

Terraform with AWS



Provisioning on AWS is quite easy and straightforward with Terraform.

Prerequisites

AWS CLI installed

The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

1. Create an EC2 instance.

2.SSH into EC2 instance

3.Install AWS CLI

```
sudo apt install awscli
```

```
ubuntu@ip-172-31-9-113:~$ sudo apt install awscli
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2 docutils-common fontconfig fontconfig-config fonts-droid-fallback fonts-noto-mono fonts-urw-base35 ghostscript groff gsfonts hicolor-icon-theme
  imagemagick imagemagick-6-common imagemagick-6.q16 libaom3 libavahi-client3 libavahi-common-data libavahi-common3 libcairo2 libcups2 libdatrie1
  libdav1d5 libde265-0 libdeflate0 libdjvulibre-text libdjvulibre21 libfftw3-double3 libfontconfig1 libgomp1 libgraphite2-3 libgs9 libgs9-common
  libharfbuzz0b libheif1 libice6 libidn12 libijs-0.35 liblilmbase25 libimagequant0 libjbig0 libjbig2dec0 libjpeg-turbo8 libjpeg8 libjxr-tools libjxr0
  liblcms2-2 liblqr-1-0 libltdl7 libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra libmagickwand-6.q16-6 libnetpbm10 libopenexr25 libopenjp2-7
  libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpaper-utils libpaper1 libpixman-1-0 libraqm0 libsm6 libthai-data libthai0 libtiff5 libwebp7
  libwebpdemux2 libwebpmux3 libwmf-lite-0.2-7 libx265-199 libxaw7 libxcb-render0 libxcb-shm0 libxmu6 libxpm4 libxrender1 libxt6 mailcap mime-support netpbm
  poppler-data psutils python3-botocore python3-dateutil python3-docutils python3-jmespath python3-olefile python3-pil python3-pygments python3-roman
  python3-rsa python3-s3transfer sgml-base x11-common xml-core
Suggested packages:
  bzip2-doc fonts-noto fonts-freefont-otf | fonts-freefont-ttf fonts-texgyre ghostscript-x imagemagick-doc autotrace cups-bsd | lpr | lprng enscript
  ffmpeg gimp gnuplot grads graphviz hp2xx html2ps libwmf-bin mplayer povray radiance sane-utils texlive-base-bin transfig ufw batch xdg-utils
  cups-common libfftw3-bin libfftw3-dev liblcms2-utils inkscape poppler-utils fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum docutils-doc fonts-linuxlibertine | ttf-linux-libertine texlive-lang-french
  texlive-latex-base texlive-latex-recommended python-pil-doc python-pygments-doc ttf-bitstream-vera sgml-base-doc debhelper
The following NEW packages will be installed:
  awscli bzip2 docutils-common fontconfig fontconfig-config fonts-droid-fallback fonts-noto-mono fonts-urw-base35 ghostscript groff gsfonts
  hicolor-icon-theme imagemagick imagemagick-6-common imagemagick-6.q16 libaom3 libavahi-client3 libavahi-common-data libavahi-common3 libcairo2 libcups2
  libdatrie1 libdav1d5 libde265-0 libdeflate0 libdjvulibre-text libdjvulibre21 libfftw3-double3 libfontconfig1 libgomp1 libgraphite2-3 libgs9
  libgs9-common libharfbuzz0b libheif1 libice6 libidn12 libijs-0.35 liblilmbase25 libimagequant0 libjbig0 libjbig2dec0 libjpeg-turbo8 libjpeg8 libjxr-tools
  libjxr0 liblcms2-2 liblqr-1-0 libltdl7 libmagickcore-6.q16-6 libmagickcore-6.q16-6-extra libmagickwand-6.q16-6 libnetpbm10 libopenexr25 libopenjp2-7
  libpango-1.0-0 libpangocairo-1.0-0 libpangoft2-1.0-0 libpaper-utils libpaper1 libpixman-1-0 libraqm0 libsm6 libthai-data libthai0 libtiff5 libwebp7
  libwebpdemux2 libwebpmux3 libwmf-lite-0.2-7 libx265-199 libxaw7 libxcb-render0 libxcb-shm0 libxmu6 libxpm4 libxrender1 libxt6 mailcap mime-support netpbm
  poppler-data psutils python3-botocore python3-dateutil python3-docutils python3-jmespath python3-olefile python3-pil python3-pygments python3-roman
  python3-rsa python3-s3transfer sgml-base x11-common xml-core
0 upgraded, 96 newly installed, 0 to remove and 30 not upgraded.
Need to get 46.0 MB of archives.
After this operation, 226 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all 1:6.0.1r16-1.1build1 [1805 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgomp1 amd64 12.1.0-2ubuntu1-22.04 [126 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libfftw3-double3 amd64 3.3.8-2ubuntu8 [770 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all 20200910-1 [6367 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fontconfig-config all 2.13.1-4.2ubuntu5 [29.1 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libfontconfig1 amd64 2.13.1-4.2ubuntu5 [131 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libaom3 amd64 3.3.0-1 [1748 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libdav1d5 amd64 0.9.2-1 [463 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libde265-0 amd64 1.0.8-1 [243 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libx265-199 amd64 3.5-2 [1170 kB]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 libheif1 amd64 1.12.0-2build1 [196 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libjbig0 amd64 2.1-3.1ubuntu0.22.04.1 [29.2 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libjpeg-turbo8 amd64 2.1.2-0ubuntu1 [134 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libjpeg8 amd64 8c-2ubuntu10 [2264 B]
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 liblcms2-2 amd64 2.12-rc1-2build2 [159 kB]
```

Check aws cli version

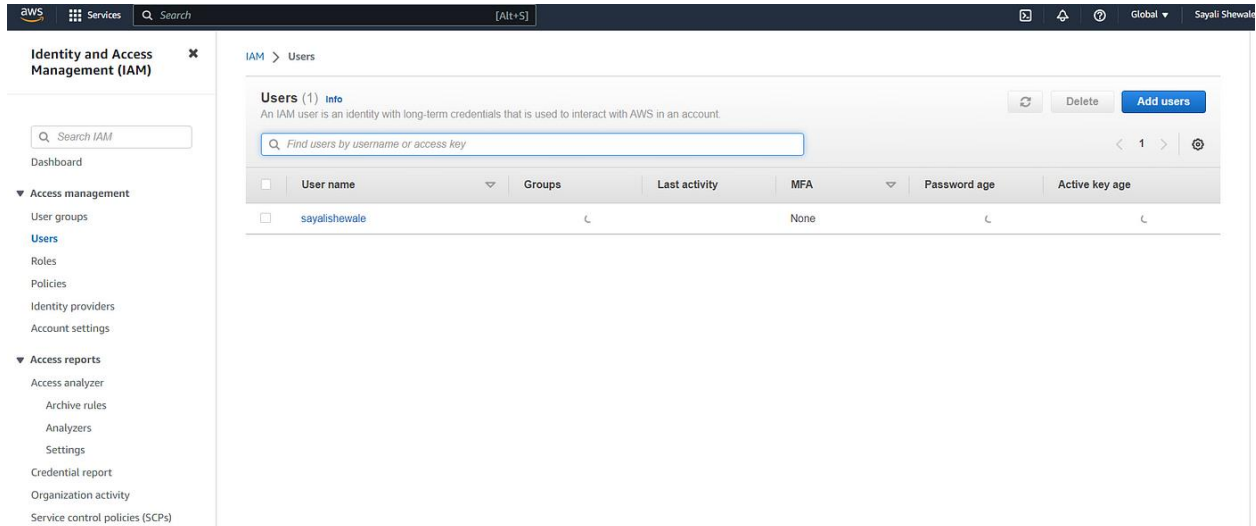
```
ubuntu@ip-172-31-9-113:~$ aws --version
aws-cli/1.22.34 Python/3.10.6 Linux/5.15.0-1031-aws botocore/1.23.34
ubuntu@ip-172-31-9-113:~$
ubuntu@ip-172-31-9-113:~$
```

AWS IAM user

IAM (Identity Access Management) AWS Identity and Access Management (IAM) is a web service that helps you securely control

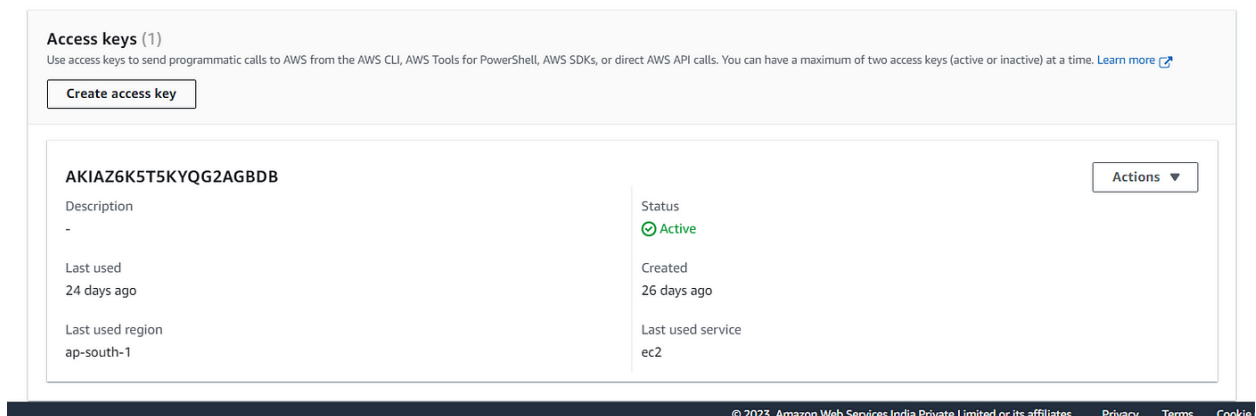
access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

Create IAM user.

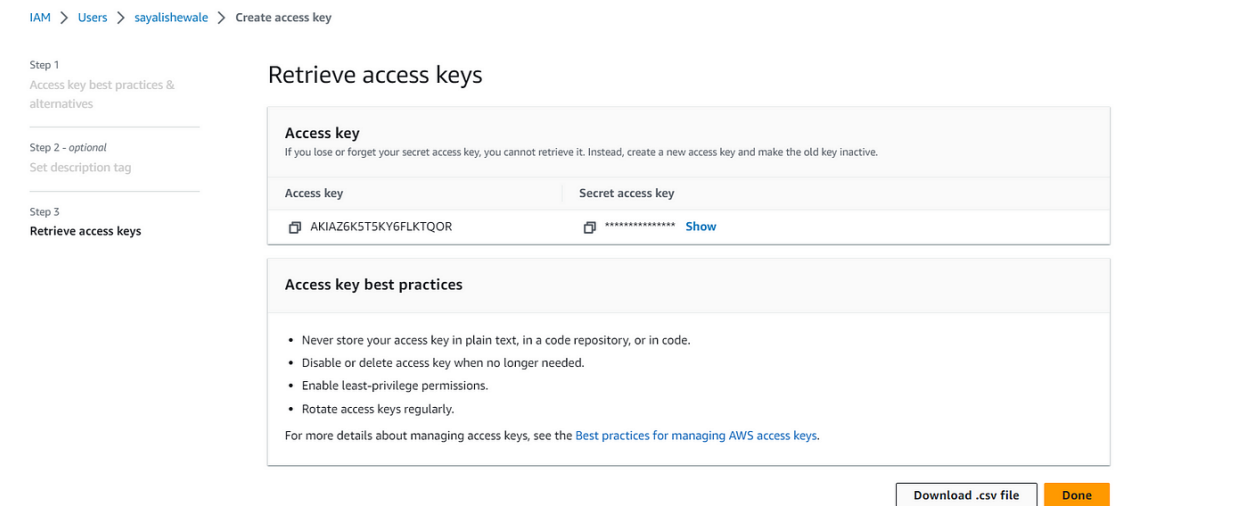
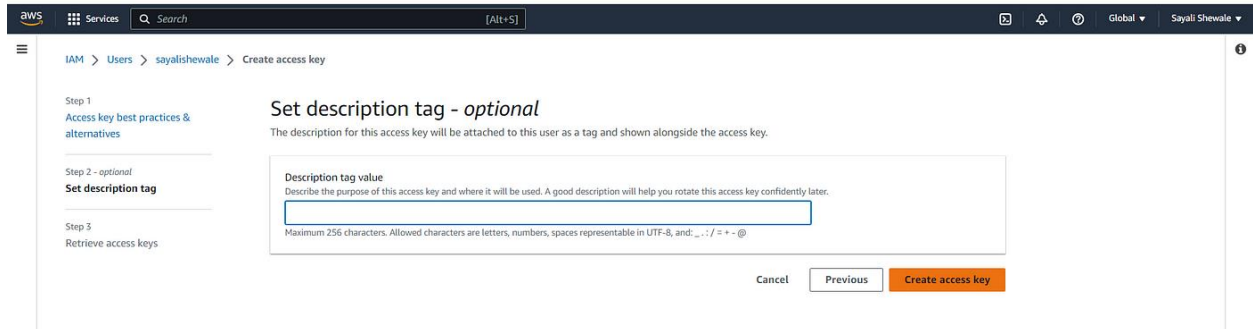
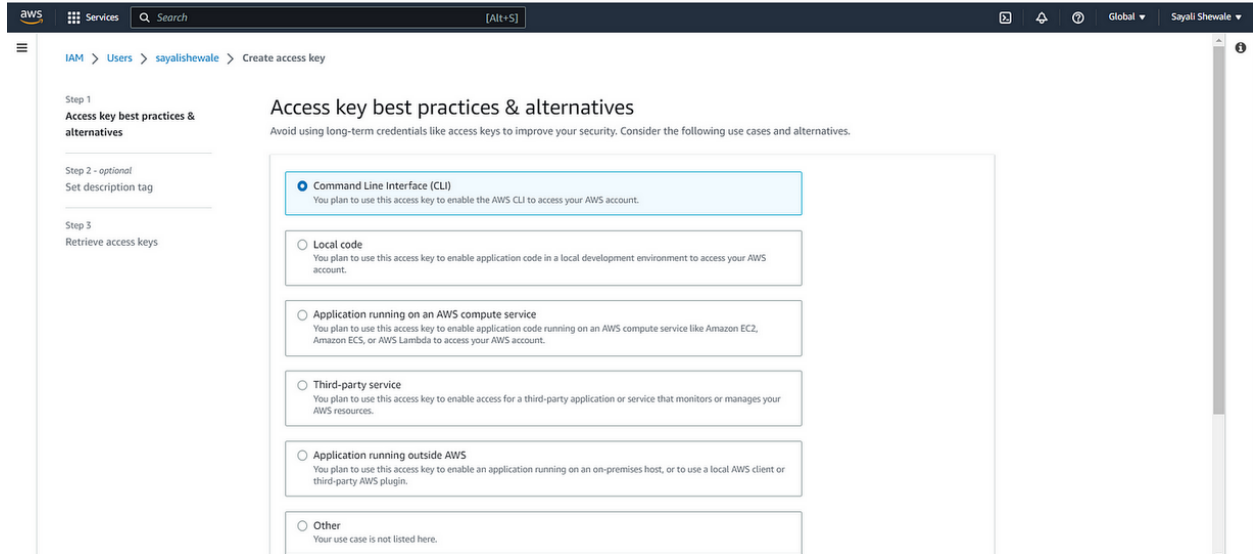


Create Access key for IAM user.

Click on ‘Create access key’



Select ‘command line interface’



In order to connect your AWS account and Terraform, you need the access keys and secret access keys exported to your machine.

```
export AWS_ACCESS_KEY_ID=<access key>
export AWS_SECRET_ACCESS_KEY=<secret access key>
```

```
ubuntu@ip-172-31-9-113:~$
ubuntu@ip-172-31-9-113:~$ export AWS_ACCESS_KEY_ID=AKIAZ6K5T5KY6FLKTQOR
ubuntu@ip-172-31-9-113:~$ export AWS_SECRET_ACCESS_KEY=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
ubuntu@ip-172-31-9-113:~$
ubuntu@ip-172-31-9-113:~$
```

Install required providers

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}
```

The terraform block defines the version of Terraform that is required to execute this configuration. In this case, it specifies that the Terraform version must be $\geq 1.2.0$.

The required_providers block declares the AWS provider and its version that Terraform will use for the resources defined in this configuration. In this case, it declares the AWS provider with the source hashicorp/aws and specifies that the version of the provider should be $\sim > 4.16$, which means any version of the AWS provider greater than or equal to 4.16 and less than 5.0 will be acceptable.

Add the region where you want your instances to be

```
provider "aws" {  
  region = "ap-south-1"  
}
```

```
ubuntu@ip-172-31-9-113:~$ cd terraform/  
ubuntu@ip-172-31-9-113:~/terraform$ cd terraform-aws/  
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ sudo vi main.tf  
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ cat main.tf  
terraform {  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = "~> 4.16"  
    }  
  }  
  required_version = ">= 1.2.0"  
}  
  
provider "aws" {  
  region = "ap-south-1"  
}  
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ |
```

Task-01

Provision an AWS EC2 instance using Terraform

```
resource "aws_instance" "aws_ec2_demo" {  
  count = 2  
  ami   = "ami-0f8ca728008ff5af4"  
  instance_type = "t2.micro"  
  tags = {  
    Name = "TerraformTestInstance"  
  }  
}
```

The resource block has a resource type of “aws_instance” and a resource name of “aws_ec2_demo”. The count parameter is set to 2, which means that two instances will be created.

The `ami` parameter specifies the Amazon Machine Image (AMI) to use for the instances. In this case, the AMI ID is “ami-of8ca728008ff5af4”.

The `instance_type` parameter specifies the type of instance to create. In this case, the instance type is “t2.micro”.

The `tags` parameter specifies metadata to attach to the instance, in this case, a tag named “Name” with the value “TerraformTestInstance”.

```
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ sudo vi main.tf
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ cat main.tf
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.16"
    }
  }
  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "ap-south-1"
}

resource "aws_instance" "aws_ec2_demo" {
  count = 2
  ami = "ami-0f8ca728008ff5af4"
  instance_type = "t2.micro"
  tags = {
    Name = "TerraformTestInstance"
  }
}
```

first initialize the working directory with the necessary plugins and modules by executing `terraform init`

```

ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.16"...
- Installing hashicorp/aws v4.59.0...
- Installed hashicorp/aws v4.59.0 (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-172-31-9-113:~/terraform/terraform-aws$

```

It will create an execution plan by analyzing the changes required to achieve the desired state of your infrastructure with terraform plan

```

ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ 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.aws_ec2_demo[0] will be created
+ resource "aws_instance" "aws_ec2_demo" {
+   ami                        = "ami-0f8ca72808ff5af4"
+   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_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
+   subnet_id                  = (known after apply)
+   tags                       = {
+     "Name" = "TerraformTestInstance"
+   }
+   tags_all                   = {
+     "Name" = "TerraformTestInstance"
+   }
}

```



```

# aws_instance.aws_ec2_demo[1] will be created
+ resource "aws_instance" "aws_ec2_demo" {
  + ami                    = "ami-0f8ca728008ff5af4"
  + 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_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
  + subnet_id               = (known after apply)
  + tags                    = {
    + "Name" = "TerraformTestInstance"
  }
  + tags_all                = {
    + "Name" = "TerraformTestInstance"
  }
  + 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   = (known after apply)
}

```

Plan: 2 to add, 0 to change, 0 to destroy.

Finally, it will apply the changes to create or update resources as needed with terraform apply.

```
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$ 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_instance.aws_ec2_demo[0] will be created
+ resource "aws_instance" "aws_ec2_demo" {
  + ami                    = "ami-0f8ca728088ff5af4"
  + 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_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
  + subnet_id              = (known after apply)
  + tags                   = {
    + "Name" = "TerraformTestInstance"
  }
  + tags_all              = {
    + "Name" = "TerraformTestInstance"
  }
  + tenancy                = (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
  + subnet_id              = (known after apply)
  + tags                   = {
    + "Name" = "TerraformTestInstance"
  }
  + tags_all              = {
    + "Name" = "TerraformTestInstance"
  }
  + 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 = (known after apply)
}
```

Plan: 2 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.aws_ec2_demo[0]: Creating...
aws_instance.aws_ec2_demo[1]: Creating...
aws_instance.aws_ec2_demo[0]: Still creating... [10s elapsed]
aws_instance.aws_ec2_demo[1]: Still creating... [10s elapsed]
aws_instance.aws_ec2_demo[0]: Still creating... [20s elapsed]
aws_instance.aws_ec2_demo[1]: Still creating... [20s elapsed]
aws_instance.aws_ec2_demo[0]: Still creating... [30s elapsed]
aws_instance.aws_ec2_demo[1]: Still creating... [30s elapsed]
aws_instance.aws_ec2_demo[0]: Creation complete after 31s [id=i-0784743f5db28fbab]
aws_instance.aws_ec2_demo[1]: Still creating... [40s elapsed]
aws_instance.aws_ec2_demo[1]: Creation complete after 41s [id=i-0d1b4a535b7be12c5]
```

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

```
ubuntu@ip-172-31-9-113:~/terraform/terraform-aws$
```

You can check two instances created using terraform.

The screenshot shows the AWS Management Console interface for the EC2 service. The top navigation bar includes the AWS logo, a search bar, and user information. The left sidebar contains a navigation menu with options like 'EC2 Dashboard', 'EC2 Global View', 'Events', 'Tags', 'Limits', and 'Instances'. The 'Instances' section is expanded, showing a list of instances. The main content area displays a table of instances with columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Three instances are listed: 'terraform-demo' (Running), 'TerraformTestInstance' (Running), and another 'TerraformTestInstance' (Initializing).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
terraform-demo	i-0f5926d246fd675bb	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1b
TerraformTestInstance	i-0784743f5db28fbab	Running	t2.micro	Initializing	No alarms	ap-south-1b
TerraformTestInstance	i-0d1b4a535b7be12c5	Running	t2.micro	Initializing	No alarms	ap-south-1b

Thank you for reading!