**Pre-Requisites**

1. Create IAM user with programmatic access and administrative access.
2. Note down the shared and secret keys
3. Note down the AMI ID of Amazon Linux AMI from Mumbai region

**Terraform Installation**

***Note****: You can install terraform either on windows or Linux. Below is the installation link.*

*Please enable the Powershell terminal within VS code*

https://www.terraform.io/downloads

1. Let’s download for Linux.
2. Select Linux tab -> Amazon Linux
3. Run 3 command provided in documentation.

e.g sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install terraform

1. Verify installation

Below are steps for windows

1. Download the windows zip from below location
2. <https://developer.hashicorp.com/terraform/downloads>
3. Unzip it
4. Set the environment variable in windows where terraform binary is store.

**Validation Steps:**

1. Run – terraform –version

**Note** – For below labs, please download sample TF code from below URL

https://ethans-tf-labs.s3.ap-south-1.amazonaws.com/ethans-tf-labs.zip

**Lab 1 – Create AWS Instance Resource (single main.tf)**

1. Make sure you download sample TF code from above mentioned URL
2. Modify main.tf with your IAM user keys and AMI id from ap-south-1 region
3. Run – terraform init
4. Run – terraform plan
5. Run – terraform apply
6. Verify instance gets create successfully
7. Run – terraform destroy
8. Verify instance gets terminated successfully

**Validation Steps:**

1. Init/plan/apply works without an error
2. Login to AWS console and verify instance is running/terminated

**Lab 2 – Create AWS Instance Resource (With different tf files)**

1. Make sure you download sample TF code from above mentioned URL
2. Refer lab 2
3. Modify dev.tfvars for AMI ID
4. Instead of passing AWS credentials hard coded, set as environment variables

e.g. In Linux, you can use below format

export AWS\_ACCESS\_KEY\_ID="accesskey"

export AWS\_SECRET\_ACCESS\_KEY="secretkey"

e.g. In VS Code terminal, on powershell terminal you can use below format

$ENV:AWS\_ACCESS\_KEY\_ID=’<Your Access Key>’

$ENV:AWS\_SECRET\_ACCESS\_KEY=’<Your Secret Key>’

1. Run – terraform init --var-file=dev.tfvars
2. Run – terraform plan --var-file=dev.tfvars
3. Run - terraform apply --var-file=dev.tfvars
4. Verify instance get created
5. Run - terraform destroy --var-file=dev.tfvars

**Validation Steps:**

1. Init/plan/apply works without an error
2. Login to AWS console and verify instance is running/terminated

**Lab 3 – Create AWS Instance Resource (With Modules)**

1. Make sure you download sample TF code from above mentioned URL
2. Refer lab 3
3. Modify dev.tfvars for AMI ID
4. We have already set the env variables as part of above lab 2.
5. Run – terraform init --var-file=dev.tfvars
6. Run – terraform plan --var-file=dev.tfvars
7. Run - terraform apply --var-file=dev.tfvars
8. Verify instance get created
9. Run - terraform destroy --var-file=dev.tfvars

**Validation Steps:**

1. Init/plan/apply works without an error
2. Login to AWS console and verify instance is running/terminated

**Lab 4 – Remote State Management (Remote Backend)**

1. Make sure you download sample TF code from above mentioned URL
2. Refer lab 4
3. Create simple bucket in ap-south-1 region with all default inputs. E.g tf-remote-backend-mumbai
4. Create dynamo DB table with same name e.g. tf-remote-backend-mumbai with primary key as LockID and type string
5. Provide above two inputs in backend.tf file
6. Modify dev.tfvars, with your AMI ID
7. AWS credentials are already set as part of lab2
8. Run – terraform init --var-file=dev.tfvars
9. Run – terraform plan --var-file=dev.tfvars
10. Run - terraform apply --var-file=dev.tfvars
11. Verify instance get created
12. Run - terraform destroy --var-file=dev.tfvars

**Validation Steps:**

1. Init/plan/apply works without an error
2. Login to AWS console and verify instance is running/terminated
3. Verify terraform.tfstate file is created in AWS S3 bucket

**Lab 5 – Terraform Provisioner**

1. Make sure you download sample TF code from above mentioned URL
2. Refer lab 5
3. Use backend configuration created in above lab4.
4. Provide above two inputs in backend.tf file
5. Modify dev.tfvars, with your AMI ID
6. Modify dev.tfvars for key\_name. e.g. aws-demo-lin-mumbai. (Do not mention key name with .pem or .ppk)
7. Modify module/ec2 – main.tf with correct private\_key
8. Copy your EC2 key pair in current folder of lab5
9. AWS credentials are already set as part of lab2
10. Run – terraform init --var-file=dev.tfvars
11. Run – terraform plan --var-file=dev.tfvars
12. Run - terraform apply --var-file=dev.tfvars
13. Verify instance get created
14. Run - terraform destroy --var-file=dev.tfvars

**Validation Steps:**

1. Init/plan/apply works without an error
2. Login to AWS console and verify instance is running/terminated
3. Verify terraform.tfstate file is created in AWS S3 bucket
4. Verify local file gets generated i.e. private\_ips.txt
5. Verify php gets installed on target EC2 server
6. Verify script gets copied and folder gets created in /tmp directory

**Lab 6 – Using Terraform Registry Modules**

1. First use below code to create s3 bucket, change your values accordingly.

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 4.0"

}

}

}

provider "aws" {

region = "ap-south-1"

access\_key = "Your access\_key"

secret\_key = "Your secret\_key"

}

resource "aws\_s3\_bucket" "example" {

bucket = "my-tf-test-bucket"

tags = {

Name = "My bucket"

Environment = "Dev"

}

}

1. Perform init/plan/apply to verify bucket gets created successfully.
2. Modify the code as below to use terraform registry modules
3. terraform {
4. required\_providers {
5. aws = {
6. source  = "hashicorp/aws"
7. version = "~> 4.0"
8. }
9. }
10. }
11. provider "aws" {
12. region     = "ap-south-1"
13. access\_key = "Access Key"
14. secret\_key = "Secret key"
15. }
16. module "s3\_bucket" {
17. source = "terraform-aws-modules/s3-bucket/aws"
18. bucket = "bucket-name"
19. acl    = "private"
20. control\_object\_ownership = true
21. object\_ownership         = "ObjectWriter"
22. versioning = {
23. enabled = true
24. }
25. }

Perform init/plan/apply to verify bucket gets created

**Lab 7 – Terraform Cloud (Basic workspace configuration)**

Please make sure, you have done registration and configuration with

<https://app.terraform.io/signup/account>

For configuration, refer other lab guide

1. Go to Github and create repository e.g. terraformcloud-aws-ec2 and create below 3 files with provided contents. Be ready with provided terraform sample files.
2. Login to your terraform cloud workspace e.g. <https://app.terraform.io/app/demo-devopscloud/workspaces>
3. Go to workspaces -> Create New One
4. Select version control workflow
5. Select Github
6. Select your repository e.g. terraformcloud-aws-ec2
7. Click on Create workspace
8. On next screen, enter AMI ID and instance\_type as t2.micro
9. Click on start new plan -> Start run
10. Run will fail as we haven’t mentioned AWS authentication keys
11. Go to workspace -> variables section and add below environment variables
    1. AWS\_ACCESS\_KEY\_ID =<Your access key> Type Env variable and mark as sensitive
    2. AWS\_SECRET\_ACCESS\_KEY =<Your Secret key>. Type env and sensitive
12. Then start plan and apply. It should create one instance