

Steps-

Creating Docker container using terraform

Step 1: Check the docker functionality

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Mikil> docker

Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\\Users\\Mikil\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\\Users\\Mikil\\.docker\\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\\Users\\Mikil\\.docker\\cert.pem")
  --tlskey string        Path to TLS key file (default
                        "C:\\Users\\Mikil\\.docker\\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder      Manage builds
  buildx*      Docker Buildx (Docker Inc., v0.8.2)
  compose*     Docker Compose (Docker Inc., v2.7.0)
  config       Manage Docker configs
  container    Manage containers
  context       Manage contexts
  extension*   Manages Docker extensions (Docker Inc., v0.2.8)
  image        Manage images
  manifest     Manage Docker image manifests and manifest lists
  network      Manage networks
  node         Manage Swarm nodes
  plugin       Manage plugins
  sbom*        View the packaged-based Software Bill Of Materials (SBOM) for an image (Anchore Inc., 0.6.0)
  scan*        Docker Scan (Docker Inc., v0.17.0)
  secret       Manage Docker secrets
  service      Manage services
  stack        Manage Docker stacks
  swarm        Manage Swarm
  system       Manage Docker
  trust        Manage trust on Docker images
  volume       Manage volumes
```

```
Windows PowerShell
PS C:\Users\Mikil> docker --version
Docker version 20.10.17, build 100c701
PS C:\Users\Mikil>
```

Step 2: Write a terraform script to create a Ubuntu Linux container

Create a new docker.tf file using Atom editor and write the following contents into it.

```
docker.tf — C:\Terraform — Atom
File Edit View Selection Find Packages Help

docker.tf
1 terraform{
2   required_providers{
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "2.13.0"
6     }
7   }
8 }
9 provider "docker"{
10  version = "~>2.7"
11  host = "npipe://///pipe/docker_engine"
12 }
13 # Pulls image
14 resource "docker_image" "ubuntu" {
15   name = "ubuntu:latest"
16 }
```

Save the file in a new directory called docker where rest of the terraform scripts are stored

Step 2: Open Command Prompt and go to Terraform_Script\docker directory where our .tf file is stored

```
Windows PowerShell
PS C:\Users\Mikil> cd..
PS C:\Users> cd..
PS C:\> cd .\Terraform\Docker
PS C:\Terraform\Docker> dir

    Directory: C:\Terraform\Docker

Mode                LastWriteTime         Length Name
----                -
-a----           28-08-2022    14:27           270 docker.tf

PS C:\Terraform\Docker>
```

Step 3: Execute Terraform Init command to initialize the resources

```
Windows PowerShell
PS C:\Terraform\Docker> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 2.7, 2.13.0"...
- Installing kreuzwerker/docker v2.13.0...
- Installed kreuzwerker/docker v2.13.0 (self-signed, key ID 24E54F214569A8A5)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

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.


Warning: Version constraints inside provider configuration blocks are deprecated

on docker.tf line 10, in provider "docker":
10: version = "~>2.7"

Terraform 0.13 and earlier allowed provider version constraints inside the provider
configuration block, but that is now deprecated and will be removed in a future
version of Terraform. To silence this warning, move the provider version constraint
into the required_providers block.


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.
PS C:\Terraform\Docker>
```

Step 4: Execute Terraform plan to see the available resources

```
Windows PowerShell
commands will detect it and remind you to do so if necessary.
PS C:\Terraform\Docker> 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:

# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
  + id          = (known after apply)
  + latest      = (known after apply)
  + name        = "ubuntu:latest"
  + output      = (known after apply)
  + repo_digest = (known after apply)
}

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

Warning: Version constraints inside provider configuration blocks are deprecated

on docker.tf line 10, in provider "docker":
10:   version = "~> 2.7"

Terraform 0.13 and earlier allowed provider version constraints inside the provider
configuration block, but that is now deprecated and will be removed in a future version
of Terraform. To silence this warning, move the provider version constraint into the
required_providers block.

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.
PS C:\Terraform\Docker>
```

Step 5: Execute Terraform apply to apply the configuration, which will automatically create and run the ubuntu Linux container based on our configuration.

```
Windows PowerShell
PS C:\Terraform\Docker> 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:

# docker_image.ubuntu will be created
+ resource "docker_image" "ubuntu" {
+   id           = (known after apply)
+   latest       = (known after apply)
+   name         = "ubuntu:latest"
+   output       = (known after apply)
+   repo_digest  = (known after apply)
+ }

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

Warning: Version constraints inside provider configuration blocks are deprecated

on docker.tf line 10, in provider "docker":
10:   version = "~> 2.7"

Terraform 0.13 and earlier allowed provider version constraints inside the provider
configuration block, but that is now deprecated and will be removed in a future version
of Terraform. To silence this warning, move the provider version constraint into the
required_providers block.

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

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Creation complete after 21s [id=sha256:df5de72bdb3b711aba4eca685b1f42c7
22cc8a1837ed3fbd548a9282af2d836dubuntu:latest]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform\Docker>
```

Docker images Before Executing Apply command:

```
PS C:\Terraform\Docker> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
PS C:\Terraform\Docker>
```

Docker images, After Executing Apply step:

```
PS C:\Terraform\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    df5de72bdb3b   3 weeks ago    77.8MB
PS C:\Terraform\Docker>
```

Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container

```
Windows PowerShell
PS C:\Terraform\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d836dubuntu:latest]

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:

# docker_image.ubuntu will be destroyed
- resource "docker_image" "ubuntu" {
  id      = "sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d836dubuntu:latest" -> null
  latest  = "sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d836d" -> null
  name    = "ubuntu:latest" -> null
  repo_digest = "ubuntu@sha256:34fea4f31bf187bc915536831fd0afc9d214755bf700b5cdb1336c82516d154e" -> null
}

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

Warning: Version constraints inside provider configuration blocks are deprecated

on docker.tf line 10, in provider "docker":
10:  version = "~> 2.7"

Terraform 0.13 and earlier allowed provider version constraints inside the provider configuration block, but that is now deprecated and will be removed in a future version of Terraform. To silence this warning, move the provider version constraint into the required_providers block.

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

docker_image.ubuntu: Destroying... [id=sha256:df5de72bdb3b711aba4eca685b1f42c722cc8a1837ed3fbd548a9282af2d836dubuntu:latest]
docker_image.ubuntu: Destruction complete after 0s

Destroy complete! Resources: 1 destroyed.
PS C:\Terraform\Docker>
```

Docker images After Executing Destroy step:

```
PS C:\Terraform\Docker> docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
PS C:\Terraform\Docker>
```


Creating S3 Bucket using terraform

Step 1: Write a Terraform Script in Atom for creating S3 Bucket on Amazon AWS

```
s3.tf
1 resource "aws_s3_bucket" "mikil"{
2     bucket = "terraform-test-bucket-mikil"
3     acl = "public-read"
4
5     tags = {
6         Name = "My bucket"
7         Enviorment = "Dev"
8     }
9 }
10
```

Create a new provider.tf file and write the following contents into it.

```
provider.tf
provider "aws"{
    access_key = "AKIA2A5SPDYNZBVARMWV"
    secret_key = "jVRwp/NQnvNS3pPrNwU/qxNqw340fQ90330X8"
    region = "us-east-1"
}
```

Save both the files in the same directory Terraform/S3

Step 2: Open Command Prompt and go to Terraform\S3 directory where our .tf files are stored

```
Windows PowerShell
PS C:\> cd .\Terraform\
PS C:\Terraform> cd .\S3\
PS C:\Terraform\S3> dir

Directory: C:\Terraform\S3

Mode                LastWriteTime         Length Name
----                -
-a----          28-08-2022   16:01             141 provider.tf
-a----          28-08-2022   15:28             164 s3.tf

PS C:\Terraform\S3>
```

Step 3: Execute Terraform Init command to initialize the resources

```
PS C:\Terraform\S3> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.28.0...
- Installed hashicorp/aws v4.28.0 (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.
PS C:\Terraform\S3>
```

Step 4: Execute Terraform plan to see the available resources

Windows PowerShell

PS C:\Terraform\S3> 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_s3_bucket.mikil will be created
+ resource "aws_s3_bucket" "mikil" {
  + acceleration_status = (known after apply)
  + acl                 = "public-read"
  + arn                 = (known after apply)
  + bucket              = "terraform-test-bucket"
  + bucket_domain_name = (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                = {
    + "Enviorment" = "Dev"
    + "Name"       = "My bucket"
  }
  + tags_all            = {
    + "Enviorment" = "Dev"
    + "Name"       = "My bucket"
  }
  + website_domain      = (known after apply)
  + website_endpoint    = (known after apply)

  + cors_rule {
    + allowed_headers = (known after apply)
    + allowed_methods = (known after apply)
    + allowed_origins = (known after apply)
    + expose_headers  = (known after apply)
    + max_age_seconds = (known after apply)
  }

  + grant {
    + id           = (known after apply)
    + permissions = (known after apply)
    + type        = (known after apply)
    + uri         = (known after apply)
  }

  + lifecycle_rule {
    + abort_incomplete_multipart_upload_days = (known after apply)
    + enabled                                = (known after apply)
    + id                                     = (known after apply)
    + prefix                                 = (known after apply)
    + tags                                  = (known after apply)

    + expiration {
      + date           = (known after apply)
      + days           = (known after apply)
      + expired_object_delete_marker = (known after apply)
    }

    + noncurrent_version_expiration {
      + days = (known after apply)
    }

    + noncurrent_version_transition {
      + days           = (known after apply)
      + storage_class = (known after apply)
    }
  }
}
```

Windows PowerShell

```
+ access_control_translation {
  + owner = (known after apply)
}

+ metrics {
  + minutes = (known after apply)
  + status  = (known after apply)
}

+ replication_time {
  + minutes = (known after apply)
  + status  = (known after apply)
}
}

+ filter {
  + prefix = (known after apply)
  + tags   = (known after apply)
}

+ source_selection_criteria {
  + sse_kms_encrypted_objects {
    + enabled = (known after apply)
  }
}
}

+ server_side_encryption_configuration {
  + rule {
    + bucket_key_enabled = (known after apply)

    + apply_server_side_encryption_by_default {
      + kms_master_key_id = (known after apply)
      + sse_algorithm      = (known after apply)
    }
  }
}

+ versioning {
  + enabled    = (known after apply)
  + mfa_delete = (known after apply)
}

+ website {
  + error_document          = (known after apply)
  + index_document          = (known after apply)
  + redirect_all_requests_to = (known after apply)
  + routing_rules            = (known after apply)
}
}
```

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

Warning: Argument is deprecated

```
with aws_s3_bucket.mikil,
on s3.tf line 3, in resource "aws_s3_bucket" "mikil":
3:   acl = "public-read"
```

Use the aws_s3_bucket_acl resource instead

(and one more similar warning elsewhere)

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.

PS C:\Terraform\S3>

Step 5: Execute Terraform apply to apply the configuration, which will automatically create an S3 bucket based on our configuration.

```
Windows PowerShell
PS C:\Terraform\S3> terraform apply

Terraform used the selected providers to generate the following execution plan.
are indicated with the following symbols:
  + create

Terraform will perform the following actions:

# aws_s3_bucket.mikil will be created
+ resource "aws_s3_bucket" "mikil" {
  + acceleration_status      = (known after apply)
  + acl                      = "public-read"
  + arn                      = (known after apply)
  + bucket                   = "terraform-test-bucket-mikil"
  + bucket_domain_name      = (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                     = {
    + "Enviornment" = "Dev"
    + "Name"        = "My bucket"
  }
  + tags_all                = {
    + "Enviornment" = "Dev"
    + "Name"        = "My bucket"
  }
  + website_domain          = (known after apply)
  + website_endpoint        = (known after apply)

  + cors_rule {
    + allowed_headers = (known after apply)
    + allowed_methods = (known after apply)
    + allowed_origins = (known after apply)
    + expose_headers  = (known after apply)
    + max_age_seconds = (known after apply)
  }

  + grant {
    + id           = (known after apply)
    + permissions = (known after apply)
    + type         = (known after apply)
    + uri          = (known after apply)
  }
}
```

```
Windows PowerShell

    }
  }

+ versioning {
  + enabled      = (known after apply)
  + mfa_delete   = (known after apply)
}

+ website {
  + error_document      = (known after apply)
  + index_document      = (known after apply)
  + redirect_all_requests_to = (known after apply)
  + routing_rules        = (known after apply)
}
}

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

Warning: Argument is deprecated

with aws_s3_bucket.mikil,
on s3.tf line 3, in resource "aws_s3_bucket" "mikil":
  3:   acl = "public-read"

Use the aws_s3_bucket_acl resource instead

(and one more similar warning elsewhere)

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_s3_bucket.mikil: Creating...
aws_s3_bucket.mikil: Creation complete after 6s [id=terraform-test-bucket-mikil]

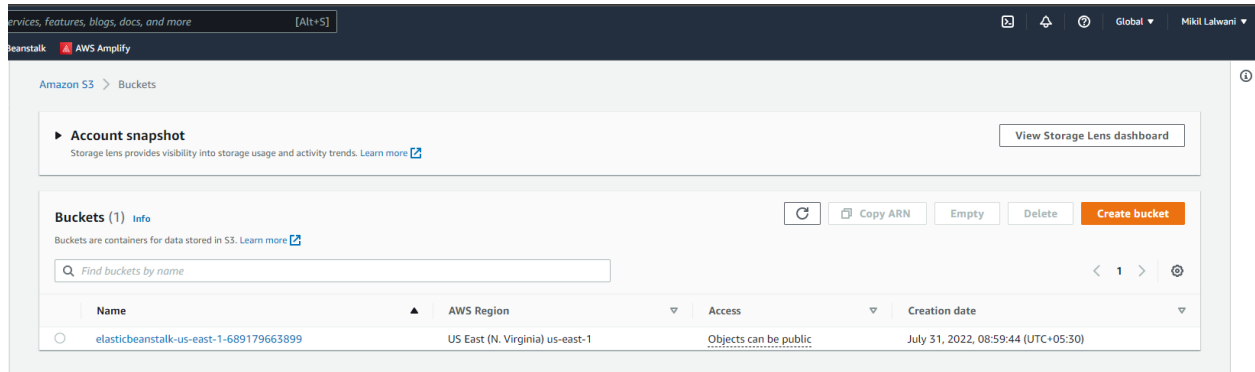
Warning: Argument is deprecated

with aws_s3_bucket.mikil,
on s3.tf line 3, in resource "aws_s3_bucket" "mikil":
  3:   acl = "public-read"

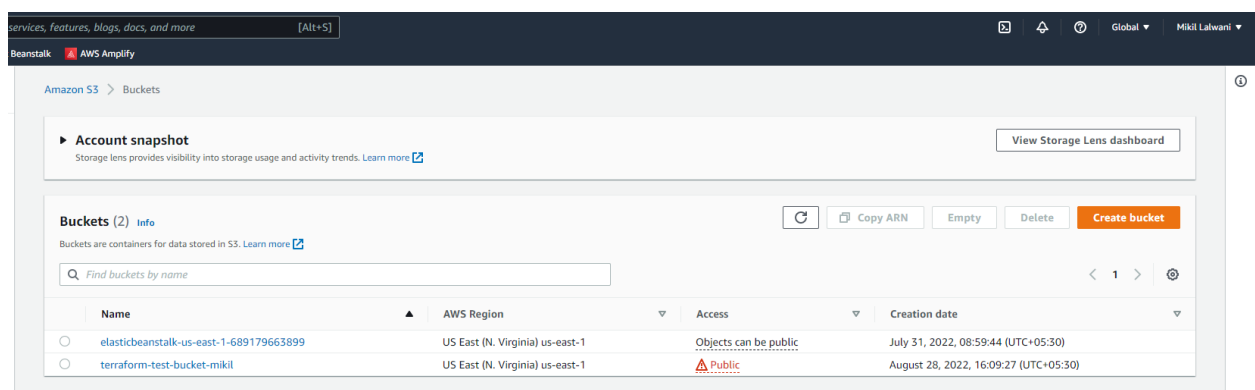
Use the aws_s3_bucket_acl resource instead

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform\S3>
```

AWS S3bucket dashboard, Before Executing Apply command:



AWS S3 Bucket dashboard, After Executing Apply step:



Step 6: Execute Terraform destroy to delete the configuration, which will automatically delete an EC2 instance

```

Windows PowerShell
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform\S3> terraform destroy
aws_s3_bucket.mikil: Refreshing state... [id=terraform-test-bucket-mikil]

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_s3_bucket.mikil will be destroyed
- resource "aws_s3_bucket" "mikil" {
  - acl = "public-read" -> null
  - arn = "arn:aws:s3:::terraform-test-bucket-mikil" -> null
  - bucket = "terraform-test-bucket-mikil" -> null
  - bucket_domain_name = "terraform-test-bucket-mikil.s3.amazonaws.com" -> null
  - bucket_regional_domain_name = "terraform-test-bucket-mikil.s3.amazonaws.com" -> null
  - force_destroy = false -> null
  - hosted_zone_id = "Z3AQBSTGFYJSTF" -> null
  - id = "terraform-test-bucket-mikil" -> null
  - object_lock_enabled = false -> null
  - region = "us-east-1" -> null
  - request_payer = "BucketOwner" -> null
  - tags = {}

```



```

- versioning {
  - enabled      = false -> null
  - mfa_delete = false -> null
}
}

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

Warning: Argument is deprecated

with aws_s3_bucket.mikil,
on s3.tf line 3, in resource "aws_s3_bucket" "mikil":
  3:   acl = "public-read"

Use the aws_s3_bucket_acl resource instead

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.

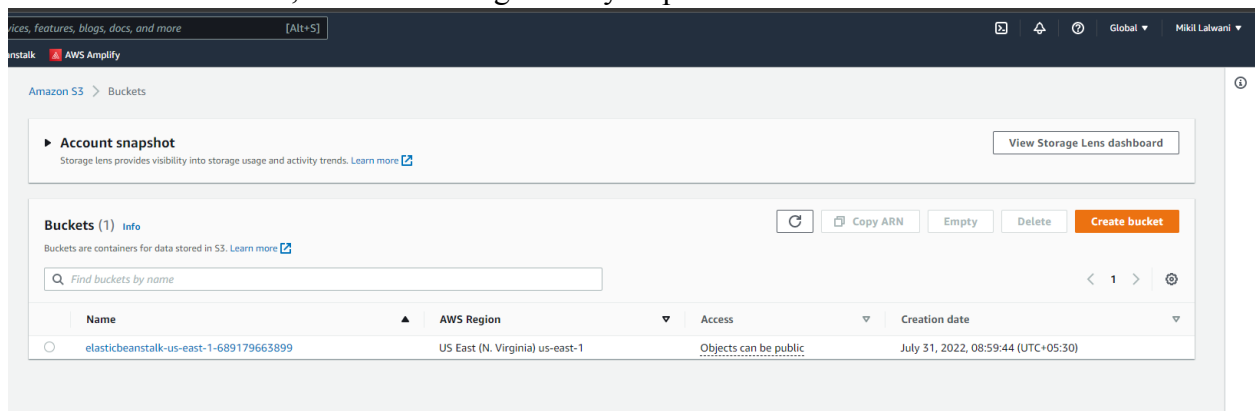
Enter a value: yes

aws_s3_bucket.mikil: Destroying... [id=terraform-test-bucket-mikil]
aws_s3_bucket.mikil: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
PS C:\Terraform\S3>

```

AWS EC2 dashboard, After Executing Destroy step:



Creating EC2 instance using terraform

Step 1: Create a Working directory called “Terraform_Scripts” in C:\ drive for storing all the Terraform scripts

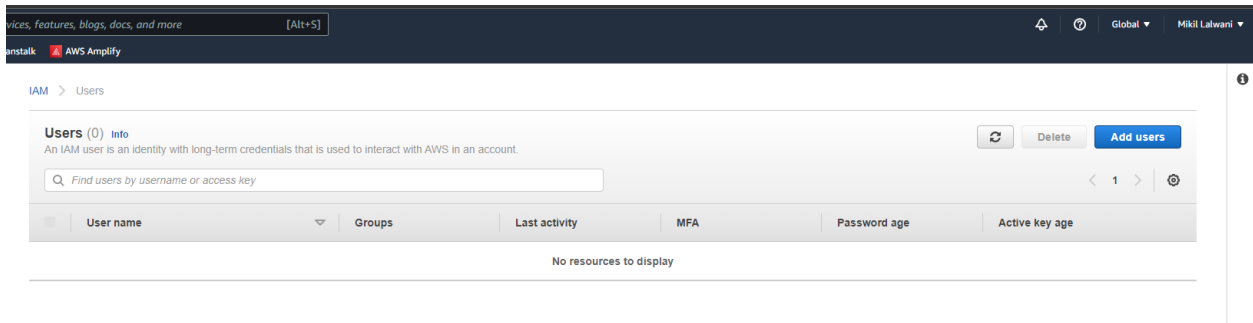
Step 2: Open Atom Editor and Open newly created folder “Terraform_Scripts” in it

Step 3: Create a new file called “EC2_on_Terraform” in it

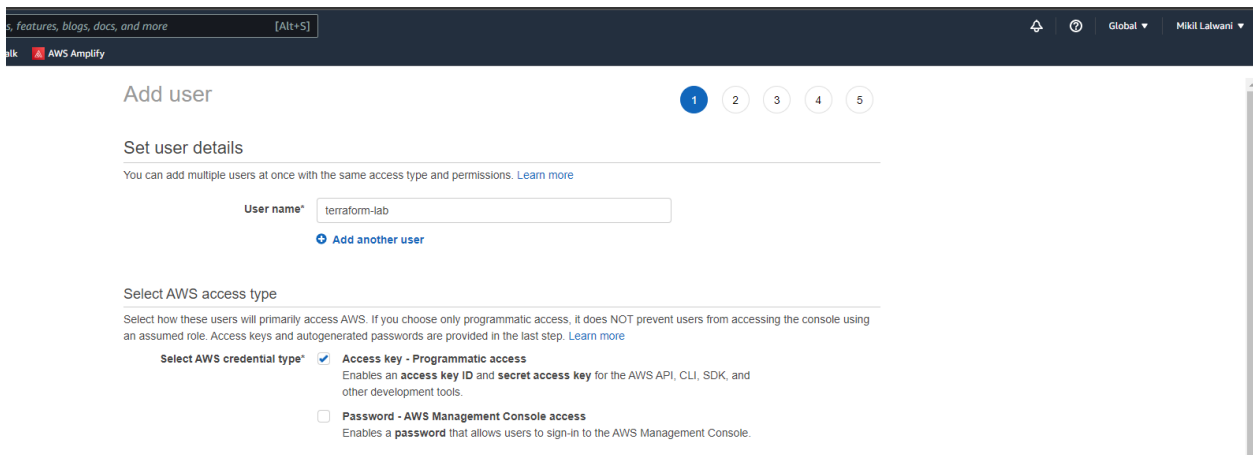
Step 4: Open AWS Console and Search for IAM to get Access Key ID and Secret Key

Open AWS Console: <https://aws.amazon.com/console>, Provide the credentials to Log in to Cloud Portal.

Now, search for IAM to get the Access Key ID and Secret Key



Create a new user Terraform_user and select option Programmatic Access then click on next for “Permissions”



Now, Create a group “Terraform_Group” and assign permission as “Administrator Access”

Now, Click on Create group followed by click on next for specifying a Tag which is Optional

Create group

Create a group and select the policies to be attached to the group. Using groups is a best-practice way to manage users' permissions by job functions, AWS service access, or your custom permissions. [Learn more](#)

Group name

terraform

Create policy

Refresh

Filter policies

Q Search

Showing 770 results

	Policy name	Type	Used as	Description
<input checked="" type="checkbox"/>	AdministratorAccess	Job function	None	Provides full access to AWS services and resources.
<input type="checkbox"/>	AdministratorAccess-Ampify	AWS managed	None	Grants account administrative permissions while explicitly allowing direct access to resources needed by Ampify applicat...
<input type="checkbox"/>	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None	Grants account administrative permissions. Explicitly allows developers and administrators to gain direct access to resou...
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusiness services
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness resources and access to related AWS Services
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS managed	None	Provide gateway execution access to AlexaForBusiness services
<input type="checkbox"/>	AlexaForBusinessLifesizeDelegatedAccessPolicy	AWS managed	None	Provide access to Lifesize AVS devices
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None	Provide access to Poly AVS devices
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	AWS managed	None	Provide read only access to AlexaForBusiness services
<input type="checkbox"/>	AmazonAPIGatewayAdministra	AWS managed	None	Provides full access to create/edit/delete APIs in Amazon API Gateway via the AWS Management Console

Cancel

Create group

o, blogs, docs, and more

[Alt+S]

Global

Mikil Lalwani

WS Amplify

Add user

12345

Set permissions

Add user to group

Copy permissions from existing user

Attach existing policies directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Add user to group

Create group

Refresh



Q Search

Showing 1 result

Group	Attached policies
<input checked="" type="checkbox"/> terraform	AdministratorAccess

es, blogs, docs, and more

[Alt+S]

GlobalMikil Lalwani


AWS Amplify

Add user

12345

Add tags (optional)

IAM tags are key-value pairs you can add to your user. Tags can include user information, such as an email address, or can be descriptive, such as a job title. You can use the tags to organize, track, or control access for this user. [Learn more](#)

Key	Value (optional)	Remove
<input type="text" value="Add new key"/>	<input type="text"/>	

You can add 50 more tags.

Cancel



Previous

Next: Review

Finally, Click on Next and Create user button to Create a New User.

ogs, docs, and more

[Alt+S]

GlobalMikil Lalwani

AWS Amplify

Add user

12345

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	terraform-lab
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The user shown above will be added to the following groups.

Type	Name
Group	terraform

Tags

No tags were added.

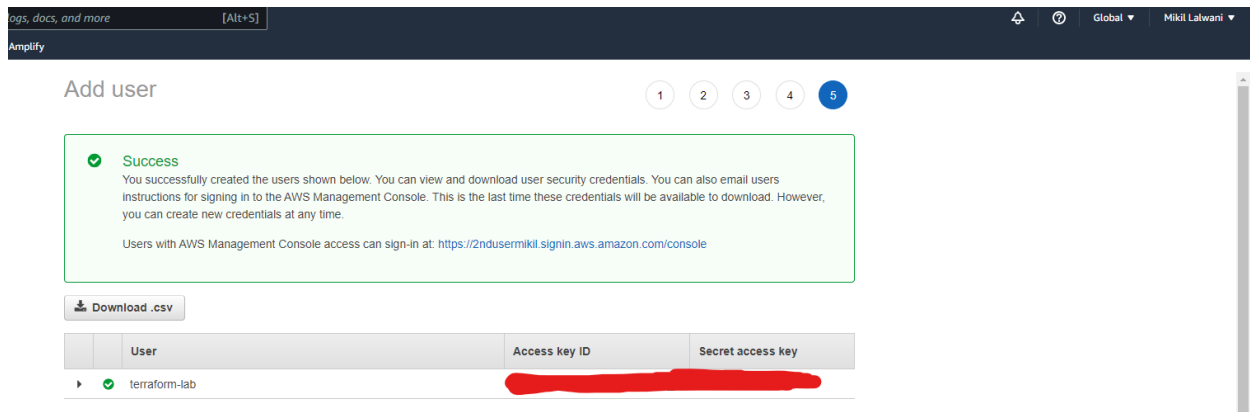
Cancel

Previous

Create user

Now, Copy the Access Key ID and Secret Access Key for using it in Terraform Script,or Download .CSV file and save it for using Access Key ID and Secret Access in Key for

Terraform Script



Step 5: Write Terraform Script in Atom for creating a EC2 instance using an automated Script

```
ec2.tf
provider "aws"{
  access_key = "AKIA2A5SPDYN54BRJVGI"
  secret_key = "7yQ0tOATfabQkKX5UTd8ldK3HAsbRPCW6OdqG"
  region = "us-east-1"
}

resource "aws_instance" "terraform-ec2"{
  ami = "ami-052efd3df9dad4825"
  instance_type = "t2.micro"
}
```

In this Script, access_key = "_____", secret_key = "_____" and ami = "_____" needs to be specified as per Operating system and EC2 instance id.

The access_key and secret_key can be used from previous step i.e. using CSV file or copied from user section of IAM service.

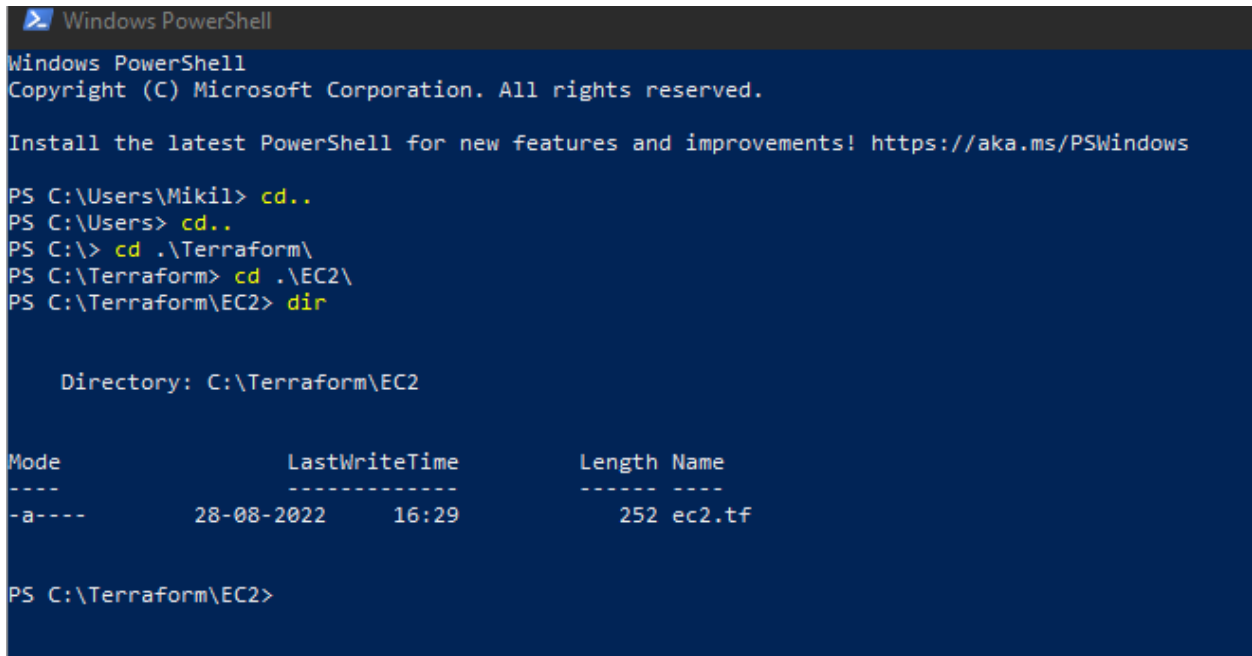
AMI stands for Amazon Machine Image which is the id of EC2 Virtual machine instance which can be copied from AWS EC2 service ami = "___"

To get AMI, First open AWS console and open EC2 service.

Click on Launch instance, which will show you list of Operating systems for which EC2 instance to be created. Copy the AMI id of an image for which instance to be created and

paste it into our terraform Script. [Note: Ami changes region to region, so see the region before copying AMI which is mentioned in the script, in our example it is us-east-1]

Step 7: Open Command Prompt and go to Terraform_Script directory where our .tf files are stored



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Mikil> cd..
PS C:\Users> cd..
PS C:\> cd .\Terraform\
PS C:\Terraform> cd .\EC2\
PS C:\Terraform\EC2> dir

        Directory: C:\Terraform\EC2

Mode                LastWriteTime         Length Name
----                -
-a----           28-08-2022   16:29             252 ec2.tf

PS C:\Terraform\EC2>
```

Step 8: Execute Terraform Init command to initialize the resources

```
Windows PowerShell
PS C:\Terraform\EC2> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.28.0...
- Installed hashicorp/aws v4.28.0 (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.
PS C:\Terraform\EC2>
```

Step 8: Execute Terraform plan to see the available resources

```
Windows PowerShell
Commands will detect it and remind you to do so if necessary.
PS C:\Terraform\EC2> 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.terraform-ec2 will be created
+ resource "aws_instance" "terraform-ec2" {
  + ami                        = "ami-052efd3df9dad4825"
  + 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)
  + 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_all                   = (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     = (known after apply)
}
```



```
Windows PowerShell

+ volume_type          = (known after apply)
}

+ enclave_options {
+ enabled = (known after apply)
}

+ ephemeral_block_device {
+ device_name = (known after apply)
+ no_device   = (known after apply)
+ virtual_name = (known after apply)
}

+ maintenance_options {
+ auto_recovery = (known after apply)
}

+ metadata_options {
+ http_endpoint          = (known after apply)
+ http_put_response_hop_limit = (known after apply)
+ http_tokens            = (known after apply)
+ instance_metadata_tags  = (known after apply)
}

+ network_interface {
+ delete_on_termination = (known after apply)
+ device_index          = (known after apply)
+ network_card_index    = (known after apply)
+ network_interface_id  = (known after apply)
}

+ private_dns_name_options {
+ enable_resource_name_dns_a_record  = (known after apply)
+ enable_resource_name_dns_aaaa_record = (known after apply)
+ hostname_type                      = (known after apply)
}

+ root_block_device {
+ delete_on_termination = (known after apply)
+ device_name           = (known after apply)
+ encrypted              = (known after apply)
+ iops                   = (known after apply)
+ kms_key_id             = (known after apply)
+ tags                   = (known after apply)
+ throughput             = (known after apply)
+ volume_id              = (known after apply)
+ volume_size            = (known after apply)
+ volume_type            = (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.
PS C:\Terraform\EC2>
```

Step 9: Execute Terraform apply to apply the configuration, which will automatically create an EC2 instance based on our configuration.

```
Windows PowerShell
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.
PS C:\Terraform\EC2> 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.terraform-ec2 will be created
+ resource "aws_instance" "terraform-ec2" {
  + ami                  = "ami-052efd3df9dad4825"
  + 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)
  + 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_all              = (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 = (known after apply)

  + capacity_reservation_specification {
    + capacity_reservation_preference = (known after apply)

    + capacity_reservation_target {
      + capacity_reservation_id = (known after apply)
      + capacity_reservation_resource_group_arn = (known after apply)
    }
  }
}
```

```
Windows PowerShell

+ virtual_name = (known after apply)
}

+ maintenance_options {
+   auto_recovery = (known after apply)
}

+ metadata_options {
+   http_endpoint           = (known after apply)
+   http_put_response_hop_limit = (known after apply)
+   http_tokens             = (known after apply)
+   instance_metadata_tags   = (known after apply)
}

+ network_interface {
+   delete_on_termination = (known after apply)
+   device_index           = (known after apply)
+   network_card_index     = (known after apply)
+   network_interface_id   = (known after apply)
}

+ private_dns_name_options {
+   enable_resource_name_dns_a_record   = (known after apply)
+   enable_resource_name_dns_aaaa_record = (known after apply)
+   hostname_type                       = (known after apply)
}

+ root_block_device {
+   delete_on_termination = (known after apply)
+   device_name            = (known after apply)
+   encrypted              = (known after apply)
+   iops                   = (known after apply)
+   kms_key_id             = (known after apply)
+   tags                   = (known after apply)
+   throughput             = (known after apply)
+   volume_id              = (known after apply)
+   volume_size            = (known after apply)
+   volume_type            = (known after apply)
}
}

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

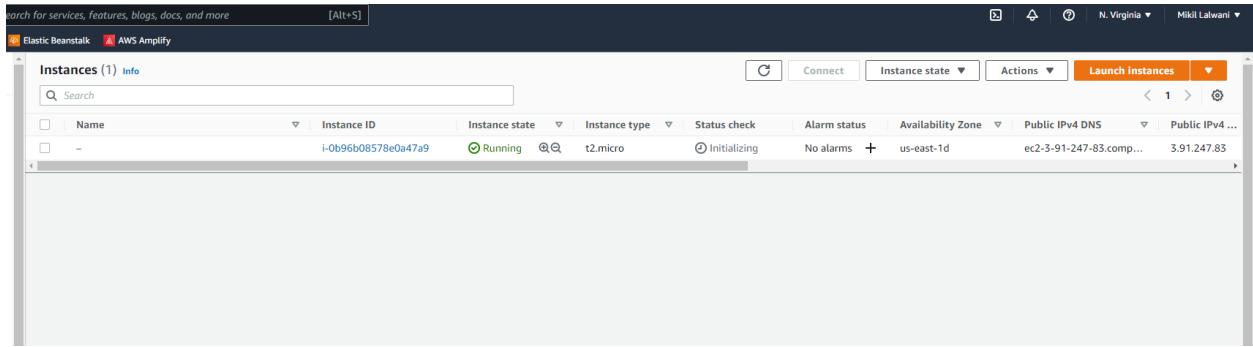
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

Enter a value: yes

aws_instance.terraform-ec2: Creating...
aws_instance.terraform-ec2: Still creating... [11s elapsed]
aws_instance.terraform-ec2: Still creating... [21s elapsed]
aws_instance.terraform-ec2: Still creating... [31s elapsed]
aws_instance.terraform-ec2: Still creating... [41s elapsed]
aws_instance.terraform-ec2: Creation complete after 46s [id=i-0b96b08578e0a47a9]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform\EC2>
```

AWS EC2 dashboard, After Executing Apply step:



Step 10: Execute Terraform destroy to delete the configuration, which will automatically delete an EC2 instance

```

Windows PowerShell
PS C:\Terraform\EC2> terraform destroy
aws_instance.terraform-ec2: Refreshing state... [id=i-0b96b08578e0a47a9]

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.terraform-ec2 will be destroyed
- resource "aws_instance" "terraform-ec2" {
  - ami                               = "ami-052efd3df9dad4825" -> null
  - arn                              = "arn:aws:ec2:us-east-1:689179663899:instance/i-0b96b08578e0a47a9" -> null
  - associate_public_ip_address      = true -> null
  - availability_zone                = "us-east-1d" -> null
  - cpu_core_count                   = 1 -> null
  - cpu_threads_per_core             = 1 -> null
  - disable_api_stop                 = false -> null
  - disable_api_termination          = false -> null
  - ebs_optimized                    = false -> null
  - get_password_data                = false -> null
  - hibernation                      = false -> null
  - id                               = "i-0b96b08578e0a47a9" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state                   = "running" -> null
  - instance_type                    = "t2.micro" -> null
  - ipv6_address_count               = 0 -> null
  - ipv6_addresses                   = [] -> null
  - monitoring                       = false -> null
  - primary_network_interface_id     = "eni-0d9b6a6b75fe3e37b" -> null
  - private_dns                      = "ip-172-31-84-94.ec2.internal" -> null
  - private_ip                       = "172.31.84.94" -> null
  - public_dns                       = "ec2-3-91-247-83.compute-1.amazonaws.com" -> null
  - public_ip                        = "3.91.247.83" -> null
  - secondary_private_ips            = [] -> null
  - security_groups                  = [
    - "default",
  ] -> null
  - source_dest_check                = true -> null
  - subnet_id                       = "subnet-08c7b710e3ed8910b" -> null
  - tags                             = {} -> null
  - tags_all                         = {} -> null
  - tenancy                          = "default" -> null
  - user_data_replace_on_change      = false -> null
  - vpc_security_group_ids           = [
    - "sg-054158c3a4acffe43",
  ] -> null

  - capacity_reservation_specification {
    - capacity_reservation_preference = "open" -> null
  }

  - credit_specification {
    - cpu_credits = "standard" -> null
  }

  - enclave_options {
    - enabled = false -> null
  }
}

```

Windows PowerShell

```
- capacity_reservation_specification {  
  - capacity_reservation_preference = "open" -> null  
}  
  
- credit_specification {  
  - cpu_credits = "standard" -> null  
}  
  
- enclave_options {  
  - enabled = false -> null  
}  
  
- maintenance_options {  
  - auto_recovery = "default" -> null  
}  
  
- metadata_options {  
  - http_endpoint = "enabled" -> null  
  - http_put_response_hop_limit = 1 -> null  
  - http_tokens = "optional" -> null  
  - instance_metadata_tags = "disabled" -> null  
}  
  
- private_dns_name_options {  
  - enable_resource_name_dns_a_record = false -> null  
  - enable_resource_name_dns_aaaa_record = false -> null  
  - hostname_type = "ip-name" -> null  
}  
  
- root_block_device {  
  - delete_on_termination = true -> null  
  - device_name = "/dev/sda1" -> null  
  - encrypted = false -> null  
  - iops = 100 -> null  
  - tags = {} -> null  
  - throughput = 0 -> null  
  - volume_id = "vol-046c8cc89df750f98" -> null  
  - volume_size = 8 -> null  
  - volume_type = "gp2" -> 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_instance.terraform-ec2: Destroying... [id=i-0b96b08578e0a47a9]  
aws_instance.terraform-ec2: Still destroying... [id=i-0b96b08578e0a47a9, 10s elapsed]  
aws_instance.terraform-ec2: Still destroying... [id=i-0b96b08578e0a47a9, 20s elapsed]  
aws_instance.terraform-ec2: Still destroying... [id=i-0b96b08578e0a47a9, 30s elapsed]  
aws_instance.terraform-ec2: Destruction complete after 32s
```

Destroy complete! Resources: 1 destroyed.

PS C:\Terraform\EC2>

AWS EC2 dashboard, After Executing Destroy step:

