Wordpress Deployment with Terraform (Task 1)

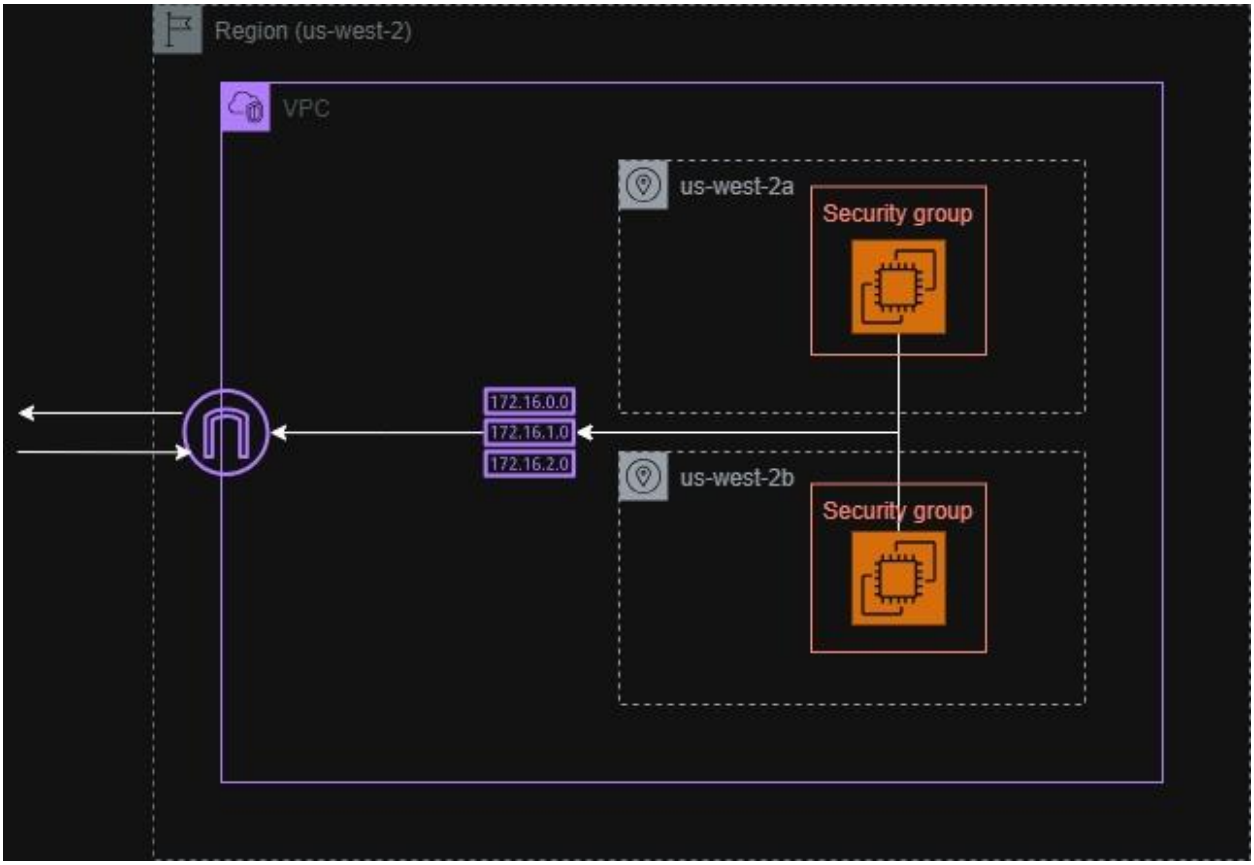
Zaeem Attique Ashar

Cloud Intern

Task Description:

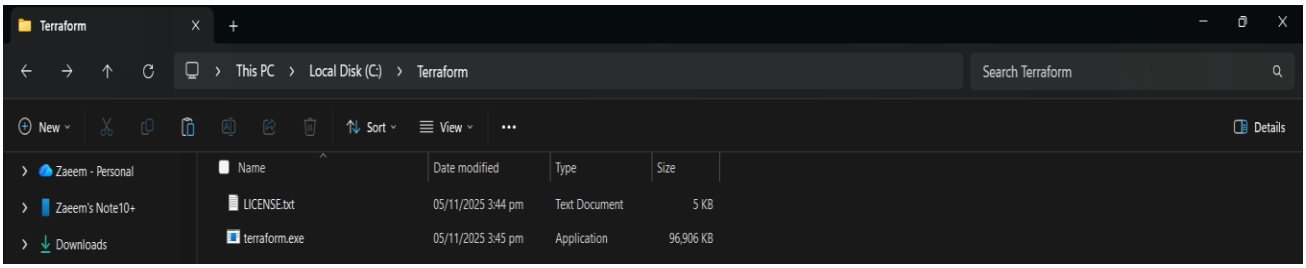
In this task is about deploying a Wordpress Website on AWS using Terraform. An EC2 instance will be provisioned with Apache, PHP, and MySQL installed on it through a user data script. Wordpress will be configured to connect to the database. The setup will include a custom VPC, **subnets, and security groups** allowing HTTP, SSH, and MySQL access, with the **public IP** output for easy access to the WordPress site.

Architecture Diagram:

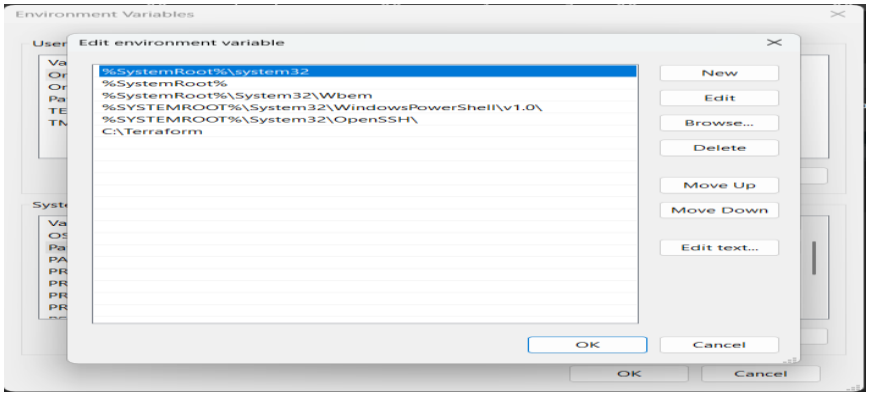


Task 1.1: Set Up Terraform Environment

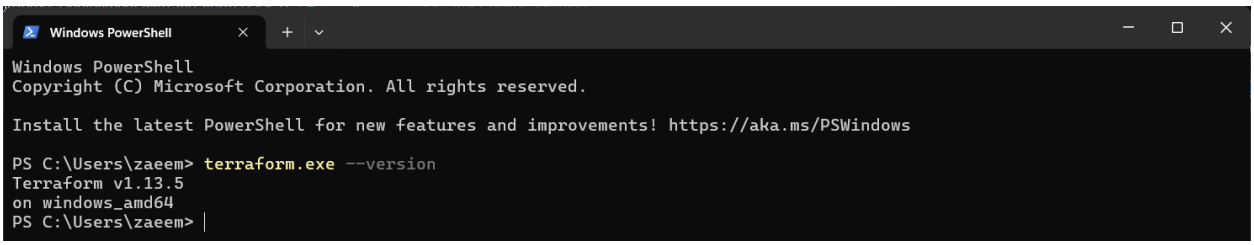
- Go to Install | Terraform | HashiCorp Developer and download the zip file according to the PC's architecture. AMD64 has been selected for this task.
- Create a folder called 'Terraform' in the C: drive and extract the contents of the downloaded zip file in that folder.



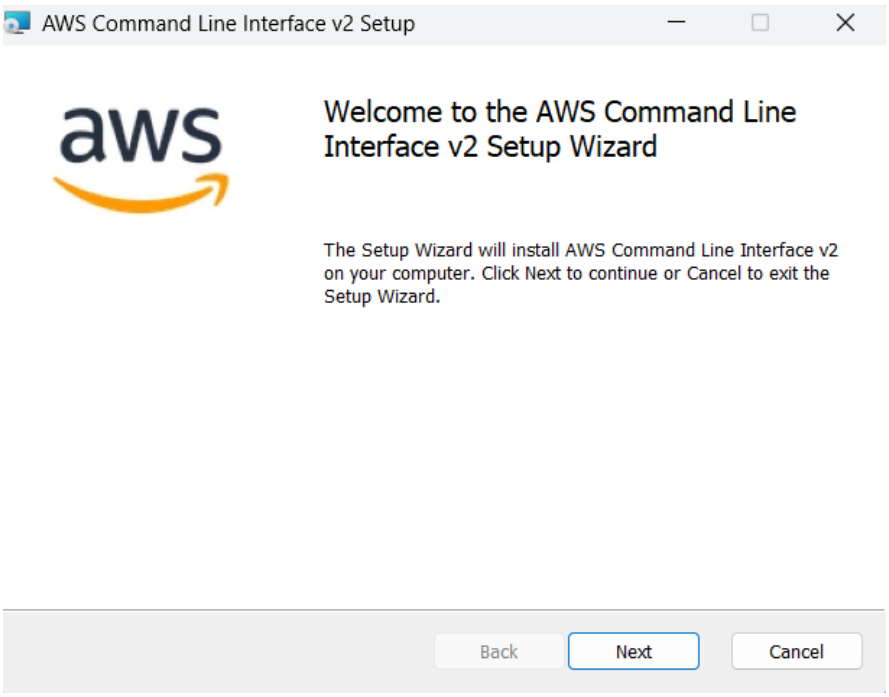
- Next, we need to add the path for the terraform.exe file to the environment valuables of the PC.



- Now we can run the command 'terraform.exe' in windows terminal to use terraform. Check the installation by checking the version.



- Now download the AWS cli from https://awscli.amazonaws.com/AWSCLIV2.msi and run the installer.



- After finishing the installation by following the onscreen instructions, open the terminal and use the command ‘aws to configure’ to enter access keys to the AWS account.

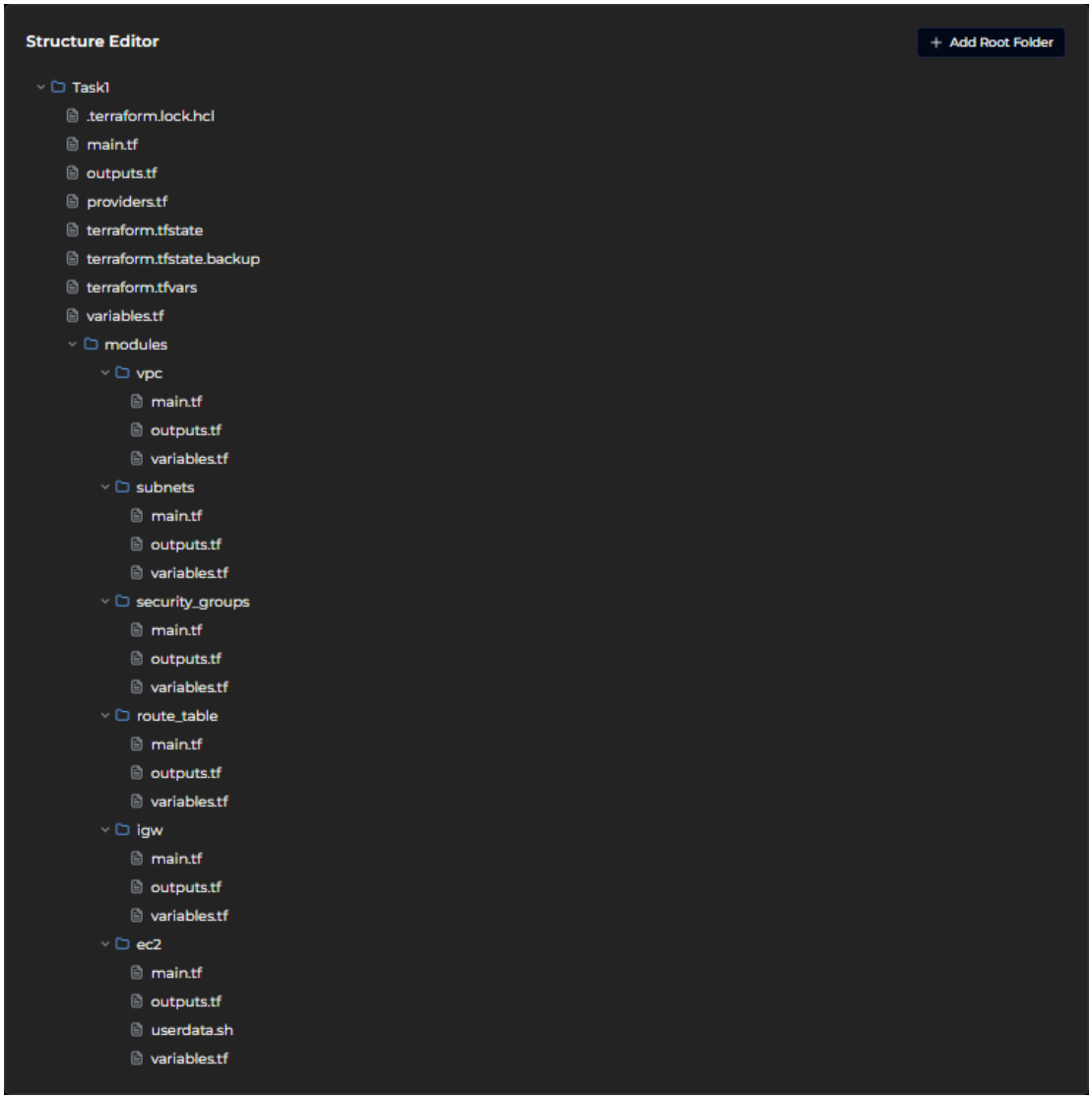
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\zaeem> aws.exe configure
AWS Access Key ID [****************7JRK]: ASIAXK73N2D72YIU55Z5
AWS Secret Access Key [****************4fb5]: EuR+O53MGoDQUnMtb9aCL4T1uWam7BGeC3XTsdrJ
AWS Session Token [None]: IQoJb3JpZ2luX2VjEOX//////////wEaCXVzLWVhc3QtMSJGMEQCIFDWL3Ie0l1bf1LXDTXaKA1T+iAriCcEA1FO4d7eAc
sNAiACrHrmCi0SHpD5PsO/aG04vY79fnXEHYWostnkSpk6MirzAgiu/////////8BEAIaDDUwNDY0OTA3Njk5MSIMxmYocWNcRXtsupVIKscC7IvSgEd6Ov
z/+1VFht8VXObwno9HI4rP0qw0pQb+xld/NM5KIXK6KPOZLxptxrbCckOGyfrvzquUhvR2us0ZRy7VIQLqSZngbrahfJHu68DRmnlTSwTHfHXQolv0sp49KW
J2mDhs5plOmCNYlRAnDJPM5N1p4F1bK625Dxj7ck8MtrSLeG+NvRYculXyUzMfPxZNuTfXjDavTFpbRaMbs0Lg7cPzm23IDgGfTkYLI66Wmk8hDoQByAuuh2
siaLAGKySU3PwbG42r5uHTblKAIUzl34VV59gH/45jmSY+HXmwoaiUslipTjFa6O/wP5DyrnK5EgF1SVKK6xKYNEFixJ+gWq+ujjqJbQ/ok3Prc+BsVX+Dab
8potfVG8ayGNg1tJoqsoU5r6oEgRs8IymJyDGJ0Fu3ZW15XY7FkheqfmrGW7BqFKJBMISTtMgGOqYBuMzsCkLuU1PMQWEdtrjyBoy4XVPwfrWkRLw46gxIRh
4YLyWVk14Tvdnp+jLwEV18lQkHJrNUHq9OetrHf+ZsNoqj72bA3i9Rg6iRK9mLm8z6pSkW16cRpPzMlDMDcYN6Jcs/aCC9tYWwXpSzFYwyJq8X5jZTLPivPP
lY2S6jm1R/QWVy1L1EDr9KLkx6IYyvgUB4+VprSEtnbbW3mxyoQ7mrwzKSfA==
Default region name [us-east-1]: us-west-2
Default output format [None]:
PS C:\Users\zaeem>
```

Task 1.2: Create Terraform Configuration files

- We will create 6 module folders for the components of Task1 and create these files for each module and root folder.

1. Main.tf
    - This file will contain the complete code used in this task
2. Variables.tf
    - This file contains the variables definitions that can be used multiple times in the main.tf.
3. Outputs.tf
    - This file will contain the outputs that are defined in the main.tf file.

- Here is a visual representation of the module directories.



Task 1.3: Infrastructure and Networking Configuration

Components needed to be defined in terraform:

- VPC:
  - CIDR: 10.0.0.0/16

```
modules > vpc > main.tf > resource "aws_vpc" "Task1_VPC_Zaeem"
1   resource "aws_vpc" "Task1_VPC_Zaeem" {
2     cidr_block = var.vpc_cidr
3     tags = {
4       Name = "Task1_VPC_Zaeem"
5     }
6   }
```

  - Output: vpc_id

```
modules > vpc > outputs.tf > output "vpc_id" > value
1   output "vpc_id" {
2     value = aws_vpc.Task1_VPC_Zaeem.id
3
4   }
```

- Internet Gateway:
  - Definition and attach to VPC

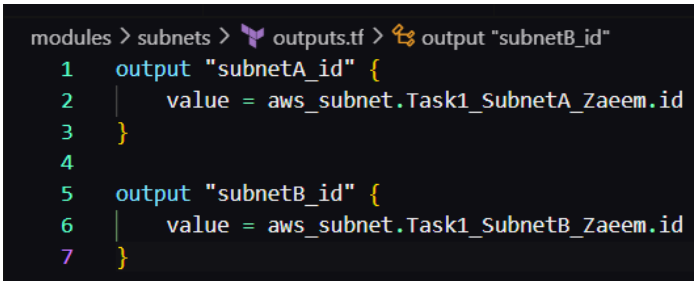
```
modules > igw > main.tf > resource "aws_internet_gateway" "Task0_igw_Zaeem" > tags > Name
1  resource "aws_internet_gateway" "Task0_igw_Zaeem" {
2    vpc_id = var.vpc_id
3
4    tags = {
5      Name = "Task1-IGW-Zaeem"
6    }
7
8  }
```

  - Outputs:

```
modules > igw > outputs.tf > output "igw_id" > value
1   output "igw_id" {
2     value = aws_internet_gateway.Task0_igw_Zaeem.id
3   }
```

- Subnets:
    - CIDR block A: 10.0.1.0/24 | CIDR block B: 10.0.2.0/24
    - Availability Zone A: us-west-2a | Availability Zone B: us-west-2b

    - Outputs:



- Route Table:
    - Route Table Destination: 0.0.0.0/0 | Target: Internet Gateway ID
    - Route Table Subnet Association with Subnet A & B

```
modules > route_table > main.tf > resource "aws_route_table" "Task1_route_table_Zaeem" > tags
resource "aws_route_table" "Task1_route_table_Zaeem" {
  vpc_id = var.vpc_id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = var.igw_id
  }

  tags = {
    Name = "Task1_route_table_Zaeem"
  }
}

resource "aws_route_table_association" "Task01_route_table_associationA_Zaeem" {
  subnet_id      = var.subnetA_id
  route_table_id = aws_route_table.Task1_route_table_Zaeem.id

}

resource "aws_route_table_association" "Task01_route_table_associationB_Zaeem" {
  subnet_id      = var.subnetB_id
  route_table_id = aws_route_table.Task1_route_table_Zaeem.id

}
```

    - Outputs:

```
modules > route_table > outputs.tf > output "route_table_id" > value
output "route_table_id" {
  value = aws_route_table.Task1_route_table_Zaeem
}
```

- Security Groups:
    - Security Group for Instance A
        - Inbound Port: 22 | Source: 0.0.0.0/0
        - Inbound Port: 80 | Source: 0.0.0.0/0
        - Outbound Port: ANY

```
modules > security_groups > main.tf > resource "aws_security_group" "Task1_sg_instanceA_Zaeem"
resource "aws_security_group" "Task1_sg_instanceA_Zaeem" {
  name        = "Task1_sg_instanceA_Zaeem"
  description = "Security group for Instance A"
  vpc_id      = var.vpc_id

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"          # -1 means all protocols
    cidr_blocks = ["0.0.0.0/0"] # allow to all destinations
  }

}
```

- Security Group for Instance B: Same as the previous one.

- EC2 Instances:
  - Instance A
    - AMI: Amazon Linux 2023
    - Instance Type: t3.micro
    - Associate Public IP: True
    - Availability Zone: us-west-2a
    - Subnet ID: Import from module
    - Security Group ID: Import from module
    - VPC ID: Import from module
    - User Data: ./userdata.sh

```
modules > ec2 > main.tf > resource "aws_instance" "Task1_EC2_A_Zaeem"
resource "aws_instance" "Task1_EC2_A_Zaeem" {
  ami                         = var.ec2_ami
  instance_type               = var.ec2_instance_type
  subnet_id                   = var.subnetA_id
  vpc_security_group_ids      = [var.sgA_id]
  associate_public_ip_address = true
  availability_zone = var.AZA
  user_data = file("${path.module}/userdata.sh")


  tags = {
    Name = "Task1_EC2_A_Zaeem"
  }
}
```

- Instance B
  - AMI: Amazon Linux 2023
  - Instance Type: t3.micro
  - Associate Public IP: True
  - Availability Zone: us-west-2b
  - Subnet ID: Import from module
  - Security Group ID: Import from module
  - VPC ID: Import from module

Task 1.4: User Data Script for installing and setting up WP & MySQL

- Below are the steps implemented in bash to achieve the objectives:

1. Yum repository update.
2. Install and enable Apache (httpd).
3. Enable php8.2 repository in amazon-linux-extras and refresh the repo.
4. Install php php-mysqlnd php-fpm php-json php-mbstring for wordpress.
5. Install mariadb server version 10.5 which supports mysql. Then enable it.
6. Set a password for the mysql root user.
7. Create database wordpress.
8. Create user wpuser@localhost and set password.
9. Grant all privileges to the user created. Flush privileges to refresh and reflect.
10. Download the wordpress .tar file from https://wordpress.org/latest.tar.gz
11. Extract and copy the folder to the directory var/www/html to be served.
12. Change the ownership of the said directory to the user called apache and assign permissions (755).
13. Rename the file wp-config-sample.php to wp-config.php
14. Edit the file wp-config.php and replace the place holders with
    a. Database Name
    b. Username
    c. Password
15. Restart httpd to start serving the wordpress webpages on port 80.

- Write the User Data code in a .sh file and place it in the ec2 module.

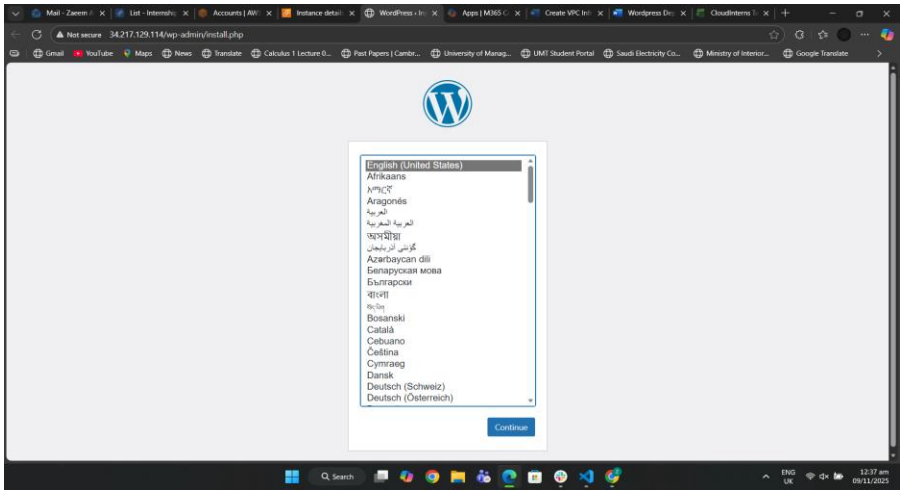
Task 1.5: Terraform project collaboration setup

- Store the .tfstate file in an S3 bucket to sync the current version of the file.
    - Create an S3 bucket.
- Use DynamoDB for state locking to avoid clashing between users.
    - Create a DynamoDB table and enter ‘LockID’ as the partition key.

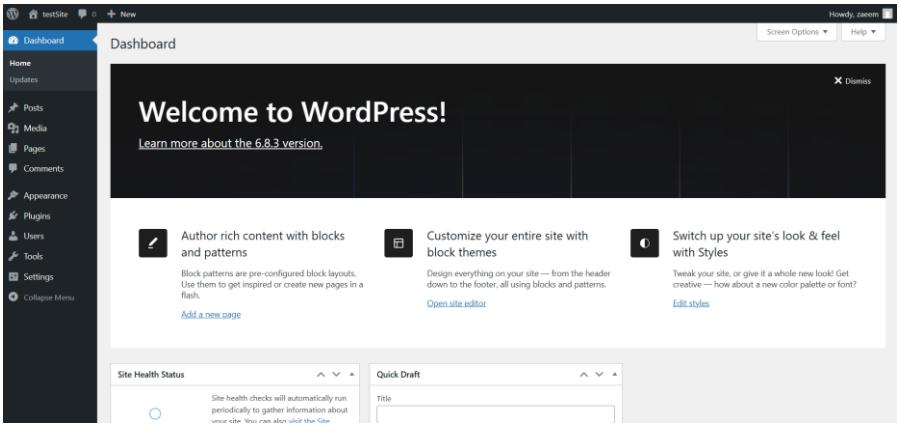
```
terraform {
  backend "s3" {
    bucket = "task1-zaeem-tfstate"
    key    = "task1/terraform/state/terraform.tfstate"
    region = "us-west-2"

    dynamodb_table = "task1-zaeem-tfstate-lock"
  }
}
```

Task 1.6: Running Tests

- Accessing the WordPress setup page on the public IP of the instance.



- WordPress Dashboard after completing the setup.



- Connecting to the instance via SSH and using systemctl to verify the services.
  - Apache (httpd)

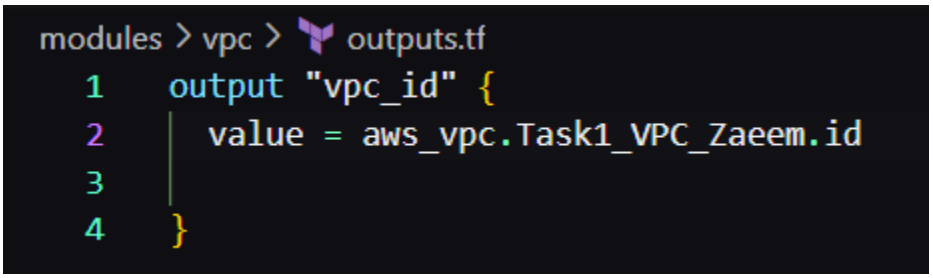
```
[ec2-user@ip-10-0-1-222 ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Sat 2025-11-08 19:36:47 UTC; 6min ago
       Docs: man:httpd.service(8)
   Main PID: 27441 (httpd)
     Status: "Total requests: 239; Idle/Busy workers 100/0;Requests/sec: 0.599; Bytes served/sec:  29KB/sec"
      Tasks: 230 (limit: 1053)
     Memory: 28.7M
        CPU: 673ms
     CGroup: /system.slice/httpd.service
             ├─27441 /usr/sbin/httpd -DFOREGROUND
             ├─27442 /usr/sbin/httpd -DFOREGROUND
             ├─27443 /usr/sbin/httpd -DFOREGROUND
             ├─27444 /usr/sbin/httpd -DFOREGROUND
             ├─27445 /usr/sbin/httpd -DFOREGROUND
             └─27657 /usr/sbin/httpd -DFOREGROUND
```

  - MariaDB (MySQL)

```
[ec2-user@ip-10-0-1-222 ~]$ sudo systemctl status mariadb.service
● mariadb.service - MariaDB 10.5 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
     Active: active (running) since Sat 2025-11-08 19:36:42 UTC; 9min ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 27295 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 12 (limit: 1053)
     Memory: 80.2M
        CPU: 1.106s
     CGroup: /system.slice/mariadb.service
             └─27295 /usr/libexec/mariadbd --basedir=/usr

Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: The second is mysql@localhost, it has no password either, but
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: you need to be the system 'mysql' user to connect.
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: After connecting you can set the password, if you would need to be
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: able to connect as any of these users with a password and without sudo
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: See the MariaDB Knowledgebase at https://mariadb.com/kb
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: Please report any problems at https://mariadb.org/jira
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: The latest information about MariaDB is available at https://mariadb.org/.
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: Consider joining MariaDB's strong and vibrant community:
Nov 08 19:36:41 ip-10-0-1-222.us-west-2.compute.internal mariadb-prepare-db-dir[27251]: https://mariadb.org/get-involved/
Nov 08 19:36:42 ip-10-0-1-222.us-west-2.compute.internal systemd[1]: Started mariadb.service - MariaDB 10.5 database server.
[ec2-user@ip-10-0-1-222 ~]$
```

Task 1.7: Challenges Faced

- Terraform Modularization
    - If we need to use a resource and its attributes in a different module, then we need to export that module in an outputs.tf file.



- Userdata Script:
    - When we write code in a userdata script, we need to make it fully automated so that no interaction is needed since we will not be given a shell for inputs.