CSCL1030 Applied Lab 1 - Managing AWS Infrastructure with Terraform

Student Name: Vinod Kumar Dhanavath
Course: CSCL1030 –CloudOps Tools and Techniques
Instructor: Joba Hassan
Date: 26-03-2025

Lab Overview:

The purpose of this lab was to use Terraform to create an AWS EC2 instance that hosts a basic web server. The goal was to ensure the infrastructure was correctly provisioned and accessible through a web browser.

Tools and Technologies:

- **Terraform (**Terraform v1.5.7) for infrastructure automation
- **AWS CLI (**aws-cli/2.24.23) for managing AWS resources
- **Git & GitHub** for version control (Used to clone the Repository)
- **SSH Keys** for secure server access

I. Steps to Deploy the Web Server

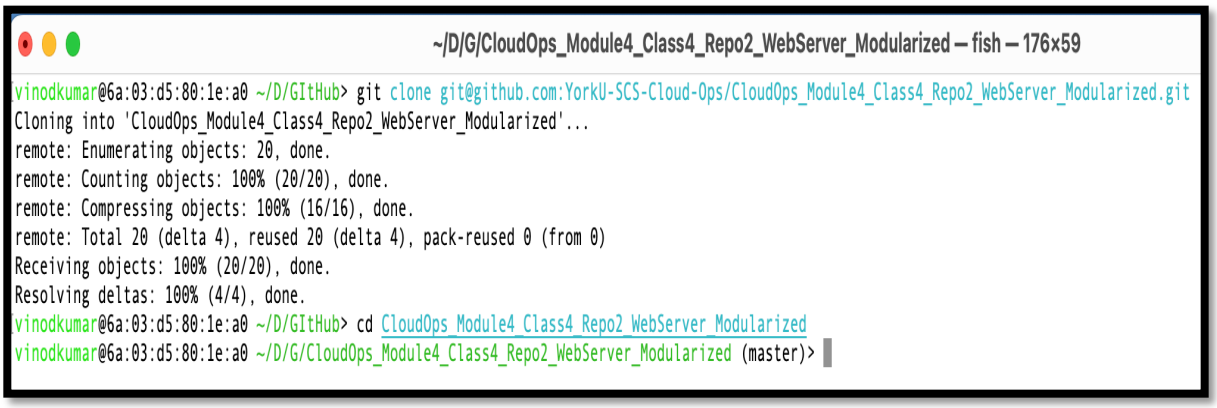
1. Cloning the Repository

To begin, the Terraform configuration files were retrieved from GitHub. This repository contained the necessary code (.tf files) to provision the AWS infrastructure. The following command was used:

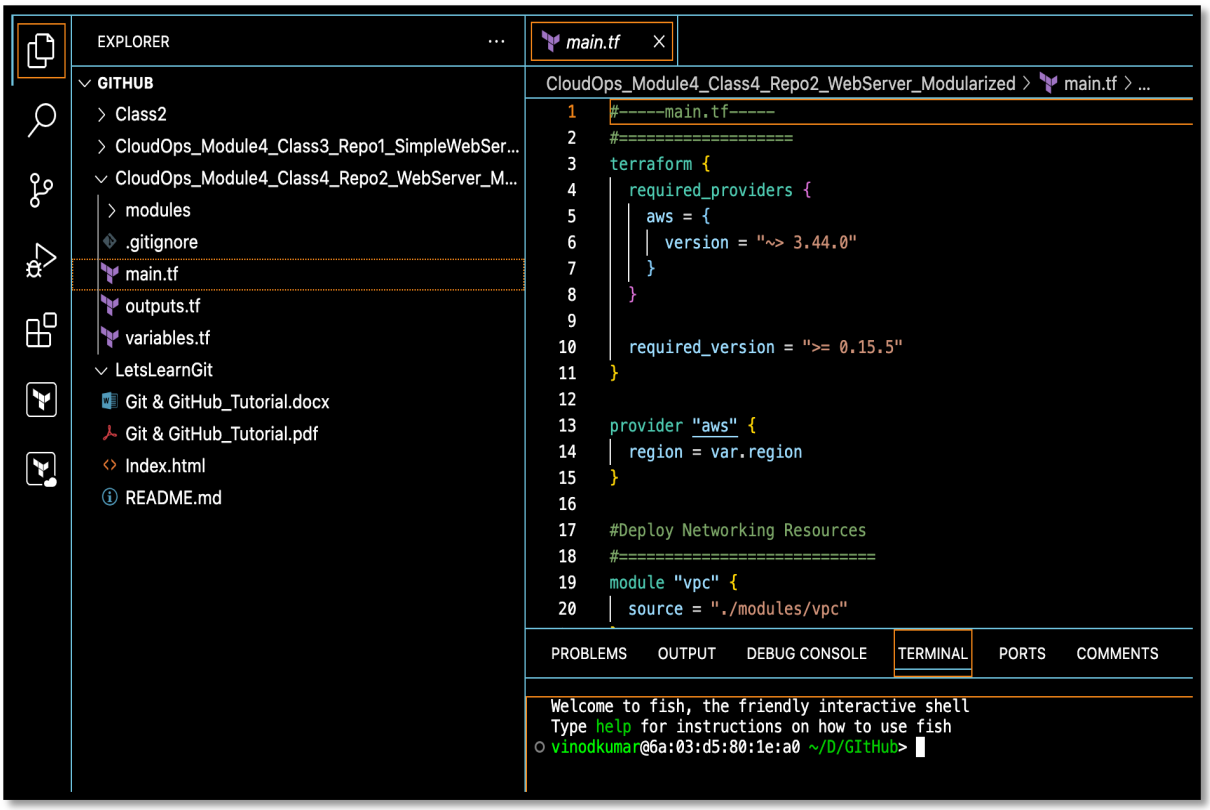
Commands:

git clone https://github.com/SMannionYorkU/CloudOps_Module4_Class4_Repo2_WebServer_Modularized.git

cd CloudOps_Module4_Class4_Repo2_WebServer_Modularized



- **Note:** I have cloned the repository and opened the folder in VS Code for easier modifications to the `.tf` files. Now, I will initialize Terraform and proceed with provisioning the AWS infrastructure. Following is the screenshot for the reference.



2. Initializing Terraform

Terraform Version Command and Output to demonstrate that Terraform was installed and configured.

```
vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)> terraform --version
Terraform v1.5.7
on darwin_arm64

Your version of Terraform is out of date! The latest version
is 1.11.2. You can update by downloading from https://www.terraform.io/downloads.html
```

Terraform was initialized to download the required provider plugins and set up the working directory. This step was executed with: **Terraform init .**

```
vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)> Terraform init

Initializing the backend...
Initializing modules...
- compute in modules/compute
- vpc in modules/vpc

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 3.44.0"...
- Installing hashicorp/aws v3.44.0...
- Installed hashicorp/aws v3.44.0 (signed by HashiCorp)

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

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.
vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)>
```

3. Validating the Configuration

Before applying the Terraform configuration, the syntax and structure of the files were checked to ensure there were no errors. The validation was performed using: **Terraform validate.**

```
vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)> terraform validate
Success! The configuration is valid.

vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)>
```

Generated a Terraform execution plan to preview the infrastructure changes before deployment using: **Terraform plan.**

```
vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)> terraform plan
module.vpc.data.aws_availability_zones.azs: Reading...
module.compute.data.aws_ssm_parameter.webserver-ami: Reading...
module.vpc.data.aws_availability_zones.azs: Read complete after 1s [id=us-east-1]
module.compute.data.aws_ssm_parameter.webserver-ami: Read complete after 1s [id=/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64]

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:

  # module.compute.aws_instance.webserver will be created
  + resource "aws_instance" "webserver" {
      + ami                                  = (sensitive value)
      + arn                                  = (known after apply)
      + associate_public_ip_address          = true
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = "webserver"
      + outpost_arn                          = (known after apply)
```

4. Deploying the Infrastructure

Terraform was executed to create the required AWS resources, including the EC2 instance using the command: **Terraform apply.**

```
vinodkumar@6a:03:d5:80:1e:a0 ~/D/G/CloudOps_Module4_Class4_Repo2_WebServer_Modularized (master)> terraform apply
module.vpc.data.aws_availability_zones.azs: Reading...
module.compute.data.aws_ssm_parameter.webserver-ami: Reading...
module.vpc.data.aws_availability_zones.azs: Read complete after 0s [id=us-east-1]
module.compute.data.aws_ssm_parameter.webserver-ami: Read complete after 0s [id=/aws/service/ami-amazon-linux-latest/al2023-ami-kernel-default-x86_64]

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:

  # module.compute.aws_instance.webserver will be created
  + resource "aws_instance" "webserver" {
```

Once, apply command executed, terraform requires confirmation as follows.

Enter the value as **“yes”.**

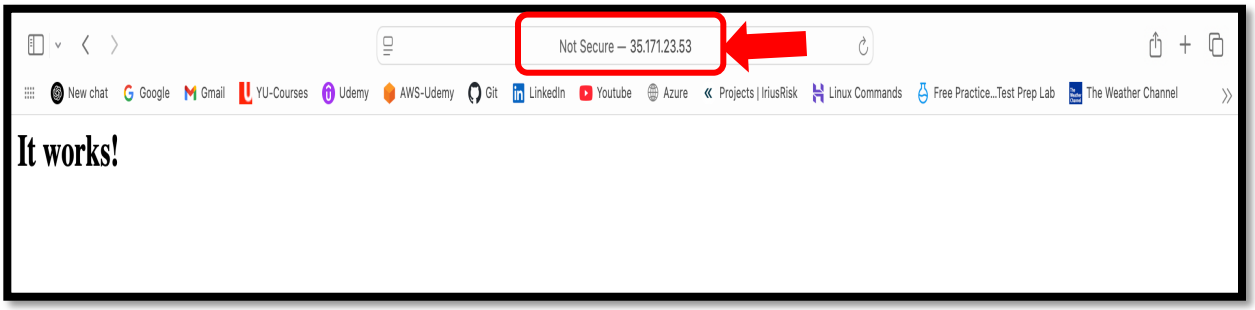
```
Changes to Outputs:
  + Apache-Webserver-Public-URL = (known after apply)

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
```

Upon successful completion, Terraform provided a **Public URL for** the web server:

5. Verifying the Web Server

a) Browser Verification

The provided Public URL was opened in a web browser to confirm that the web server was running. Following is the output a webpage displaying **"It Works!"**

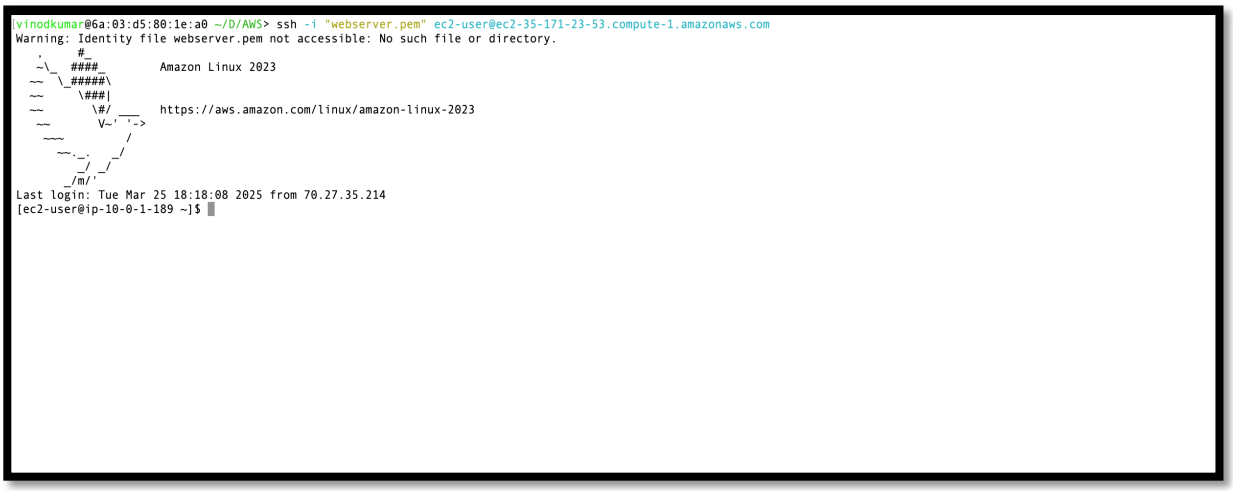


b) AWS CLI Verification

- The following screenshot demonstrates that I have successfully installed and configured the AWS Command Line Interface (CLI) on my system by generating **Access key and ID** from the AWS Console. The output confirms the following:
- AWS CLI is Installed – The version check (aws --version) confirms that AWS CLI version **2.24.23** is installed on a system running macOS **(Darwin/x86_64).**
- AWS CLI is Configured – The aws configure list command verifies that the CLI is set up with valid AWS credentials, including the access key, secret key, and default region (**us-east-1**).

- AWS CLI Can Query EC2 Instances – The aws ec2 describe-instances command successfully retrieves and displays information about my EC2 instances, showing their Instance IDs, Public IPs, and States. Required Instance id (**i-047dda929ed3a4837)** is running.

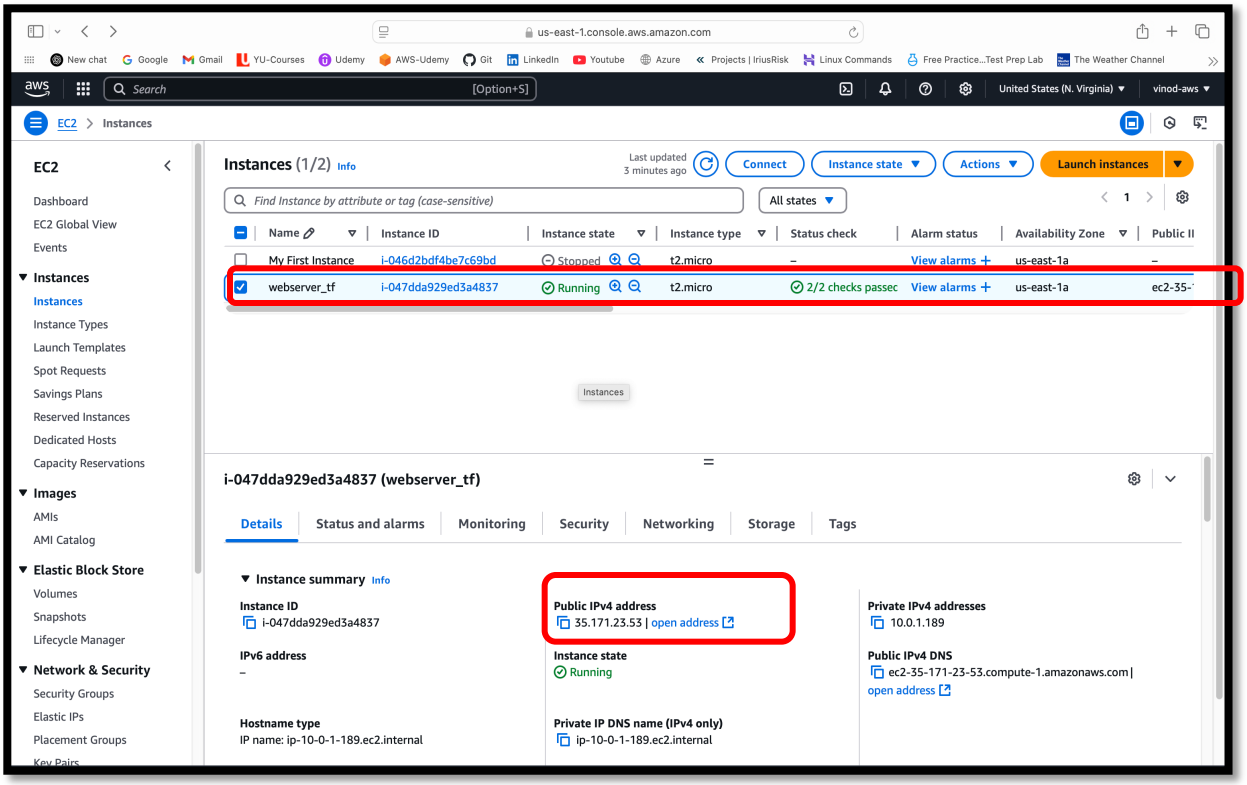
Executed the command: **ssh -i "webserver.pem" ec2-user@ec2-35-171-23-53.compute-1.amazonaws.com,** to connect the webserver using SSH key from AWS CLI. Following is the screenshot for the reference.



6. **Confirming AWS Deployment**

   The AWS Management Console was accessed to verify that the EC2 instance was running. The instance details, including its public IP address, were reviewed to ensure they matched the Terraform output.

   Following the screenshot that displays the instance name: **webserver_tf** created and running.



7. Conclusion

Terraform successfully provisioned an EC2 instance, deployed a web server, and assigned a public IP. The server was accessible via a web browser and verified through AWS CLI. This demonstrated how Infrastructure as Code (IaC) simplifies cloud resource management.

8. Submitted Deliverables

- Screenshot of Terraform and AWS CLI versions
- Screenshot of Terraform apply output (Public URL only)
- Screenshot of the web browser displaying "It Works!"
- Screenshot of AWS CLI output confirming the instance is running
- Screenshot of AWS EC2 instance details in the AWS Console

9. References:

https://registry.terraform.io/providers/hashicorp/aws/latest

https://learn.continue.yorku.ca/mod/lesson/view.php?id=603345

Thank you.