Lesson 06 Lesson-End Project

Provisioning an EC2 Instance Using Terraform

Project Agenda: To automate the provisioning of an AWS EC2 instance using Terraform for centralized infrastructure management and enhanced access control

Description: You are a DevOps Engineer in an IT company. Your company wants you to automate an infrastructure to manage all the running services from one place and provide an access model control based on the organization, teams, and users. This will help the team in collaboration and centralize documentation.

Tools required: Terraform and AWS CLI

Prerequisites: You must have Terraform and AWS CLI installed in your local system.

Expected Deliverables: Configure AWS in your local system and launch an EC2 instance using Terraform.

Steps to be followed:

1. Configure your machine
2. Create a Terraform scripts
3. Run your Terraform scripts

NOTE: Ensure you have security credentials to authenticate your AWS Account

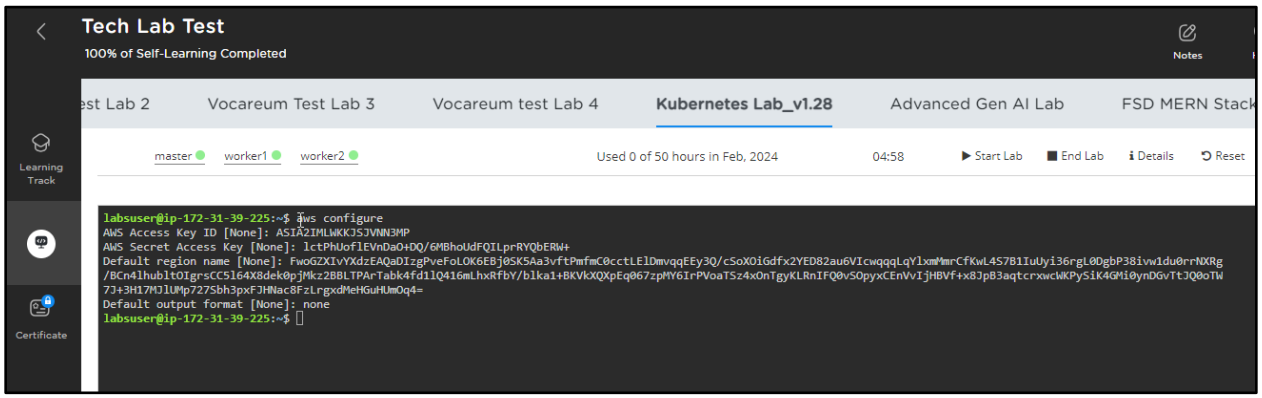
Step 1: Configure your machine

1.1 Execute the following command to set up your AWS credentials as environment Variables; the playbook needs these at runtime

aws configure

Enter your AWS access key
Enter your AWS secret access key

Region: us-east-1
Default output format: none



Note: The configuration process stores your credentials in a file at ~/.aws/credentials on MacOS and Linux or %UserProfile%\.aws\credentials on Windows.

1.2 Run the **terraform --version** command for version verification

```
labsuser@ip-172-31-39-225:~$ terraform --version
Terraform v1.7.2
on linux_amd64
```

1.3 Run the following commands in the given sequence to set up your AWS CLI:
pip install awscli
sudo apt-get update

```
labsuser@ip-172-31-39-225:~$ pip install awscli
sudo apt-get update
Command 'pip' not found, but can be installed with:
sudo apt install python3-pip
Hit:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://apt.releases.hashicorp.com jammy InRelease
Ign:6 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:7 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:9 https://dl.google.com/linux/chrome/deb stable InRelease
Hit:8 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  InRelease
Hit:10 https://ppa.launchpadcontent.net/mozillateam/ppa/ubuntu jammy InRelease
```

Step 2: Create a Terraform scripts

2.1 Run the following commands:

Create a new folder:

mkdir ec2demo

Move to the folder:

cd ec2demo

Create cred.tf file:

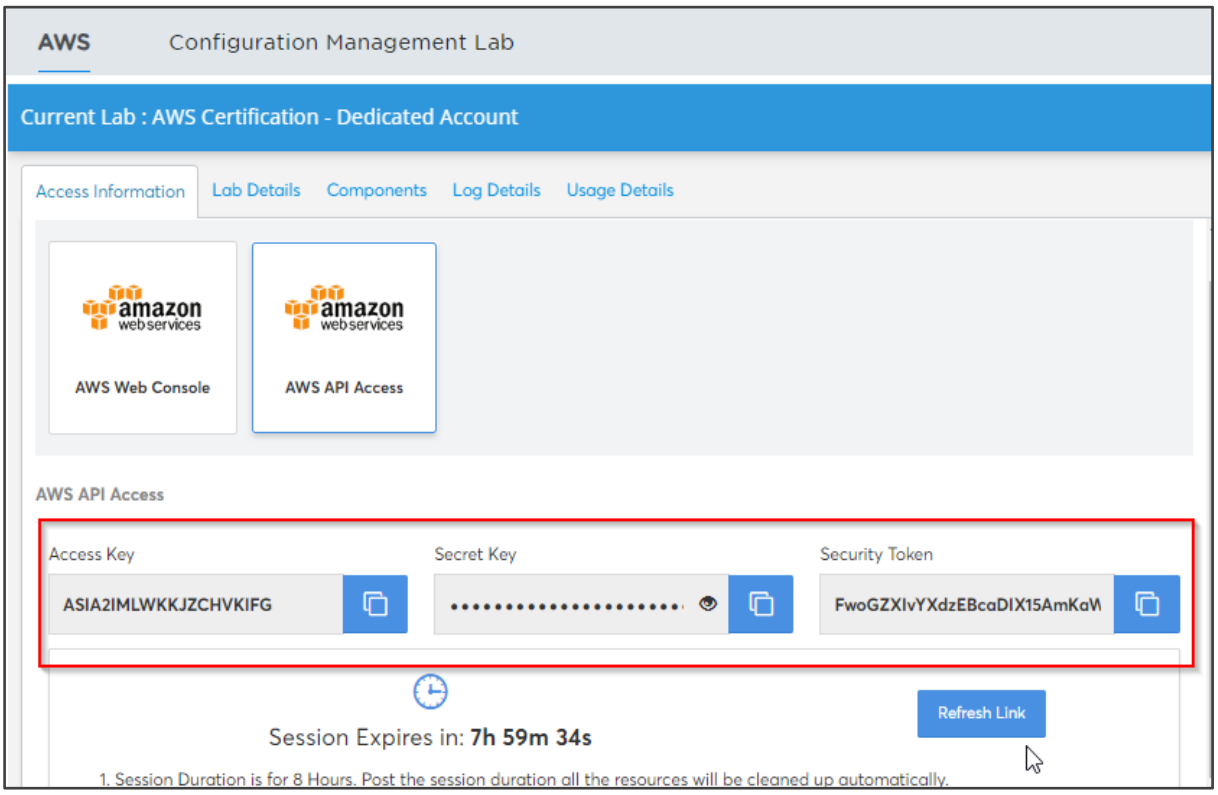
nano cred.tf

Enter the following code inside the **nano cred.tf** file and provide all the details:

```
provider "aws" {
access_key = "AWS_LAB_ACCESS_KEY"
secret_key = "AWS_LAB_SECRET_KEY"
token = “AWS_LAB_SECURITY_TOKEN”
region = "us-east-1"
}
```

```
labsuser@ip-172-31-39-225:~$ mkdir ec2demo
labsuser@ip-172-31-39-225:~$ cd ec2demo
labsuser@ip-172-31-39-225:~/ec2demo$ nano cred.tf
```

Note: Use the credentials given in your AWS lab and replace the placeholders of Access Key, Security Key, and Security Token with the corresponding values



2.2 Create a file to define your infrastructure and add the following code inside it:
Create a file using the below command:
nano ec2.tf

```
labsuser@ip-172-31-39-225:~/ec2demo$ nano ec2.tf
```

Enter the below code inside the file **ec2.tf**:

```
resource "aws_instance" "example" {
ami           = "ami-2757f631"
instance_type = "t2.micro"
}
```

```
  GNU nano 4.8                                                    ec2.tf
resource "aws_instance" "example" {

  ami           = "ami-2757f631"

  instance_type = "t2.micro"

}
```

Step 3: Run your Terraform scripts

3.1 Enter the **terraform init** command to initialize the directory

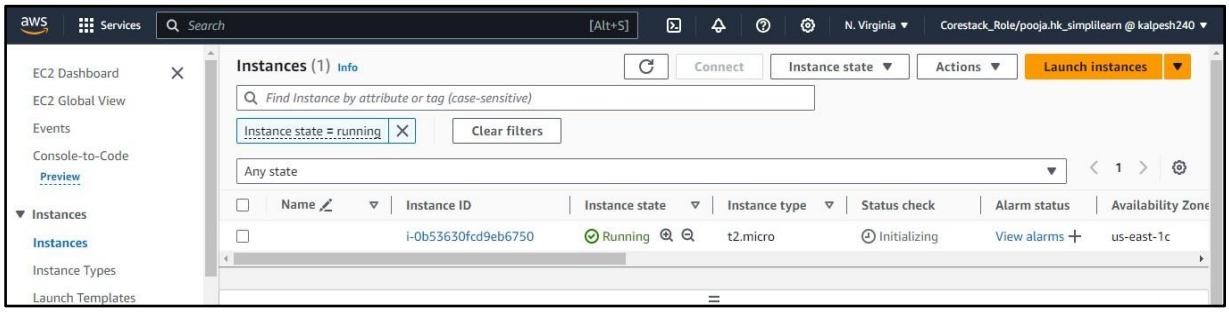
```
labsuser@ip-172-31-39-225:~/ec2demo$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.35.0...
- Installed hashicorp/aws v5.35.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record
selections it made above. Include this file in your version cont
so that Terraform can guarantee to make the same selections by d
you run "terraform init" in the future.

Terraform has been successfully initialized!
```

3.2 Use the following command to apply the configuration:
terraform apply

```
labsuser@ip-172-31-39-225:~/ec2demo$ 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.example will be created
  + resource "aws_instance" "example" {
      + ami                                  = "ami-2757f631"
      + arn                                  = (known after apply)
```

Note: Terraform will pause and wait for your approval before proceeding.
Enter a value: **Yes**

3.3 Verify the creation of an EC2 instance in your AWS management console



By following these steps, you have successfully automated the provisioning of an AWS EC2 instance using Terraform to facilitate centralized infrastructure management and streamlined access control.