

# TERRAFORM ASSIGNMENT 5

Name: Vikram

Assignment: Automated Apache Deployment with Terraform

## Problem Statement

Perform the following using Terraform:

1. Destroy previous deployments
2. Create a script to install Apache2
3. Run this script on a newly created EC2 instance
4. Store the public IP address of the instance in a local file

## Steps Performed

### Step 1 – Destroy Previous Infrastructure

`terraform destroy`

```
ubuntu@ip-172-31-4-222:~/terraform-vpc$ terraform destroy -auto-approve
aws_vpc.myvpc: Refreshing state... [id=vpc-0cce6baffa296b58a]
aws_internet_gateway.myigw: Refreshing state... [id=igw-044be3d237d4036b8]
aws_subnet.mysubnet: Refreshing state... [id=subnet-0e52bbebe53852a82]
aws_security_group.mymsg: Refreshing state... [id=sg-014ac4368c4a49629]
aws_route_table.myrt: Refreshing state... [id=rtb-007380ea785574bbb]
aws_instance.myec2: Refreshing state... [id=i-05afb9b0bc223c89a]
aws_route_table_association.rta: Refreshing state... [id=rtbassoc-032635badfb5925ca]

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.myec2 will be destroyed
- resource "aws_instance" "myec2" {
  - ami                  = "ami-0f5fcd9bd140e4ab7" -> null
  - arn                  = "arn:aws:ec2:us-east-2:062250062838:instance/i-05afb9b0bc223c89a" -> null
  - associate_public_ip_address = true -> null
  - availability_zone     = "us-east-2a" -> null
  - disable_api_stop      = false -> null
  - disable_api_termination = false -> null
  - ebs_optimized         = false -> null
  - force_destroy         = false -> null
  - get_password_data      = false -> null
  - hibernation           = false -> null
  - id                    = "i-05afb9b0bc223c89a" -> null
}
```

```
Plan: 0 to add, 0 to change, 7 to destroy.
aws_route_table_association.rta: Destroying... [id=rtbassoc-032635badfb5925ca]
aws_instance.myec2: Destroying... [id=i-05afb9b0bc223c89a]
aws_route_table_association.rta: Destruction complete after 0s
aws_route_table.myrt: Destroying... [id=rtb-007380ea785574bbb]
aws_route_table.myrt: Destruction complete after 1s
aws_internet_gateway.myigw: Destroying... [id=igw-044be3d237d4036b8]
aws_instance.myec2: Still destroying... [id=i-05afb9b0bc223c89a, 00m10s elapsed]
aws_internet_gateway.myigw: Still destroying... [id=igw-044be3d237d4036b8, 00m10s elapsed]
aws_instance.myec2: Still destroying... [id=i-05afb9b0bc223c89a, 00m20s elapsed]
aws_internet_gateway.myigw: Still destroying... [id=igw-044be3d237d4036b8, 00m20s elapsed]
aws_internet_gateway.myigw: Destruction complete after 27s
aws_instance.myec2: Still destroying... [id=i-05afb9b0bc223c89a, 00m30s elapsed]
aws_instance.myec2: Destruction complete after 40s
aws_security_group.mymsg: Destroying... [id=sg-014ac4368c4a49629]
aws_subnet.mysubnet: Destroying... [id=subnet-0e52bbebe53852a82]
aws_subnet.mysubnet: Destruction complete after 1s
aws_security_group.mymsg: Destruction complete after 1s
aws_vpc.myvpc: Destroying... [id=vpc-0cce6baffa296b58a]
aws_vpc.myvpc: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.
ubuntu@ip-172-31-4-222:~/terraform-vpc$
```

---

## Step 2 – Create Apache Installation Script

`nano install_apache.sh`

paste this:

`#!/bin/bash`

`sudo apt update -y`

`sudo apt install apache2 -y`

`sudo systemctl start apache2`

`sudo systemctl enable apache2`

save and exit

run

`chmod +x install_apache.sh`

```
ubuntu@ip-172-31-4-222:~$ nano install_apache.sh
```

```
GNU nano 7.2
```

```
#!/bin/bash
sudo apt update -y
sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
```

```
ubuntu@ip-172-31-4-222:~$ chmod +x install_apache.sh
ubuntu@ip-172-31-4-222:~$
```

---

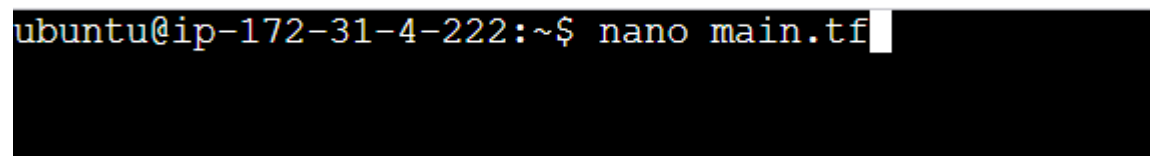
### Step 3 – Terraform Configuration

nano main.tf

paste this:

```
provider "aws" {  
    region = "us-east-2"  
}  
  
resource "aws_instance" "apache_server" {  
    ami          = "ami-0f5fcd140e4ab7"  
    instance_type = "t3.micro"  
  
    user_data = file("install_apache.sh")  
  
    tags = {  
        Name = "terraform-apache-server"  
    }  
}  
  
resource "local_file" "ip_output" {  
    content = aws_instance.apache_server.public_ip  
    filename = "server_ip.txt"  
}
```

Save and exit



```
ubuntu@ip-172-31-4-222:~$ nano main.tf
```

```
GNU nano 7.2
provider "aws" {
  region = "us-east-2"
}

resource "aws_instance" "apache_server" {
  ami          = "ami-0f5fcdafb140e4ab7"
  instance_type = "t3.micro"

  user_data = file("install_apache.sh")

  tags = {
    Name = "terraform-apache-server"
  }
}

resource "local_file" "ip_output" {
  content  = aws_instance.apache_server.public_ip
  filename = "server_ip.txt"
}
```

## Step 4 – Deploy Infrastructure

`terraform init`

`terraform apply`

```
ubuntu@ip-172-31-4-222:~$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/local...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/local v2.6.1...
- Installed hashicorp/local v2.6.1 (signed by HashiCorp)
- Installing hashicorp/aws v6.27.0...
- Installed hashicorp/aws v6.27.0 (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-172-31-4-222:~$
```

```
ubuntu@ip-172-31-4-222:~$ terraform validate
Success! The configuration is valid.

ubuntu@ip-172-31-4-222:~$
```

```
ubuntu@ip-172-31-4-222:~$ terraform apply -auto-approve
```

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

- + create

Terraform will perform the following actions:

```
# aws_instance.apache_server will be created
+ resource "aws_instance" "apache_server" {
  + ami                    = "ami-0f5fcd140e4ab7"
  + arn                    = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone       = (known after apply)
  + disable_api_stop        = (known after apply)
  + disable_api_termination = (known after apply)
  + ebs_optimized           = (known after apply)
  + enable_primary_ipv6     = (known after apply)
  + force_destroy           = false
  + 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           = "t3.micro"
```

```
+ file_permission = "0777"
+ filename        = "server_ip.txt"
+ id              = (known after apply)
}
```

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

```
aws_instance.apache_server: Creating...
aws_instance.apache_server: Still creating... [00m10s elapsed]
aws_instance.apache_server: Creation complete after 13s [id=i-0ca17011326e4b0ed]
local_file.ip_output: Creating...
local_file.ip_output: Creation complete after 0s [id=f7406bbe3e70a6398fe67ded3b9ba0b4c5020a0a]
```

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

```
ubuntu@ip-172-31-4-222:~$
```

i-0ca0c7a99fa1c8500 (terraform)

## Verification

Open in browser:

[http://<EC2\\_PUBLIC\\_IP>](http://<EC2_PUBLIC_IP>)

Verify IP stored locally:

[cat server\\_ip.txt](#)

The screenshot shows the AWS Management Console 'Instances' page. It lists two EC2 instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
terraform	i-0ca0c7a99fa1c8500	Running	t3.small	3/3 checks passed	View alarms +	us-east-2a	ec2-18-118
terraform-apache-server	i-0ca17011326e4b0ed	Running	t3.micro	3/3 checks passed	View alarms +	us-east-2c	ec2-3-138-1

The 'terraform-apache-server' instance is selected, and its details are shown below:

- Instance ID:** i-0ca17011326e4b0ed
- Public IPv4 address:** 3.138.189.21 | [open address](#)
- Private IPv4 addresses:** 172.31.34.144
- Instance state:** Running
- Public DNS:** [ec2-3-138-189-21.us-east-2.compute.amazonaws.com](#)



```
ubuntu@ip-172-31-4-222:~$ cat server_ip.txt
3.138.189.21ubuntu@ip-172-31-4-222:~$
```

## Conclusion

Successfully automated EC2 provisioning, Apache installation, and public IP storage using Terraform.