Module-6: Infrastructure Automation using Terraform

Demo Document - 1

© Brain4ce Education Solutions Pvt. Ltd.

Creating and destroying an EC2 instance using Terraform

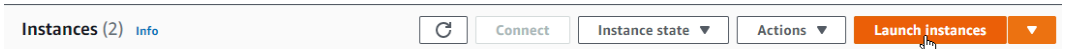
1. Create a new text file using the editor of your choice with .tf extension

   Syntax: vi filename.tf
2. Edit the file with the following configuration

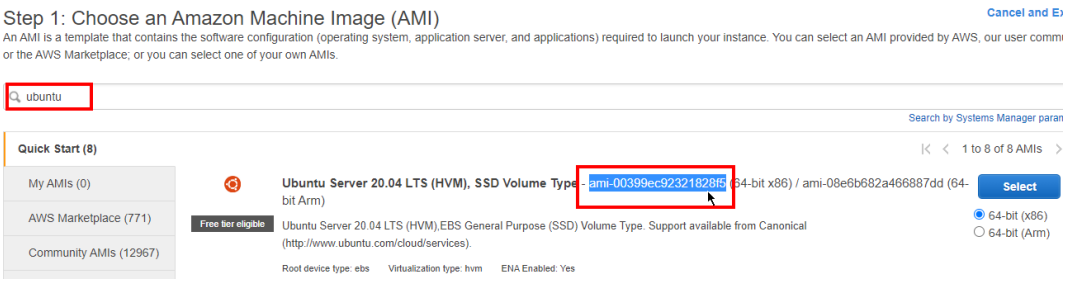
```
# Configuring provider
provider "aws" {
  region = "us-east-2"
  access_key = "access-key"
  secret_key = secret-key"
}

# Deploying an ec2 instance
resource "aws_instance" "Terraform-instance1" {
  ami           = "ami-<id>"
  instance_type = "t2.micro"
  tags = {
    Name = "terra-instance1"
  }
}
```

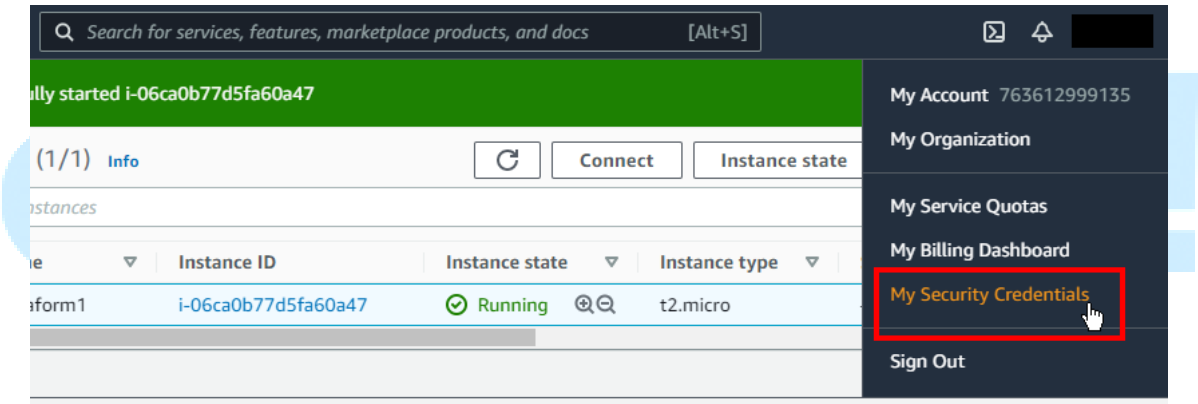
3. To get the valid ami for your instance, the easy way is to click the launch instance button on you ec2 console



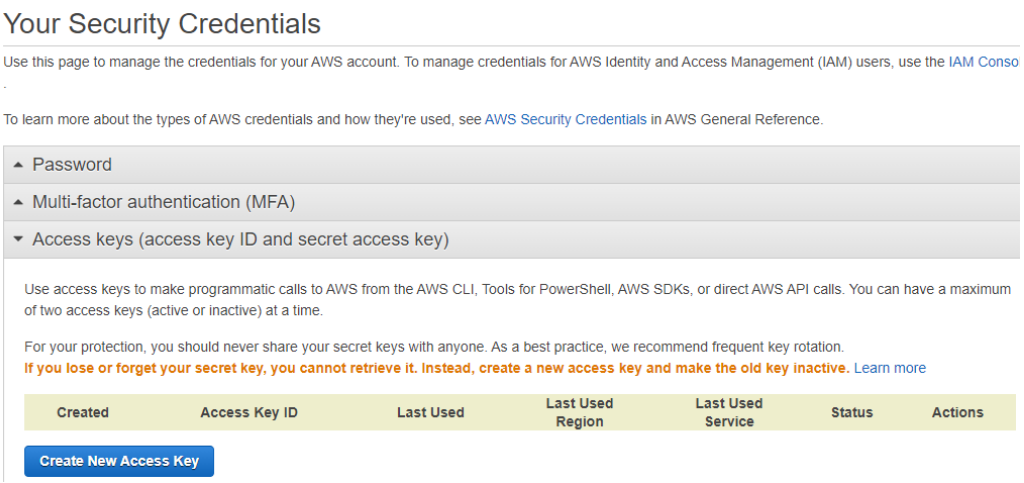
Then search the image you want and copy the ami value given for it



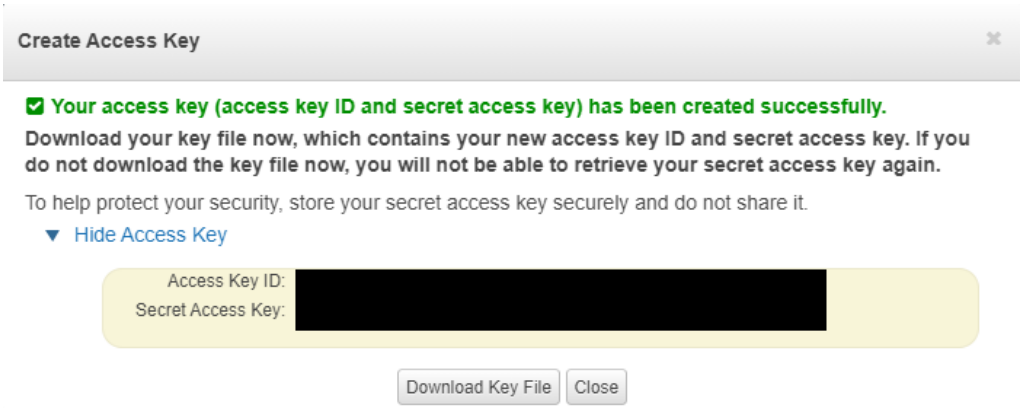
4. Now, to get the access and the secret key for terraform to access your aws console Click on My Security Credentials on your AWS console under your username



5. Click on Access Keys and then click on Create New Access Key



6. A pop-up window like below will appear. Copy and paste the access key and secret key values in your terraform configuration



7. After adding the keys, your configuration is ready. Now we can initialize terraform using the init command

   Syntax: terraform init

```
edureka@kmaster:~/Desktop/terraformDemos$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
^[[A- Using previously-installed hashicorp/aws v3.46.0

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.
```

8. Now run the plan command to check the changes the configuration is going to make

   Syntax: terraform plan

```
edureka@kmaster:~/Desktop/terraformDemos$ 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.Terraform-instance1 will be created
  + resource "aws_instance" "Terraform-instance1" {
      + ami                                  = "ami-0d8d212151031f51c"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
```

9. After verifying the changes, you can go ahead and apply them using the apply command

   Syntax: terraform apply

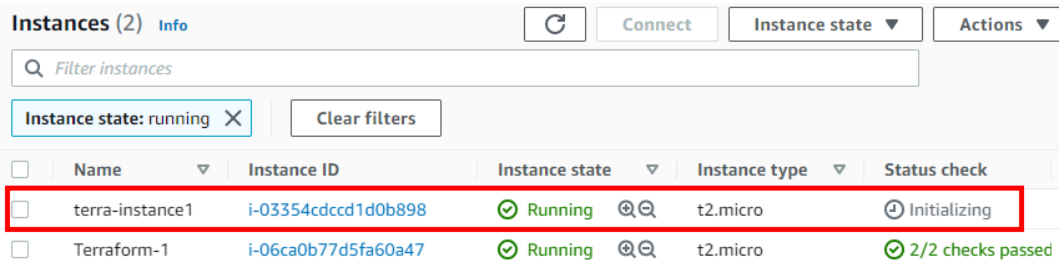
```
edureka@kmaster:~/Desktop/terraformDemos$ terraform apply

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.Terraform-instance1 will be created
  + resource "aws_instance" "Terraform-instance1" {
      + ami                                  = "ami-0d8d212151031f51c"
      + 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)
      + get_password_data                    = false
```

10. We can verify if the instance has been provisioned on the aws console



11. You can destroy the instance using the following command
    Syntax: terraform destroy

```
edureka@kmaster:~/Desktop/terraformDemos$ terraform destroy
aws_instance.Terraform-instance1: Refreshing state... [id=i-08f691a8913f42764]

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.Terraform-instance1 will be destroyed
  - resource "aws_instance" "Terraform-instance1" {
```

edureka!