

Terraform AWS Infrastructure with Jenkins CI/CD

Complete Project Documentation

This document contains the complete steps followed during the creation, automation, and deployment of AWS infrastructure using Terraform modules and Jenkins CI/CD pipeline. It also explains all files, modules, code structure, and AWS resources created.

At each step, insert screenshots exactly where mentioned.

1. Project Overview

This project provisions a complete AWS infrastructure using Terraform and automates the deployment using Jenkins CI/CD pipeline.

AWS resources created:

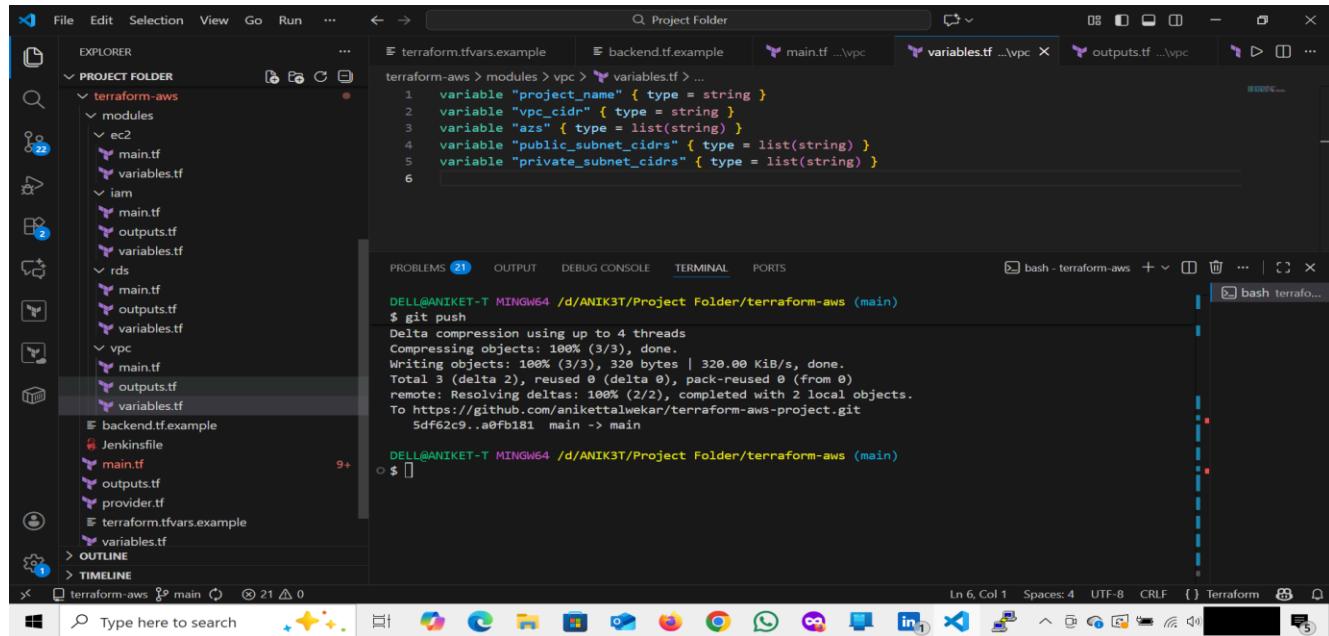
1. VPC
2. Two Public Subnets
3. Two Private Subnets
4. Internet Gateway
5. NAT Gateway with Elastic IP
6. Route Tables (Public and Private)
7. Security Groups
8. RDS MySQL Instance
9. IAM Role and Instance Profile
10. EC2 Instances (2 instances in public subnets)

Automation:

1. Terraform code stored on GitHub
2. Jenkins pipeline pulls code
3. Terraform Init
4. Terraform Plan
5. Terraform Apply
6. Infrastructure creation
7. Outputs printed

2. Create Project Folder Structure

Create a folder in VS Code:



The project structure:

```
terraform-aws-project/
  versions.tf
  provider.tf
  variables.tf
  main.tf
  outputs.tf
  terraform.tfvars.example
  backend.tf.example
```

```
modules/
  vpc/
    main.tf
    variables.tf
    outputs.tf
```

```
iam/
  main.tf
  variables.tf
  outputs.tf
```

```
ec2/
  main.tf
  variables.tf
```

```
rds/
  main.tf
  variables.tf
  outputs.tf
```

3. Root Module Files Explanation

The screenshot shows a code editor interface with multiple tabs open. On the left, the 'EXPLORER' panel displays the 'PROJECT FOLDER' structure, which includes modules for ec2, iam, rds, and vpc, along with Jenkinsfile, main.tf, outputs.tf, provider.tf, variables.tf, backend.tf.example, and terraform.tfvars.example. The right side of the screen has three main code editors:

- provider.tf:** Contains the AWS provider configuration with the 'aws' provider block.
- versions.tf:** Contains the Terraform version requirement and provider version requirement.
- main.tf:** Contains the root-level variable declarations for project_name, region, vpc_cidr, public_subnets, private_subnets, instance_type, db_username, db_password, and key_name.

The bottom status bar shows the terminal output of a git push command, indicating successful commit and push to a GitHub repository.

versions.tf

Specifies Terraform version and AWS provider version.

provider.tf

Configures AWS provider with region.

variables.tf

Contains all root-level variables used across modules:

- project_name
- region
- vpc_cidr
- public_subnets
- private_subnets
- instance_type
- db_username
- db_password
- key_name (ec2 key pair)

This screenshot shows the same code editor environment as the previous one, but the focus is on the 'variables.tf' file. The 'EXPLORER' panel shows the same project structure. The right side of the screen displays the 'variables.tf' file content, which defines variables for project_name, region, vpc_cidr, azs, public_subnet_cidrs, and private_subnet_cidrs.

main.tf

Calls all modules (vpc, iam, ec2, rds) and passes variables to each module.

```
provider "aws" {
    region = "us-west-2"
}

module "vpc" {
    source      = "./modules/vpc"
    project_name = var.project_name
    vpc_cidr   = var.vpc_cidr
    azs        = var.azs
    public_subnet_cidrs = var.public_subnet_cidrs
    private_subnet_cidrs = var.private_subnet_cidrs
}

module "iam" {
    source      = "./modules/iam"
    project_name = var.project_name
}

module "ec2" {
    source      = "./modules/ec2"
    project_name = var.project_name
    subnet_ids = module.vpc.public_subnet_ids
    ami        = data.aws_ami.ubuntu.id
    instance_type = var.instance_type
    key_name   = var.key_name
    iam_instance_profile = module.iam.instance_profile_name
    vpc_id     = module.vpc.vpc_id
}

module "rds" {
    source      = "./modules/rds"
    project_name = var.project_name
}
```

outputs.tf

Prints final resource outputs:

- vpc_id
- public_subnet_ids
- private_subnet_ids
- rds_endpoint
- ec2_public_ips

```
output "vpc_id" {
    value = module.vpc.vpc_id
}

output "public_subnet_ids" {
    value = module.vpc.public_subnet_ids
}

output "private_subnet_ids" {
    value = module.vpc.private_subnet_ids
}

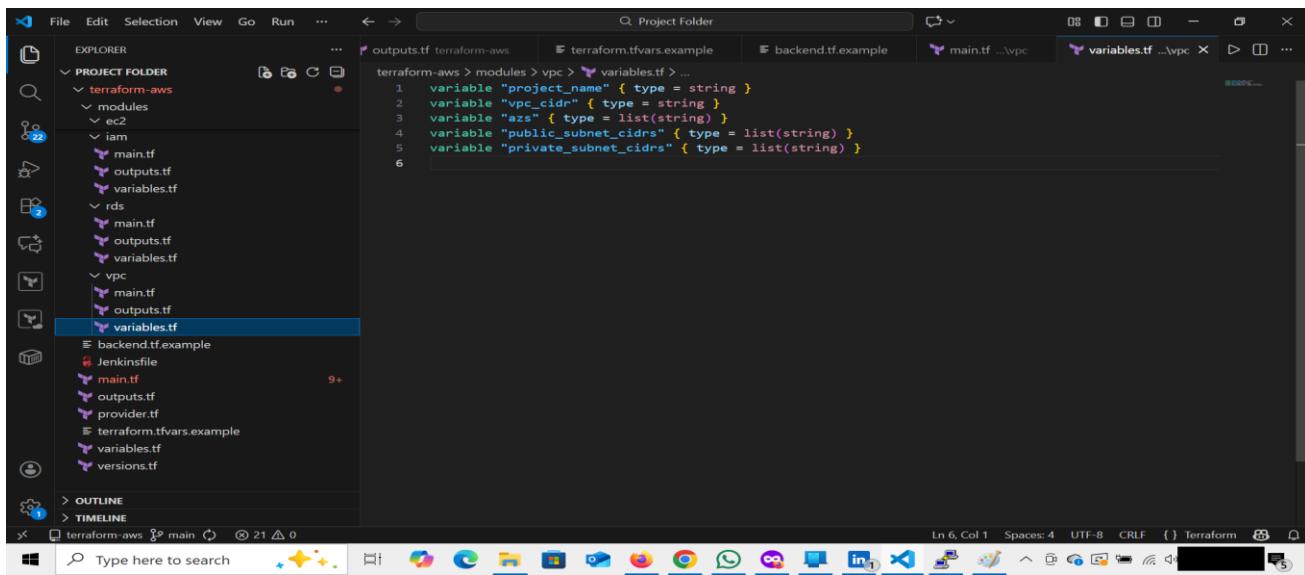
output "ec2_public_ips" {
    value = module.ec2.public_ips
}

output "rds_endpoint" {
    value = module.rds.db_endpoint
}
```

4. VPC Module Files

vpc/variables.tf

Defines inputs required for VPC creation.

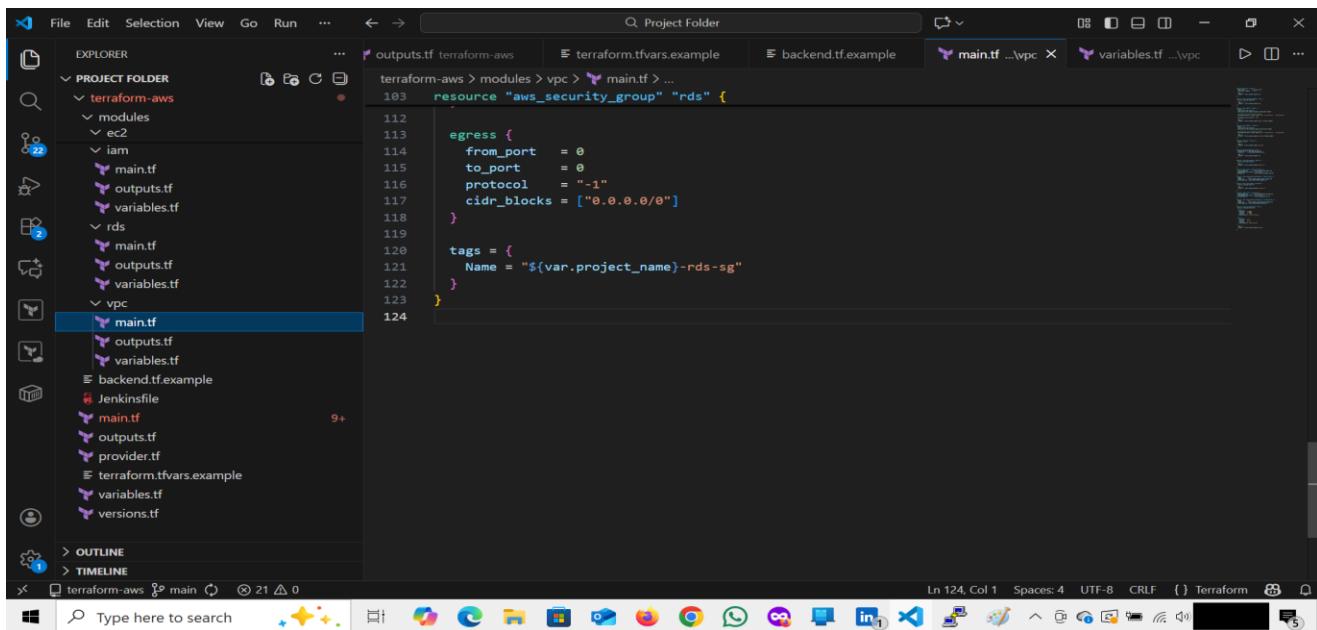


```
variable "project_name" { type = string }
variable "vpc_cidr" { type = string }
variable "azs" { type = list(string) }
variable "public_subnet_cidrs" { type = list(string) }
variable "private_subnet_cidrs" { type = list(string) }
```

vpc/main.tf

Creates the entire VPC architecture:

- VPC
- Subnets
- Internet Gateway
- NAT Gateway
- Route Tables



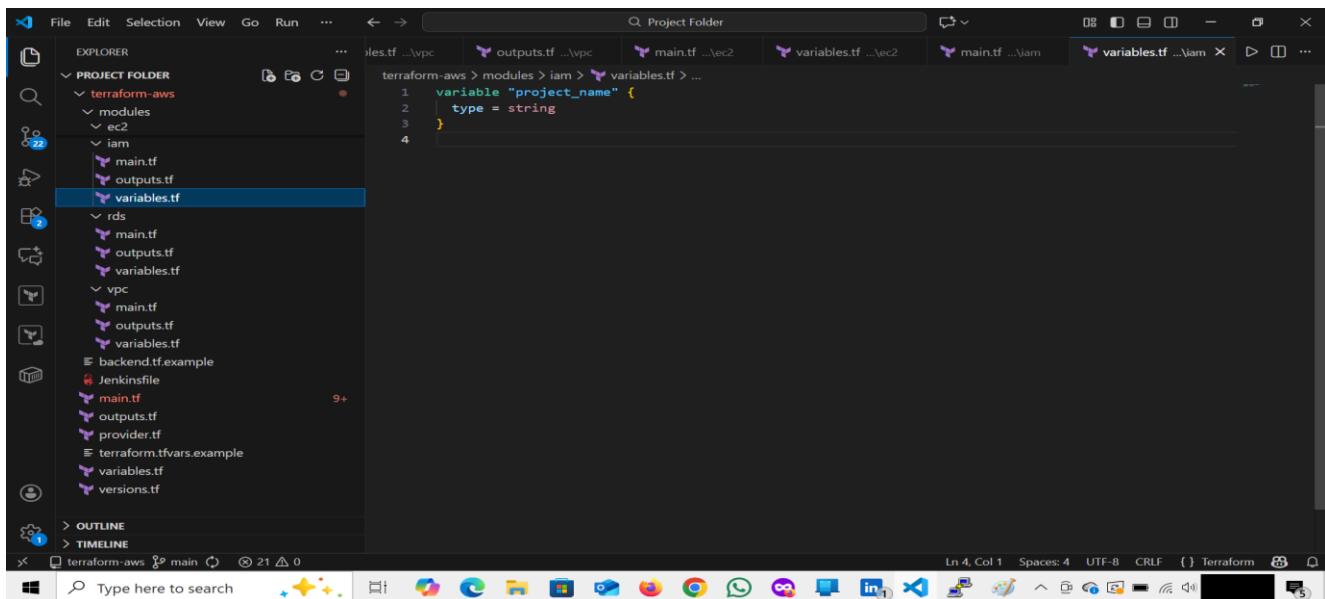
```
resource "aws_security_group" "rds" {
    egress {
        from_port   = 0
        to_port     = 0
        protocol    = "-1"
        cidr_blocks = ["0.0.0.0/0"]
    }

    tags = {
        Name = "${var.project_name}-rds-sg"
    }
}
```

5. IAM Module

iam/variables.tf

Defines project_name.



The screenshot shows the Visual Studio Code interface with the following details:

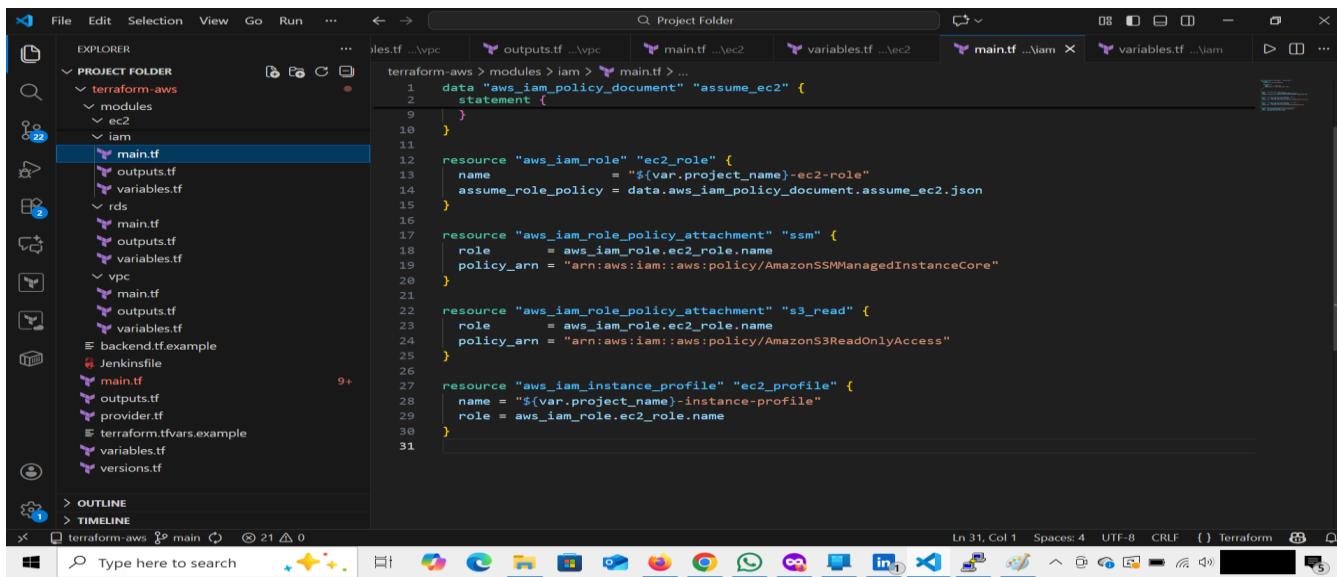
- File Explorer:** Shows the project structure under "PROJECT FOLDER". The "variables.tf" file in the "iam" module is selected and highlighted in blue.
- Editor:** Displays the contents of the "variables.tf" file:

```
variable "project_name" {
  type = string
}
```
- Bottom Bar:** Includes a search bar ("Type here to search"), a ribbon of icons, and the Terraform extension icon.

iam/main.tf

Creates:

- IAM role
- IAM instance profile
- IAM policy attachments (S3 ReadOnly, SSM Managed)



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "PROJECT FOLDER". The "main.tf" file in the "iam" module is selected and highlighted in blue.
- Editor:** Displays the contents of the "main.tf" file:

```
data "aws_iam_policy_document" "assume_ec2" {
  statement {
    ...
  }
}

resource "aws_iam_role" "ec2_role" {
  name      = "${var.project_name}-ec2-role"
  assume_role_policy = data.aws_iam_policy_document.assume_ec2.json
}

resource "aws_iam_role_policy_attachment" "ssm" {
  role      = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore"
}

resource "aws_iam_role_policy_attachment" "s3_read" {
  role      = aws_iam_role.ec2_role.name
  policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
}

resource "aws_iam_instance_profile" "ec2_profile" {
  name      = "${var.project_name}-instance-profile"
  role     = aws_iam_role.ec2_role.name
}
```
- Bottom Bar:** Includes a search bar ("Type here to search"), a ribbon of icons, and the Terraform extension icon.

iam/outputs.tf

Outputs the instance profile.

```
File Edit Selection View Go Run ... < > Project Folder utstf ...vpc main.tf ...ec2 variables.tf ...iam main.tf ...iam variables.tf ...iam outputs.tf ...iam
EXPLORER PROJECT FOLDER terraform-aws modules ec2 iam main.tf outputs.tf variables.tf vpc main.tf outputs.tf variables.tf backend.tf.example Jenkinsfile main.tf outputs.tf provider.tf terraform.tfvars.example variables.tf versions.tf
> OUTLINE > TIMELINE
terraform-aws main 21 △ 0
1 output "instance_profile_name" {
2   value = aws_iam_instance_profile.ec2_profile.name
3 }
4
```

The screenshot shows a code editor window with several tabs at the top: 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', etc. Below the tabs is a search bar and a 'Project Folder' dropdown. The main area displays a file named 'outputs.tf' which contains a single line of Terraform code: 'output "instance_profile_name" { value = aws_iam_instance_profile.ec2_profile.name }'. The code editor interface includes a left sidebar labeled 'EXPLORER' with a 'PROJECT FOLDER' section containing 'terraform-aws', 'modules', 'ec2', 'iam', and other files like 'main.tf', 'variables.tf', 'vpc', 'rds', etc. At the bottom, there's an 'OUTLINE' and 'TIMELINE' panel, and a taskbar with various icons.

6. EC2 Module

ec2/variables.tf

Defines inputs such as:

- ami
- instance_type
- subnet_ids
- iam_instance_profile
- key_name

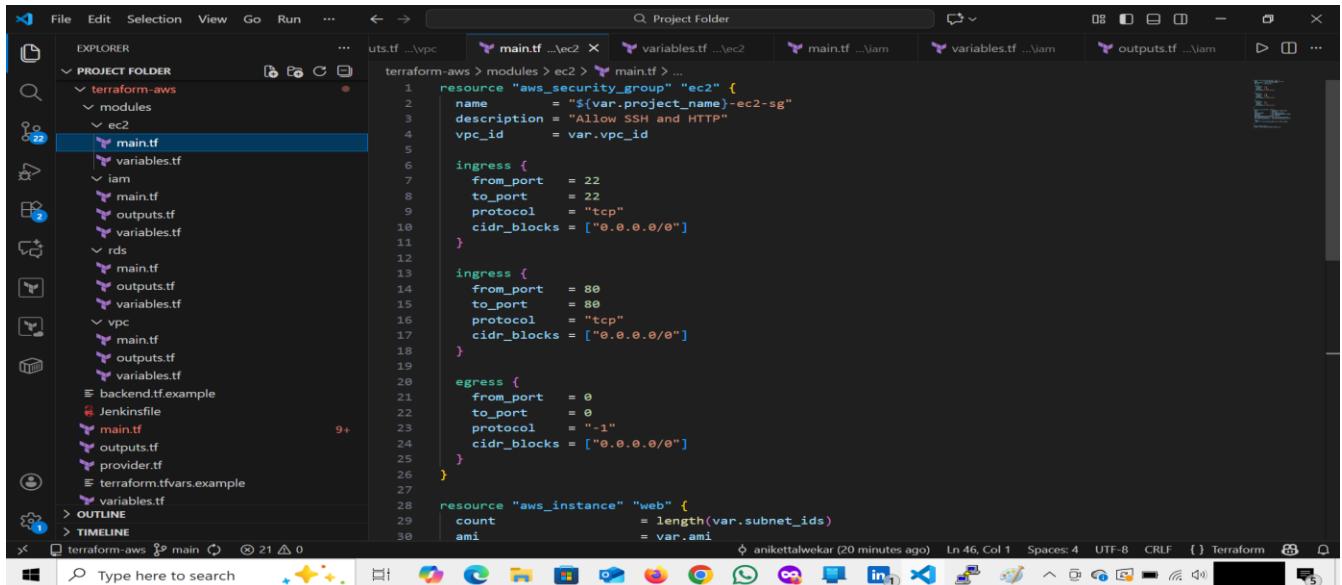
```
File Edit Selection View Go Run ... < > Project Folder utstf ...vpc main.tf ...ec2 variables.tf ...ec2 main.tf ...iam variables.tf ...iam outputs.tf ...iam
EXPLORER PROJECT FOLDER terraform-aws modules ec2 variables.tf variable "iam_instance_profile"
1 variable "project_name" { type = string }
2 variable "subnet_ids" { type = list(string) }
3 variable "ami" { type = string }
4 variable "instance_type" { type = string }
5 variable "key_name" { type = string }
6 variable "iam_instance_profile" { type = string }
7 variable "vpc_id" { type = string }
8
9+
main.tf outputs.tf provider.tf terraform.tfvars.example variables.tf
backend.tf.example Jenkinsfile
main.outputs.provider.terraform
> OUTLINE > TIMELINE
terraform-aws main 21 △ 0
aniketbalwekar (2 hours ago) Ln 6, Col 1 Spaces: 4 UTF-8 CRLF ( ) Terraform
```

The screenshot shows a code editor window with tabs for 'variables.tf' in the 'ec2' module. The code defines several variables: 'project_name' (string), 'subnet_ids' (list of strings), 'ami' (string), 'instance_type' (string), 'key_name' (string), and 'iam_instance_profile' (string). The code editor interface is similar to the previous screenshot, with a left sidebar showing the project structure and a taskbar at the bottom.

ec2/main.tf

Creates:

- Security group for EC2
- Two EC2 instances in two public subnets



The screenshot shows a code editor with the main.tf file selected in the left sidebar under the PROJECT FOLDER. The main.tf file contains Terraform configuration for creating an AWS security group and two EC2 instances. The code includes resource declarations for aws_security_group and aws_instance, along with ingress and egress rules.

```
resource "aws_security_group" "ec2" {
  name     = "${var.project_name}-ec2-sg"
  description = "Allow SSH and HTTP"
  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"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

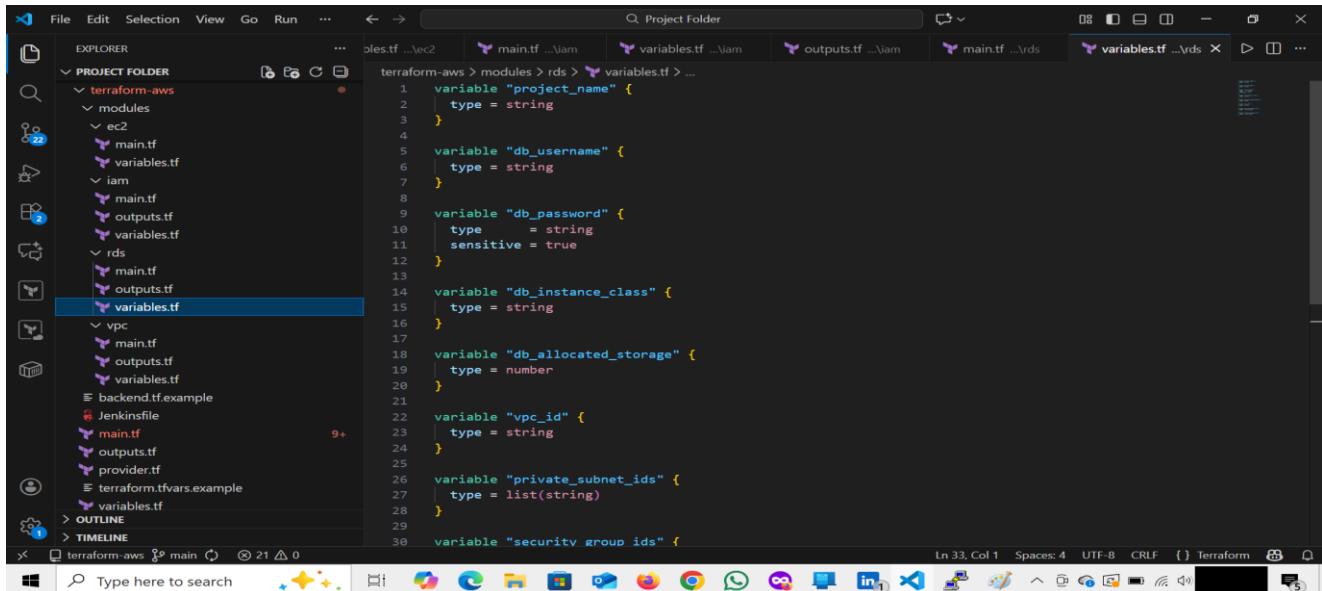
resource "aws_instance" "web" {
  count      = length(var.subnet_ids)
  ami        = var.ami
}
```

7. RDS Module

rds/variables.tf

Defines:

- db_username
- db_password
- private_subnet_ids



The screenshot shows a code editor with the variables.tf file selected in the left sidebar under the PROJECT FOLDER. The variables.tf file defines several variables used for configuring an AWS RDS instance, including project_name, db_username, db_password, db_instance_class, db_allocated_storage, vpc_id, private_subnet_ids, and security_group_ids.

```
variable "project_name" {
  type = string
}

variable "db_username" {
  type = string
}

variable "db_password" {
  type = string
  sensitive = true
}

variable "db_instance_class" {
  type = string
}

variable "db_allocated_storage" {
  type = number
}

variable "vpc_id" {
  type = string
}

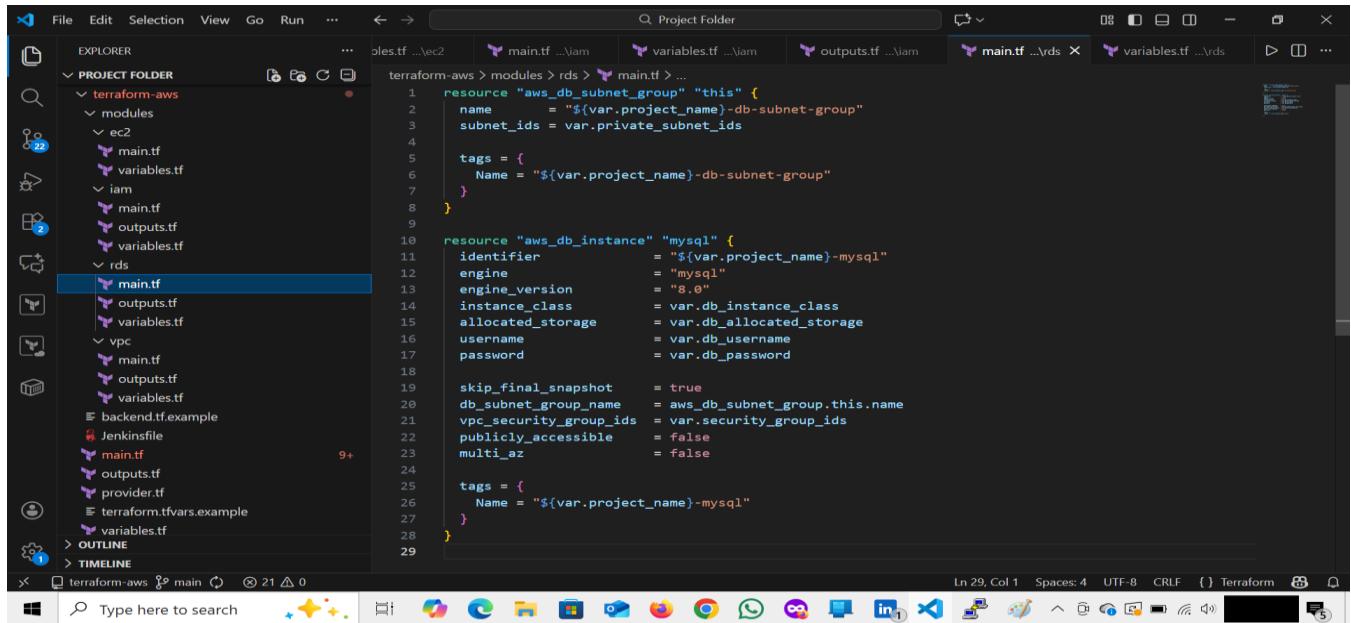
variable "private_subnet_ids" {
  type = list(string)
}

variable "security_group_ids" {
```

rds/main.tf

Creates:

- DB subnet group
- RDS MySQL instance



The screenshot shows the VS Code interface with the Terraform main.tf file open. The code defines an AWS DB subnet group and an AWS DB instance (MySQL). The subnet group is named based on the project name, and the instance has specific engine, version, and security group configurations. It also specifies a final snapshot and multi-AZ support.

```
resource "aws_db_subnet_group" "this" {
  name      = "${var.project_name}-db-subnet-group"
  subnet_ids = var.private_subnet_ids

  tags = {
    Name = "${var.project_name}-db-subnet-group"
  }
}

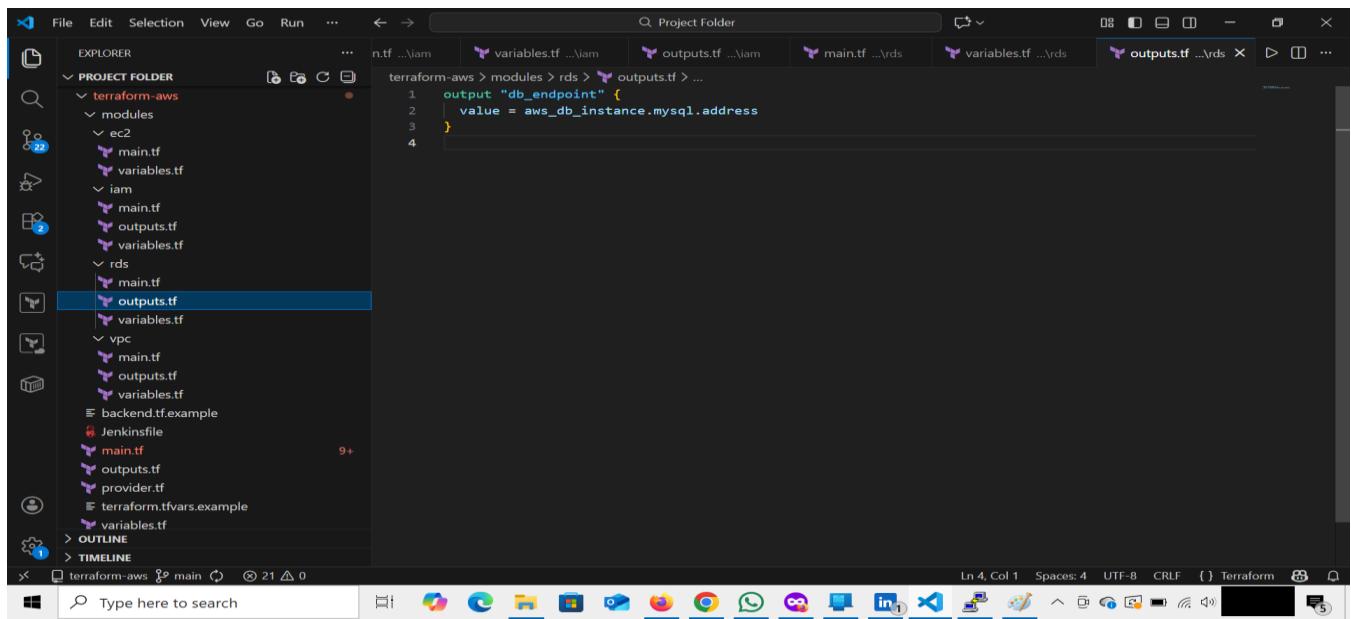
resource "aws_db_instance" "mysql" {
  identifier      = "${var.project_name}-mysql"
  engine          = "mysql"
  engine_version = "8.0"
  instance_class  = var.db_instance_class
  allocated_storage = var.db_allocated_storage
  username        = var.db_username
  password        = var.db_password

  skip_final_snapshot = true
  db_subnet_group_name = aws_db_subnet_group.this.name
  vpc_security_group_ids = var.security_group_ids
  publicly_accessible = false
  multi_az           = false

  tags = {
    Name = "${var.project_name}-mysql"
  }
}
```

rds/outputs.tf

Outputs database endpoint.



The screenshot shows the VS Code interface with the Terraform outputs.tf file open. It contains a single output block that outputs the address of the AWS DB instance.

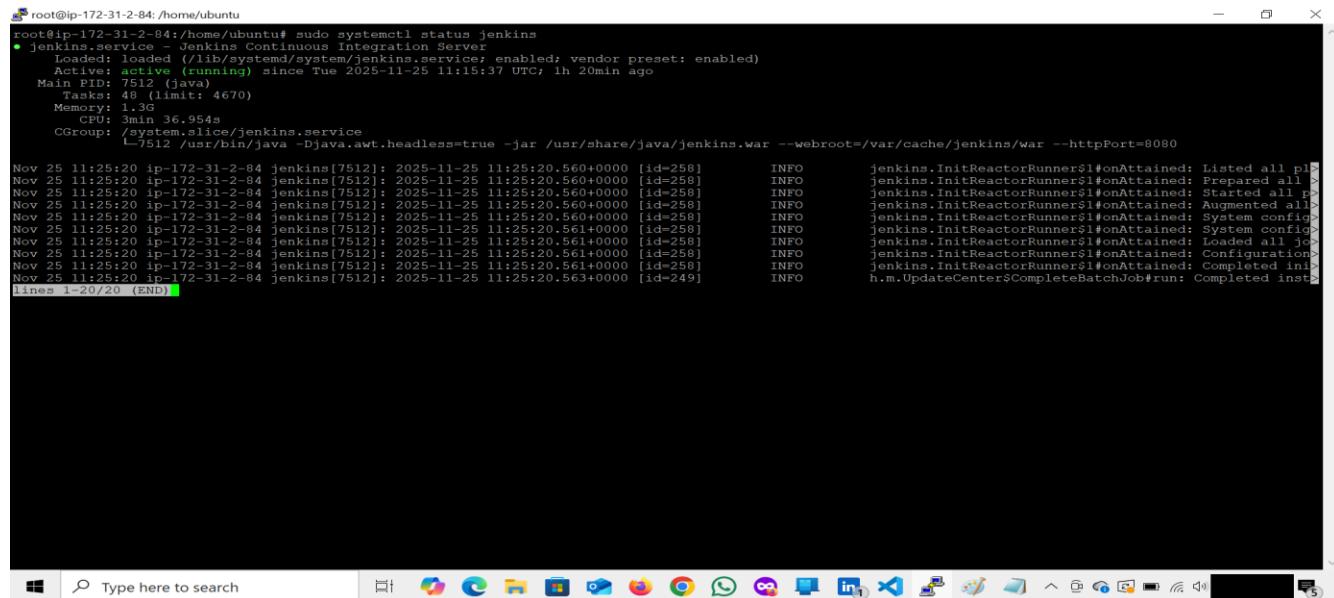
```
output "db_endpoint" {
  value = aws_db_instance.mysql.address
}
```

8. Push Code to GitHub

Commands used:

```
git init
git add .
git commit -m "Initial Terraform project"
git branch -M main
git remote add origin https://github.com/anikettalwekar/terraform-aws-project.git
git push -u origin main
```

9. Create Jenkins EC2 Server



A screenshot of a Windows Command Prompt window titled 'root@ip-172-31-2-84: /home/ubuntu'. The window displays log output from a Jenkins service running on port 8080. The logs show Jenkins starting up, loading configuration, and initializing reactor runners. The command entered was 'sudo systemctl status jenkins'.

```
root@ip-172-31-2-84:/home/ubuntu$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2025-11-25 11:15:37 UTC; 1h 20min ago
       Main PID: 7512 (java)
          Tasks: 48 (limit: 4670)
        Memory: 1.304GiB
         CPU: 3min 36.954s
        CGroup: /system.slice/jenkins.service
                 └─7512 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.560+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Listed all pl...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.560+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Prepared all ...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.560+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Started all ...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.560+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Registered all ...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.560+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: System config...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.561+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: System config...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.561+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Loaded all jo...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.561+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Configuration...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.561+0000 [id=258]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed ini...
Nov 25 11:25:20 ip-172-31-2-84 jenkins[7512]: 2025-11-25 11:25:20.563+0000 [id=249]      INFO      h.m.UpdateCenter$CompleteBatchJob#run: Completed inst...
lines 1-20/20 (END)
```

Install Java and Jenkins

```
sudo apt update
sudo apt install fontconfig openjdk-21-jre -y
java -version
```

```
sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
```

```
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" \
| sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
```

```
sudo apt update
sudo apt install jenkins -y
```

```
sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl status Jenkins
```

10. Add AWS Credentials in Jenkins

Credentials type used:

AWS Credentials (Access Key + Secret Key)

The screenshot shows the Jenkins 'Credentials' management page. It lists two entries under the 'System' store, both with '(global)' domain. The first entry is 'aws-creds' with ID 'aws-creds' and name 'AKIAUHLT6FP6FIAWB2KJ'. The second entry is 'db-pass' with ID 'db-pass' and name 'RDS MySQL DB password'. Below this, there's a section titled 'Stores scoped to Jenkins' which shows a single 'System' store with '(global)' domain.

11. Create Jenkins Pipeline

Insert Screenshot: 21-jenkins-job-list.png

Pipeline uses Jenkinsfile stored in GitHub.

The screenshot shows the Jenkins 'Configuration' page for the 'terraform-aws-deploy' job. Under the 'Pipeline' tab, it is set to 'Pipeline script from SCM'. The 'SCM' dropdown is set to 'Git' and the repository URL is 'https://github.com/anikettalwekar/terraform-aws-project.git'. The 'Credentials' dropdown is set to '- none -'. At the bottom, there are 'Save' and 'Apply' buttons.

12. Jenkinsfile Workflow

Steps executed:

1. Checkout code from GitHub
2. Terraform Init
3. Terraform Plan
4. Terraform Apply
5. Print outputs

13. Jenkins Build Output

The screenshot shows the Jenkins interface for the 'terraform-aws-deploy' job. The top navigation bar includes links for 'Status', 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Stages', 'Rename', 'Pipeline Syntax', and 'Credentials'. The main content area displays a list of builds, with the last completed build (#9) highlighted. Below the build list are 'Permalinks' for various build events.

Build	Status	Time Ago
#9	Success	30 min ago
#8	Success	34 min ago
#7	Success	30 min ago
#6	Success	30 min ago
#5	Success	30 min ago
#4	Success	30 min ago
#3	Success	30 min ago
#2	Success	30 min ago
#1	Success	30 min ago

The screenshot shows the Jenkins interface for the 'terraform-aws-deploy' job, specifically the console output for build #9. The left sidebar lists options like 'Status', 'Changes', 'Console Output' (which is selected), 'Edit Build Information', 'Delete build '#9', 'Timings', 'Git Build Data', 'Pipeline Overview', 'Restart from Stage', 'Replay', 'Pipeline Steps', 'Workspaces', and 'Previous Build'. The right pane displays the terminal logs for the build, starting with the command 'git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/terraform-aws-deploy/.git # timeout=10'.

```
Started by user aniket
Obtained Jenkinsfile from git https://github.com/anikettalwekar/terraform-aws-project.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/terraform-aws-deploy
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/terraform-aws-deploy/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/anikettalwekar/terraform-aws-project.git # timeout=10
Fetching upstream changes from https://github.com/anikettalwekar/terraform-aws-project.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/anikettalwekar/terraform-aws-project.git
```

The screenshot shows a Jenkins job named "terraform-aws-deploy" with a build number of 9. The "Console Output" tab is selected. The output displays the Terraform configuration code and its execution results. The configuration defines private and public subnet IDs, an RDS endpoint, and a VPC ID. The pipeline stage is completed successfully.

```

private_subnet_ids = [
    "subnet-0e1da3414964c9ef6",
    "subnet-0e514b3b5ded2939d",
]
public_subnet_ids = [
    "subnet-044d7857d7e01dc1a",
    "subnet-0f2b7acc855c70e3b",
]
rds_endpoint = "demo-mysql.c9qu6gs06l19.ap-south-1.rds.amazonaws.com"
vpc_id = "vpc-00ca71705391531ef"
[Pipeline] }
[Pipeline] // withAWS
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Final Terraform outputs included:

- EC2 public IPs
- VPC ID
- Subnet IDs
- RDS Endpoint

14. AWS Console Validation

VPC

The screenshot shows the AWS VPC console with two VPCs listed: "demo-vpc" and a default VPC. Both are in an "Available" state. The "demo-vpc" was created by terraform. The interface includes a search bar, a table with columns for Name, VPC ID, State, Encryption controls, Encryption control, Block Public, and IPv4 range, and a "Create VPC" button.

Name	VPC ID	State	Encryption controls	Encryption control	Block Public...	IPv4
-	vpc-02311120e9f9a1a85	Available	-	-	Off	172
demo-vpc	vpc-00ca71705391531ef	Available	-	-	Off	10.0

Select a VPC above

Subnets

The screenshot shows the AWS VPC Subnets page. At the top, there are tabs for 'Console Home', 'VPC | ap-south-1', 'Databases | Aurora and...', 'Create access key | IAM', 'Instances | EC2 | ap-so...', and 'Private browsing'. The main heading is 'Subnets (4) Info'. A search bar says 'Find subnets by attribute or tag' and a filter says 'VPC : vpc-00ca71705391531ef'. The table has columns: Name, Subnet ID, State, VPC, Block Public..., and IPv4 CIDR. The data is as follows:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
demo-private-0	subnet-0e1da3414964c9ef6	Available	vpc-00ca71705391531ef dem...	Off	10.0.101.0/24
demo-public-0	subnet-044d7857d7e01dc1a	Available	vpc-00ca71705391531ef dem...	Off	10.0.1.0/24
demo-public-1	subnet-0f2b7acc855c70e3b	Available	vpc-00ca71705391531ef dem...	Off	10.0.2.0/24
demo-private-1	subnet-0e514b3b5ded2939d	Available	vpc-00ca71705391531ef dem...	Off	10.0.102.0/24

Select a subnet

The screenshot shows the AWS CloudShell interface. At the top, there are tabs for 'CloudShell', 'Feedback', and 'Console Mobile App'. The main search bar says 'Type here to search'. Below it is a toolbar with various icons.

Internet Gateway

The screenshot shows the AWS VPC Internet Gateways page. At the top, there are tabs for 'Console Home', 'VPC | ap-south-1', 'Databases | Aurora and...', 'Create access key | IAM', 'Instances | EC2 | ap-so...', and 'Private browsing'. The main heading is 'Internet gateways (1) Info'. A search bar says 'Find internet gateways by attribute or tag' and a filter says 'VPC ID : vpc-00ca71705391531ef'. The table has columns: Name, Internet gateway ID, State, and VPC ID. The data is as follows:

Name	Internet gateway ID	State	VPC ID
demo-igw	igw-0878b0aa50792095a	Attached	vpc-00ca71705391531ef dem...

In the left sidebar, under 'Virtual private cloud', there is a section for 'Internet gateways' which includes 'Egress-only internet gateways', 'DHCP option sets', and 'Elastic IPs'. A note at the bottom says 'Select an internet gateway above'.

NAT Gateway

The screenshot shows the AWS VPC NAT gateways console. On the left, a sidebar lists various VPC components under 'Virtual private cloud'. The main area displays a table titled 'NAT gateways (1)'. The table has columns for Name, NAT gateway ID, Connectivity..., State, State message, and Availability. One row is shown, corresponding to the 'demo-nat' gateway.

Name	NAT gateway ID	Connectivity...	State	State message	Availability
demo-nat	nat-0d66a5d6778c5199d	Public	Available	-	Zonal

Route Tables

The screenshot shows the AWS VPC Route tables console. The sidebar lists 'Route tables' under 'Virtual private cloud'. The main area displays a table titled 'Route tables (3)'. The table has columns for Name, Route table ID, Explicit subnet associ..., Edge associations, and Main. Three route tables are listed: 'demo-public-rt' (rtb-079a305ab60d49954), 'demo-private-rt' (rtb-001af23527230ebd3), and a third unnamed entry (rtb-01eeb93fcdd8beebd). The unnamed entry is marked as the 'Main' route table.

Name	Route table ID	Explicit subnet associ...	Edge associations	Main
demo-public-rt	rtb-079a305ab60d49954	2 subnets	-	No
demo-private-rt	rtb-001af23527230ebd3	2 subnets	-	No
-	rtb-01eeb93fcdd8beebd	-	-	Yes

EC2 Instances

The screenshot shows the AWS EC2 Instances console. The sidebar lists 'Instances' under 'EC2'. The main area displays a table titled 'Instances (3)'. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IP. Three instances are listed: 'jenkins-server' (i-07eca8d302124760f), 'demo-web-1' (i-02642283848ca18da), and 'demo-web-0' (i-0015971ccaa49aada). All instances are in a 'Running' state.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
jenkins-server	i-07eca8d302124760f	Running	t2.medium	2/2 checks passed	View alarms	ap-south-1b	ec2-43-2
demo-web-1	i-02642283848ca18da	Running	t3.micro	3/3 checks passed	View alarms	ap-south-1b	-
demo-web-0	i-0015971ccaa49aada	Running	t3.micro	3/3 checks passed	View alarms	ap-south-1a	-

RDS Instance and Endpoint

The screenshot shows the AWS RDS console for the 'demo-mysql' database. The 'Summary' section displays the DB identifier (demo-mysql), status (Available), role (Instance), engine (MySQL Community), and region (ap-south-1a). The 'Connectivity & security' tab is selected, showing the endpoint (demo-mysql.c9qu6gs06l19.ap-south-1.rds.amazonaws.com), availability zone (ap-south-1a), and VPC security group (demo-rds-sg). The security group is marked as active.

Security Groups

The screenshot shows the AWS VPC console displaying a list of security groups. There are three entries:

Name	Security group ID	Security group name	VPC ID	Description
-	sg-04caaf65eaa97b2cc	default	vpc-00ca71705391531ef	default VPC secur
demo-rds-sg	sg-0bb69d0a456b28dec	demo-rds-sg	vpc-00ca71705391531ef	Managed by Terra
-	sg-01c6b0bc0207bfd3f	demo-ec2-sg	vpc-00ca71705391531ef	Allow SSH and HT

15. Cleanup Instructions

To delete the infrastructure: terraform destroy

16. Conclusion of the Project

This project demonstrates a complete, production-style AWS automation workflow using Terraform modules and Jenkins CI/CD. All major AWS foundational services were created and managed entirely through Infrastructure as Code, showing how real DevOps teams build consistent and repeatable environments.

The project covered the following key capabilities:

1. Used modular Terraform design to structure VPC, IAM, EC2, and RDS components in separate reusable modules.
2. Built a complete AWS network architecture including VPC, subnets, route tables, and Internet Gateway, NAT Gateway, and security groups.
3. Implemented IAM roles and instance profiles following least-privilege security practices.
4. Provisioned EC2 instances and RDS MySQL database through Terraform modules.
5. Stored the entire Terraform codebase in a GitHub repository for version control.
6. Used Jenkins to fully automate Terraform commands (init, plan, apply) using a pipeline stored as code.
7. Managed AWS credentials securely through Jenkins credentials system.
8. Validated infrastructure by checking outputs and AWS console resources.
9. Followed real-world debugging and error-solving steps, including IAM issues, key pair issues, and variable configurations.

This project reflects a strong understanding of DevOps workflows and follows the same process used by cloud teams in actual organizations.