

# **Hashicorp Terraform**

by Mr. Mahendran Selvakumar

## **EC2 Instance Management with Terraform: Creation, Scaling, and Deletion**

Name: Sooriya N

Class: III CSE C

Organized by KPR Institute of Engineering and Technology  
Department of Computer Science and Engineering

## Prerequisites:

### 1. Install Terraform:

- Download Terraform for Windows from the [Terraform official website](#).
- Add Terraform to your system's PATH.

### 2. Install AWS CLI:

- Download and install AWS CLI from the [official website](#).
- Configure AWS CLI by running: “aws configure”

Enter your AWS access key, secret key, default region, and output format.

### 3. Set up an IAM user in AWS with permissions for EC2.

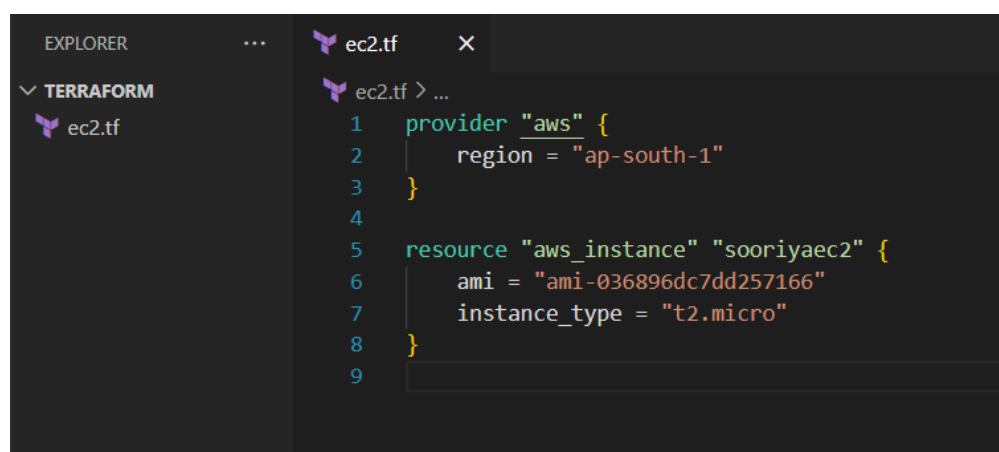
### 4. Install a Code Editor like Visual Studio Code to write your Terraform files.

## 1. Creating a Basic EC2 Instance

### 1. Prepare the Environment:

- Create a working directory (e.g., TERRAFORM) and navigate to it.
- Inside the directory, create a file named ec2.tf.

### 2. Write the Configuration:



```
1 provider "aws" {
2     region = "ap-south-1"
3 }
4
5 resource "aws_instance" "sooriyaec2" {
6     ami = "ami-036896dc7dd257166"
7     instance_type = "t2.micro"
8 }
9
```

The **PROVIDER BLOCK** specifies the cloud provider Terraform will interact with. In this case, it's AWS.

The **REGION ARGUMENT** determines the AWS region where the resources will be created (e.g., ap-south-1 for Mumbai). Replace this with your preferred region.

The **RESOURCE BLOCK** defines the resource to be created in AWS. In this case, it's an `aws_instance`, which represents an EC2 instance.

The name "sooriyaec2" is a label which I choose to identify this resource in the Terraform configuration.

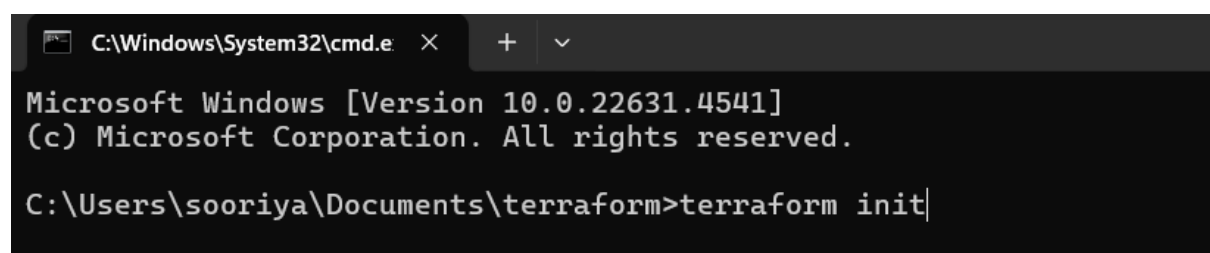
The **AMI ID** represents the operating system and configuration of the instance. Replace it with an AMI valid for your chosen region.

You can find the AMI ID in the **AWS Management Console > EC2 Dashboard > AMIs** under the **Images** section.

**INSTANCE TYPES** determine the computing power and memory of the instance. The t2.micro type is often used for free-tier-eligible instances.

### 3. Initialize Terraform:

Run the following command to download the necessary provider plugins:

A screenshot of a Windows Command Prompt window. The title bar shows the path 'C:\Windows\System32\cmd.e' and standard window controls. The command prompt displays the Windows version '10.0.22631.4541' and copyright information. The user is in the directory 'C:\Users\sooriya\Documents\terraform' and has entered the command 'terraform init'.

The **terraform init command** initializes your Terraform working directory, setting up the backend and preparing it for managing resources. It downloads the provider plugins, such as the AWS provider, required to interact with cloud services defined in your configuration. This step ensures that your environment is ready for further commands like `plan` and `apply`, making it an essential first step in any Terraform workflow.

```
C:\Users\sooriya\Documents\terraform>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.78.0...
- Installed hashicorp/aws v5.78.0 (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.

```
C:\Users\sooriya\Documents\terraform>
```

#### 4. Plan and Apply Changes:

“terraform plan” creates an execution plan, showing what actions Terraform will perform to achieve the desired state described in your configuration.

```
C:\Users\sooriya\Documents\terraform>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.sooriyac2 will be created
+ resource "aws_instance" "sooriyac2" {
  ami           = "ami-036896dc7dd257166"
  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_termination_protection = (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)
  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
  spot_instance_request_id = (known after apply)
  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 (known after apply)

  cpu_options (known after apply)

  ebs_block_device (known after apply)

  enclave_options (known after apply)

  ephemeral_block_device (known after apply)

  instance_market_options (known after apply)

  maintenance_options (known after apply)

  metadata_options (known after apply)

  network_interface (known after apply)

  private_dns_name_options (known after apply)

  root_block_device (known after apply)
}

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

“**terraform apply**” command applies the execution plan, creating or modifying resources in your cloud environment.

Terraform will prompt for confirmation (type yes), after which it provisions the resources.

```
C:\Users\sooriya\Documents\terraform>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.sooriyaec2 will be created
+ resource "aws_instance" "sooriyaec2" {
  + ami                    = "ami-036896dc7dd257166"
  + 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)
  + 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
  + spot_instance_request_id = (known after apply)
  + 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)
}
```

```
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.sooriyaec2: Creating...
aws_instance.sooriyaec2: Still creating... [10s elapsed]
aws_instance.sooriyaec2: Creation complete after 13s [id=i-02467e7a12abe71dd]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

C:\Users\sooriya\Documents\terraform>|
```

After running terraform apply, go to the **EC2 Dashboard** in the AWS Management Console and navigate to **Instances**. Here, you can find your newly created instance listed by its **Instance ID**. Click on the instance to view its details, such as public IP, instance type, and AMI ID, confirming it matches your configuration.

Instances (1) Info									
Find Instance by attribute or tag (case-sensitive)									
All states									
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
<input type="checkbox"/>		i-02467e7a12abe71dd	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-13-233-154-216.ap...	13.233

This EC2 instance can be used to run applications, host websites, store data, or serve as a backend for your cloud-based services. You can also integrate it with other AWS services to build a scalable and secure infrastructure.

## 2. Creating Multiple EC2 Instances

### 1. Edit the Configuration for Multiple Instances:

Use the count parameter in ec2.tf to create multiple instances:

```

EXPLORER
└─ TERRAFORM
  └─ .terraform
    └─ .terraform.lock.hcl
    └─ ec2.tf
    └─ terraform.tfstate
    └─ terraform.tfstate.backup

ec2.tf
1 provider "aws" {
2   region = "ap-south-1"
3 }
4
5 resource "aws_instance" "sooriyaec2" {
6   count = 2
7   ami = "ami-036896dc7dd257166"
8   instance_type = "t2.micro"
9 }
10

```

### 2. Apply the Changes:

Run “terraform apply” to apply the changes and provision the resources, such as creating multiple EC2 instances, as defined in the Terraform configuration.

### terraform plan

```

C:\Users\sooriya\Documents\terraform>terraform plan
aws_instance.sooriyaec2[0]: Refreshing state... [id=i-02467e7a12abe71dd]

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.sooriyaec2 has moved to aws_instance.sooriyaec2[0]
resource "aws_instance" "sooriyaec2" {
  id          = "i-02467e7a12abe71dd"
  tags        = {}
  # (39 unchanged attributes hidden)
  # (8 unchanged blocks hidden)
}

# aws_instance.sooriyaec2[1] will be created

```

## terraform apply

```
C:\Users\sooriya\Documents\terraform>terraform apply
aws_instance.sooriyaec2[0]: Refreshing state... [id=i-02467e7a12abe71dd]

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.sooriyaec2 has moved to aws_instance.sooriyaec2[0]
resource "aws_instance" "sooriyaec2" {
  id           = "i-02467e7a12abe71dd"
  tags        = {}
  # (39 unchanged attributes hidden)

  # (8 unchanged blocks hidden)
}

# aws_instance.sooriyaec2[1] will be created
```

```
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.sooriyaec2[1]: Creating...
aws_instance.sooriyaec2[1]: Still creating... [10s elapsed]
aws_instance.sooriyaec2[1]: Creation complete after 12s [id=i-0c65a028442e20fa2]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

C:\Users\sooriya\Documents\terraform>
```

After running terraform apply, verify the updates by checking the EC2 instance details in the EC2 Dashboard for the new instance type and Public IPv4 DNS.

Instances (2) Info

Last updated less than a minute ago

Connect

Instance state

Actions

Launch instances

Find Instance by attribute or tag (case-sensitive)

All states

< 1 >

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
<input type="checkbox"/>		i-02467e7a12abe71dd	Running	t2.micro	2/2 checks passed	View alarms +	ap-south-1b	ec2-13-203-105-61.ap-...	13.203
<input type="checkbox"/>		i-0c65a028442e20fa2	Running	t2.micro	Initializing	View alarms +	ap-south-1b	ec2-52-66-246-242.ap-...	52.66.:

## 3. Destroying AWS Resources

To delete all resources managed by Terraform, run: “terraform destroy”

```
C:\Users\sooriya\Documents\terraform>terraform destroy
aws_instance.sooriyaec2[1]: Refreshing state... [id=i-0c65a028442e20fa2]
aws_instance.sooriyaec2[0]: Refreshing state... [id=i-02467e7a12abe71dd]

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.sooriyaec2[0] will be destroyed
- resource "aws_instance" "sooriyaec2" {
  ami           = "ami-036896dc7dd257166" -> null
  arn           = "arn:aws:ec2:ap-south-1:886436965715:instance/i-02467e7a12abe71dd" -> null
  associate_public_ip_address = true -> null
  availability_zone = "ap-south-1b" -> null
}
```

```
# aws_instance.sooriyaec2[1] will be destroyed
- resource "aws_instance" "sooriyaec2" {
  ami           = "ami-036896dc7dd257166" -> null
  arn           = "arn:aws:ec2:ap-south-1:886436965715:instance/i-0c65a028442e20fa2" -> null
  associate_public_ip_address = true -> null
  availability_zone = "ap-south-1b" -> null
}
```

Terraform will prompt for confirmation (type yes), after which it will delete the specified resources.

```

Plan: 0 to add, 0 to change, 2 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.sooriyaec2[1]: Destroying... [id=i-0c65a028442e20fa2]
aws_instance.sooriyaec2[0]: Destroying... [id=i-02467e7a12abe71dd]
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 10s elapsed]
aws_instance.sooriyaec2[0]: Still destroying... [id=i-02467e7a12abe71dd, 10s elapsed]
aws_instance.sooriyaec2[0]: Still destroying... [id=i-02467e7a12abe71dd, 20s elapsed]
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 20s elapsed]
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 30s elapsed]
aws_instance.sooriyaec2[0]: Still destroying... [id=i-02467e7a12abe71dd, 30s elapsed]
aws_instance.sooriyaec2[0]: Destruction complete after 30s
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 40s elapsed]
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 50s elapsed]
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 1m0s elapsed]
aws_instance.sooriyaec2[1]: Still destroying... [id=i-0c65a028442e20fa2, 1m10s elapsed]
aws_instance.sooriyaec2[1]: Destruction complete after 1m11s

Destroy complete! Resources: 2 destroyed.

C:\Users\sooriya\Documents\terraform>

```

Go to the **EC2 Dashboard** and check the **Instances** section to ensure the terminated instances are no longer listed. The **instance status** will show as "terminated" for deleted instances, confirming their removal from your account.

Instances (2) <small>Info</small>									
Find Instance by attribute or tag (case-sensitive)				All states	<small>Last updated less than a minute ago</small> <span>Connect</span> <span>Instance state</span> <span>Actions</span> <span>Launch instances</span>				
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public
<input type="checkbox"/>		i-02467e7a12abe71dd	Terminated	t2.micro	–	<a href="#">View alarms +</a>	ap-south-1b	–	–
<input type="checkbox"/>		i-0c65a028442e20fa2	Terminated	t2.micro	–	<a href="#">View alarms +</a>	ap-south-1b	–	–

## Troubleshooting Tips

- **Instance Creation Failures:** Ensure that your AWS credentials and region settings are correct. If there's an issue with the AMI ID, verify that it's valid for your selected region.
- **Permissions Errors:** Ensure that the IAM user/role has the necessary permissions to manage EC2 instances.