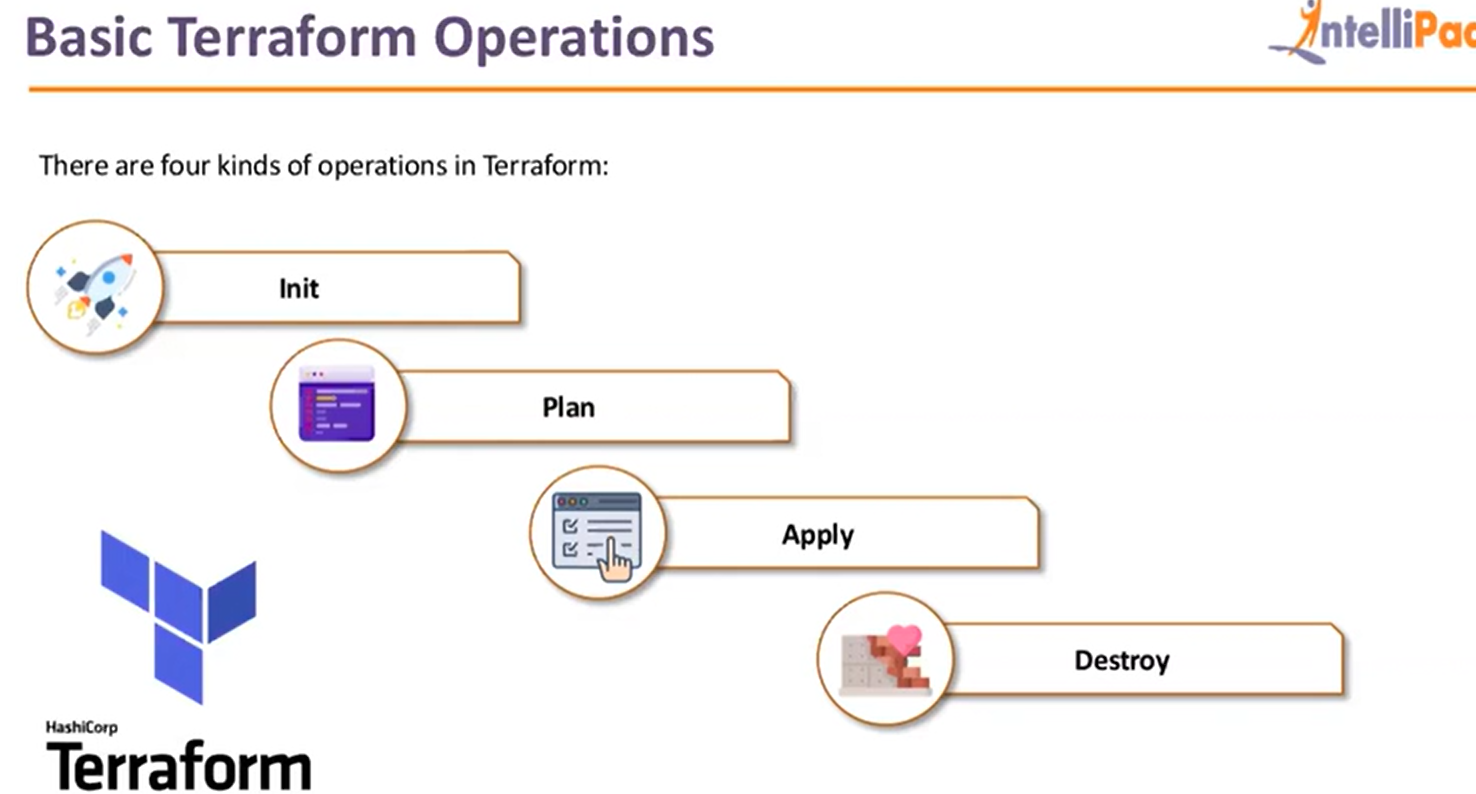
# Terraform

Infrastructure as a code (Iaac)

<https://www.hashicorp.com/>

HCL or json language

<https://www.terraform.io/cli/install/apt>



<https://registry.terraform.io/browse/providers>

<https://www.terraform.io/language/providers/configuration>

ubuntu@ip-172-31-31-36:~$ mkdir terraform

ubuntu@ip-172-31-31-36:~$ cd te

terraform/ testrepo/

ubuntu@ip-172-31-31-36:~$ cd te

terraform/ testrepo/

ubuntu@ip-172-31-31-36:~$ cd te

terraform/ testrepo/

ubuntu@ip-172-31-31-36:~$ cd terraform/

ubuntu@ip-172-31-31-36:~/terraform$ ls

ubuntu@ip-172-31-31-36:~/terraform$ touch main.tf

ubuntu@ip-172-31-31-36:~/terraform$ vi main.tf

provider "aws" {

region = "us-east-1"

}

ubuntu@ip-172-31-31-36:~/terraform$ terraform init

Initializing the backend...

Initializing provider plugins...

- Finding latest version of hashicorp/aws...

- Installing hashicorp/aws v3.70.0...

- Installed hashicorp/aws v3.70.0 (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.

ubuntu@ip-172-31-31-36:~/terraform$ ls

main.tf

ubuntu@ip-172-31-31-36:~/terraform$ ls -al

total 20

drwxrwxr-x 3 ubuntu ubuntu 4096 Dec 22 11:21 .

drwxr-xr-x 20 ubuntu ubuntu 4096 Dec 22 11:21 ..

drwxr-xr-x 3 ubuntu ubuntu 4096 Dec 22 11:21 .terraform

-rw-r--r-- 1 ubuntu ubuntu 1077 Dec 22 11:21 .terraform.lock.hcl

-rw-rw-r-- 1 ubuntu ubuntu 42 Dec 22 11:20 main.tf

ubuntu@ip-172-31-31-36:~/terraform$ tree

.

└── main.tf

0 directories, 1 file

ubuntu@ip-172-31-31-36:~/terraform$ tree -a

.

├── .terraform

│   └── providers

│   └── registry.terraform.io

│   └── hashicorp

│   └── aws

│   └── 3.70.0

│   └── linux\_amd64

│   └── terraform-provider-aws\_v3.70.0\_x5

├── .terraform.lock.hcl

└── main.tf

7 directories, 3 files

ubuntu@ip-172-31-31-36:~/terraform$ cat .terraform.lock.hcl /////for version stability

# This file is maintained automatically by "terraform init".

# Manual edits may be lost in future updates.

ubuntu@ip-172-31-31-36:~/terraform$ terraform fmt ////for indendation

ubuntu@ip-172-31-31-36:~/terraform$ cat main.tf

provider "aws" {

region = "us-east-1"

}

ubuntu@ip-172-31-31-36:~/terraform$ vim main.tf

ubuntu@ip-172-31-31-36:~/terraform$ cat main.tf

provider "aws" {

region = "us-east-1"

}

ubuntu@ip-172-31-31-36:~/terraform$ terraform fmt

main.tf

ubuntu@ip-172-31-31-36:~/terraform$ cat main.tf

provider "aws" {

region = "us-east-1"

}

provider "registry.terraform.io/hashicorp/aws" {

version = "3.70.0"

hashes = [

"h1:E5IKHXzPGGSizZM5rHKzNCzpwQ7lWPXmmJnms82uzDk=",

"zh:0af710e528e21b930899f0ac295b0ceef8ad7b623dd8f38e92c8ec4bc7af0321",

"zh:4cabcd4519c0aae474d91ae67a8e3a4a8c39c3945c289a9cf7c1409f64409abe",

"zh:58da1a436facb4e4f95cd2870d211ed7bcb8cf721a4a61970aa8da191665f2aa",

"zh:6465339475c1cd3c16a5c8fee61304dcad2c4a27740687d29c6cdc90d2e6423d",

"zh:7a821ed053c355d70ebe33185590953fa5c364c1f3d66fe3f9b4aba3961646b1",

"zh:7c3656cc9cc1739dcb298e7930c9a76ccfce738d2070841d7e6c62fbdae74eef",

"zh:9d9da9e3c60a0c977e156da8590f36a219ae91994bb3df5a1208de2ab3ceeba7",

"zh:a3138817c86bf3e4dca7fd3a92e099cd1bf1d45ee7c7cc9e9773ba04fc3b315a",

"zh:a8603044e935dfb3cb9319a46d26276162c6aea75e02c4827232f9c6029a3182",

"zh:aef9482332bf43d0b73317f5909dec9e95b983c67b10d72e75eacc7c4f37d084",

"zh:fc3f3cad84f2eebe566dd0b65904c934093007323b9b85e73d9dd4535ceeb29d",

]

}

ubuntu@ip-172-31-31-36:~/terraform$ terraform validate ///before compiling

Success! The configuration is valid.

ubuntu@ip-172-31-31-36:~/terraform$ vi main.tf

ubuntu@ip-172-31-31-36:~/terraform$ cat main.tf

provide "aws" {

region = "us-east-1"

}

ubuntu@ip-172-31-31-36:~/terraform$ terraform validate

╷

│ Error: Unsupported block type

│

│ on main.tf line 1:

│ 1: provide "aws" {

│

│ Blocks of type "provide" are not expected here. Did you mean "provider"?

╵

## Create awc vpc using terraform

ubuntu@ip-172-31-31-36:~/terraform$ vi main.tf

ubuntu@ip-172-31-31-36:~/terraform$ cat main.tf

provider "aws" {

region = "us-west-1"

}

resource "aws\_vpc" "myvpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "VPC"

}

}

ubuntu@ip-172-31-31-36:~/terraform$ terraform fmt

main.tf

ubuntu@ip-172-31-31-36:~/terraform$ cat main.tf

provider "aws" {

region = "us-west-1"

}

resource "aws\_vpc" "myvpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "VPC"

}

}

ubuntu@ip-172-31-31-36:~/terraform$ terraformvalidate

terraformvalidate: command not found

ubuntu@ip-172-31-31-36:~/terraform$ terraform validate

Success! The configuration is valid.

ubuntu@ip-172-31-31-36:~/terraform$ terraform plan

╷

│ Error: error configuring Terraform AWS Provider: no valid credential sources for Terraform AWS Provider found.

│

│ Please see https://registry.terraform.io/providers/hashicorp/aws

│ for more information about providing credentials.

│

│ Error: NoCredentialProviders: no valid providers in chain

│ caused by: EnvAccessKeyNotFound: failed to find credentials in the environment.

│ SharedCredsLoad: failed to load profile, .

│ EC2RoleRequestError: no EC2 instance role found

│ caused by: EC2MetadataError: failed to make EC2Metadata request

│ <?xml version="1.0" encoding="iso-8859-1"?>

│ <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

│ "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

│ <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

│ <head>

│ <title>404 - Not Found</title>

│ </head>

│ <body>

│ <h1>404 - Not Found</h1>

│ </body>

│ </html>

│

│ status code: 404, request id:

│

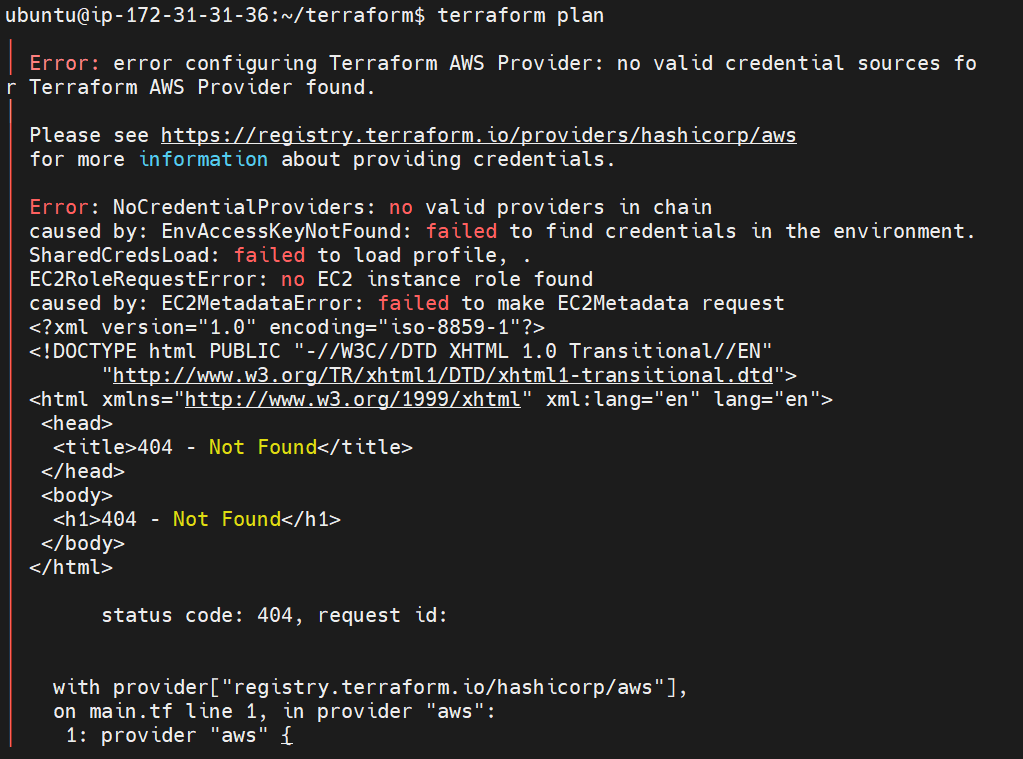
│

│ with provider["registry.terraform.io/hashicorp/aws"],

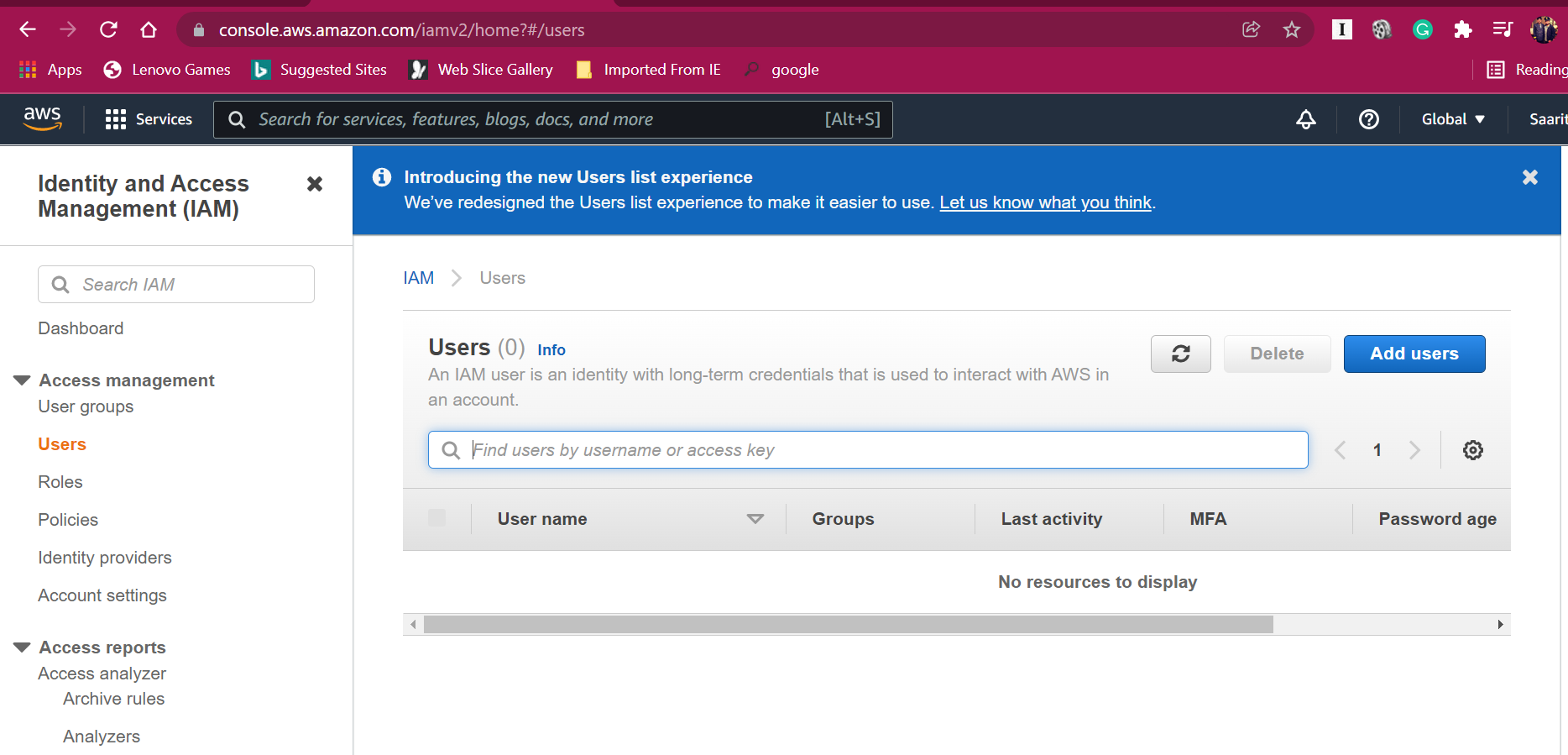
│ on main.tf line 1, in provider "aws":

│ 1: provider "aws" {

│



So, create a user on AWS console for ur account



Vi main.tf

provider "aws" {

region = "us-west-1"

access\_key="AKIAXQEM2AZM7T6N2CA3"

secret\_key="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"

}

resource "aws\_vpc" "myvpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "VPC"

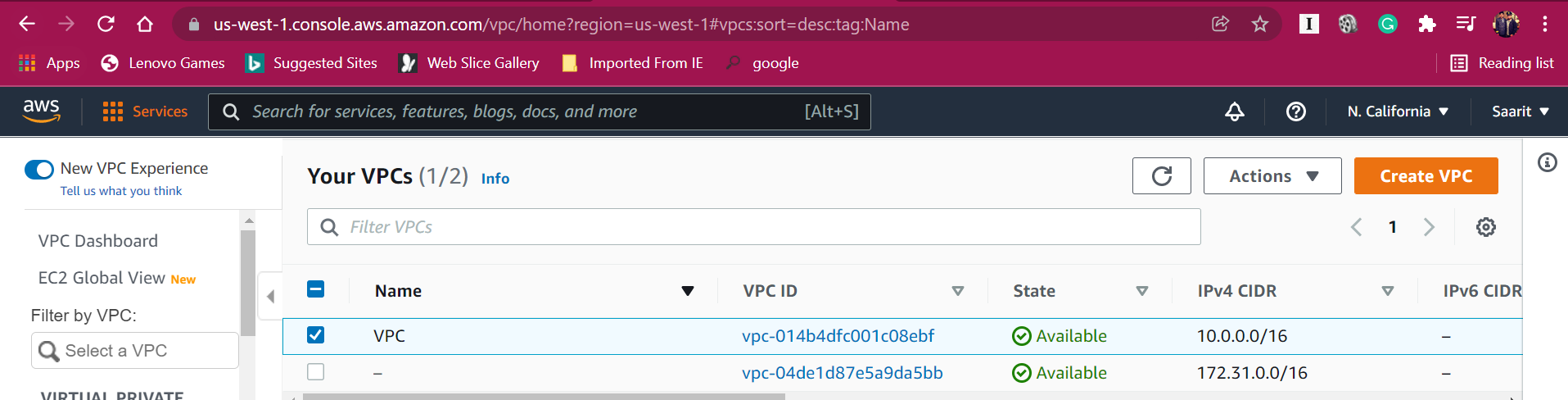
}

}

ubuntu@ip-172-31-31-36:~/terraform$ terraform plan

ubuntu@ip-172-31-31-36: ~/terraform$ terraform apply

yes



<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance>

ubuntu@ip-172-31-31-36:~/terraform$ ls

1. main.tf terraform.tfstate

//// never edit terraform.tfstate

## Creating EC2 instance

Vi main.tf

provider "aws" {

region = "us-west-1"

access\_key="AKIAXQEM2AZM7T6N2CA3"

secret\_key="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"

}

resource "aws\_vpc" "myvpc" {

cidr\_block = "10.0.0.0/16"

instance\_tenancy = "default"

tags = {

Name = "VPC"

}

}

resource "aws\_subnet" "my\_subnet" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-west-1c"

tags = {

Name = "tf-example"

}

}

resource "aws\_subnet" "my\_subnet\_new" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-west-1c"

tags = {

Name = "tf-example"

}

resource "aws\_network\_interface" "foo" {

subnet\_id = aws\_subnet.my\_subnet.id

tags = {

Name = "primary\_network\_interface"

}

}

resource "aws\_instance" "foo" {

ami = "ami-01f87c43e618bf8f0"

instance\_type = "t2.micro"

network\_interface {

network\_interface\_id = aws\_network\_interface.foo.id

device\_index = 0

}

}

terraform init

terraform fmt

terraform validate

terraform plan

terraform apply --auto-approve

<https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance>

Create key pair in AWS named saarit

rm –rf .terraform

cat /dev/null > main.tf

vi main



terraform init

terraform fmt

terraform validate

terraform plan

terraform apply --auto-approve

ubuntu@ip-172-31-31-36:~/terraform$ vi saarit ////copy rsa private key from AWS console key pairs

ubuntu@ip-172-31-31-36:~/terraform$ chmod 400 saarit

ubuntu@ip-172-31-31-36:~/terraform$ ssh -i saarit ubuntu@public\_ip

ubuntu@ip-172-31-31-36:~/terraform$ terraform destroy --auto-approve

## For multiple instances



terraform init

terraform fmt

terraform validate

terraform plan

terraform apply --auto-approve

terraform destroy --auto-approve

## Adding variable file to avoid hard coding:





rm –rf .terraform

cat /dev/null > main.tf

terraform init

terraform fmt

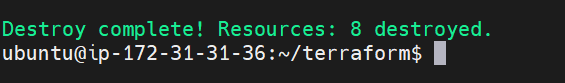
terraform validate

terraform plan

terraform apply --auto-approve

ssh -i saarit ubuntu@public\_ip

terraform destroy --auto-approve



## Output printing

ubuntu@ip-172-31-31-36:~/terraform$ vi myout.tf



ubuntu@ip-172-31-31-36:~/terraform$ terraform apply --auto-approve

aws\_vpc.myvpc: Refreshing state... [id=vpc-0258cb1bf8a94a8d9]

aws\_internet\_gateway.nyigw: Refreshing state... [id=igw-06a43406ee65e67b7]

aws\_security\_group.security\_group: Refreshing state... [id=sg-036773de0be742b0f]

aws\_subnet.mysubnet: Refreshing state... [id=subnet-03f371febed731730]

aws\_route\_table.myrt: Refreshing state... [id=rtb-0855c04eb185f5c28]

aws\_instance.myinstance[0]: Refreshing state... [id=i-06308ae0d8577fb4b]

aws\_route\_table\_association.rtassociation: Refreshing state... [id=rtbassoc-065122ad8eab7faea]

Changes to Outputs:

+ instance\_public\_ip = [

+ "54.176.13.228",

]

You can apply this plan to save these new output values to the Terraform state, without changing any real infrastructure.

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

Outputs:

instance\_public\_ip = [

"54.176.13.228",

]

vpc\_id = "vpc-0258cb1bf8a94a8d9"

## Best practice

Rather than main.tf only, many files are created separately such as instance.tf, variable.tf, myout.tf, security.tf, provider.tf, gateway.tf etc.

Also all the variables are defined in a file rather than hard coded.

## If terraform.tfstate terraform.tfstate.backup are deleted

ubuntu@ip-172-31-31-36:~/terraform$ ls

1 main.tf main.tf\_backup myout.tf saarit terraform.tfstate terraform.tfstate.backup variable.tf

ubuntu@ip-172-31-31-36:~/terraform$ rm -rf terraform.tfstate terraform.tfstate.backup

ubuntu@ip-172-31-31-36:~/terraform$ ls

1 main.tf main.tf\_backup myout.tf saarit variable.tf

ubuntu@ip-172-31-31-36:~/terraform$ terraform plan

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:

+ instance\_public\_ip = [

+ (known after apply),

]

+ vpc\_id = (known after apply)

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

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.

ubuntu@ip-172-31-31-36:~/terraform$ terraform plan -out myplan

ubuntu@ip-172-31-31-36:~/terraform$ ls

main.tf main.tf\_backup myout.tf myplan saarit variable.tf

ubuntu@ip-172-31-31-36:~/terraform$ terraform apply myplan

ubuntu@ip-172-31-31-36:~/terraform$ terraform destroy

## multiple user/ version control n lock concept

### User1

ubuntu@ip-172-31-31-36:~/terraform$ terraform apply --auto-approve

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.myinstance[0] will be created

………………………………………………………….

### At the same time

### User 2

ubuntu@ip-172-31-31-36:~/terraform$ terraform apply --auto-approve

╷

│ Error: Error acquiring the state lock

│

│ Error message: resource temporarily unavailable

│ Lock Info:

│ ID: 39608d51-82c2-3175-980a-6edc46d751b6

│ Path: terraform.tfstate

│ Operation: OperationTypeApply

│ Who: ubuntu@ip-172-31-31-36

│ Version: 1.1.2

│ Created: 2022-01-03 04:52:58.896540312 +0000 UTC

│ Info:

│

│

│ Terraform acquires a state lock to protect the state from being written

│ by multiple users at the same time. Please resolve the issue above and try

│ again. For most commands, you can disable locking with the "-lock=false"

│ flag, but this is not recommended.

This is because of ………………………. .terraform.tfstate.lock.info ///which stays there until the actions are complete and avoid conflicts.

ubuntu@ip-172-31-31-36:~/terraform$ ll

total 96

drwxrwxr-x 3 ubuntu ubuntu 4096 Jan 3 04:54 ./

drwxr-xr-x 20 ubuntu ubuntu 4096 Jan 3 04:51 ../

-rw-r--r-- 1 ubuntu ubuntu 12288 Dec 29 05:19 .main.tf.swo

-rw-r--r-- 1 ubuntu ubuntu 12288 Dec 29 05:17 .main.tf.swp

drwxr-xr-x 3 ubuntu ubuntu 4096 Dec 29 04:52 .terraform/

-rw-r--r-- 1 ubuntu ubuntu 1077 Dec 22 11:21 .terraform.lock.hcl

-rw------- 1 ubuntu ubuntu 206 Jan 3 04:54 .terraform.tfstate.lock.info

-rw-rw-r-- 1 ubuntu ubuntu 42 Dec 22 11:30 1

-rw-rw-r-- 1 ubuntu ubuntu 1778 Jan 3 04:21 main.tf

-rw-rw-r-- 1 ubuntu ubuntu 1010 Dec 29 04:50 main.tf\_backup

-rw-rw-r-- 1 ubuntu ubuntu 133 Jan 3 04:29 myout.tf

-rw-rw-r-- 1 ubuntu ubuntu 4705 Jan 3 04:41 myplan

-r-------- 1 ubuntu ubuntu 1675 Dec 29 05:38 saarit

-rw-rw-r-- 1 ubuntu ubuntu 12396 Jan 3 04:53 terraform.tfstate

-rw-rw-r-- 1 ubuntu ubuntu 156 Jan 3 04:53 terraform.tfstate.backup

-rw-rw-r-- 1 ubuntu ubuntu 176 Dec 29 06:10 variable.tf

## Using S3 bucket and lock concept

* Create a S3 bucket
* Create backend.tf as

ubuntu@ip-172-31-31-36:~/terraform$ cat backend.tf

terraform {

backend "s3" {

bucket = "saarit123"

key = "dev/terraform.tfstate"

region = "us-west-1"

access\_key = "AKIAXQEM2AZM7T6N2CA3"

secret\_key = "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"

}

}

* terraform init
* terraform fmt
* terraform validate
* terraform plan -out s3plan
* terraform apply s3plan
* No longer lock is present in local

ubuntu@ip-172-31-31-36:~/terraform$ ll

total 112

drwxrwxr-x 3 ubuntu ubuntu 4096 Jan 3 05:14 ./

drwxr-xr-x 20 ubuntu ubuntu 4096 Jan 3 05:13 ../

-rw-r--r-- 1 ubuntu ubuntu 12288 Dec 29 05:19 .main.tf.swo

-rw-r--r-- 1 ubuntu ubuntu 12288 Dec 29 05:17 .main.tf.swp

drwxr-xr-x 3 ubuntu ubuntu 4096 Jan 3 05:13 .terraform/

-rw-r--r-- 1 ubuntu ubuntu 1077 Dec 22 11:21 .terraform.lock.hcl

-rw-rw-r-- 1 ubuntu ubuntu 42 Dec 22 11:30 1

-rw-rw-r-- 1 ubuntu ubuntu 237 Jan 3 05:13 backend.tf

-rw-rw-r-- 1 ubuntu ubuntu 1778 Jan 3 04:21 main.tf

-rw-rw-r-- 1 ubuntu ubuntu 1010 Dec 29 04:50 main.tf\_backup

-rw-rw-r-- 1 ubuntu ubuntu 129 Jan 3 05:13 myout.tf

-rw-rw-r-- 1 ubuntu ubuntu 4705 Jan 3 04:41 myplan

-rw-rw-r-- 1 ubuntu ubuntu 4733 Jan 3 05:03 newplan

-rw-rw-r-- 1 ubuntu ubuntu 5367 Jan 3 05:14 s3plan

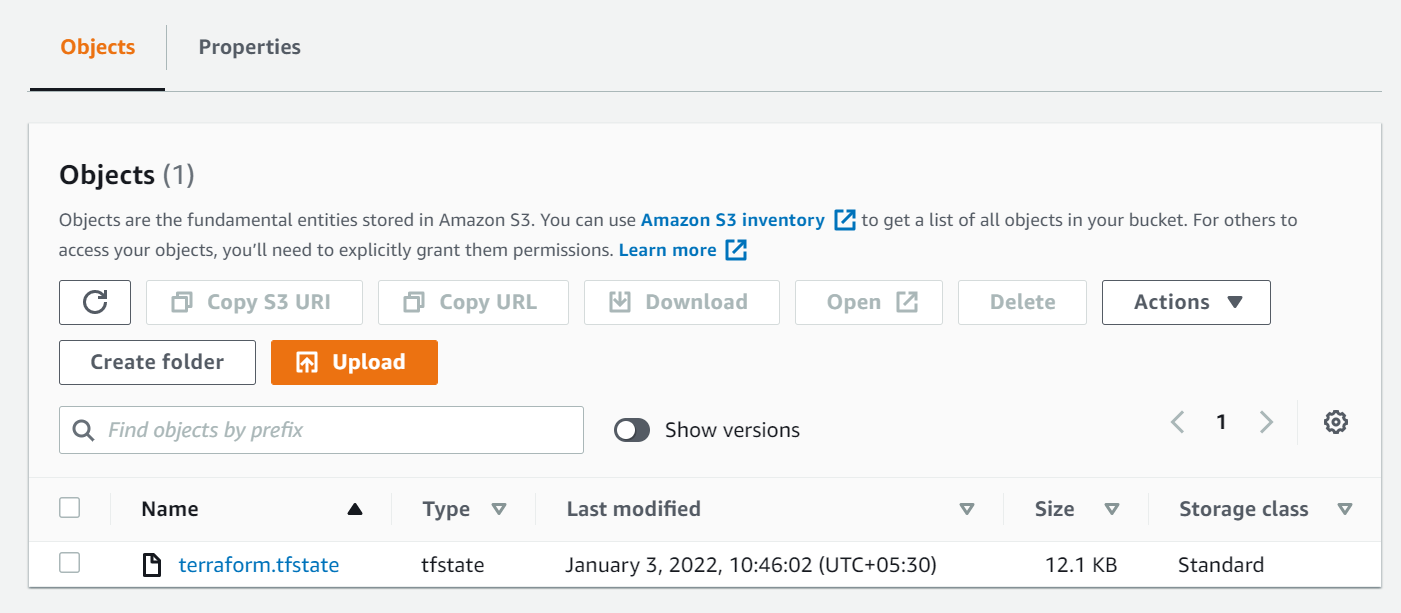
-r-------- 1 ubuntu ubuntu 1675 Dec 29 05:38 saarit

-rw-rw-r-- 1 ubuntu ubuntu 156 Jan 3 05:05 terraform.tfstate

-rw-rw-r-- 1 ubuntu ubuntu 12397 Jan 3 05:05 terraform.tfstate.backup

-rw-rw-r-- 1 ubuntu ubuntu 176 Dec 29 06:10 variable.tf

Rather the lock is at S3



## For\_each logic (set type)

*resource "aws\_instance" "myinstance" {*

*for\_each = toset(var.instance\_name)*

*ami = var.ami\_id*

*instance\_type = var.instance\_type*

*associate\_public\_ip\_address = true*

*key\_name = "saarit"*

*vpc\_security\_group\_ids = [aws\_security\_group.security\_group.id]*

*subnet\_id = aws\_subnet.mysubnet.id*

*tags = {*

*Name = each.value*

And in variable.tf file

….add…..

*variable "instance\_name" {*

*default = ["vm1","vm2"]*

*}*