Automating AWS Infrastructure Provisioning with Terraform: A Step-by-Step Guide

In this blog post, we'll explore how to use Terraform, a powerful infrastructure as code (IaC) tool, to automate the provisioning of an Amazon Web Services (AWS) environment. Specifically, we'll walk you through the process of creating an EC2 instance, installing and enabling the Apache HTTP Server (httpd), and attaching an Elastic Block Store (EBS) volume to the instance. By the end of this guide, you'll have a fully functional web server running in the cloud, all managed through Terraform's declarative configuration language.

Prerequisites:

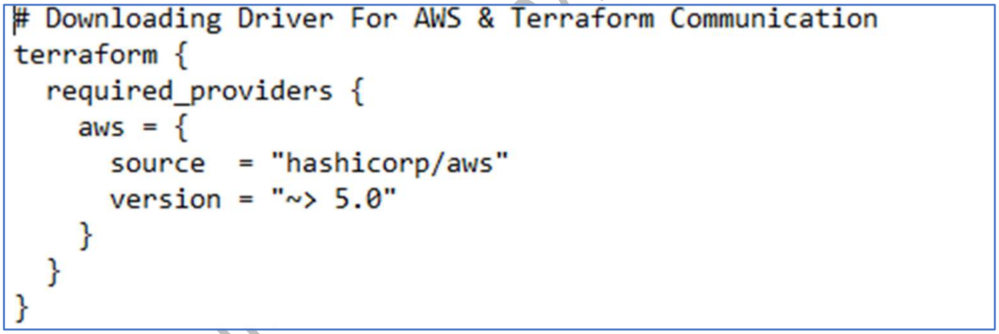
Before we begin, make sure you have the following prerequisites in place:

- **An AWS account with appropriate access credentials.**
- **Terraform installed on your local machine.**

Start by creating a new directory for your Terraform project. Inside this directory, create a file named main.tf. This file will contain the configuration for your AWS resources.

After this run the command in the Command line where you have stored this code. So, it will download the required Driver/plugins.

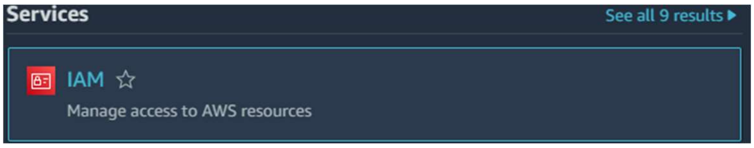
terraform init



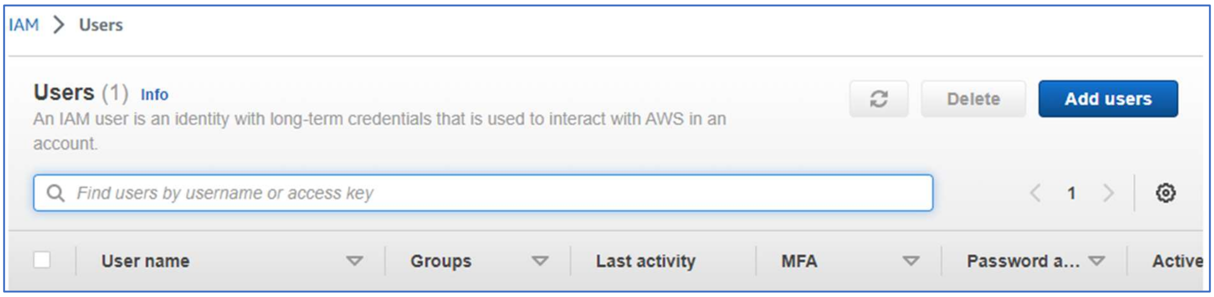
```
# Downloading Driver For AWS & Terraform Communication
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}
```

For Authentication in AWS, either you can create new User or you can use the root User. But it is not recommended to use Root User. So here I am creating the New User.

Go To AWS -> Search For IAM ->



Click on Add User on right Top Corner



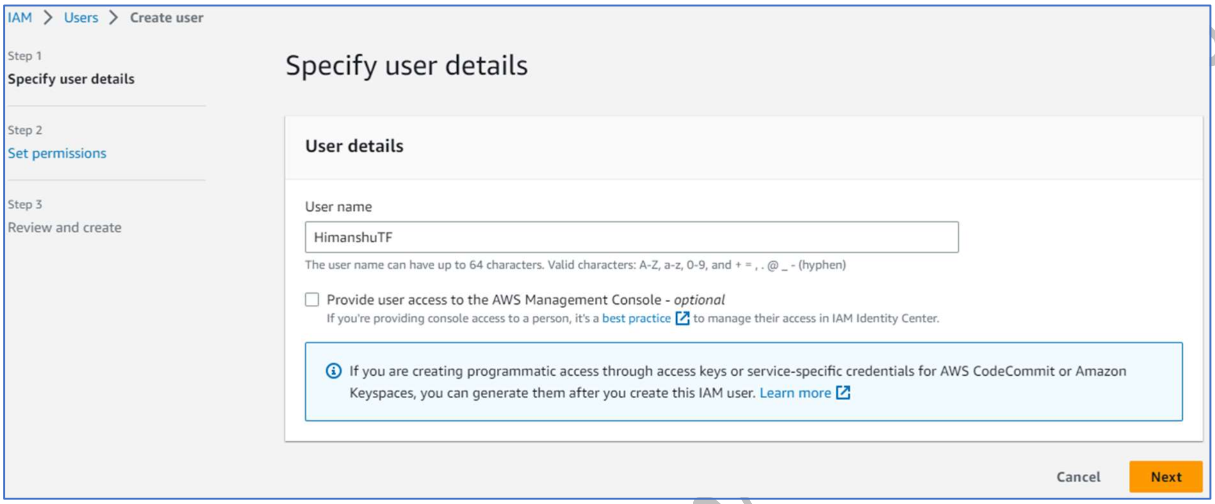
IAM > Users

Users (1) Info

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

□	User name	Groups	Last activity	MFA	Password a...	Active

Provide the Username. Here I have given HimanshuTF.



IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

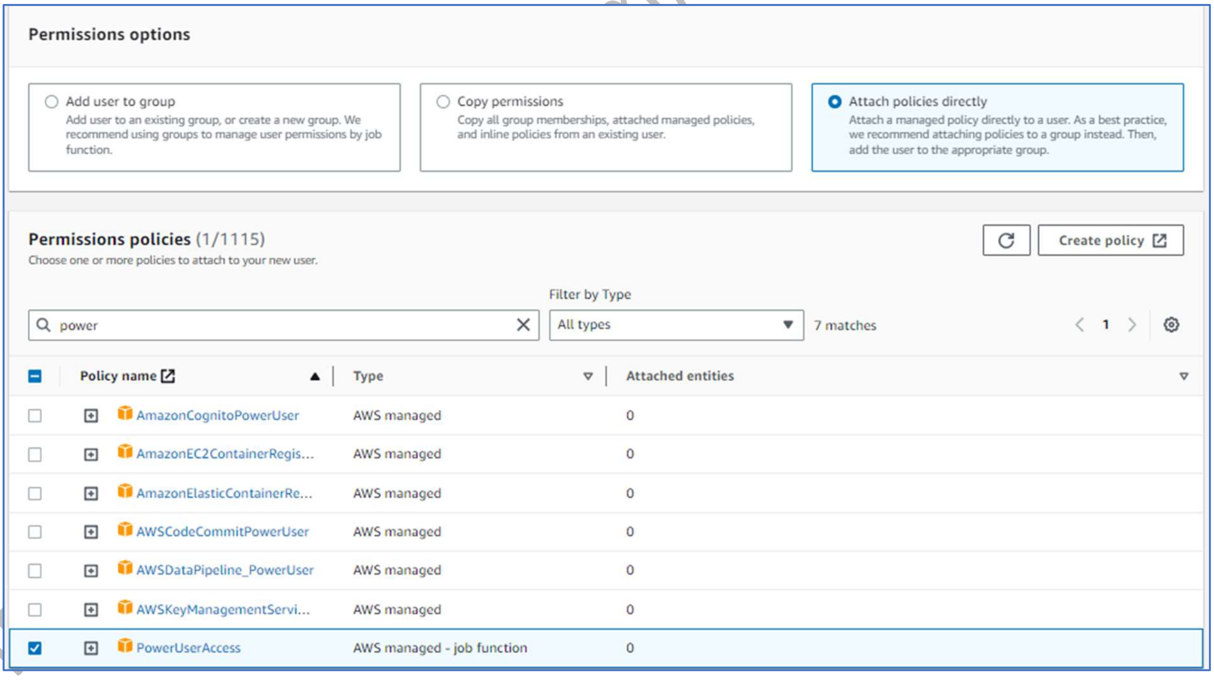
User name: HimanshuTF

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

□ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

Here I have Given the Permission for PowerUserAcess which is equivalent to Root Account.



Permissions options

○ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

● Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

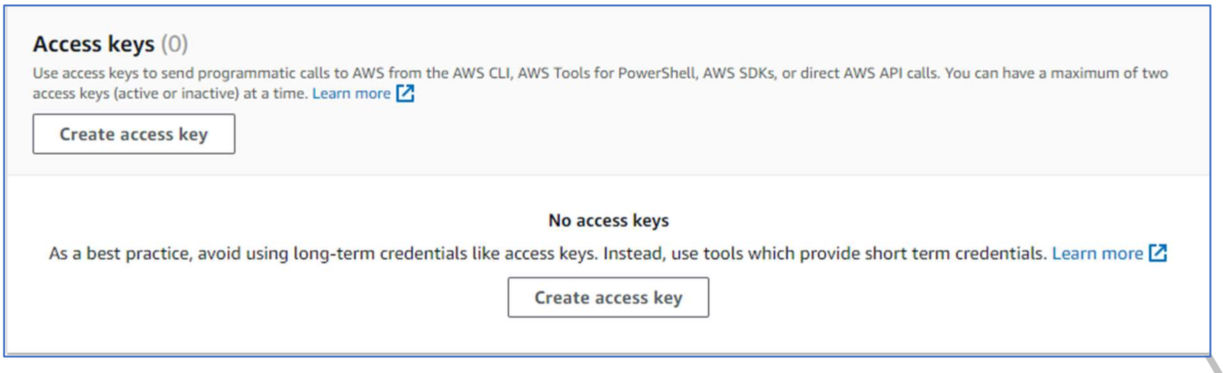
Permissions policies (1/1115)
Choose one or more policies to attach to your new user.

Search: power
Filter by Type: All types
7 matches

	Policy name	Type	Attached entities
□	AmazonCognitoPowerUser	AWS managed	0
□	AmazonEC2ContainerRegis...	AWS managed	0
□	AmazonElasticContainerRe...	AWS managed	0
□	AWSCodeCommitPowerUser	AWS managed	0
□	AWSDataPipeline_PowerUser	AWS managed	0
□	AWSKeyManagementServi...	AWS managed	0
☑	PowerUserAccess	AWS managed - job function	0

After that, User is created. Now, select the user and go to the Security Credentials Section To create Access and Secret Key which we require in Terraform.

Click on Create Access Key



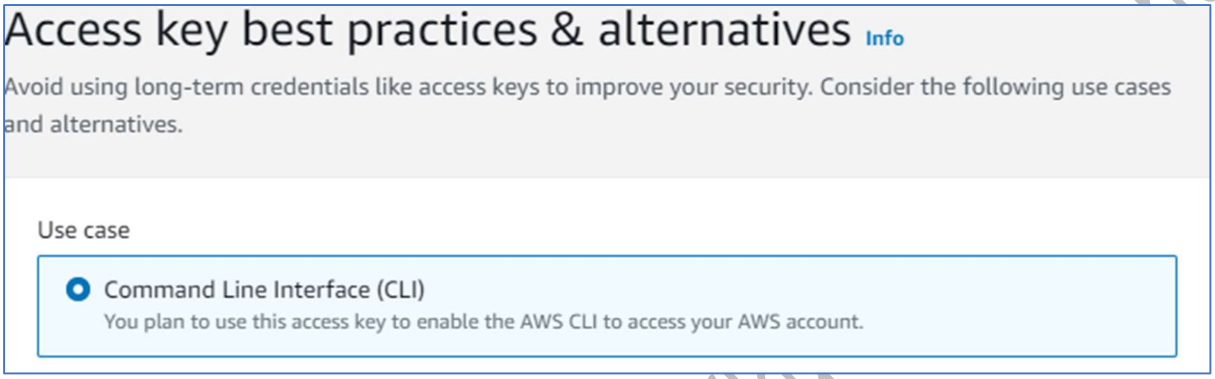
Access keys (0)

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

No access keys

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

Then Based on my Use Case I have chosen CLI.



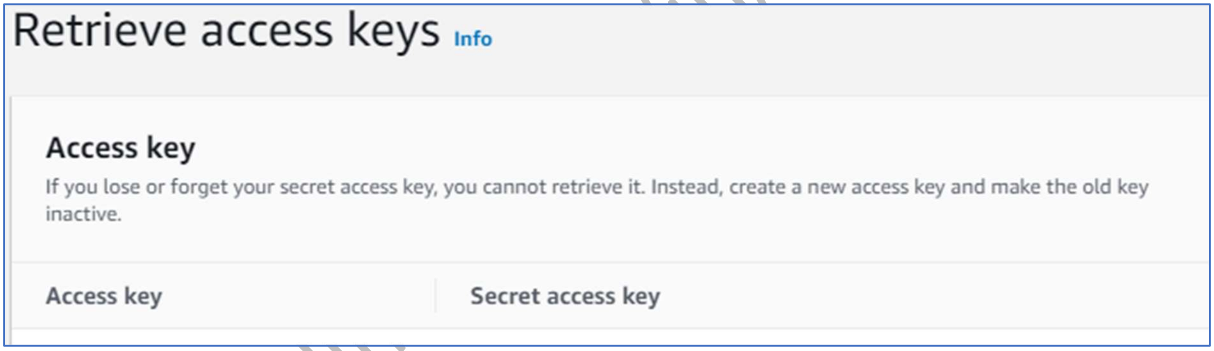
Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

Then Download You Secret and Access Key.



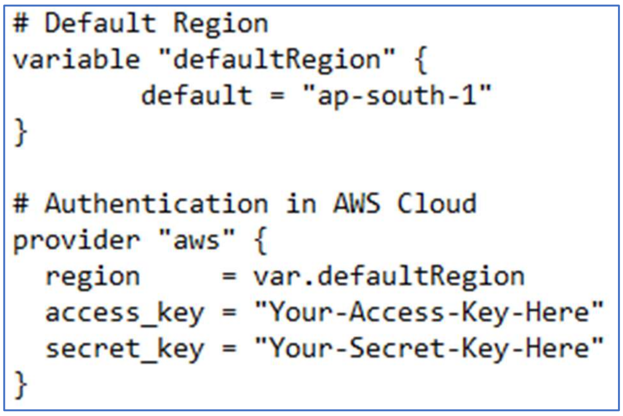
Retrieve access keys Info

Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key

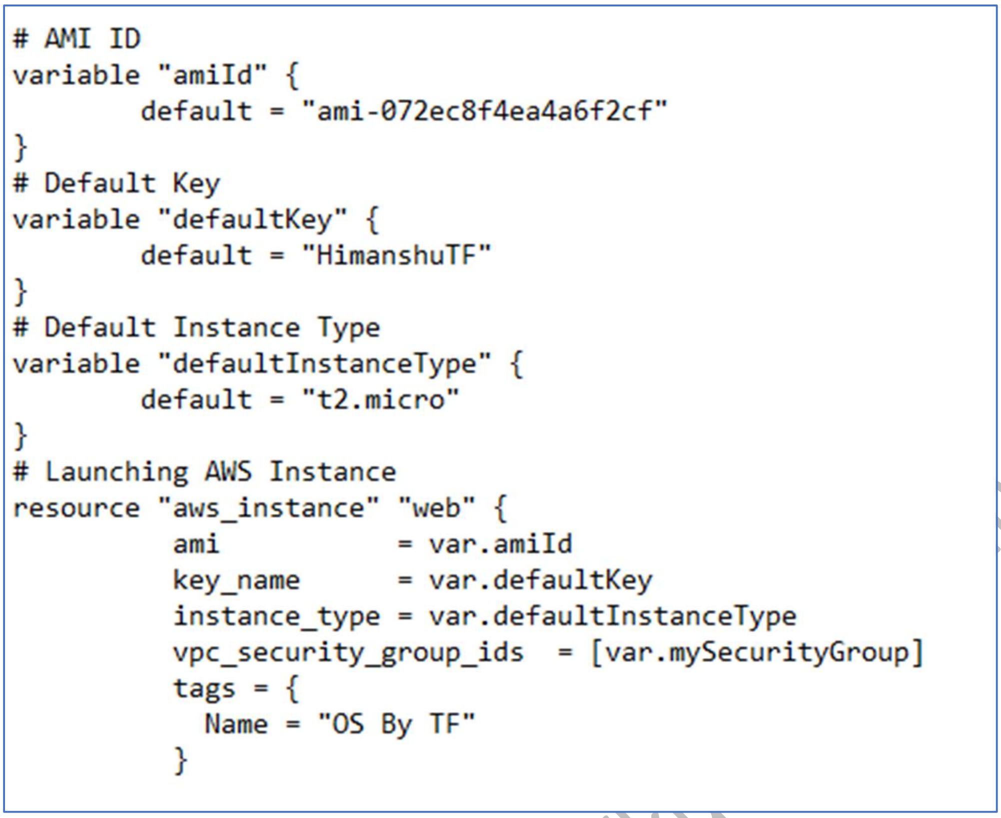
Now, For Authentication we can directly pass our access & secret key **WHICH IS NOT A GOOD PRACTICE.** Or we can use AWS CLI to login to the AWS account. But here, I am passing the access Key & secret Key.



```
# Default Region
variable "defaultRegion" {
        default = "ap-south-1"
}

# Authentication in AWS Cloud
provider "aws" {
  region     = var.defaultRegion
  access_key = "Your-Access-Key-Here"
  secret_key = "Your-Secret-Key-Here"
}
```

Here I Have created one variable for Default Region where I have stored the Value **ap-south-1.** By this way we can connect to the AWS Account.

After Login in the AWS, I am going to launch the OS.



```
# AMI ID
variable "amiId" {
        default = "ami-072ec8f4ea4a6f2cf"
}
# Default Key
variable "defaultKey" {
        default = "HimanshuTF"
}
# Default Instance Type
variable "defaultInstanceType" {
        default = "t2.micro"
}
# Launching AWS Instance
resource "aws_instance" "web" {
          ami           = var.amiId
          key_name      = var.defaultKey
          instance_type = var.defaultInstanceType
          vpc_security_group_ids  = [var.mySecurityGroup]
          tags = {
            Name = "OS By TF"
          }
```

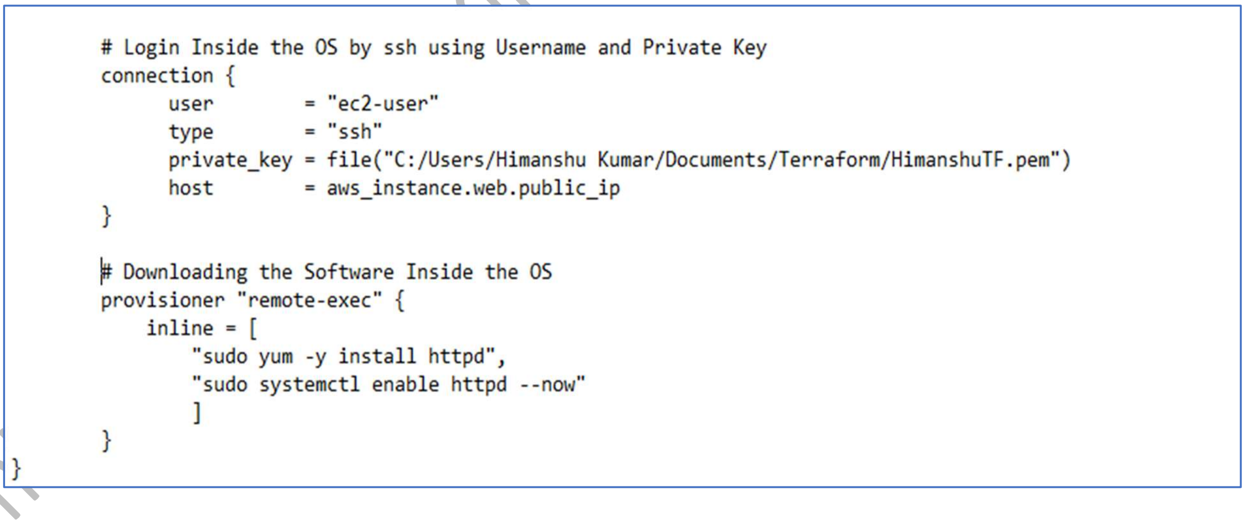
For This Here I Required Amazon Machine Image (AMI) ID: This is used for OS that I am going to Use.

Key Name that I have already created in the AWS Account.

Instance Type which is **t2.micro**.

And In the VPC Security Group I have allowed the permission for **All Traffics, HTTP, HTTPS, SSH**.

I have stored all these information inside the user Defined Variable.

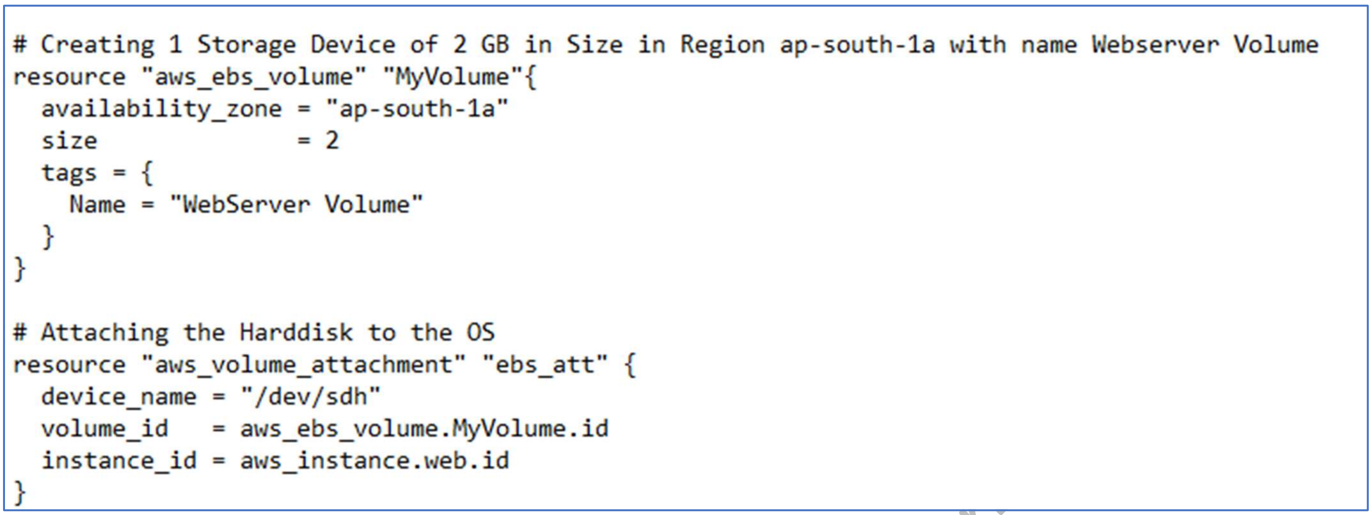


```
        # Login Inside the OS by ssh using Username and Private Key
        connection {
              user        = "ec2-user"
              type        = "ssh"
              private_key = file("C:/Users/Himanshu Kumar/Documents/Terraform/HimanshuTF.pem")
              host        = aws_instance.web.public_ip
        }

        # Downloading the Software Inside the OS
        provisioner "remote-exec" {
            inline = [
                "sudo yum -y install httpd",
                "sudo systemctl enable httpd --now"
                ]
        }
}
```

After Login inside the OS. I would like to launch the webserver. For this I need to do the Login inside the OS. For that I am going to establish the SSH Connection and here I am using the username as **ec2-user** and private key is stored in the Local OS. So, I have given the path for that. And here I am fetching the IP of the OS which can be retrieved later after launching the OS which is I mentioned in **host.**

After that I would like to create this OS for webserver purpose. For this I require some software which is httpd and after installing the software we need to start their services. Which I have performed inside the provisioner block.

In this OS, I would like to add some Extra Storage Device. For this, I am going to create 1 Hard Disk of size 2 GB. The things that we should keep in mind that. If we are launching our OS in ap-south-1a then we need to create the Hard Disk in the Same region so we can attach our HD in the OS, else if we create the HD in the different region and storage in different region then we cannot connect both the things together.



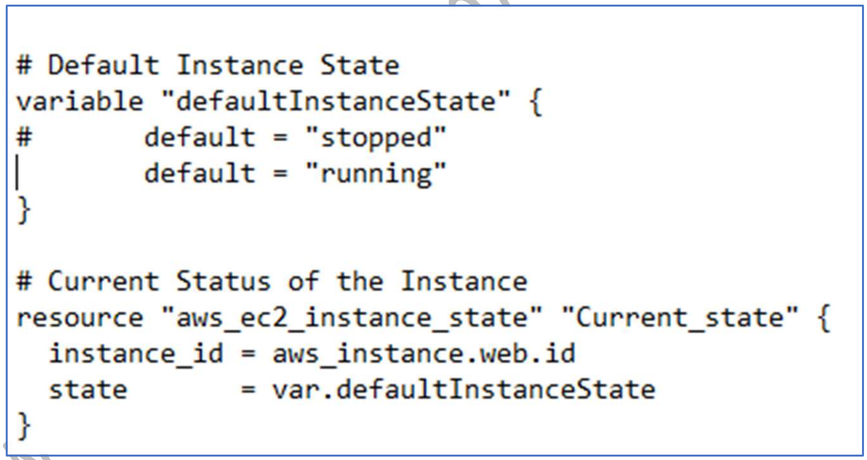
```
# Creating 1 Storage Device of 2 GB in Size in Region ap-south-1a with name Webserver Volume
resource "aws_ebs_volume" "MyVolume"{
  availability_zone = "ap-south-1a"
  size              = 2
  tags = {
    Name = "WebServer Volume"
  }
}

# Attaching the Harddisk to the OS
resource "aws_volume_attachment" "ebs_att" {
  device_name = "/dev/sdh"
  volume_id   = aws_ebs_volume.MyVolume.id
  instance_id = aws_instance.web.id
}
```

After Creating the Hard Disk, we need to attach our Hard Disk for that we need to mention the

Volume ID: It contains the details about the Hard disk.

Instance ID: It contains the details about the Instance.

After that we need to create one device where we would like to attach.



```
# Default Instance State
variable "defaultInstanceState" {
#        default = "stopped"
         default = "running"
}

# Current Status of the Instance
resource "aws_ec2_instance_state" "Current_state" {
  instance_id = aws_instance.web.id
  state       = var.defaultInstanceState
}
```

It is Not good practice to shut down the OS manually for this we can control our Instance state by changing the code, we can start and stop our Instance based on our requirement.

Once Everything is done, we need to run: **terraform validate.**

Terraform validate will check the code. If the code has any issues, then it will help us to find the error.

If the Code does not have any error, then we need to Deploy our infrastructure for this we can use:

terraform apply -auto-approve

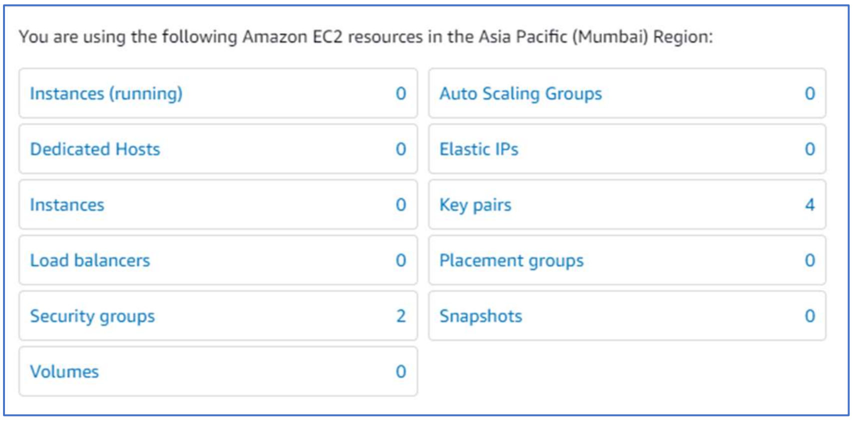
This will help us to reach our desired state.

-auto-approve: It will automatically approve the command.

Dangerous Command: It can wipe the entire Infrastructure in one go, so at the time of running be careful.

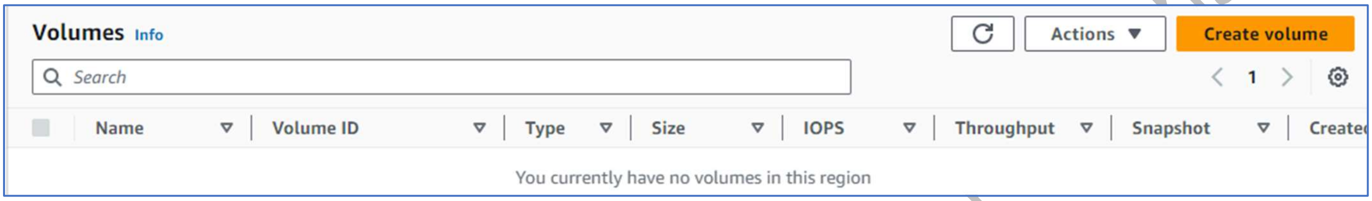
For destroying everything use **terraform destroy**

Current State Of my Infrastructure, 0 Instance is Launched & Not have any Volumes.



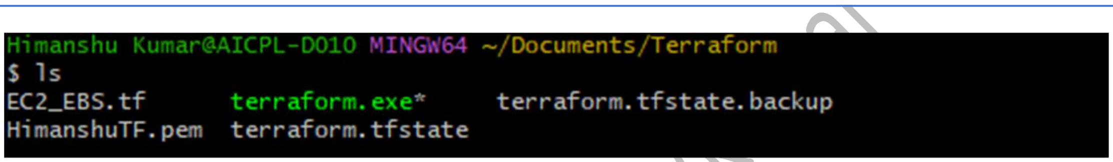
You are using the following Amazon EC2 resources in the Asia Pacific (Mumbai) Region:

Instances (running)	0	Auto Scaling Groups	0
Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	4
Load balancers	0	Placement groups	0
Security groups	2	Snapshots	0
Volumes	0		

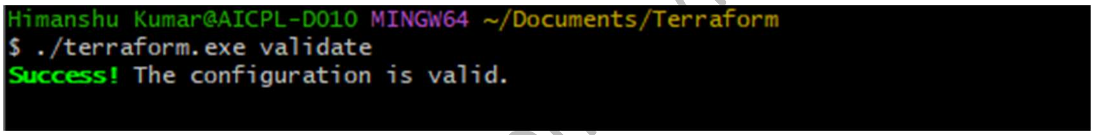


□	Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot	Create

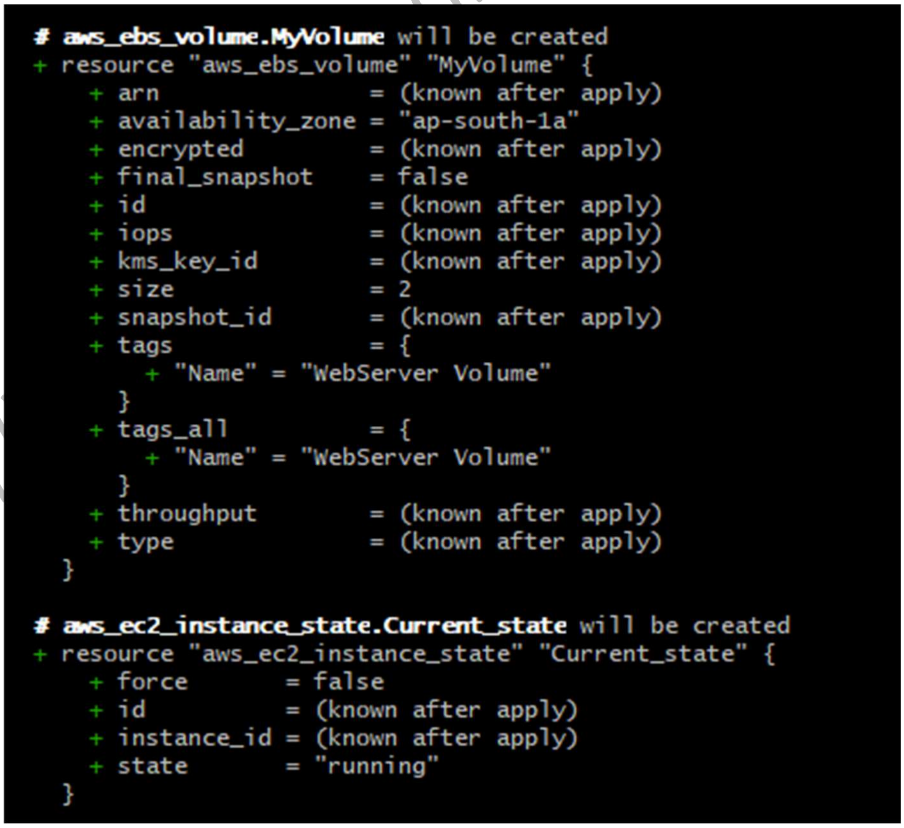
You currently have no volumes in this region



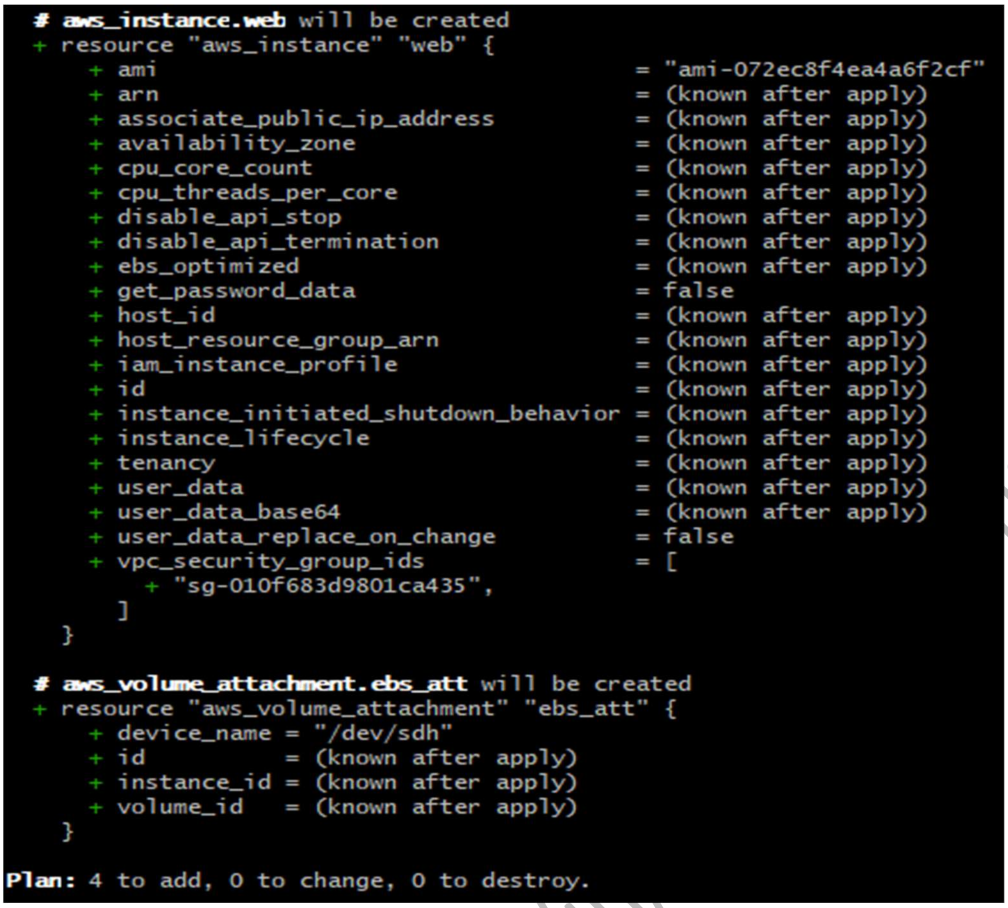
```
Himanshu Kumar@AICPL-D010 MINGW64 ~/Documents/Terraform
$ ls
EC2_EBS.tf       terraform.exe*     terraform.tfstate.backup
HimanshuTF.pem   terraform.tfstate
```



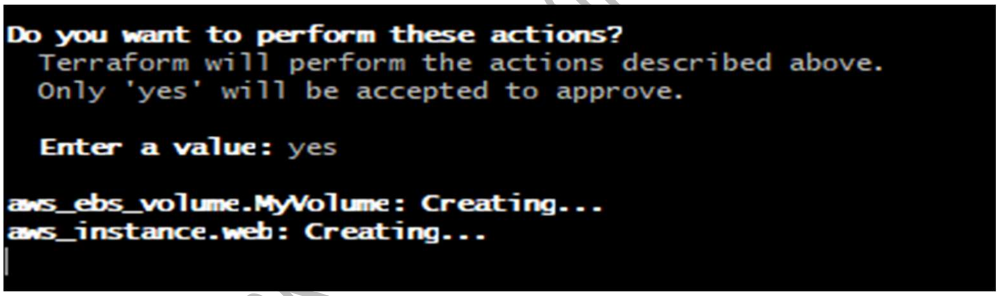
```
Himanshu Kumar@AICPL-D010 MINGW64 ~/Documents/Terraform
$ ./terraform.exe validate
Success! The configuration is valid.
```



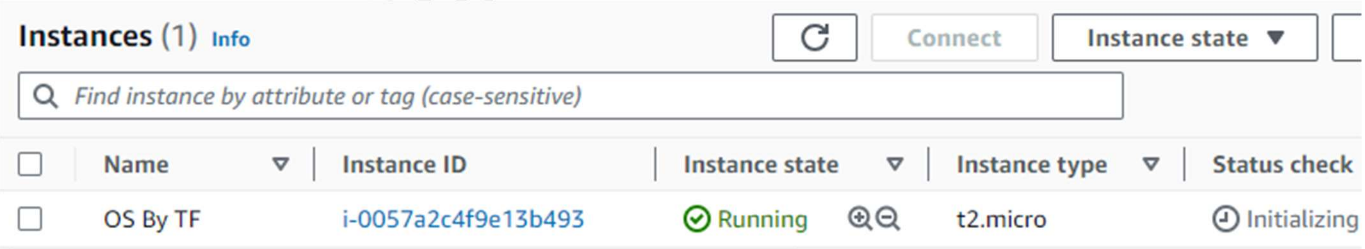
```
# aws_ebs_volume.MyVolume will be created
+ resource "aws_ebs_volume" "MyVolume" {
    + arn               = (known after apply)
    + availability_zone = "ap-south-1a"
    + encrypted         = (known after apply)
    + final_snapshot    = false
    + id                = (known after apply)
    + iops              = (known after apply)
    + kms_key_id        = (known after apply)
    + size              = 2
    + snapshot_id       = (known after apply)
    + tags              = {
        + "Name" = "WebServer Volume"
      }
    + tags_all          = {
        + "Name" = "WebServer Volume"
      }
    + throughput        = (known after apply)
    + type              = (known after apply)
  }

# aws_ec2_instance_state.Current_state will be created
+ resource "aws_ec2_instance_state" "Current_state" {
    + force       = false
    + id          = (known after apply)
    + instance_id = (known after apply)
    + state       = "running"
  }
```



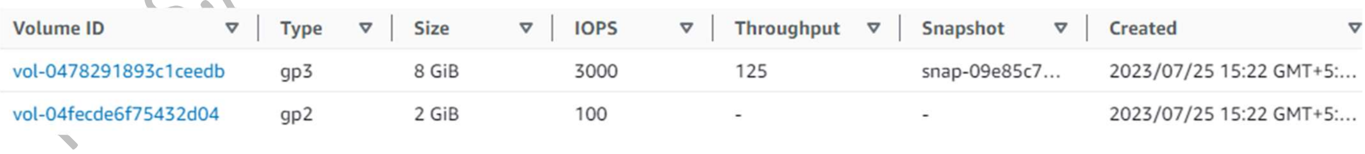
```
  # aws_instance.web will be created
  + resource "aws_instance" "web" {
      + ami                                  = "ami-072ec8f4ea4a6f2cf"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (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)
      + tenancy                              = (known after apply)
      + user_data                            = (known after apply)
      + user_data_base64                     = (known after apply)
      + user_data_replace_on_change          = false
      + vpc_security_group_ids               = [
          + "sg-010f683d9801ca435",
        ]
    }

  # aws_volume_attachment.ebs_att will be created
  + resource "aws_volume_attachment" "ebs_att" {
      + device_name = "/dev/sdh"
      + id          = (known after apply)
      + instance_id = (known after apply)
      + volume_id   = (known after apply)
    }

Plan: 4 to add, 0 to change, 0 to destroy.
```



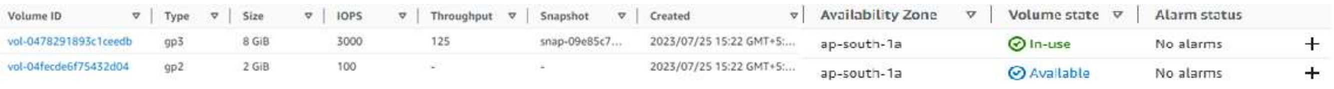
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_ebs_volume.MyVolume: Creating...
aws_instance.web: Creating...
```



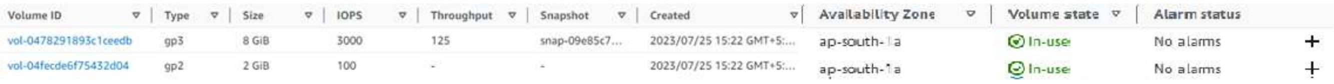
□	Name	Instance ID	Instance state	Instance type	Status check
□	OS By TF	i-0057a2c4f9e13b493	✓ Running	t2.micro	Initializing



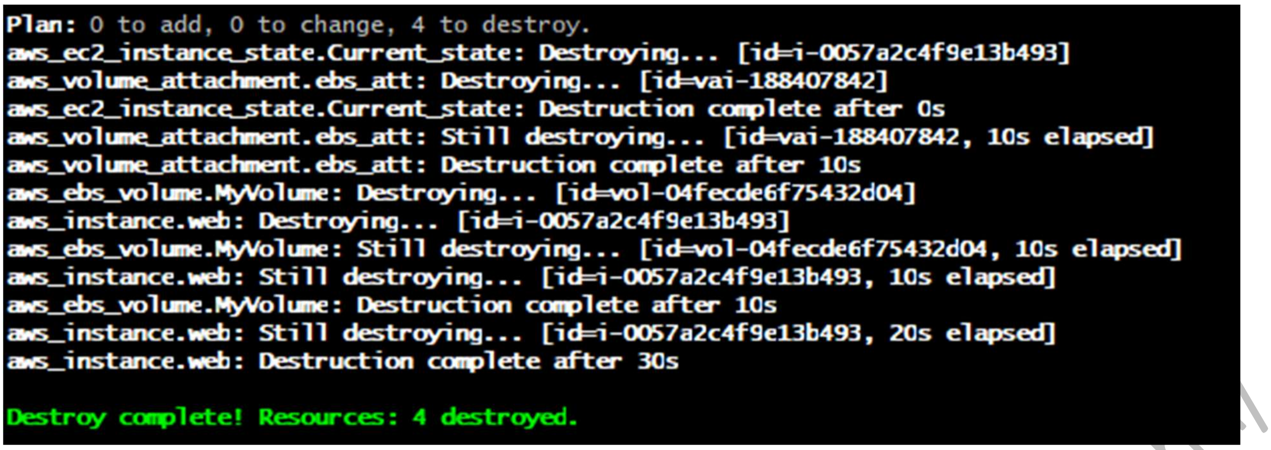
Volume ID	Type	Size	IOPS	Throughput	Snapshot	Created
vol-0478291893c1ceedb	gp3	8 GiB	3000	125	snap-09e85c7...	2023/07/25 15:22 GMT+5:...
vol-04fecde6f75432d04	gp2	2 GiB	100	-	-	2023/07/25 15:22 GMT+5:...



Volume ID	Type	Size	IOPS	Throughput	Snapshot	Created	Availability Zone	Volume state	Alarm status
vol-0478291893c1ceedb	gp3	8 GiB	3000	125	snap-09e85c7...	2023/07/25 15:22 GMT+5:...	ap-south-1a	✓ In-use	No alarms
vol-04fecde6f75432d04	gp2	2 GiB	100	-	-	2023/07/25 15:22 GMT+5:...	ap-south-1a	✓ Available	No alarms



Volume ID	Type	Size	IOPS	Throughput	Snapshot	Created	Availability Zone	Volume state	Alarm status
vol-0478291893c1ceedb	gp3	8 GiB	3000	125	snap-09e85c7...	2023/07/25 15:22 GMT+5:...	ap-south-1a	✓ In-use	No alarms
vol-04fecde6f75432d04	gp2	2 GiB	100	-	-	2023/07/25 15:22 GMT+5:...	ap-south-1a	✓ In-use	No alarms



```
Plan: 0 to add, 0 to change, 4 to destroy.
aws_ec2_instance_state.Current_state: Destroying... [id=i-0057a2c4f9e13b493]
aws_volume_attachment.ebs_att: Destroying... [id=vai-188407842]
aws_ec2_instance_state.Current_state: Destruction complete after 0s
aws_volume_attachment.ebs_att: Still destroying... [id=vai-188407842, 10s elapsed]
aws_volume_attachment.ebs_att: Destruction complete after 10s
aws_ebs_volume.MyVolume: Destroying... [id=vol-04fecde6f75432d04]
aws_instance.web: Destroying... [id=i-0057a2c4f9e13b493]
aws_ebs_volume.MyVolume: Still destroying... [id=vol-04fecde6f75432d04, 10s elapsed]
aws_instance.web: Still destroying... [id=i-0057a2c4f9e13b493, 10s elapsed]
aws_ebs_volume.MyVolume: Destruction complete after 10s
aws_instance.web: Still destroying... [id=i-0057a2c4f9e13b493, 20s elapsed]
aws_instance.web: Destruction complete after 30s

Destroy complete! Resources: 4 destroyed.
```