Task Overview

This document outlines the implementation of AWS infrastructure using Terraform, including EC2 instance and S3 bucket setup with modular architecture and output variables.

Deliverables Completed

- AWS Infrastructure via Terraform (EC2 + S3)
- Implementation of output variables
- Modular Terraform architecture
- Remote state management using S3 backend

Tools Used

- **Terraform**: Infrastructure as Code tool
- **AWS CLI**: Command-line interface for AWS services
- **AWS Console**: Web-based AWS management interface
- **VS Code/Text Editor**: For writing Terraform configuration files

Project Structure

```
dummy_ec2/
├── main.tf           ← Module calls and main configuration
├── providers.tf      ← AWS provider configuration
├── backend.tf        ← S3 backend configuration
├── variables.tf      ← Variable declarations
├── outputs.tf        ← Output declarations
├── terraform.tfvars  ← Variable values
└── modules/ec2/      ← EC2 module
    ├── main.tf
    ├── variables.tf
    └── outputs.tf
```

Step-by-Step Implementation

Step 1: Project Setup and Initial Configuration

1.1 Created Project Directory Structure

- Created `dummy_ec2/` root directory
- Set up `modules/ec2/` subdirectory for modular architecture

1.2 Configured AWS Provider Created `providers.tf` with AWS provider configuration:

```
provider "aws" {
    region = "us-east-1"
}
```

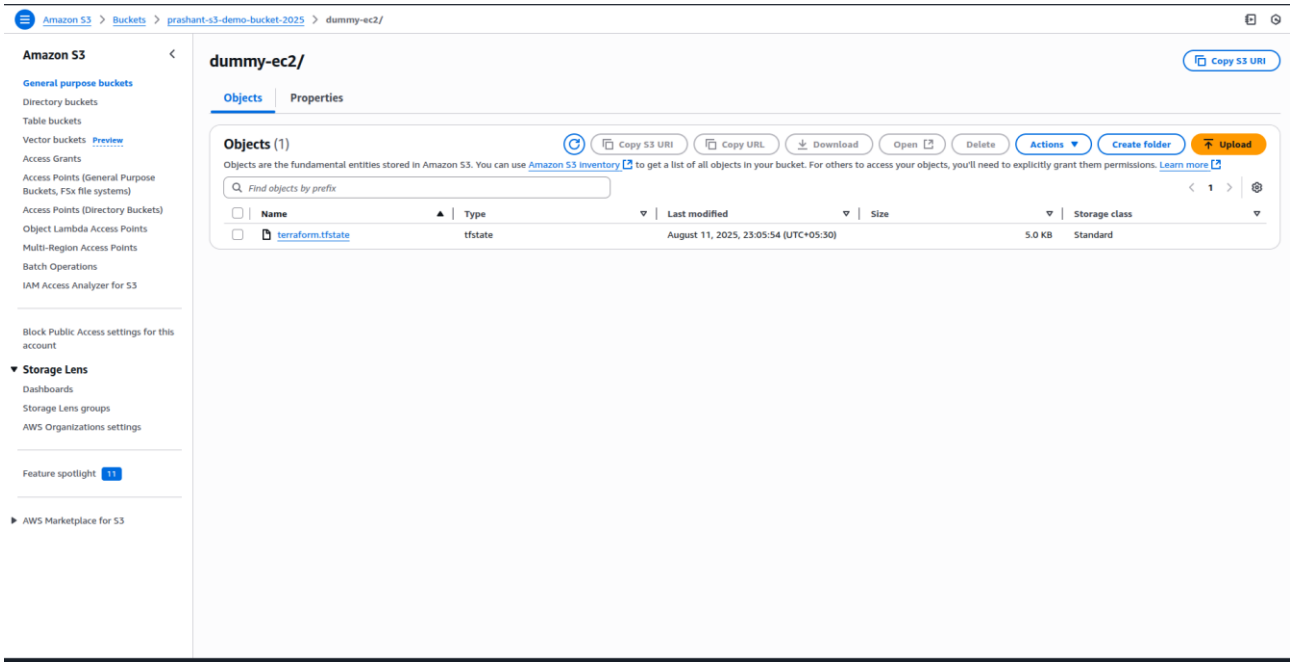
Step 2: S3 Backend Configuration

2.1 Created S3 Bucket for State Management Used AWS CLI to create S3 bucket for remote state storage:

```
aws s3 mb s3://prashant-s3-demo-bucket
```

2.2 Configured Terraform Backend Created `backend.tf` for remote state management:

```
terraform {
  backend "s3" {
    bucket = "prashant-s3-demo-bucket-2025"
    key    = "dummy-ec2/terraform.tfstate"
    region = "us-east-1"
    encrypt = true
  }
}
```



Step 3: EC2 Module Development

3.1 Created EC2 Module Structure Developed modular architecture with separate files:

modules/ec2/main.tf:

```
resource "aws_instance" "example_instance" {
    ami           = var.ami_value
    instance_type = var.instance_type_value
    subnet_id     = var.subnet_id_value

}
```

modules/ec2/variables.tf:

```
variable "ami_value" {
    description = "AMI ID for the EC2 instance"
    type        = string
```

```
}

variable "instance_type_value" {
    description = "Instance type for EC2"
    type        = string
}

variable "subnet_id_value" {
    description = "Subnet ID for EC2 placement"
    type        = string
}
```

modules/ec2/outputs.tf:

```
output "public_ip" {
    description = "Public IP of the EC2 instance"
    value       = aws_instance.example_instance.public_ip
}

output "instance_id" {
    description = "ID of the EC2 instance"
    value       = aws_instance.example_instance.id
}

output "private_ip" {
    description = "Private IP of the EC2 instance"
    value       = aws_instance.example_instance.private_ip
}
```

Step 4: Root Module Configuration

4.1 Main Configuration Created `main.tf` in root directory to call the EC2 module:

```
module "ec2_instance" {
    source = "./modules/ec2"
    ami_value = "ami-0e001c9271cf7f3b9"
    instance_type_value = "t2.micro"
    subnet_id_value = "subnet-0654a462a6ac7ea72"
}
```

4.2 Output Variables Implementation Created `outputs.tf` to expose module outputs:

```
output "ec2_instance" {
    value = {
        public_ip = module.ec2_instance.public_ip
        instance_id = module.ec2_instance.instance_id
        private_ip = module.ec2_instance.private_ip
    }
}
```

4.3 Variable Configuration Set up `terraform.tfvars` with actual values for the infrastructure.

Step 5: Infrastructure Deployment

5.1 Terraform Initialization

```
terraform init
```

```
→ terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v6.8.0

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.
```

5.2 Infrastructure Planning

`terraform plan`

```
      + 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)
      + outpost_arn                         = (known after apply)
      + password_data                       = (known after apply)
      + placement_group                     = (known after apply)
      + placement_partition_number          = (known after apply)
      + primary_network_interface_id        = (known after apply)
      + private_dns                         = (known after apply)
      + private_ip                          = (known after apply)
      + public_dns                          = (known after apply)
      + public_ip                           = (known after apply)
      + region                              = "us-east-1"
      + secondary_private_ips               = (known after apply)
      + security_groups                     = (known after apply)
      + source_dest_check                   = true
      + spot_instance_request_id            = (known after apply)
      + subnet_id                           = "subnet-0654a462a6ac7ea72"
      + tags_all                            = (known after apply)
      + tenancy                             = (known after apply)
      + user_data_base64                    = (known after apply)
      + user_data_replace_on_change         = false
      + vpc_security_group_ids              = (known after apply)

      + capacity_reservation_specification (known after apply)

      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + ec2_instance = {
      + instance_id = (known after apply)
      + private_ip  = (known after apply)
      + public_ip   = (known after apply)
    }
```

5.3 Infrastructure Deployment

`terraform apply`

```
      + tags_all                           = (known after apply)
      + tenancy                            = (known after apply)
      + user_data_base64                   = (known after apply)
      + user_data_replace_on_change        = false
      + vpc_security_group_ids             = (known after apply)

      + capacity_reservation_specification (known after apply)

      + cpu_options (known after apply)

      + ebs_block_device (known after apply)

      + enclave_options (known after apply)

      + ephemeral_block_device (known after apply)

      + instance_market_options (known after apply)

      + maintenance_options (known after apply)

      + metadata_options (known after apply)

      + network_interface (known after apply)

      + private_dns_name_options (known after apply)

      + root_block_device (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + ec2_instance = {
      + instance_id = (known after apply)
      + private_ip  = (known after apply)
      + public_ip   = (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.ec2_instance.aws_instance.example_instance: Creating...
module.ec2_instance.aws_instance.example_instance: Still creating... [00m10s elapsed]
module.ec2_instance.aws_instance.example_instance: Still creating... [00m20s elapsed]
module.ec2_instance.aws_instance.example_instance: Still creating... [00m30s elapsed]
module.ec2_instance.aws_instance.example_instance: Creation complete after 39s [id=i-081056ca5f66c9db6]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

ec2_instance = {
  "instance_id" = "i-081056ca5f66c9db6"
  "private_ip" = "10.0.3.24"
  "public_ip" = ""
}
```

5.4 Infrastructure Destroy

terraform destroy

```
      - enclave_options {
          - enabled = false -> null
        }

      - maintenance_options {
          - auto_recovery = "default" -> null
        }

      - metadata_options {
          - http_endpoint               = "enabled" -> null
          - http_protocol_ipv6          = "disabled" -> null
          - http_put_response_hop_limit = 2 -> null
          - http_tokens                 = "required" -> null
          - instance_metadata_tags      = "disabled" -> null
        }

      - private_dns_name_options {
          - enable_resource_name_dns_a_record    = false -> null
          - enable_resource_name_dns_aaaa_record = false -> null
          - hostname_type                        = "ip-name" -> null
        }

      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/sda1" -> null
          - encrypted             = false -> null
          - iops                  = 3000 -> null
          - tags                  = {} -> null
          - tags_all              = {} -> null
          - throughput            = 125 -> null
          - volume_id             = "vol-0e2b0e9cb8703f859" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp3" -> null
          # (1 unchanged attribute hidden)
        }
    }

Plan: 0 to add, 0 to change, 1 to destroy.

Changes to Outputs:
  - ec2_instance = {
      - instance_id = "i-06623ed69c35f2bcd"
      - private_ip  = "10.0.3.178"
      - public_ip   = ""
    } -> null

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.ec2_instance.aws_instance.example_instance: Destroying... [id=i-06623ed69c35f2bcd]
module.ec2_instance.aws_instance.example_instance: Still destroying... [id=i-06623ed69c35f2bcd, 00m10s elapsed]
module.ec2_instance.aws_instance.example_instance: Still destroying... [id=i-06623ed69c35f2bcd, 00m20s elapsed]
module.ec2_instance.aws_instance.example_instance: Still destroying... [id=i-06623ed69c35f2bcd, 00m30s elapsed]
module.ec2_instance.aws_instance.example_instance: Destruction complete after 36s

Destroy complete! Resources: 1 destroyed.
```

5.5 Verification of Outputs Screenshot Location: *[Insert screenshot showing output variables after apply]*

Step 6: AWS Console Verification

6.1 EC2 Instance Verification

- Verified EC2 instance creation in AWS Console
- Confirmed instance specifications (Ubuntu 22.04 LTS, t2.micro)