**Lesson 11 Lesson-End Project**

**Build Infrastructure**

**Project Agenda:** To build an AWS Infrastructure

**Description:** Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently. You can declare infrastructure components in configuration files that are then used by Terraform to provision, adjust, and tear down infrastructure in various cloud providers. In this project, you will provision an EC2 instance on Amazon Web Services (AWS). EC2 instances are virtual machines running on AWS and are a common component of many infrastructure projects.

**Tools required:** Terraform, AWS CLI

**Prerequisites:** You must have Terraform installed in the lab to proceed. You can refer to Lesson 10 Demo 1 to install and set up Terraform. Also, you must have AWS CLI installed to configure AWS.

**Expected Deliverables:**

Write configuration to describe infrastructure

Initialize the directory

Apply the configuration

**Steps to be followed:**

1. Configuring the AWS CLI from the terminal
2. Writing your first configuration to define a single AWS EC2 instance
3. Initializing the directory
4. Formatting and validating the configuration
5. Creating the infrastructure
6. Validating the creation of EC2 instance

**Step 1: Configuring the AWS CLI from the terminal**

1.1 Confirm the availability of Terraform by running the below command:

***Terraform -version***

1.2 AWS CLI should be installed in your system. However, you can run the below commands to install AWS CLI:

***pip install awscli***

***sudo apt-get update***

Text, letter

Description automatically generated

1.3 Execute the following command to set up your AWS credentials as environment variables:

***aws configure***



1.4 Follow the prompts to input your AWS Access Key ID and Secret Access Key

**Enter your AWS access key**:

**Enter your AWS secret access key**:

**Region**: us-east-1

**Default output format**: none

1.5 The configuration process stores your credentials in a file at **~/.aws/credentials.** Navigate to the directory and add the security token of your AWS lab as we are using an AWS user account

***cd .aws***

***vi credentials***

Graphical user interface, text, application, Word, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**To save and exit**: Press Esc to enter Command mode, and then type **:wq** to save and quit the file

1.6 Alternatively, you can also export the AWS credentials by using the below commands:

**Note**: You can get the values from AWS API Access under your AWS Account.

**export AWS\_ACCESS\_KEY\_ID**=<Your\_AWS\_Access\_Key>

**export AWS\_SECRET\_ACCESS\_KEY**=<Your\_AWS\_Secret\_Access\_Key>

**AWS\_SESSION\_TOKEN**=<Your\_AWS\_Security\_Token>

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, Word, email

Description automatically generated

**Step 2: Writing your first configuration to define a single AWS EC2 instance**

2.1 Create a directory for your configuration:

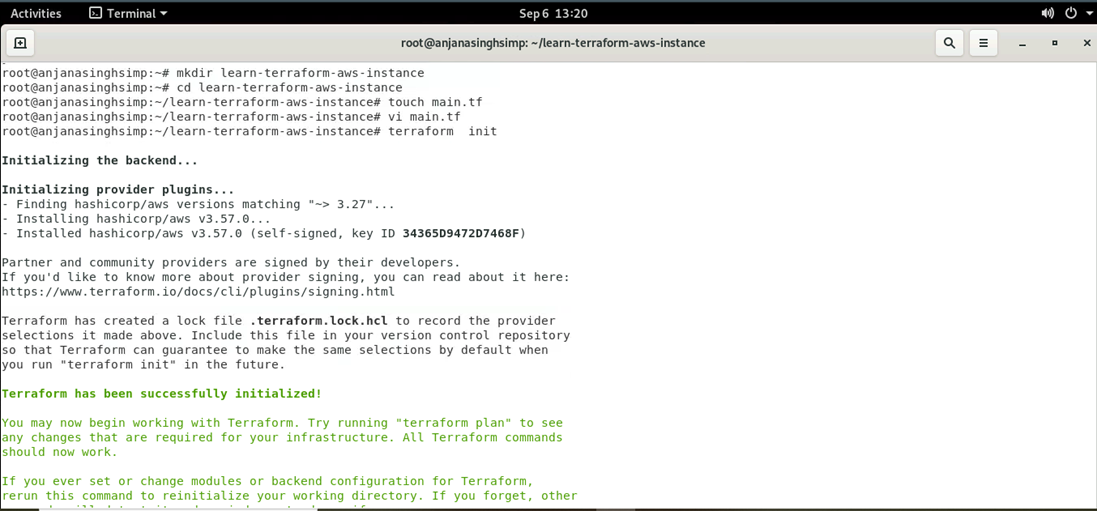
***mkdir learn-terraform-aws-instance***

2.2Change into the directory:

***cd learn-terraform-aws-instance***

2.3 Create a file to define your infrastructure:

***touch main.tf***

****

2.4 Open **main.tf** in your text editor, paste the configuration given below, and save the file:

***vi main.tf***

**Configuration:**

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 3.27"

}

}

required\_version = ">= 0.14.9"

}

provider "aws" {

profile = "default"

region = "us-east-1"

shared\_credentials\_file = "/root/.aws/creds"

}

resource "aws\_instance" "app\_server" {

ami = " ami-029bfac3973c1bda1"

instance\_type = "t2.micro"

tags = {

Name = "ExampleAppServerInstance"

}

}

Graphical user interface, text, application, email

Description automatically generated

**To save and exit**: Press Esc to enter Command mode, and then type **:wq** to save and quit the file

**Step 3: Initializing the directory**

3.1 Initialize the directory with ***terraform init***

**Note**: Initializing a configuration directory, downloads and installs the providers defined in the configuration, which in this case is the AWS provider

Graphical user interface, text, application

Description automatically generated

**Step 4: Formatting and validating the configuration**

4.1 We recommend using consistent formatting in all your configuration files. Thefollowing command automatically updates configurations in the current directory for readability and consistency:

***Terraform fmt***

4.2 The below command makes sure the configuration is syntactically valid and internally consistent:

***terraform validate***

**Note**: If the configuration provided above is valid, terraform will return a success message.

**Step 5: Creating the infrastructure**

* 1. Run ***terraform plan*** command to prepare your configuration

Graphical user interface

Description automatically generated

* 1. Apply the configuration now with the ***terraform apply*** command

Terraform will print output like what is shown below. We have truncated some of the output to save space.

Table

Description automatically generated

* 1. Terraform will now pause and wait for your approval before proceeding. The plan is acceptable, so type yes at the confirmation prompt to proceed

Executing the plan will take a few minutes since Terraform waits for the EC2 instance to become available.

Table

Description automatically generated

**Step 6: Validating the creation of EC2 instance**

6.1 Login to your AWS Console having the same credentials you must configure here to build the infrastructure:

Graphical user interface, text, application

Description automatically generated

* 1. You will see a new instance running. You can validate the instance by the instance name or ID.

Graphical user interface, text, application, Word

Description automatically generated