Detailed Report

1. Terraform Initialization

Command: `terraform init`

- **Purpose**: Initializes Terraform in the working directory.
- **Output**: Initializes provider plugins and downloads necessary modules.

2. Terraform Configuration

Files: `main.tf`, `variables.tf`, `outputs.tf`, etc.

- **Purpose**: Defines the infrastructure components using Terraform configuration files.
- **Details**:
  - **`main.tf`**: Defines AWS resources like VPC, subnets, EC2 instances, security groups, etc.
  - **`variables.tf`**: Declares input variables used in `main.tf`.
  - **`outputs.tf`**: Declares output values to be displayed after deployment.

3. Terraform Plan Generation

Command: `terraform plan`

- **Purpose**: Generates an execution plan describing what Terraform will do.
- **Output**: Detailed listing of changes, including resources to add, change, or destroy.

Example Changes:

- **AWS VPC**: Creates a VPC named `KCVPC` with CIDR block `10.0.0.0/16`.
- **Subnets**: Defines public (`PublicSubnet`) and private (`PrivateSubnet`) subnets.
- **Route Tables**: Configures public and private route tables with appropriate routes.
- **Security Groups**: Sets up security groups (`PublicSecurityGroup`, `PrivateSecurityGroup`) with specific rules.
- **NACLs**: Defines network ACLs (`PublicSubnetNACL`, `PrivateSubnetNACL`) for inbound and outbound traffic control.
- **EC2 Instances**: Plans creation of Ubuntu EC2 instances (`PublicInstance`, `PrivateInstance`) in respective subnets.

4. Terraform Apply

Command: `terraform apply`

- **Purpose**: Applies the execution plan to make changes to infrastructure.
- **Process**:

    - Creates AWS resources as per `terraform plan`.
    - Displays real-time status of resource creation.
    - Outputs final state and details of created resources.

5. Post-Deployment Verification

Actions:

- **Access**: Verify access to EC2 instances (`PublicInstance`, `PrivateInstance`) via SSH.
- **Connectivity**: Confirm connectivity between instances in public and private subnets.
- **Functionality**: Test application functionality hosted on EC2 instances (e.g., web server on `PublicInstance`, database on `PrivateInstance`).

6. Terraform Destroy

Command: `terraform destroy`

- **Purpose**: Destroys all resources created by Terraform.
- **Use Cases**:
    - Clean up resources after testing or when not needed.
    - Ensure no lingering costs from unused infrastructure.