

Module-11: Provisioning Infrastructure using Terraform Part-I

Demo Document - 3

edureka!

edureka!

© Brain4ce Education Solutions Pvt. Ltd.

DEMO-3: Writing and Running a Terraform Configuration

Information about the providers and their resources can be found here:

<https://registry.terraform.io/browse/providers>

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

Syntax: vi filename.tf

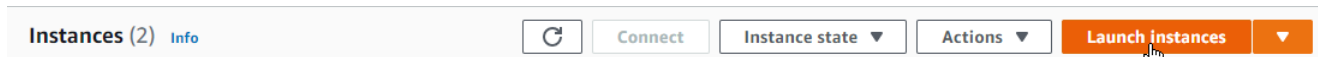
2. Edit the file with the following configuration

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      # optional
      version = "~> 3.0"
    }
  }
}

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

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

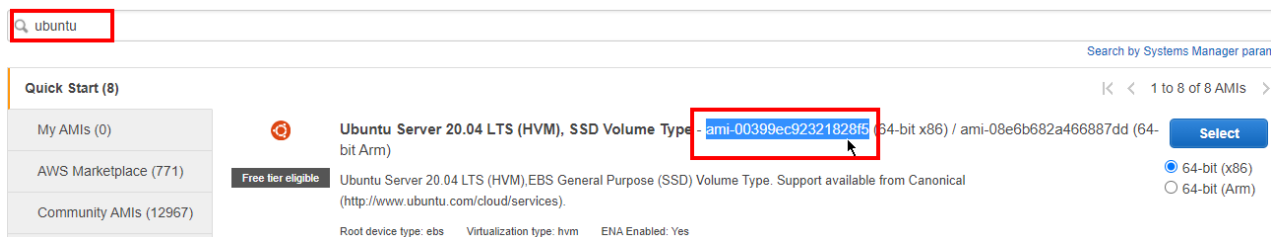
- To get the valid ami for your instance, the easy way is to click the launch instance button on you ec2 console



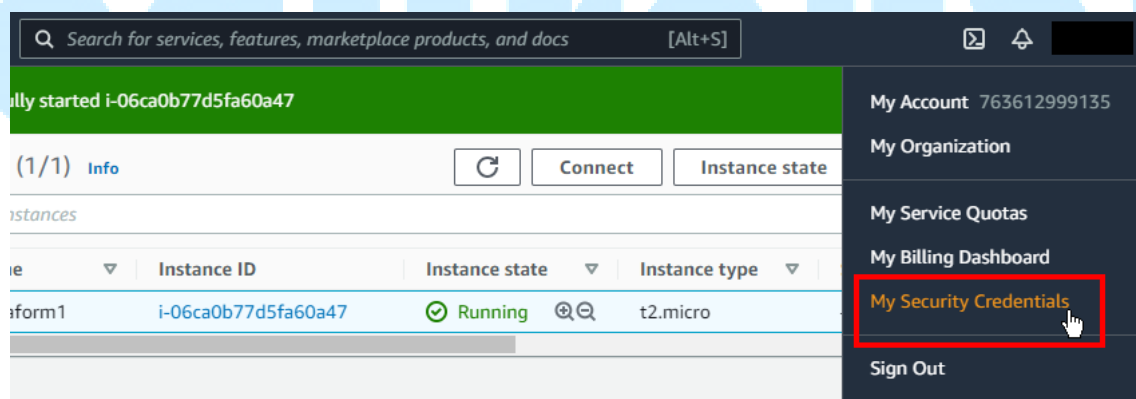
Then search the image you want and copy the ami value given for it

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace, or you can select one of your own AMIs.



- 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



- Click on Access Keys and then click on Create New Access Key

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the [IAM Console](#).

To learn more about the types of AWS credentials and how they're used, see [AWS Security Credentials](#) in AWS General Reference.

▲ Password

▲ Multi-factor authentication (MFA)

▼ Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.
If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. [Learn more](#)

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
---------	---------------	-----------	------------------	-------------------	--------	---------

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

Create Access Key

✔ Your access key (access key ID and secret access key) has been created successfully.

Download your key file now, which contains your new access key ID and secret access key. If you do not download the key file now, you will not be able to retrieve your secret access key again.

To help protect your security, store your secret access key securely and do not share it.

▼ Hide Access Key

Access Key ID:

Secret Access Key:

Download Key File

Close

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

Syntax: terraform init

```
ubuntu@ip-172-31-19-127:~/terra$ terraform init
```

```
Initializing the backend...
```

```
Initializing provider plugins...
```

```
- Finding hashicorp/aws versions matching "~> 3.0"...\n- Installing hashicorp/aws v3.39.0...\n- Installed hashicorp/aws v3.39.0 (signed by HashiCorp)
```

```
Terraform has created a lock file .terraform.lock.hcl to record the provider\nselections it made above. Include this file in your version control repository\nso that Terraform can guarantee to make the same selections by default when\nyou run "terraform init" in the future.
```

```
Terraform has been successfully initialized!
```

```
You may now begin working with Terraform. Try running "terraform plan" to see\nany changes that are required for your infrastructure. All Terraform commands\nshould now work.
```

```
If you ever set or change modules or backend configuration for Terraform,\nre-run this command to reinitialize your working directory. If you forget, other\ncommands 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

```
ubuntu@ip-172-31-19-127:~/terra$ terraform plan
```

```
Terraform used the selected providers to generate the following execution plan.\nfollowing symbols:
```

```
+ create
```

```
Terraform will perform the following actions:
```

```
# aws_instance.Terraform-instance1 will be created\n+ resource "aws_instance" "Terraform-instance1" {\n  + ami                        = "ami-00399ec92321828f5"\n  + arn                      = (known after apply)\n  + associate_public_ip_address = (known after apply)\n  + availability_zone         = (known after apply)\n  + cpu_core_count           = (known after apply)\n  + cpu_threads_per_core     = (known after apply)\n  + get_password_data         = false\n  + host_id                  = (known after apply)\n  + id                       = (known after apply)\n  + instance_initiated_shutdown_behavior = (known after apply)\n  + instance_state            = (known after apply)\n  + instance_type             = "t2.micro"
```

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

Syntax: terraform apply

```
Plan: 1 to add, 0 to change, 0 to destroy.

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.Terraform-instancetype: Creating...
aws_instance.Terraform-instancetype: Still creating... [10s elapsed]
aws_instance.Terraform-instancetype: Still creating... [20s elapsed]
aws_instance.Terraform-instancetype: Creation complete after 21s [id=i-03354cdccd1d0b898]
```

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

Instances (2) Info						Refresh	Connect	Instance state ▼	Actions ▼
<input type="text" value="Filter instances"/>									
Instance state: running X		Clear filters							
<input type="checkbox"/>	Name ▼	Instance ID	Instance state ▼	Instance type ▼	Status check				
<input type="checkbox"/>	terra-instance1	i-03354cdccd1d0b898	✓ Running 🔍	t2.micro	🕒 Initializing				
<input type="checkbox"/>	Terraform-1	i-06ca0b77d5fa60a47	✓ Running 🔍	t2.micro	✓ 2/2 checks passed				