PROG 8870 -- Final Project: Deploying AWS Infrastructure with Terraform and CloudFormation

Title: AWS Infrastructure Automation with Terraform and CloudFormation

Name: Suman Kumari Jakhar

Prof: Vikas Vattikonda

Overview

In this project, students will utilize Terraform and CloudFormation to create a scalable AWS infrastructure. The project will demonstrate a multi-service environment including S3 Buckets, EC2 Instances, and RDS
Database Instances, applying Infrastructure as Code (IaC) best practices across diLerent tools.

Project Scope

You are required to complete deployment tasks across six modules using Terraform and CloudFormation, ensuring:

- Private and secure AWS storage and compute resources.
- Automated and modular infrastructure.
- Proper documentation, reusability, and dynamic configuration. - A live demo showcasing resource provisioning.

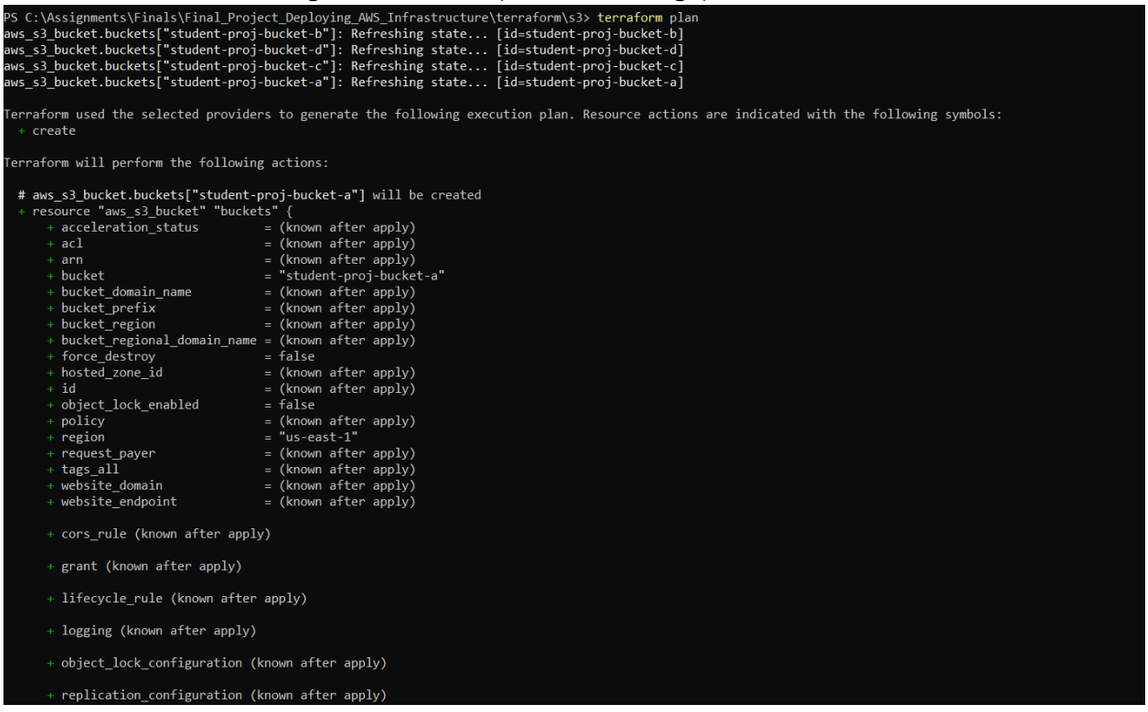
Tasks and Deliverables:

1. S3 Bucket Setup

- Using Terraform:

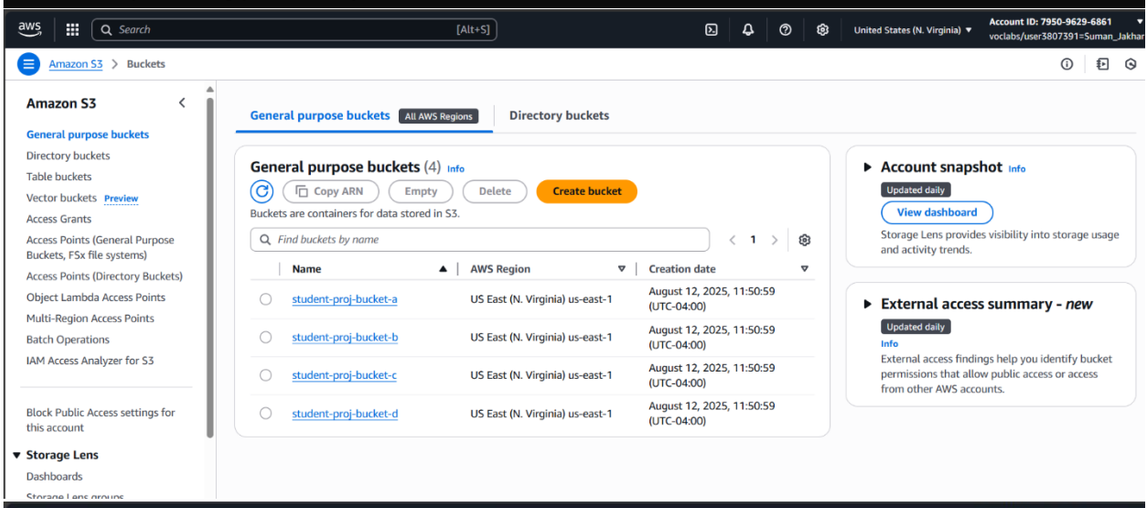
o Create 4 Private S3 Buckets without public access.

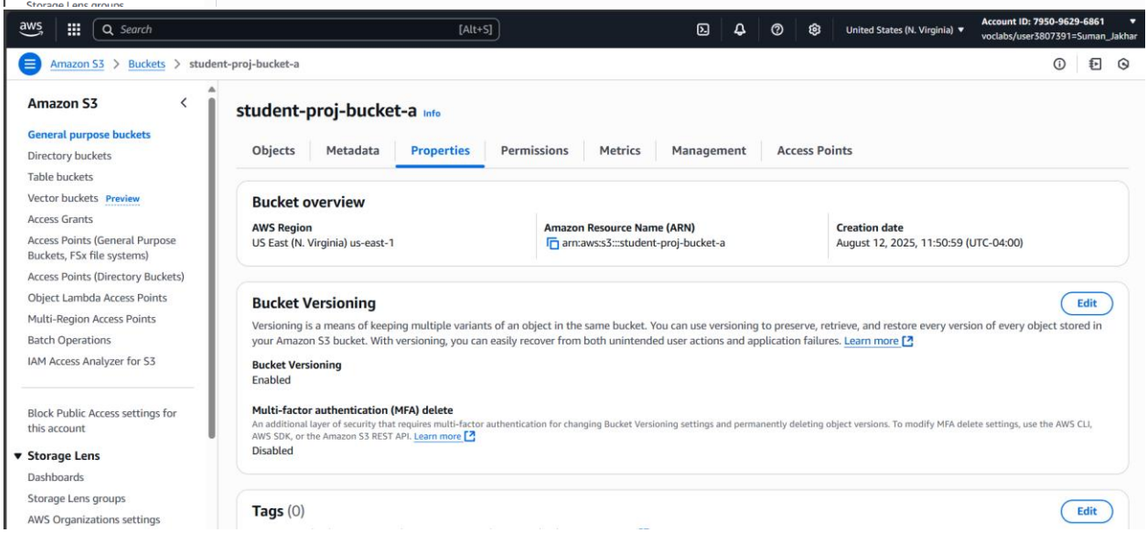
o Enable versioning on all buckets (Bonus Challenge).

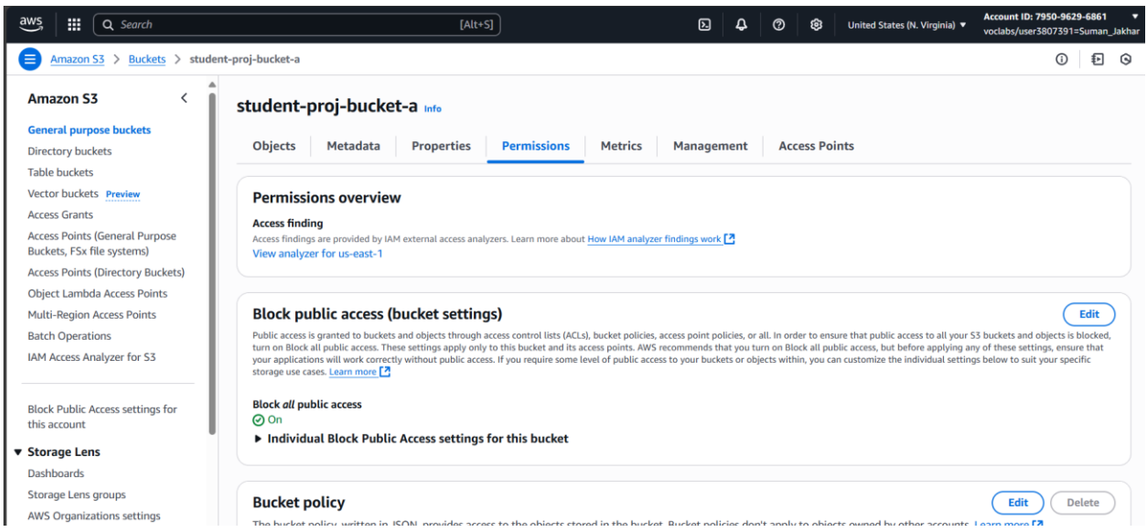


```
PS C:\Assignments\Finals\Final_Project_Deploying_AWS_Infrastructure\terraform\s3> terraform plan
aws_s3_bucket.buckets["student-proj-bucket-b"]: Refreshing state... [id=student-proj-bucket-b]
aws_s3_bucket.buckets["student-proj-bucket-d"]: Refreshing state... [id=student-proj-bucket-d]
aws_s3_bucket.buckets["student-proj-bucket-c"]: Refreshing state... [id=student-proj-bucket-c]
aws_s3_bucket.buckets["student-proj-bucket-a"]: Refreshing state... [id=student-proj-bucket-a]

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:

  # aws_s3_bucket.buckets["student-proj-bucket-a"] will be created
  + resource "aws_s3_bucket" "buckets" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "student-proj-bucket-a"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_region               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = false
      + policy                      = (known after apply)
      + region                      = "us-east-1"
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)

      + lifecycle_rule (known after apply)

      + logging (known after apply)

      + object_lock_configuration (known after apply)

      + replication_configuration (known after apply)
```

```
        + mfa_delete = (known after apply)
        + status     = "Enabled"
      }
  }

# aws_s3_bucket_versioning.versioning["student-proj-bucket-b"] will be created
+ resource "aws_s3_bucket_versioning" "versioning" {
    + bucket = (known after apply)
    + id     = (known after apply)
    + region = "us-east-1"

    + versioning_configuration {
        + mfa_delete = (known after apply)
        + status     = "Enabled"
      }
  }

# aws_s3_bucket_versioning.versioning["student-proj-bucket-c"] will be created
+ resource "aws_s3_bucket_versioning" "versioning" {
    + bucket = (known after apply)
    + id     = (known after apply)
    + region = "us-east-1"

    + versioning_configuration {
        + mfa_delete = (known after apply)
        + status     = "Enabled"
      }
  }

# aws_s3_bucket_versioning.versioning["student-proj-bucket-d"] will be created
+ resource "aws_s3_bucket_versioning" "versioning" {
    + bucket = (known after apply)
    + id     = (known after apply)
    + region = "us-east-1"

    + versioning_configuration {
        + mfa_delete = (known after apply)
        + status     = "Enabled"
      }
  }

Plan: 16 to add, 0 to change, 0 to destroy.
```

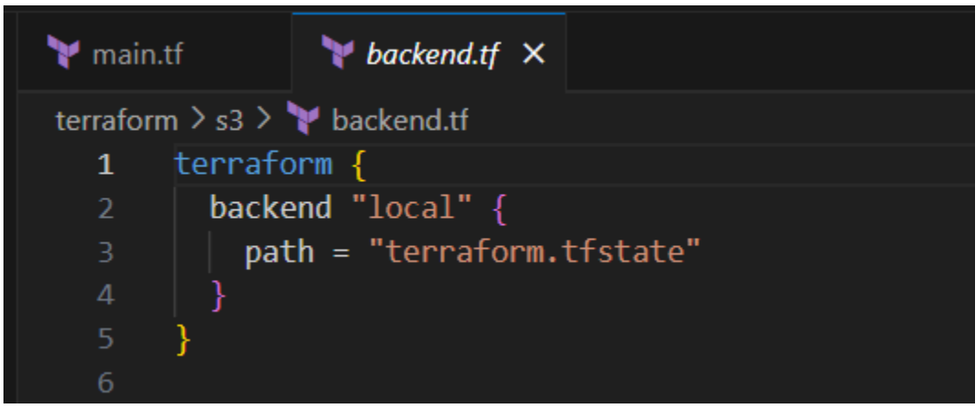




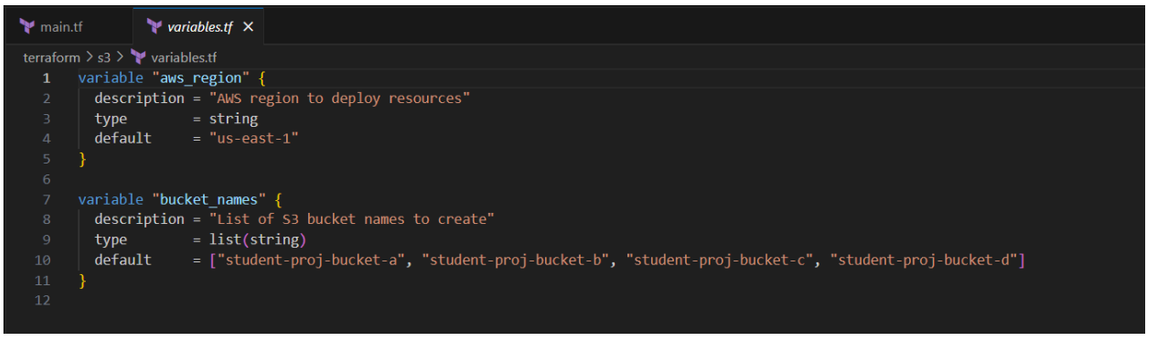


Code Snippet:

Backend.tf



Variables.tf



Terraform.tfvars

Main.tf

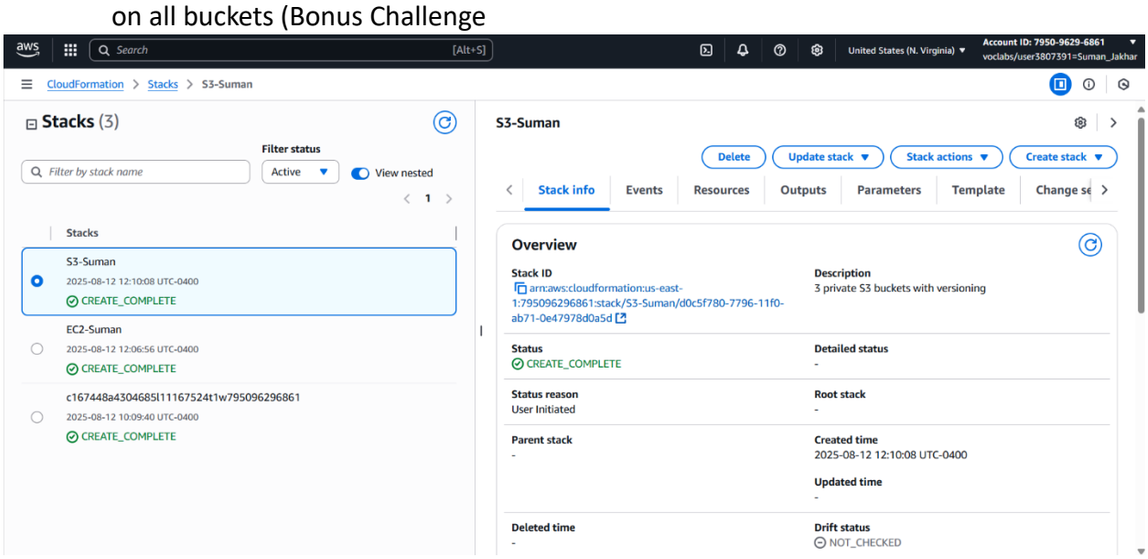
main.tf ...\ec2 | main.tf ...\s3

terraform > s3 > main.tf

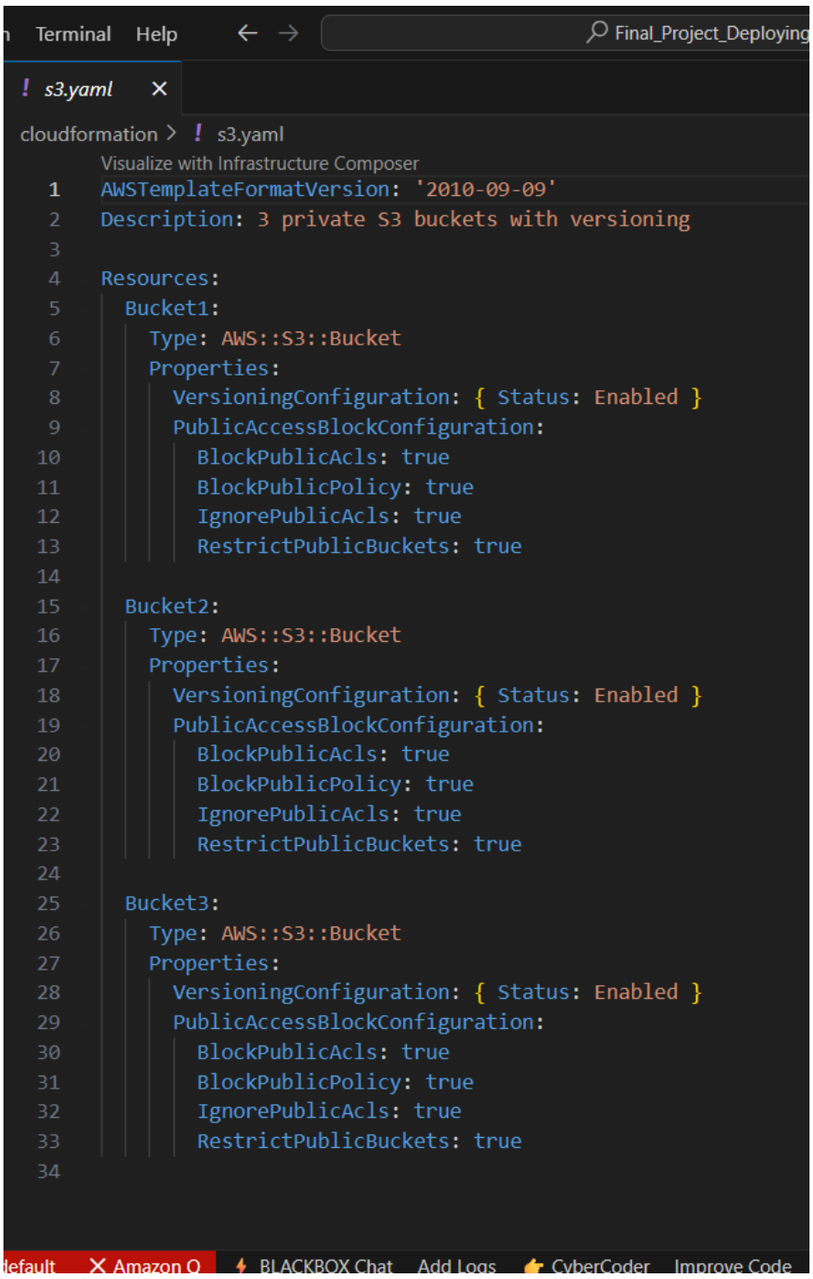
```
terraform {
  required_providers {
    aws = { source = "hashicorp/aws" }
  }
}

provider "aws" {
  region = var.aws_region
}

resource "aws_s3_bucket" "buckets" {
  for_each      = toset(var.bucket_names)
  bucket        = each.value
  object_lock_enabled = false

  lifecycle {
    ignore_changes = [object_lock_configuration]
  }
}

resource "aws_s3_bucket_acl" "private" {
  for_each = aws_s3_bucket.buckets
  bucket   = each.value.id
  acl      = "private"
}

resource "aws_s3_bucket_public_access_block" "block" {
  for_each                = aws_s3_bucket.buckets
  bucket                  = each.value.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}

resource "aws_s3_bucket_versioning" "versioning" {
  for_each = aws_s3_bucket.buckets
  bucket   = each.value.id
  versioning_configuration {
    status = "Enabled"
  }
}
```

- Using CloudFormation:

o Create 3 Private S3 Buckets with PublicAccessBlockConfiguration. o Enable versioning on all buckets (Bonus Challenge



Code Snippet:



2. VPC and EC2 Instance

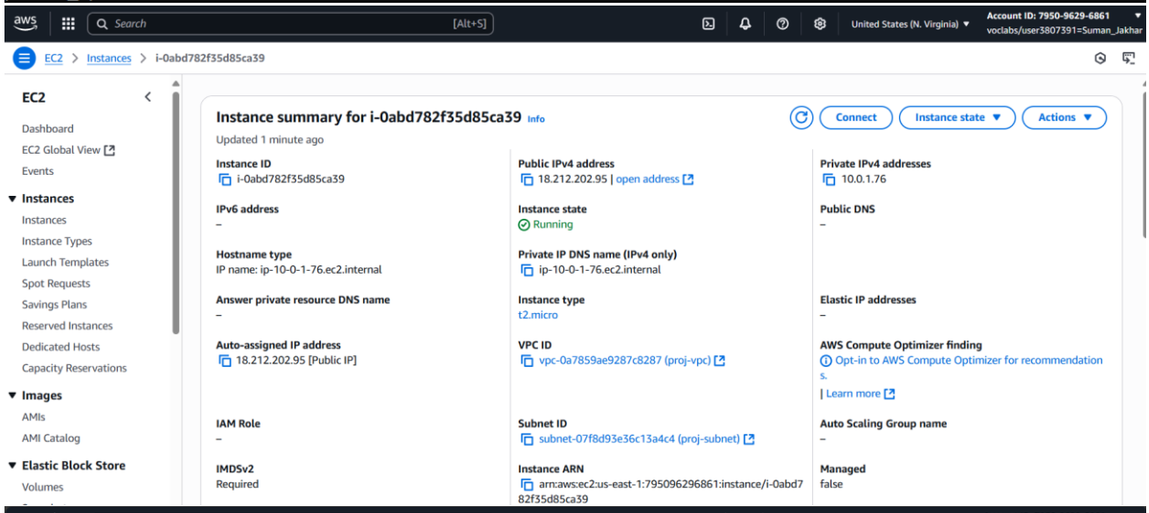
- Using Terraform:

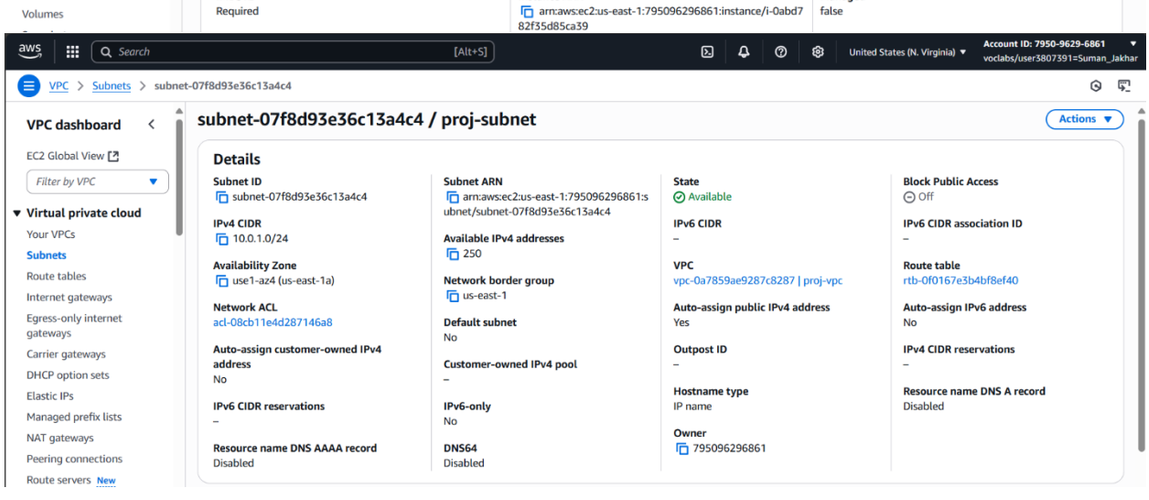
o Set up an EC2 instance inside a custom VPC.

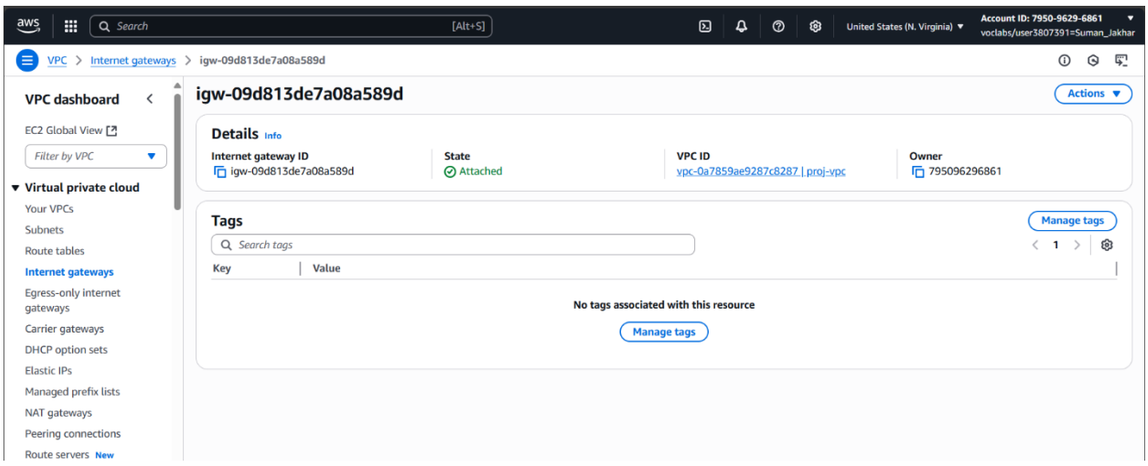
o Use dynamic variables for AMI ID and instance type. o Enable public IP and allow SSH access (port 22).

```
PS C:\Assignments\Finals\Final_Project_Deploying_AWS_Infrastructure\terraform\ec2> terraform apply

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:

  # aws_instance.ec2 will be created
  + resource "aws_instance" "ec2" {
      + ami                                  = "ami-0de716d6197524dd9"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = true
      + availability_zone                    = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + enable_primary_ipv6                  = (known after apply)
      + force_destroy                        = false
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (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                             = (known after apply)
      + monitoring                           = (known after apply)
```

```
aws_vpc.vpc: Creating...
aws_vpc.vpc: Creation complete after 2s [id=vpc-0a7859ae9287c8287]
aws_internet_gateway.igw: Creating...
aws_subnet.subnet: Creating...
aws_security_group.sg: Creating...
aws_internet_gateway.igw: Creation complete after 0s [id=igw-09d813de7a08a589d]
aws_route_table.rt: Creating...
aws_route_table.rt: Creation complete after 1s [id=rtb-0f0167e3b4bf8ef40]
aws_security_group.sg: Creation complete after 3s [id=sg-0b5c93f99be30f89e]
aws_subnet.subnet: Still creating... [10s elapsed]
aws_subnet.subnet: Creation complete after 11s [id=subnet-07f8d93e36c13a4c4]
aws_route_table_association.assoc: Creating...
aws_instance.ec2: Creating...
aws_route_table_association.assoc: Creation complete after 0s [id=rtbassoc-0a9002f52126b7d07]
aws_instance.ec2: Still creating... [10s elapsed]
aws_instance.ec2: Creation complete after 13s [id=i-0abd782f35d85ca39]

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

Outputs:

public_ip = "18.212.202.95"
```



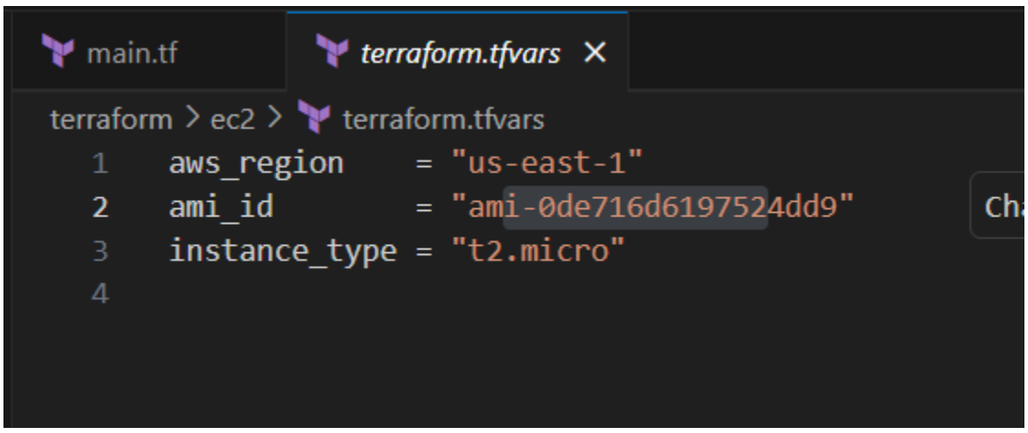




Code Snippet:

Variables.tf:

Terraform.tfvars



Backend.tf

main.tf | backend.tf

terraform > ec2 > backend.tf

```hcl
terraform {
  backend "local" {
    path = "terraform.tfstate"
  }
}
```

Main.tf

main.tf | backend.tf

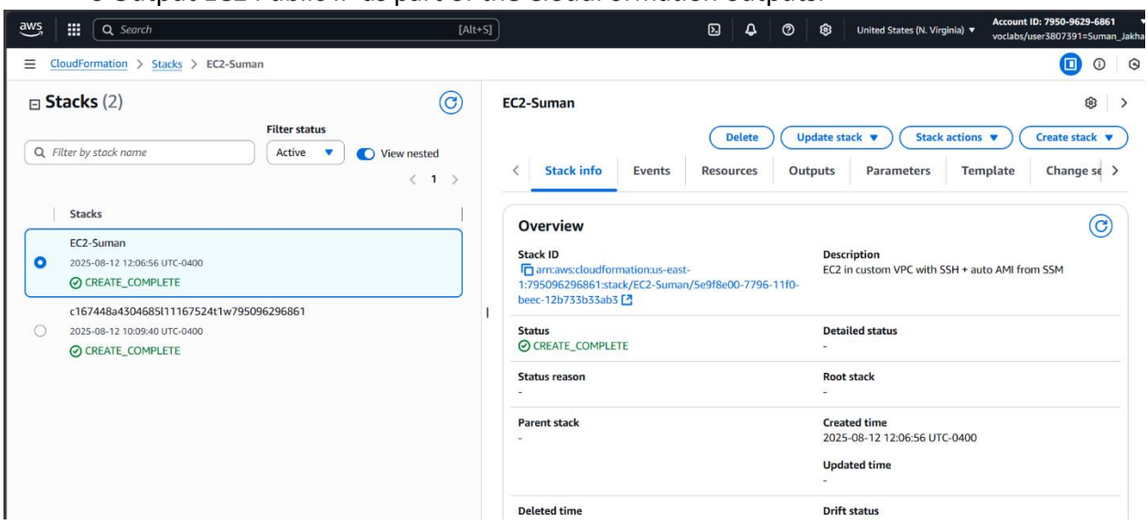
terraform > ec2 > main.tf

```hcl
terraform {
  required_providers {
    aws = { source = "hashicorp/aws" }
  }
}

provider "aws" {
  region = var.aws_region
}

resource "aws_vpc" "vpc" {
  cidr_block = var.vpc_cidr
  tags = { Name = "proj-vpc" }
}

resource "aws_subnet" "subnet" {
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = var.subnet_cidr
  map_public_ip_on_launch = true
  availability_zone       = "us-east-1a"
  tags = { Name = "proj-subnet" }
}

resource "aws_internet_gateway" "igw" { vpc_id = aws_vpc.vpc.id }

resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
}

resource "aws_route_table_association" "assoc" {
  subnet_id      = aws_subnet.subnet.id
  route_table_id = aws_route_table.rt.id
}

resource "aws_security_group" "sg" {
  vpc_id = aws_vpc.vpc.id

  ingress {
    from_port   = 22
    to_port     = 22
    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"]
  }
}

resource "aws_instance" "ec2" {
  ami                         = var.ami_id
  instance_type               = var.instance_type
  subnet_id                   = aws_subnet.subnet.id
  vpc_security_group_ids      = [aws_security_group.sg.id]
  associate_public_ip_address = true
}

output "public_ip" { value = aws_instance.ec2.public_ip }
```

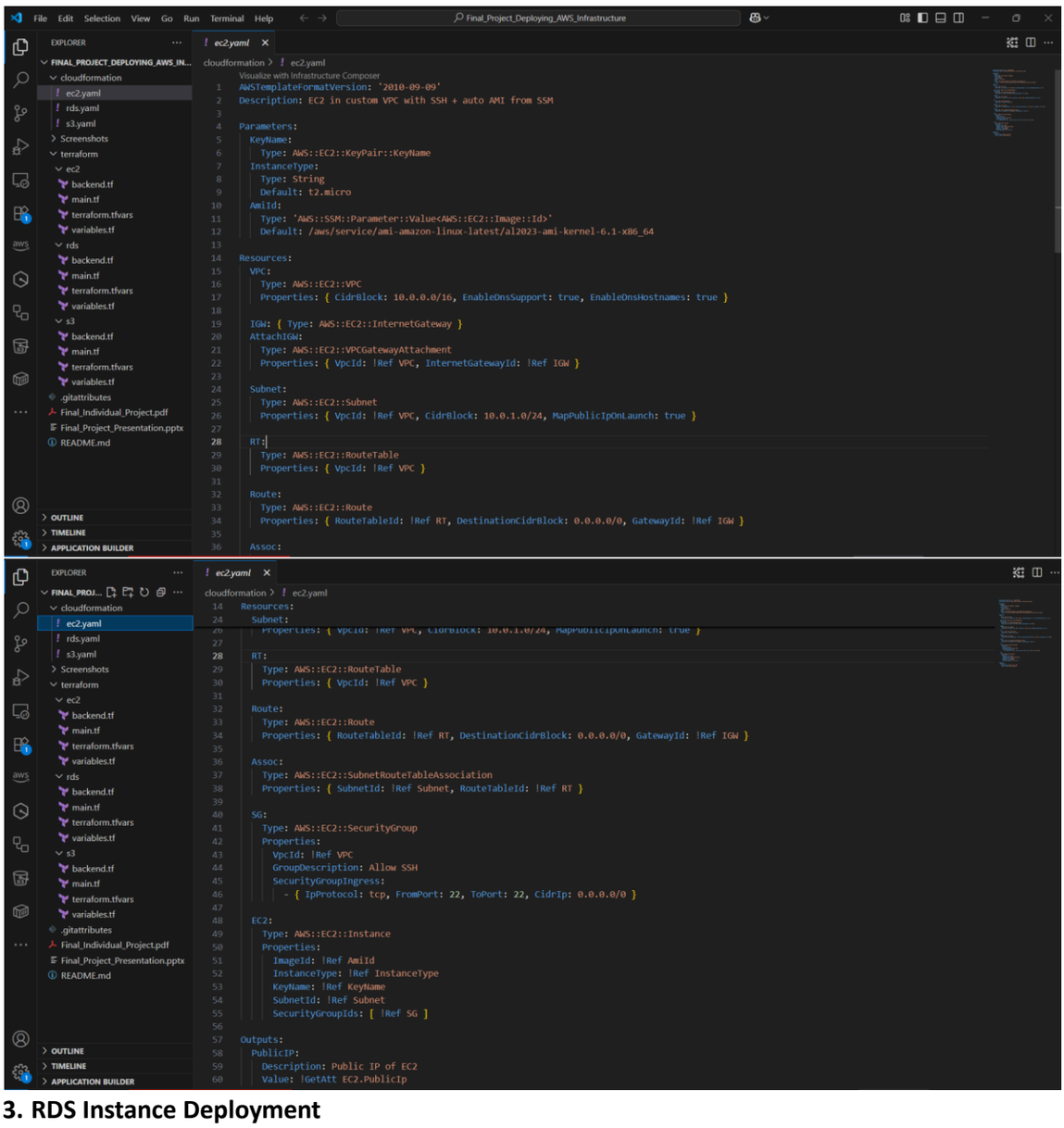
- Using CloudFormation:

o Deploy EC2 Instance with YAML-based configuration. o Attach necessary networking components (IGW, Route Tables).

o Output EC2 Public IP as part of the CloudFormation outputs.



Code snippet:



3. RDS Instance Deployment

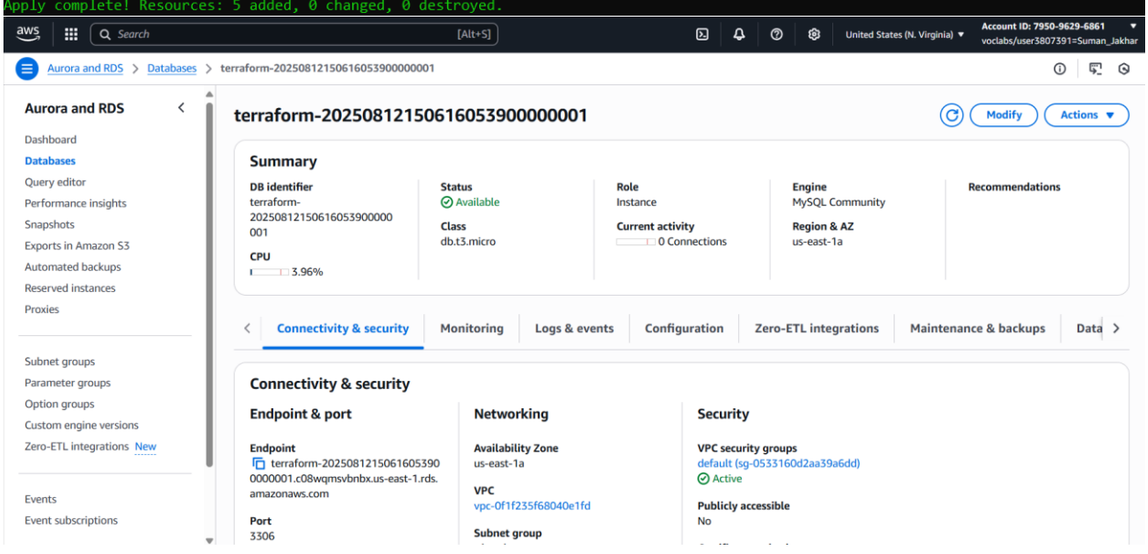
- Using Terraform:

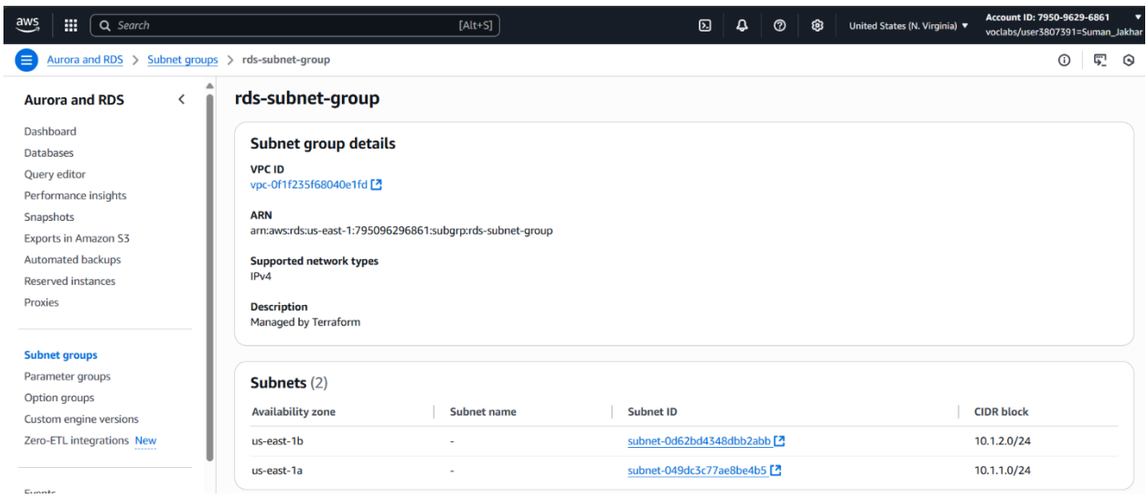
o Create a MySQL RDS Database with db.t3.micro instance type. o Define database name, username, password via input variables.

o Deploy into a dedicated DB Subnet Group.

```
PS C:\Assignments\Finals\Final_Project_Deploying_AWS_Infrastructure\terraform\rds> terraform apply

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:

  # aws_db_instance.mysql will be created
  + resource "aws_db_instance" "mysql" {
      + address                               = (known after apply)
      + allocated_storage                     = 20
      + apply_immediately                     = false
      + arn                                   = (known after apply)
      + auto_minor_version_upgrade            = true
      + availability_zone                     = (known after apply)
      + backup_retention_period               = (known after apply)
      + backup_target                         = (known after apply)
      + backup_window                         = (known after apply)
      + ca_cert_identifier                    = (known after apply)
      + character_set_name                    = (known after apply)
      + copy_tags_to_snapshot                 = false
      + database_insights_mode                = (known after apply)
      + db_name                               = "Suman_db"
      + db_subnet_group_name                  = "rds-subnet-group"
      + dedicated_log_volume                  = false
      + delete_automated_backups              = true
      + domain_fqdn                           = (known after apply)
      + endpoint                              = (known after apply)
      + engine                                = "mysql"
      + engine_lifecycle_support              = (known after apply)
      + engine_version                        = (known after apply)
      + engine_version_actual                 = (known after apply)
      + hosted_zone_id                        = (known after apply)
      + id                                    = (known after apply)
      + identifier                            = (known after apply)
      + identifier_prefix                     = (known after apply)
      + instance_class                        = "db.t3.micro"
      + iops                                  = (known after apply)
      + kms_key_id                            = (known after apply)
```

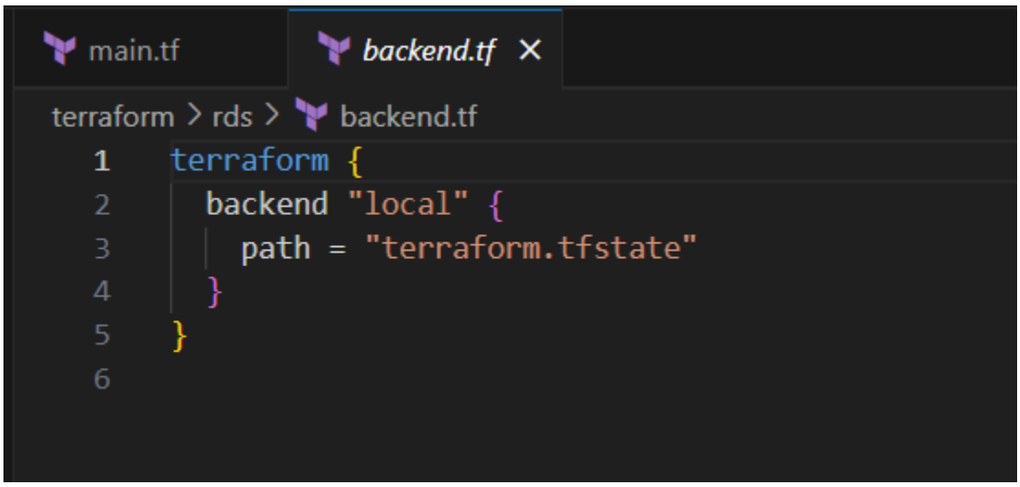
```
aws_vpc.rds_vpc: Creating...
aws_vpc.rds_vpc: Creation complete after 2s [id=vpc-0f1f235f68040e1fd]
aws_subnet.db_subnet_b: Creating...
aws_subnet.db_subnet_a: Creating...
aws_subnet.db_subnet_b: Creation complete after 1s [id=subnet-0d62bd4348dbb2abb]
aws_subnet.db_subnet_a: Creation complete after 1s [id=subnet-049dc3c77ae8be4b5]
aws_db_subnet_group.db_group: Creating...
aws_db_subnet_group.db_group: Creation complete after 1s [id=rds-subnet-group]
aws_db_instance.mysql: Creating...
aws_db_instance.mysql: Still creating... [10s elapsed]
aws_db_instance.mysql: Still creating... [20s elapsed]
aws_db_instance.mysql: Still creating... [30s elapsed]
aws_db_instance.mysql: Still creating... [40s elapsed]
aws_db_instance.mysql: Still creating... [50s elapsed]
aws_db_instance.mysql: Still creating... [1m0s elapsed]
aws_db_instance.mysql: Still creating... [1m10s elapsed]
aws_db_instance.mysql: Still creating... [1m20s elapsed]
aws_db_instance.mysql: Still creating... [1m30s elapsed]
aws_db_instance.mysql: Still creating... [1m40s elapsed]
aws_db_instance.mysql: Still creating... [1m50s elapsed]
aws_db_instance.mysql: Still creating... [2m0s elapsed]
aws_db_instance.mysql: Still creating... [2m10s elapsed]
aws_db_instance.mysql: Still creating... [2m20s elapsed]
aws_db_instance.mysql: Still creating... [2m30s elapsed]
aws_db_instance.mysql: Still creating... [2m40s elapsed]
aws_db_instance.mysql: Still creating... [2m50s elapsed]
aws_db_instance.mysql: Still creating... [3m0s elapsed]
aws_db_instance.mysql: Still creating... [3m10s elapsed]
aws_db_instance.mysql: Still creating... [3m20s elapsed]
aws_db_instance.mysql: Still creating... [3m30s elapsed]
aws_db_instance.mysql: Still creating... [3m40s elapsed]
aws_db_instance.mysql: Still creating... [3m50s elapsed]
aws_db_instance.mysql: Still creating... [4m0s elapsed]
aws_db_instance.mysql: Still creating... [4m10s elapsed]
aws_db_instance.mysql: Creation complete after 4m15s [id=db-WA2KIK6WHYTED4AIL7P7S7HTSQ]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```





Code Snippet:

Backend.tf



Variables.tf

main.tf | variables.tf

terraform > rds > variables.tf

```
variable "aws_region" {
  description = "AWS region to deploy resources"
  type        = string
  default     = "us-east-1"
}

variable "db_name" {
  description = "Database name"
  type        = string
  default     = "suman_db"
}

variable "db_user" {
  description = "Database username"
  type        = string
  default     = "admin"
}

variable "db_password" {
  description = "Database password"
  type        = string
  default     = "Suman123!"
  sensitive   = true
}
```

Terraform.tfvars

main.tf | terraform.tfvars

terraform > rds > terraform.tfvars

```
aws_region  = "us-east-1"
db_name     = "Suman_db"
db_user     = "admin"
db_password = "Suman123!"
```

Main.tf

main.tf ...\ec2 | main.tf ...\rds

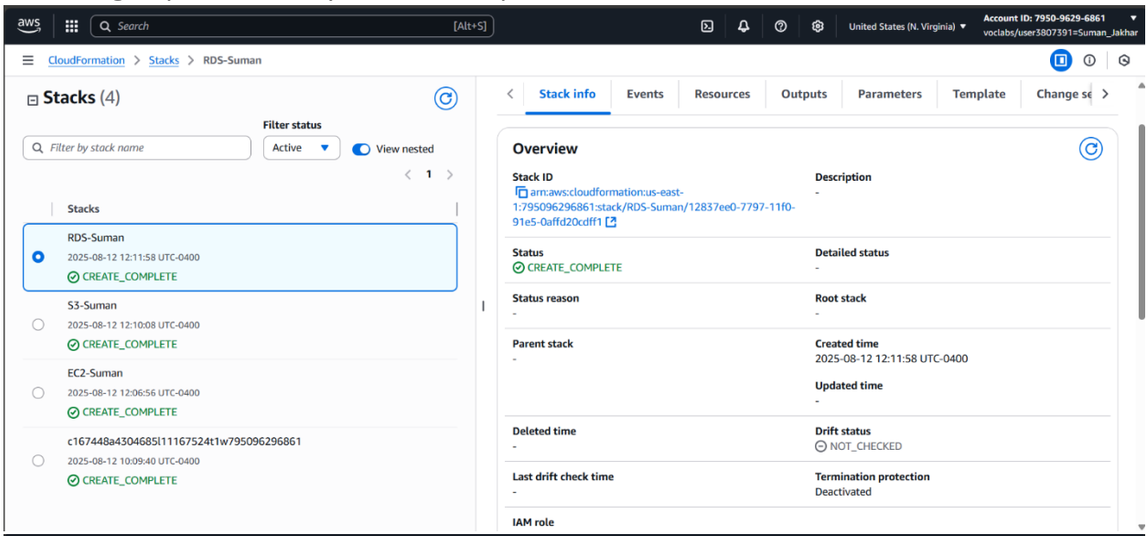
terraform > rds > main.tf

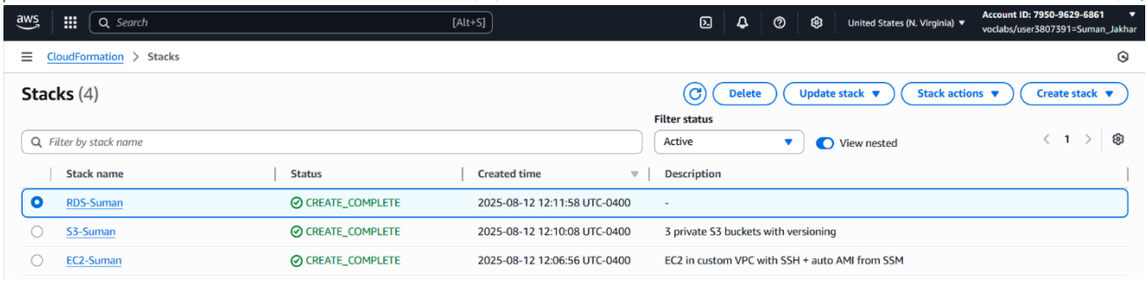
```hcl
terraform {
  required_providers {
    aws = { source = "hashicorp/aws" }
  }
}

provider "aws" {
  region = var.aws_region
}

resource "aws_vpc" "rds_vpc" {
  cidr_block = "10.1.0.0/16"
}

resource "aws_subnet" "db_subnet_a" {
  vpc_id            = aws_vpc.rds_vpc.id
  cidr_block        = "10.1.1.0/24"
  availability_zone = "us-east-1a"
}

resource "aws_subnet" "db_subnet_b" {
  vpc_id            = aws_vpc.rds_vpc.id
  cidr_block        = "10.1.2.0/24"
  availability_zone = "us-east-1b"
}

resource "aws_db_subnet_group" "db_group" {
  name       = "rds-subnet-group"
  subnet_ids = [aws_subnet.db_subnet_a.id, aws_subnet.db_subnet_b.id]
}

resource "aws_db_instance" "mysql" {
  allocated_storage    = 20
  engine               = "mysql"
  instance_class       = "db.t3.micro"
  db_name              = var.db_name
  username             = var.db_user
  password             = var.db_password
  skip_final_snapshot  = true
  db_subnet_group_name = aws_db_subnet_group.db_group.name
  publicly_accessible  = false
}
```

- Using CloudFormation:

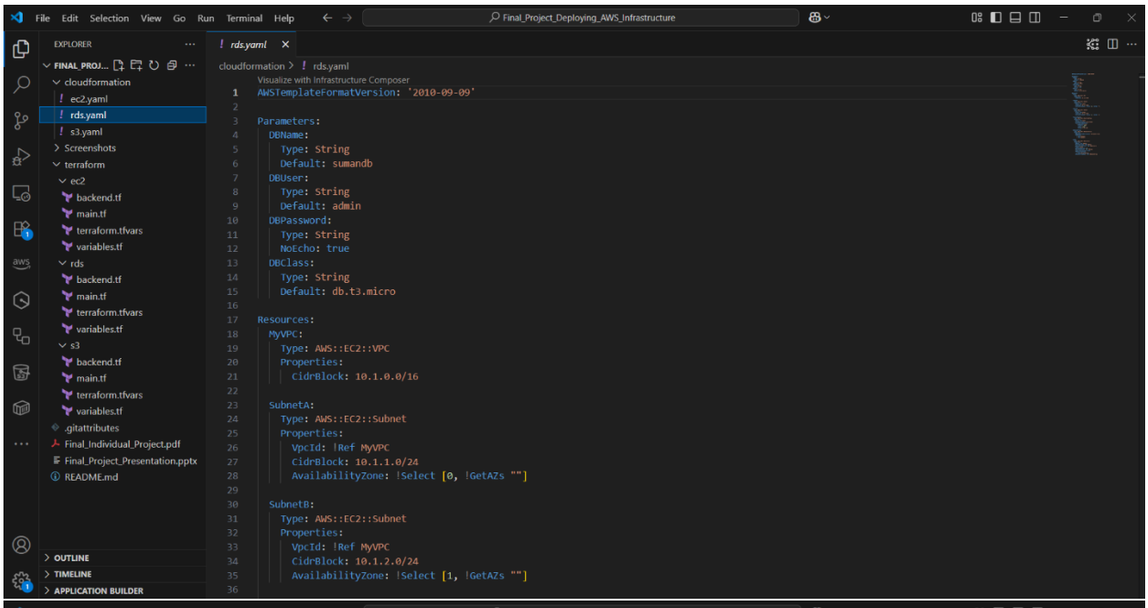
o Deploy RDS instance using YAML templates.

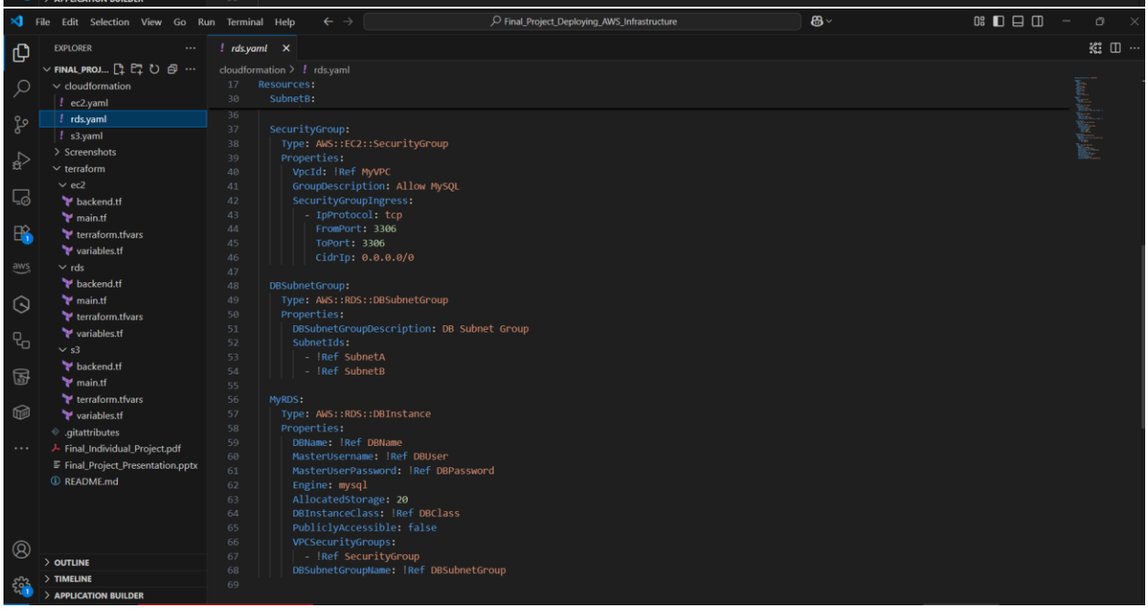
o Ensure public access is enabled (for this project only). o Configure security groups to allow MySQL traLic on port 3306.





Code Snippet:





4. Dynamic Configuration to Avoid hardcoding all values. o Use variables files (variables.tf, .tfvars) and CloudFormation Parameters where applicable.

5. Backend/State Management

o Store Terraform state file locally on your laptop. o Use AWS CLI or AWS Console for CloudFormation stack deployment.

6. GitHub Repository

- Push your Terraform code and CloudFormation YAML files to a GitHub repository.
- Your repository must include:
  - main.tf (Terraform configuration) ○ variables.tf (Variables definition) o terraform.tfvars (Variables values; sensitive data should not be pushed to GitHub) o backend.tf (Backend configuration) o CloudFormation YAML files for S3, EC2, and RDS. o README.md (Documentation)
- Share your GitHub repository URL along with the submission document.

7. Presentation/Demo

- Prepare a **5-10 minute presentation** explaining:
  - Your code structure and implementation.

- The AWS infrastructure you created.
- Key features or challenges you encountered.
- How your Terraform code and CloudFormation ensures modularity and best practices.

- **Live Demo:**
  - Run your Terraform configuration (terraform init, terraform plan, terraform apply).
  - Create and run CloudFormation Stack using YAML's. Show the resources created in the AWS Management Console (e.g., the S3 bucket, EC2 instance, VPC, RDS, etc.).
  - Demonstrate the use of your tfvars file, YAML and backend configuration.

Submission Details • Submission Items: One document containing below items

1. GitHub repository link.
2. Screenshot(s) showing:
   - S3 Buckets created and versioning enabled.
   - EC2 Instances launched with Public IP.
   - RDS Instances running.
   - Terraform and CloudFormation code snippets.
   - Terraform apply or CloudFormation deployment outputs.
3. **Terraform Files:**
   - main.tf, provider.tf, variables.tf, vars.tfvars
4. **CloudFormation Templates:**
   - YAML files for S3, EC2, and RDS.
5. Clear and concise documentation in the README.md (in GitHub repo) 6. Power Point Presentation(PPT) slides for demo.

Assessment Criteria (Total: 30 points)

Weightage in Final grade: 35% 1. **Functionality** (10 points) ○ Are all resources deployed correctly?

2. **Best Practices** (5 points) ○ Use of variables and tfvars files. ○ Proper backend configuration.
   ○ Dynamic configuration, clean code.
3. **Documentation** (5 points) ○ Is the README.md clear and comprehensive? ○ Can a third party replicate the setup using your documentation?
   ○ Are comments added in the code?
4. **Presentation/Demo** (10 points)
   ○ Was the presentation well-structured and informative? ○ Did the demo showcase the infrastructure and code eLectively?
   ○ Were challenges and solutions explained clearly during presentation in the class?
   ○ Was the presentation well-structured and informative?

Resources

- **Terraform Documentation:** https://registry.terraform.io/
- **AWS Free Tier:** https://aws.amazon.com/free/
- **GitHub Guides:** https://guides.github.com/