Lab 2 - Manage AWS Infrastructure with Terraform

Contents

Lab Evidence

Your report on this lab will be the basis for your grade. The purpose of the report is to demonstrate to your instructor that you have completed the lab successfully and understood the material. Please include the following screenshots as part of your evidence.

- Terraform and AWS CLI command line outputs for version commands
- `webserver-Public-URL` and the URL assigned from the end of the Terraform apply output. Please do not include all of the output from apply, just the `webserver-Public-URL` value
- The Cloud Computing Course webpage in your browser, including the address bar showing the URL
- Provisioned Infrastructure as viewed through your AWS account, including the public IP of your webserver, under Instances.

A portion of the grade will be allocated to the quality of the report. The report should be well formatted, concise and to the point.

Learning Goals

Demonstrate and execute steps for managing AWS Infrastructure with Terraform.

Pre-requisite

- Created AWS free tier accounts account (Check provided AWS instructions in Moodle - 02.Create AWS Free Tier Account.docx).https://github.com/

Introduction

You study during this week how Infrastructure as Code (IaC) works and how you can use Terraform as IaC tool. Let's work through a scenario to manage AWS infrastructure using Terraform.

Demonstrate Steps for standing up AWS Infrastructure using Terraform

To accomplish this, you are going to follow these steps:

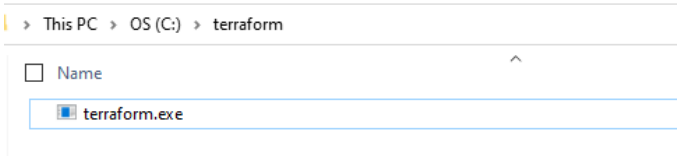
- Step 1: Download and configure Terraform on your local machine
- Step 2: AWS Free Tier Account
- Step 3: Install and Configure Visual Studio Code
- Step 4: Source Control Management
- Step 5: Terraform Structure
- Step 6: Terraform Main
- Step 7: Terraform Virtual Private Cloud (VPC) Module
- Step 8: Terraform Compute (EC2) Module
- Step 9: Terraform Commands

Step 1: Download and configure Terraform on your local machine

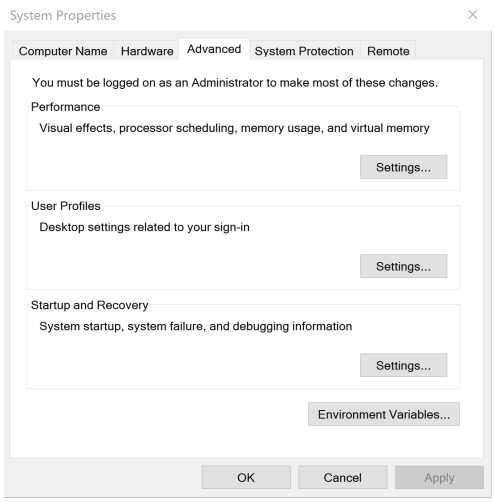
You will use Terraform for multiple labs, so having a functioning Terraform environment will be critical to your success with these labs.

Windows OS

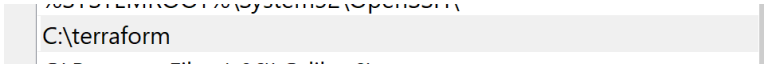
- Download Terraform link
- Configure Terraform on your local machine for your OS link
- For Windows OS, please follow these instructions
    - Extract the zip file content to c:\terraform



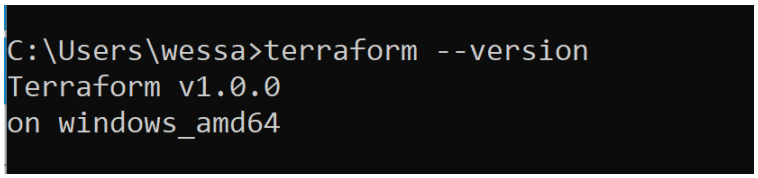
    - Go to System Properties -> Environment Variables



    - In **System Variables** edit the **Path**
    - Select **New** and add **c:\terraform**



    - Now open Command Prompt (CMD) and type **terraform --version**
    - You should have the latest Terraform version 1.0.0



Mac OS

For Mac OS, if you are running a newer Mac Book Pro M1 chip, you may have some challenges due to availability of a supported plugin from Hashicorp. As new versions of plugins are

released, we have seen that a working Mac version lag behind availability of versions for other operating systems.
Please follow these instructions for Mac OS and if this is not working for you, please provide details of the problem (including chip details) on the Lab 2 discussion forum.

- /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
- export PATH=/opt/homebrew/bin:$PATH
- brew tap hashicorp/tap
- brew install hashicorp/tap/terraform
- brew update
- brew upgrade hashicorp/tap/terraform
- terraform --version

- If you have problems with the version of Terraform installed on your mac, you will see a message like this when you run terraform init

```
Error: Incompatible provider version

Provider registry.terraform.io/hashicorp/template v2.2.0 does not have a
package available for your current platform, darwin_arm64.

Provider releases are separate from Terraform CLI releases, so not all
providers are available for all platforms. Other versions of this provider
may have different platforms supported.
```

To resolve this issue, you need to remove the latest version installed by brew and move to a specific version, for which there is mac support.

This should just be a matter of getting a zip file from Hashicorp releases, as follows:

1. Navigate to https://releases.hashicorp.com/terraform.
2. Click on terraform_0.15.5
3. Download the linked file terraform_0.15.5_darwin_amd64.zip to a folder
4. Type the command unzip terraform_0.15.5_darwin_amd64.zip

   ```
   Archive:  terraform_0.15.5_darwin_amd64.zip
   replace terraform? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
         inflating: terraform
   ```

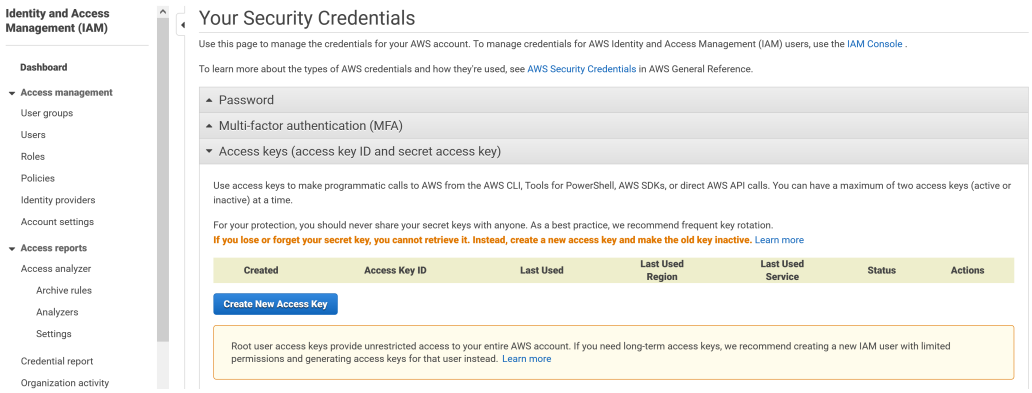
5. Type the command sudo mv terraform /usr/local/bin/
6. Verify the terraform version terraform –version. You should see Terraform v0.15.5
7. If not, then you may need to do more to remove the latest version, including
   a. brew uninstall terraform

b. brew remove terraform

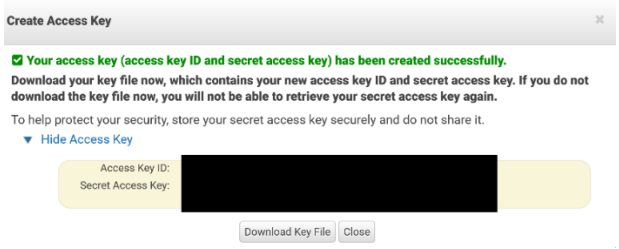
c. which terraform – to determine if terraform has remained, in which case you may need to find the terraform install and remove manually.

Step 2: AWS Free Tier Account

- Create AWS Free Tier Account link (Follow the instructions provided in Moodle)
- Create IAM Access Keys
  - Login to your AWS Account
  - After you login go to the link
  - Select Access Keys



  - Click Create New Access. Keep these keys in a secure place (key vault if possible), never share it with anyone, and don't lose it.



- Download and install AWS CLI
  - AWS CLI for your OS link
  - Here is a demonstration for AWS CLI installation on Windows OS
    - Download the latest version of the AWS CLI link
    - Install the AWS CLI MSI by following the instructions
    - After installation completion open Command Prompt (CMD) and type **aws --version**
    - You should have the latest AWS CLI version, greater than 2.2.8

```
C:\Users\wessa>aws --version
aws-cli/2.2.8 Python/3.8.8 Windows/10 exe/AMD64 prompt/off

C:\Users\wessa>
```

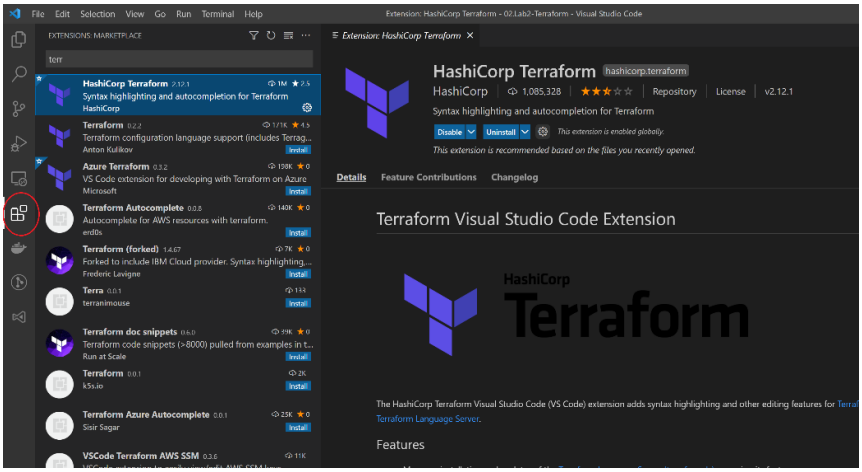
  - Here is a demonstration for AWS CLI installation on Mac OS

        - curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
        - After installation completion open terminal and type **aws --version**
        - You should have the latest AWS CLI version, greater than 2.2.8
    -
    -
    -
    -
    -
    - Configure the Access Keys
        - Open Command Prompt (CMD) and type **aws configure**
        - Copy and paste the keys you saved from the previous steps
            - Access Key ID: blablablaxsekfnbhwfbkfbuers
            - Secret Access Key: blablablau3brfhvrjelngfjlvwehdbvgeddvhkb
            - Default region name: Press Enter (Leave it empty, you could specify a region if you would like)
            - Default output format: Press Enter (Leave it empty, you could specify an output format if you would like)

Step 3: Install and Configure Visual Studio Code

I prefer using Visual Studio Code (It is free and powerful), but you could use any other editor such as Atom, Notepad ++, Notepad, Vim, etc.

- Download Visual Studio Code for your OS link
- Quick overview of Visual Studio Code
    - Why VS Code? link
    - Get started with Visual Studio Code link
    - Extensions for Visual Studio Code link
    - Personalize Visual Studio Code link
- Once you download and installed Visual Studio Code.
    - Click on the marketplace extensions on left side circled in RED
    - Search for HashiCorp Terraform extension
    - Install it (Feel free to select and install other extensions if you would likes)



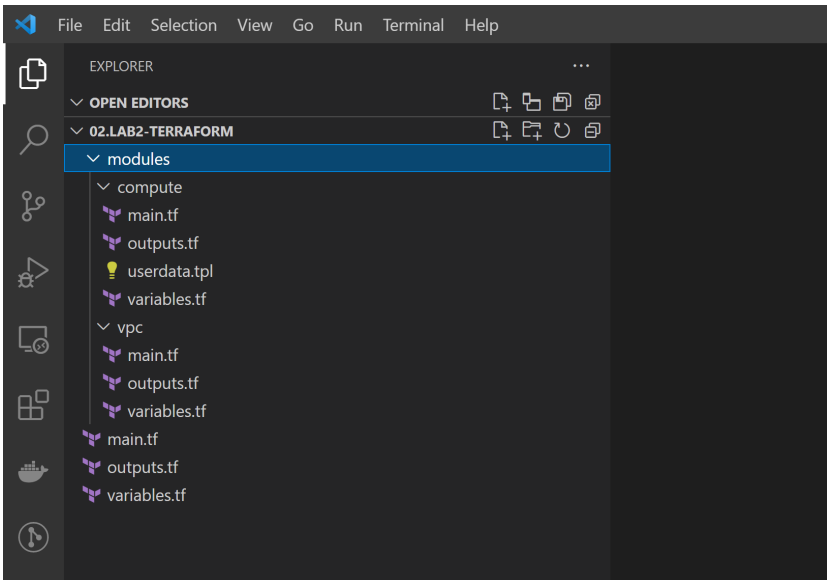
- Now Open Lab3-Terraform in Visual Studio Code
  - Download Terraform Code from Moodle
  - Extract it to your local machine
  - Open it in the Visual Studio Code File -> Open Folder -> locate the extract folder on your local machine
  - You could also Save Workspace to add folders and projects later (Optional) File -> Save Workspace as & File -> Add Folder to WorkSpace

Step 4: Source Control Management

- It is highly recommended to version your source code using your Git repository
- Experiment with the code provided and enhance it following all the best practices you studied this week

Step 5: Terraform Structure

- As you can see the code is modularized.



- You have main, outputs, and variables which calls and interact with vpc and compute modules
- As you will see the outputs of the vpc module will be used by the compute module
- Terraform executes code in files with .tf extension
- Terraform looks for providers in the Terraform providers registry link

Step 6: Terraform Main

- Define terraform required providers in our case AWS and Terraform required version >= 0.15.5 (Remember you installed V1.0.0. Terraform can break between different versions)
  - Understand Terraform Code
    - **provider** is a reserved key word
    - **aws** is the provider's name
    - Inside the curly braces is the configuration parameters

main.tf

main.tf > terraform > required_version

```
#-----main.tf-----
#====================
terraform {
  required_providers {
    aws = {
      version = "~> 3.44.0"
    }
  }
  required_version = ">= 0.15.5"
}

provider "aws" {
  region = var.region
}

```

- The specified aws region in that case it is a variable provided in variables.tf with default region us-east-1.

main.tf | variables.tf

variables.tf > variable "region"

```
#-----variables.tf-----
#========================
variable "region" {
  type    = string
  default = "us-east-1"
}
```

- This is the call for module vpc for deploying the AWS networking resources located under modules/vpc

```
#Deploy Networking Resources
#============================
module "vpc" {
  source = "./modules/vpc"
}
```

- This is the call for module compute for deploying AWS EC2 instances resources located under module/compute.

```
#Deploy Compute Resources
#============================
module "compute" {
  source          = "./modules/compute"
  subnets         = module.vpc.public_subnets
  security_group  = module.vpc.public_sg
  subnet_ips      = module.vpc.subnet_ips
}
```

These are the variables defined in variables.tf under compute module

It depends on the outputs of vpc module located in module/vpc outputs.tf

- Outputs.tf contains the EC2 public IP which is an output from the compute module and will be displayed once your resources provisioning completes using terraform plan

Step 7: Terraform Virtual Private Cloud (VPC) Module

- The specified aws region in that case it is a variable provided in module/vpc variables.tf with default region us-east-1.

- Get all available AZ's in VPC for this region in case you would like to deploy your resources to multiple Availability Zones (AZ's). we will be deploying to one AZ as you will see later in this code
  - Understand Terraform Code
    - **data** is a reserved key word which fetch data
    - "**aws_availability_zones**" is the resource provided by Terraform provider (in that case AWS)
    - "**azs**" is the user provided arbitrary resource name
    - Inside the curly braces the data source arguments

- Create a VPC.
  - Understand Terraform Code
    - **resource** is a reserved key word
    - "**aws_vpc**" is the resource provided by Terraform provider (in that case AWS)
    - "**tf_vpc**" is the user provided arbitrary resource name
    - Inside the curly braces is the resource configuration arguments
      - The CIDR block range for your VPC network
      - Enabling both dns and hostnames
      - Finally tagging the resource

main.tf

modules > vpc > main.tf > ...

```
#Create VPC in us-east-1
#=======================
resource "aws_vpc" "tf_vpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags = {
    Name = "Terraform-VPC"
  }
}
```

- Create Internet Gateway (IGW)
  - Understand Terraform Code
    - To access the vpc id resource you use the resource provided by Terraform provider.the user provided arbitrary resource name.id (aws_vpc.tf_vpc.id)

```
#Create IGW in us-east-1
#=========================
resource "aws_internet_gateway" "tf_igw" {
  vpc_id = aws_vpc.tf_vpc.id
  tags = {
    Name = "Terraform-Gateway"
  }
}
```

- Create Public route table
  - This creates a public route table for your VPC through the previously created Internet Gateway. CIDR block 0.0.0.0/0 simply means any IP for

main.tf

modules > vpc > main.tf > ...

```
#Create public route table in us-east-1
#======================================
resource "aws_route_table" "tf_public_route" {
  vpc_id = aws_vpc.tf_vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.tf_igw.id
  }
  tags = {
    Name = "Terraform-Public-RouteTable"
  }
}
```

- Create Subnet
  - Understand Terraform Code
    - **element** is a function which retrieve a single element from a list. In that case it retrieves first AZ index 0 from the list of AZs
  - Creating a Subnet CIDR block 10.0.1.0/24 within the VPC CIDR block defined early
  - Finally, we associate the subnet with the route table created early

main.tf

modules > vpc > main.tf > ...

```
#Create subnet#1 in us-east-1
#================================
resource "aws_subnet" "tf_public_subnet" {
  availability_zone = element(data.aws_availability_zones.azs.names, 0)
  vpc_id            = aws_vpc.tf_vpc.id
  cidr_block        = "10.0.1.0/24"
  tags = {
    Name = "Terraform-Subnet"
  }
}

resource "aws_route_table_association" "tf_public_assoc" {
  subnet_id          = aws_subnet.tf_public_subnet.id
  route_table_id     = aws_route_table.tf_public_route.id
}
```

- Create Security Groups
  - Here we create ingress and egress rules
  - Ingress allows inbound traffic to your VPC from the internet on tcp ports 22 and 80, you could limit the traffic from specific CIDR instead of leaving it open for security 0.0.0.0/0 or even limit from specific IP address
  - Egress allows access from the VPC to the outside world (any ports 0 and protocols -1)

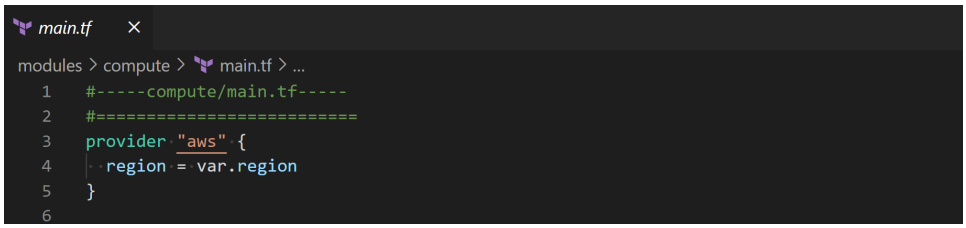
main.tf

modules > vpc > main.tf > ...

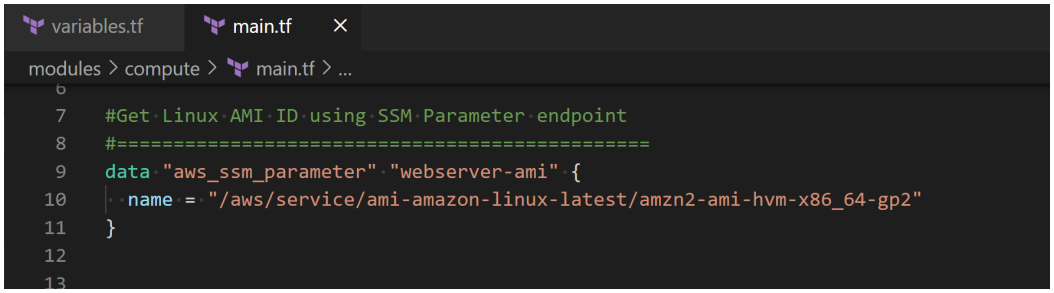
```
#Create SG for allowing TCP/80 & TCP/22
#====================================
resource "aws_security_group" "tf_public_sg" {
  name        = "tf_public_sg"
  description = "Used for access to the public instances"
  vpc_id      = aws_vpc.tf_vpc.id

  #SSH
  ingress {
    description = "Allow SSH traffic"
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  #HTTP
  ingress {
    description = "allow traffic from TCP/80"
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
  tags = {
    Name = "Terraform-SecurityGroup"
  }
}
```

Step 8: Terraform Compute Module

- The specified aws region in that case it is a variable provided in module/compute variables.tf with default region us-east-1.



- Get Linux AMI ID using SSM Parameter endpoint
  - This AWS endpoint to get all the Amazon Machine Images (AMIs) in a region (In our case it is us-east-1 region defined in the variables.tf)



- Template File
  - Understand Terraform Code
    - **file** is a function that reads the content in a file at the given path (userdata.pl) and returns them as a string

- userdata.pl
  - This a straightforward Linux Bash script to update linux, install packages, deploy a static website, and start Apache http server (You may want to go back to the provided Linux premier)

userdata.tpl

modules > compute > userdata.tpl

```
#!/bin/bash
sleep 20s
sudo yum update -y
sudo yum install httpd -y
sudo yum install wget -y
sudo yum install unzip -y
cd /var/www/html
wget https://github.com/wessamabdelwahab/CSCC1030/archive/master.zip
unzip master.zip
mv CSCC1030-master/*.* /var/www/html/
rm master.zip
service httpd start
```

- Create and bootstrap webserver
  - Understand Terraform Code
    - **data.aws_ssm_parameter.webserver-ami.value** this the value got from the SSM Parameter endpoint
    - **instance_type** it is very important not to change the value t2.micro to remain within the AWS free tier, otherwise you might encounter a cost and your credit card will be billed (Be Careful)
    - **associate_public_ip_address=true** this will associate a public IP address to the EC2 instance
    - **[var.security_group] and var.subnets**
    - **data.template_file.user-init.rendered** renders a template from a template string

main.tf

modules > compute > main.tf > resource "aws_instance" "webserver"

```
#Create and bootstrap webserver
#==============================
resource "aws_instance" "webserver" {
  ami                         = data.aws_ssm_parameter.webserver-ami.value
  instance_type               = "t2.micro"
  associate_public_ip_address = true
  vpc_security_group_ids      = [var.security_group]
  subnet_id                   = var.subnets
  user_data                   = data.template_file.user-init.rendered
  tags = {
    Name = "webserver"
  }
}
```

Step 9: Terraform Commands

- **terraform init**: Initialize a Terraform working directory
  - Locate your working directory (Lab2-Terraform) and initialize terraform **terraform init**

```
Lab2-Terraform>terraform init
Initializing modules...

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/template from the dependency lock file
- Using previously-installed hashicorp/aws v3.44.0
- Using previously-installed hashicorp/template v2.2.0

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.
```

- **terraform validate**: Validate your terraform

```
Lab2-Terraform>terraform validate
Success! The configuration is valid.
```

- **terraform fmt**: Rewrite's config files to canonical format
- **terraform plan**: Creates an execution plan for previewing your infrastructure. You could use -out option to save this plan

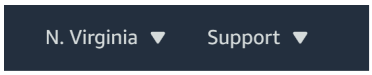
```
 # module.vpc.aws_vpc.tf_vpc will be created
 + resource "aws_vpc" "tf_vpc" {
     + arn                              = (known after apply)
     + assign_generated_ipv6_cidr_block = false
     + cidr_block                       = "10.0.0.0/16"
     + default_network_acl_id           = (known after apply)
     + default_route_table_id           = (known after apply)
     + default_security_group_id        = (known after apply)
     + dhcp_options_id                  = (known after apply)
     + enable_classiclink               = (known after apply)
     + enable_classiclink_dns_support   = (known after apply)
     + enable_dns_hostnames             = true
     + enable_dns_support               = true
     + id                               = (known after apply)
     + instance_tenancy                 = "default"
     + ipv6_association_id              = (known after apply)
     + ipv6_cidr_block                  = (known after apply)
     + main_route_table_id              = (known after apply)
     + owner_id                         = (known after apply)
     + tags                             = {
         + "Name" = "Terraform-VPC"
       }
     + tags_all                         = {
         + "Name" = "Terraform-VPC"
       }
   }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + Webserver-Public-URL = (known after apply)
```

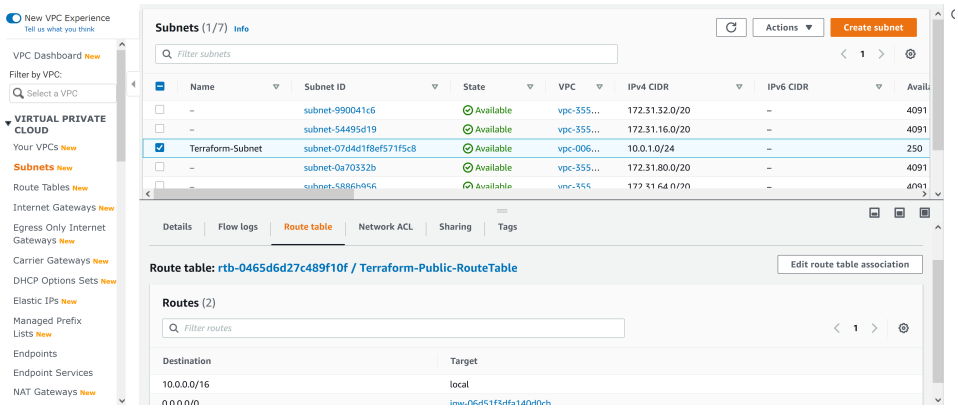
- **terraform apply**: Provision terraform managed infrastructure. You must confirm by tying **yes** if you would like to continue and perform the actions described to provision your infrastructure resources

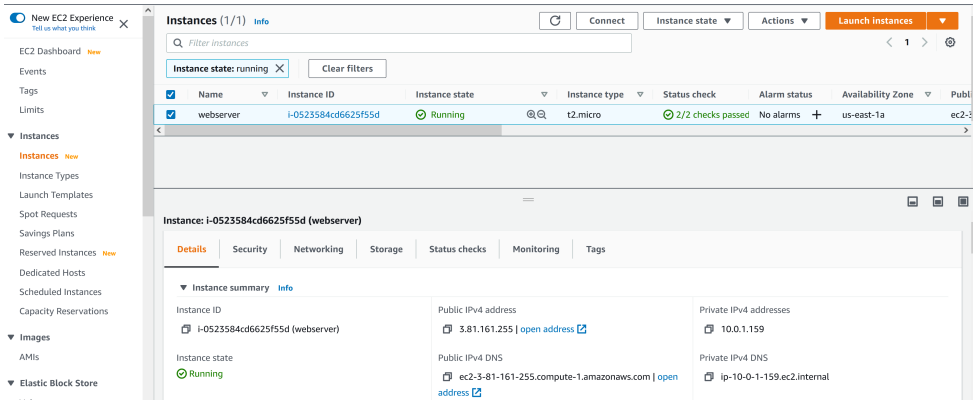
```
Changes to Outputs:
  + Webserver-Public-URL = (known after apply)

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

module.vpc.aws_vpc.tf_vpc: Creating...
module.vpc.aws_vpc.tf_vpc: Still creating... [10s elapsed]
module.vpc.aws_vpc.tf_vpc: Creation complete after 16s [id=vpc-00661c0d21b334c69]
module.vpc.aws_internet_gateway.tf_igw: Creating...
module.vpc.aws_subnet.tf_public_subnet: Creating...
module.vpc.aws_security_group.tf_public_sg: Creating...
module.vpc.aws_subnet.tf_public_subnet: Creation complete after 2s [id=subnet-07d4d1f8ef571f5c8]
module.vpc.aws_internet_gateway.tf_igw: Creation complete after 3s [id=igw-06d51f3dfa140d0cb]
module.vpc.aws_route_table.tf_public_route: Creating...
module.vpc.aws_route_table.tf_public_route: Creation complete after 2s [id=rtb-0465d6d27c489f10f]
module.vpc.aws_route_table_association.tf_public_assoc: Creating...
module.vpc.aws_security_group.tf_public_sg: Creation complete after 5s [id=sg-01f5a21335e5baa26]
module.compute.aws_instance.webserver: Creating...
module.vpc.aws_route_table_association.tf_public_assoc: Creation complete after 1s [id=rtbassoc-073e2584927a578a6]
module.compute.aws_instance.webserver: Still creating... [10s elapsed]
module.compute.aws_instance.webserver: Still creating... [20s elapsed]
module.compute.aws_instance.webserver: Still creating... [30s elapsed]
module.compute.aws_instance.webserver: Still creating... [40s elapsed]
module.compute.aws_instance.webserver: Still creating... [50s elapsed]
module.compute.aws_instance.webserver: Still creating... [1m0s elapsed]
module.compute.aws_instance.webserver: Still creating... [1m10s elapsed]
module.compute.aws_instance.webserver: Still creating... [1m20s elapsed]
module.compute.aws_instance.webserver: Creation complete after 1m30s [id=i-0523584cd6625f55d]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

Webserver-Public-URL = "http://3.81.161.255"
```

Now you can access the website from the Cloud Computing course using the URL provided in the output

- You can also login to your AWS account and see the provisioned infrastructure resources. You must select N.Virginia which is us-east-1 from the top right corner







- Once you are done with the lab you must destroy the provisioned infrastructure resources to avoid any risks of encountering cost in the future.
- **terraform destroy**: Destroy terraform managed infrastructure. You must confirm by tying **yes**

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

module.compute.aws_instance.webserver: Destroying... [id=i-0523584cd6625f55d]
module.vpc.aws_route_table_association.tf_public_assoc: Destroying... [id=rtbassoc-073e2584927a578a6]
module.vpc.aws_route_table_association.tf_public_assoc: Destruction complete after 1s
module.vpc.aws_route_table.tf_public_route: Destroying... [id=rtb-0465d6d27c489f10f]
module.vpc.aws_route_table.tf_public_route: Destruction complete after 1s
module.vpc.aws_internet_gateway.tf_igw: Destroying... [id=igw-06d51f3dfa140d0cb]
module.compute.aws_instance.webserver: Still destroying... [id=i-0523584cd6625f55d, 10s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-06d51f3dfa140d0cb, 10s elapsed]
module.compute.aws_instance.webserver: Still destroying... [id=i-0523584cd6625f55d, 20s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-06d51f3dfa140d0cb, 20s elapsed]
module.compute.aws_instance.webserver: Still destroying... [id=i-0523584cd6625f55d, 30s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-06d51f3dfa140d0cb, 30s elapsed]
module.compute.aws_instance.webserver: Still destroying... [id=i-0523584cd6625f55d, 40s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-06d51f3dfa140d0cb, 40s elapsed]
module.compute.aws_instance.webserver: Still destroying... [id=i-0523584cd6625f55d, 50s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Still destroying... [id=igw-06d51f3dfa140d0cb, 50s elapsed]
module.vpc.aws_internet_gateway.tf_igw: Destruction complete after 53s
module.compute.aws_instance.webserver: Still destroying... [id=i-0523584cd6625f55d, 1m0s elapsed]
module.compute.aws_instance.webserver: Destruction complete after 1m4s
module.vpc.aws_subnet.tf_public_subnet: Destroying... [id=subnet-07d4d1f8ef571f5c8]
module.vpc.aws_security_group.tf_public_sg: Destroying... [id=sg-01f5a21335e5baa26]
module.vpc.aws_security_group.tf_public_sg: Destruction complete after 1s
module.vpc.aws_subnet.tf_public_subnet: Destruction complete after 1s
module.vpc.aws_vpc.tf_vpc: Destroying... [id=vpc-00661c0d21b334c69]
module.vpc.aws_vpc.tf_vpc: Destruction complete after 0s

Destroy complete! Resources: 7 destroyed.
```

- terraform state file: Store state about your managed infrastructure and configuration. In a real-world scenario, you want to keep this file safe and even store it in multiple locations (ex. AWS S3)

Conclusion

Congrats on creating your first Infrastructure as Code (IaC) using Terraform!