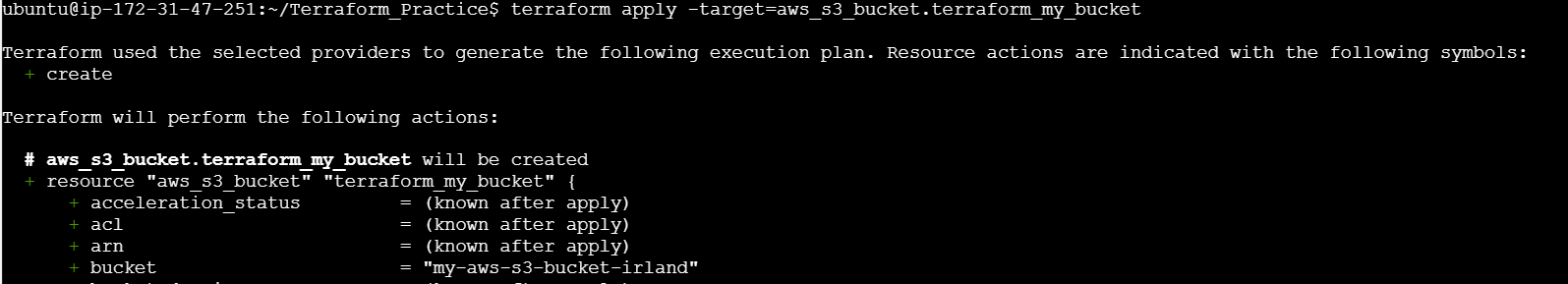


**Terraform state list**----shows all terraform resources

**S3 bucket**

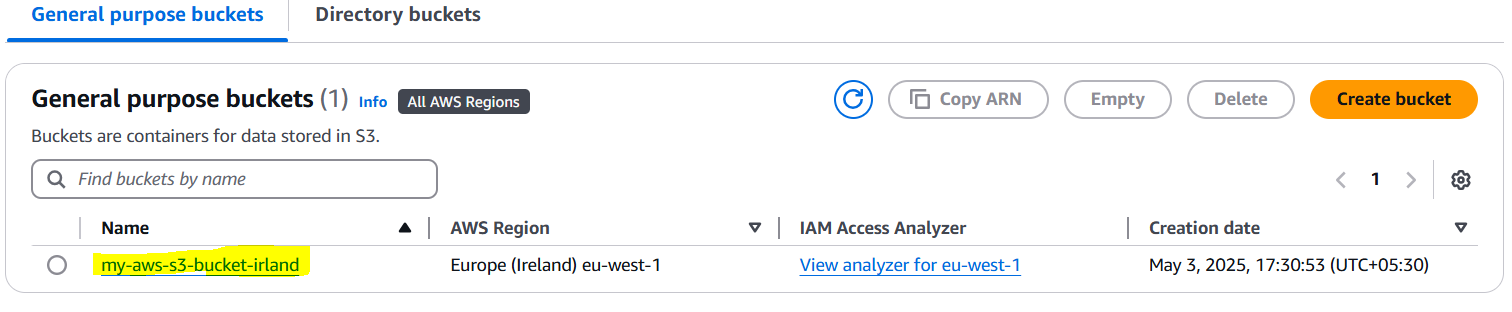








On console-



Delete particular resource from terraform--- **terraform destroy –target=terraform\_resource\_name**

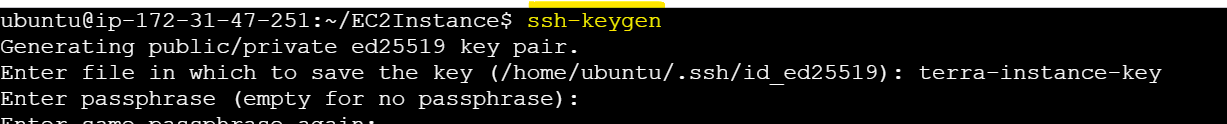
Resource\_type.refrance\_name.attribute -🡪**Interpolation**

**Ex.** aws\_default\_vpc.default.id

Given from resource “aws\_default\_vpc” “default”{}

**EC2 Instance creation-**

**Created public and private key for instance** which we want to create using cmd – ssh-keygen

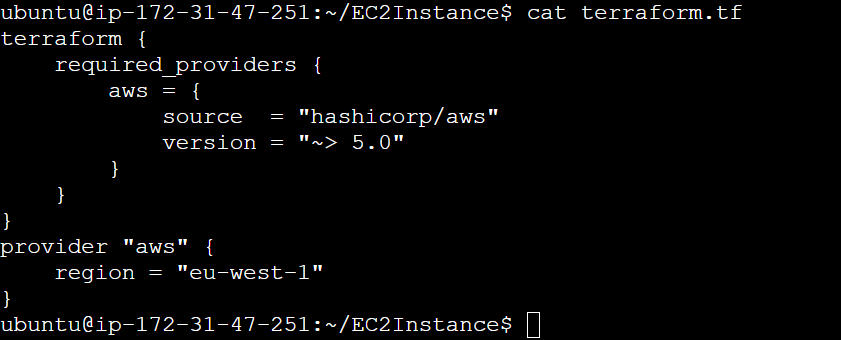


Generated keys-

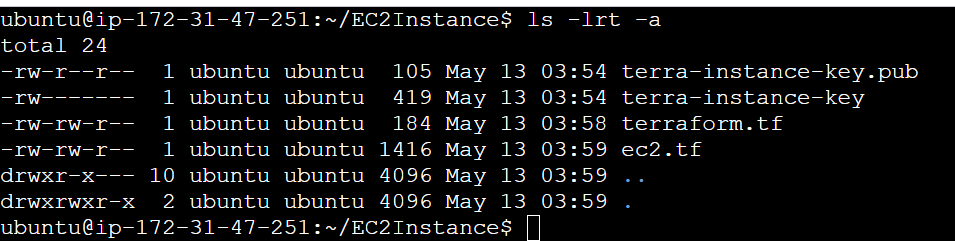
A black background with white text and yellow letters

AI-generated content may be incorrect.

**Created terafrom.tf and Added providers to access aws account-**



**Code for Ec2 instance –**



**Initializing terform into new directory-**

**A screen shot of a computer

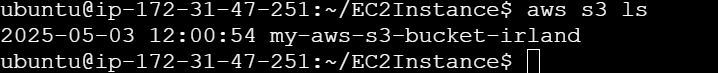
AI-generated content may be incorrect.**

**.terraform directory will be created-**

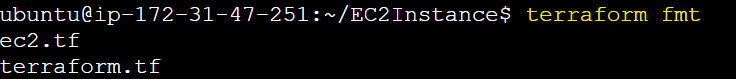
A screen shot of a computer

AI-generated content may be incorrect.

Terraform configuration already present on server so no using terraform config cmd, checked s3 buckets on aws account –



**Terraform fmt** cmd will format the .tf files-



**Terrfaomr plan-**

**A black screen with white text

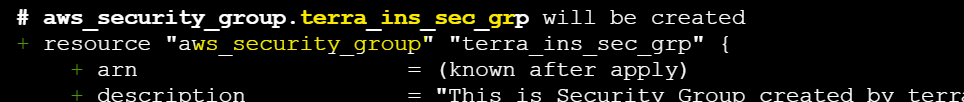
AI-generated content may be incorrect.**

**A screen shot of a black screen

AI-generated content may be incorrect.**

**A black background with white text

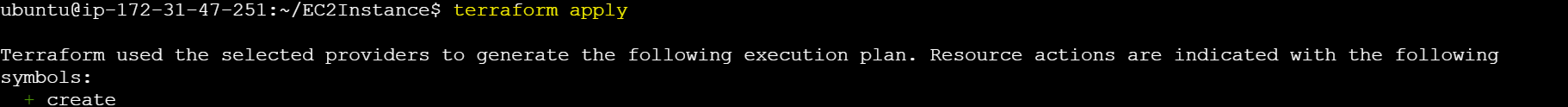
AI-generated content may be incorrect.**

****

**A black background with white text

AI-generated content may be incorrect.**

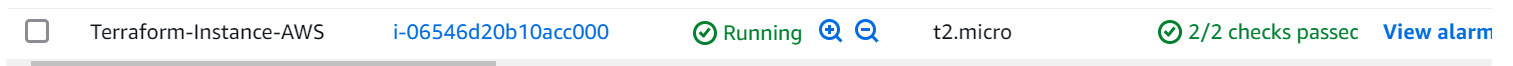
**Terraform apply-**

****

A screen shot of a computer

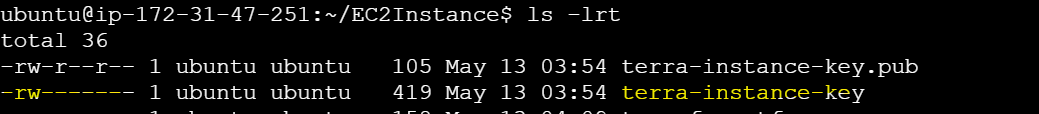
AI-generated content may be incorrect.

Instance created on region, here instance created-



Connecting newly created ec2 instance using ssh cmd-

Changed permission of private key stored in folder



A screenshot of a computer

AI-generated content may be incorrect.

Ssh cmd-

A black screen with white text

AI-generated content may be incorrect.

A black background with white numbers

AI-generated content may be incorrect.

Logging off from new ec2-



**1.Resource Block-** creating any type of resources like aws ec2, s3 bucket, azure also we can create

**2. Terraform Block**- used for terraform configuration

**3. Variable block –**

Ex- variable “aws\_ec2\_inst\_ami\_id” {

description= “This description”

default = “ami-id”

}

While accessing this variable use var keyword-

var. aws\_ec2\_inst\_ami\_id

**4.Data block-**for terraform fetching data

Ex. data “aws\_ami” “os\_image”{

Owners=[“”]

most\_recent= true

filter {

name = ”state”

values = [“available”]

}

filter{

name = “name”

values= “ubuntu/images/\*amd64\*”//condition

}

//architecture = “x86\_64”

}

While accessing the value here in ex we are using ami for instance

ami=data.aws\_ami.os\_image.id

we can condition also using filter

**5.Output Block**

Ex. output “ec2\_public\_ip” {

value=aws\_instance.my\_instance.public\_ip

} //my\_instance---is ref name

**State management**

**terraform state list**- displays all resources available/craeted

AWS

Terraform

Terraform apply Instance state

= running

**terraform state show** ‘aws\_instance.instance ref’—displays the particular resource state, this ex will show instance state as running. Also terraform.tfstate file contains all resources information and their state , other info also present.

If on aws we manually stopped EC2 instance then aws console shows state as ‘Stopped’, but at terraform still sc2 state was ‘Running’. Means both state is differs.

**terraform apply refresh-only,** it will updated the state in terraform.tfstate file

**terraform refresh** cmd is deprecated, now not using in industry also.

**Terminated the newly created ec2 instance from dashboard-**

**A screenshot of a computer

AI-generated content may be incorrect.**

Instance state displayed running in terraform.tfstate file-

****

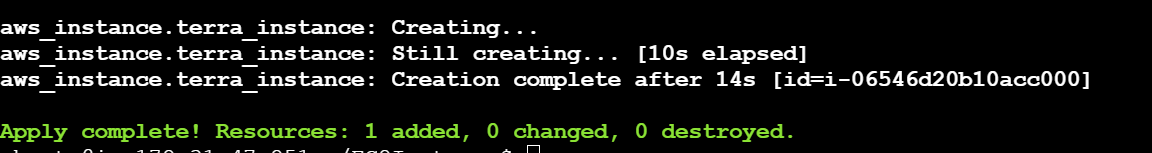
After refresh-

A computer screen with white text

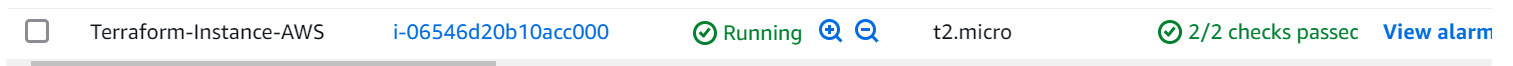
AI-generated content may be incorrect.

Now terraform.tfstate file updated and instance entry not present, only vpc, security group ,key pair info displayed

After doing apply again , only the new instance created-



dashboard



State in state file-

A computer screen with white text

AI-generated content may be incorrect.

**State Locking-** we have single source of truth which is saved in S3 bucket. This bucket any one can access for this we must use DynamoDB which will create a lockID. Means if one user using the terraform, that we use access terraform state file stored in s3 bucket, no other user can use it until that LockID is removed. Once that LockID deleted or state file lock removed other user will access that state file and main infrastructure code. Similar kind of **mutual exclusive.** This S3 bucket and DynamoDB is called **Remote Backend.**

**terraform apply -auto-approve** cmd will not show the message that do want to approve it direct approve and apply changes

**locals** variables-

we can store some value in local like variables. This defined in variables.tf file

EX. locals {

instances {

//Option 1 – for defining values- key value pair

“instance\_1”= “my\_instance\_1”,

“instance\_2” = “my\_instance\_2”

//Option 2- name- iteraable format array is iterabale

[“instance\_1”,”instance\_2”]

}

}

This will iterate using for\_each argument –

For\_each= local.instances

To access value for iteration-

Suppose name=each.value

**GitHub** is source code management tool, used for version control.