**Pratik Agrawal**

**500123601**

**Devops B2**

# Lab Exercise 4–Provisioning an EC2 Instance on AWS

**Prerequisites: Terraform Installed: Make sure you have Terraform installed on your machine. Follow the official installation guide if needed.**

AWS Credentials: Ensure you have AWS credentials (Access Key ID and Secret Access Key) configured. You can set them up using the AWS CLI or by setting environment variables.

**Exercise Steps:**

**Step 1: Create a New Directory:**

Create a new directory for your Terraform configuration:

**“Terraform-Demo”**

**Step 2: Create Terraform Configuration File (main.tf):**

Create a file named main.tf with the following content:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}
provider "aws" {
  region     = "ap-south-1"
```

```
  access_key = "your IAM access key"
  secret_key = "your secret access key"
}
```

C: > Users > prati > OneDrive > Documents > Devsecops lab > T-DEMO > main.tf

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.31.0"
    }
  }
}

provider "aws" {
  region     = "[redacted]"
  access_key = "[redacted]"
  secret_key = "AtYOfiE4ClNJLt80yGxo7yZxqgbhEZ4dIQZF9nOT"
}

```

This script defines an AWS provider and provisions an EC2 instance.

**Step 3: Initialize Terraform:**

Run the following command to initialize your Terraform working directory:

```
terraform init
```

```
PS C:\Users\prati\OneDrive\Documents\Devsecops lab\T-DEMO> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.31.0"...
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (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.
```

**Step 4: Create Terraform Configuration File for EC2 instance (instance.tf):**

Create a file named instnace.tf with the following content:

```
resource "aws_instance" "My-instance" {
      ami = "ami-03f4878755434977f"
      instance_type = "t2.micro"
   tags = {
    Name = "MY-EC2-Instnace"
   }
}
```

**Step 5: Review Plan:**

Run the following command to see what Terraform will do:

```
terraform plan
```

Review the plan to ensure it aligns with your expectations.


**Step 6: Apply Changes:**

Apply the changes to create the AWS resources:

```
terraform apply
```

Type yes when prompted.

```
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.My-instance: Creating...
aws_instance.My-instance: Still creating... [00m10s elapsed]
aws_instance.My-instance: Creation complete after 14s [id=i-0c67be31b3a648f22]

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

**Step 7: Verify Resources:**

After the terraform apply command completes, log in to your AWS Management Console and navigate to the EC2 dashboard. Verify that the EC2 instance has been created.

**Step 8: Cleanup Resources:**

When you are done experimenting, run the following command to destroy the created resources:

```
terraform 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.My-instance: Destroying... [id=i-0c67be31b3a648f22]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 00m10s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 00m20s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 00m30s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 00m40s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 00m50s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 01m00s elapsed]
aws_instance.My-instance: Still destroying... [id=i-0c67be31b3a648f22, 01m10s elapsed]
aws_instance.My-instance: Destruction complete after 1m12s
```

Type yes when prompted.

Notes:

Customize the instance.tf file to provision different AWS resources.

Explore the Terraform AWS provider documentation for additional AWS resources and configuration options.

Always be cautious when running terraform destroy to avoid accidental resource deletion.

This exercise provides a basic introduction to using Terraform with the AWS provider. Feel free to explore more complex Terraform configurations and resources based on your needs.