AWS Infrastructure Provisioning Using Terraform

- Technologies Used
- Terraform for Infrastructure as Code (IaC)
- AWS Provider
- AWS Services:
- VPC
- Subnets
- Internet Gateway
- Route Tables
- EC2 Instances
- Security Groups
- S3 Bucket

Folder Structure

```
terraform_project/
├── main.tf              # Main configuration file
├── variables.tf         # All input variables
├── terraform.tfvars     # Actual values for variables
└── provider.tf          # AWS provider & region
```

AWS Credentials Setup (No Hardcoding)

Avoid hardcoding Access Key/Secret Key and db password .Use AWS CLI or environment variables.

Step 1: Configure AWS CLI

Terminal

```
ramsha@worker1:~/terraform_project$ aws configure
```

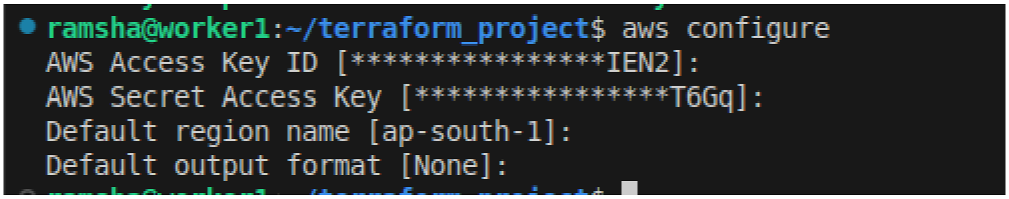
Fill in:
- Access Key ID
- Secret Access Key
- Region (e.g. us-east-1)
- Output format: json

AWS stores credentials securely at:
~/.aws/credentials
~/.aws/config

\`\`\`

Step 2: Use Environment Variables

```
export AWS_ACCESS_KEY_ID="your-access-key"
export AWS_SECRET_ACCESS_KEY="your-secret-key"
export AWS_DEFAULT_REGION="ap-south-1"
```



```
ramsha@worker1:~/terraform_project$ aws configure
AWS Access Key ID [****************IEN2]:
AWS Secret Access Key [****************T6Gq]:
Default region name [ap-south-1]:
Default output format [None]:
```

Architecture Overview

The setup includes:

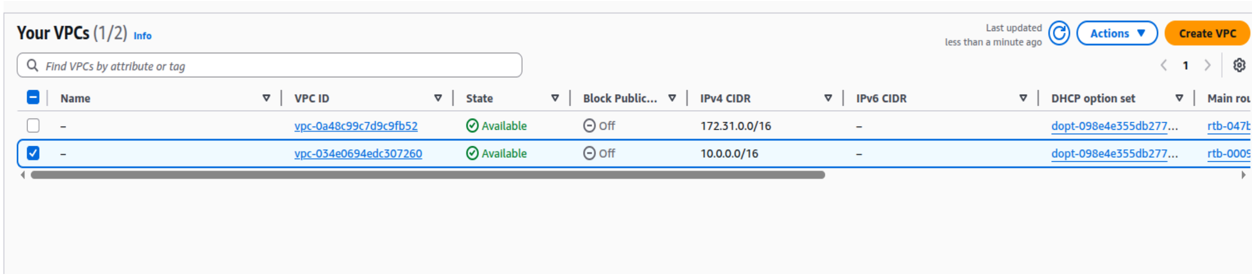
1. VPC with CIDR block
2. Subnets (for EC2)
3. Internet Gateway and Route Table
4. Security Groups (Allow SSH, HTTP, MySQL)
5. 2 EC2 Instances
6. S3 Bucket

Step-by-Step Terraform Resource Explanation

Create a VPC (main.tf)

hcl

```
resource "aws_vpc" "myvpc" {
  cidr_block = "10.0.0.0/16"
  enable_dns_hostnames = true
  enable_dns_support = true
}
```



Your VPCs (1/2) Info

	Name	VPC ID	State	Block Public...	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main rou
□	–	vpc-0a48c99c7d9c9fb52	✓ Available	⊖ Off	172.31.0.0/16	–	dopt-098e4e355db277...	rtb-047b
☑	–	vpc-034e0694edc307260	✓ Available	⊖ Off	10.0.0.0/16	–	dopt-098e4e355db277...	rtb-0009

Subnets

hcl

```
resource "aws_subnet" "subnet1" {
  vpc_id = aws_vpc.myvpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "ap-south-1a"
}

resource "aws_subnet" "subnet2" {
  vpc_id = aws_vpc.myvpc.id
  cidr_block = "10.0.2.0/24"
  availability_zone = "ap-south-1b"
}
```

Internet Gateway & Route Table

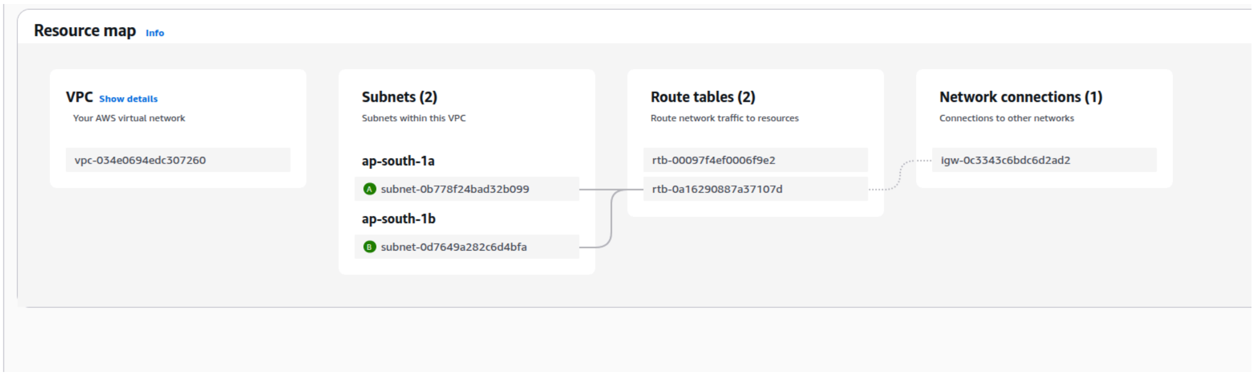
hcl

```
resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.myvpc.id
}

resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
}

resource "aws_route_table_association" "assoc1" {
  subnet_id = aws_subnet.subnet1.id
  route_table_id = aws_route_table.rt.id
}
```



Resource map Info

VPC Show details
Your AWS virtual network
vpc-034e0694edc307260

Subnets (2)
Subnets within this VPC
ap-south-1a
A subnet-0b778f24bad32b099
ap-south-1b
B subnet-0d7649a282c6d4bfa

Route tables (2)
Route network traffic to resources
rtb-00097f4ef0006f9e2
rtb-0a16290887a37107d

Network connections (1)
Connections to other networks
igw-0c3343c6bdc6d2ad2

Security Group:

hcl

```
resource "aws_security_group" "allow_tls" {
  name   = "websg"
  vpc_id = aws_vpc.vpc_proj.id

  tags = {
    Name = "web-sg"
  }
}

# Ingress Rule - Allow HTTP traffic from anywhere (IPv4)
resource "aws_vpc_security_group_ingress_rule" "allow_http_ipv4" {
  security_group_id = aws_security_group.allow_tls.id
  cidr_ipv4         = "0.0.0.0/0"         # fixed typo from 0.0.0/0 to correct block
  from_port         = 80
  to_port           = 80
  ip_protocol       = "tcp"
}

# Ingress Rule - Allow SSH traffic (port 22) from anywhere (IPv6)
resource "aws_vpc_security_group_ingress_rule" "allow_ssh_ipv6" {
  security_group_id = aws_security_group.allow_tls.id
  cidr_ipv6         = "::/0"
  from_port         = 22
  to_port           = 22
  ip_protocol       = "tcp"   # changed from "ssh" to correct value "tcp"
}
```

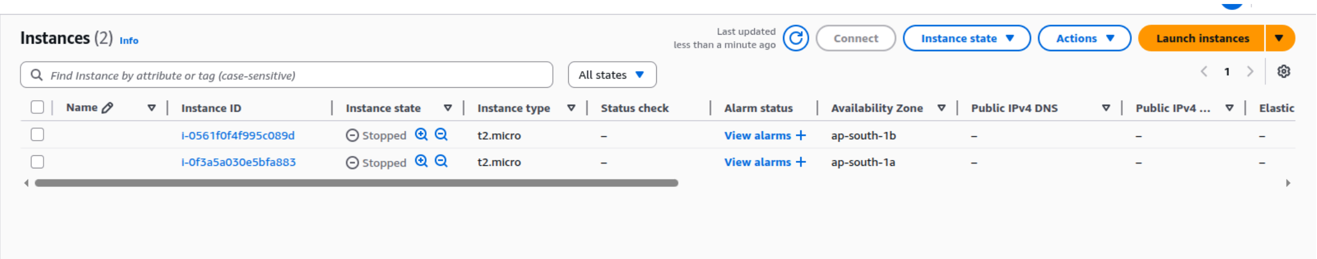
```
# Egress Rule - Allow all IPv4 traffic
resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" {
  security_group_id = aws_security_group.allow_tls.id
  cidr_ipv4         = "0.0.0.0/0"
  ip_protocol       = "-1"
}

# Egress Rule - Allow all IPv6 traffic
resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv6" {
  security_group_id = aws_security_group.allow_tls.id
  cidr_ipv6         = "::/0"
  ip_protocol       = "-1"
}
```

EC2 Instances

```
resource "aws_instance" "ec2_instance" {
  ami   = "ami-0f918f7e67a3323f0"
  instance_type  = "t2.micro"
  vpc_security_group_ids = [aws_security_group.allow_tls.id]  # Correct reference
  subnet_id    = aws_subnet.subnet1.id              # Correct subnet ID
  user_data     = file("userdata.sh")  # Make sure file exists
}

resource "aws_instance" "ec2_instance2" {
  ami     = "ami-0f918f7e67a3323f0"
  instance_type   = "t2.micro"
  vpc_security_group_ids = [aws_security_group.allow_tls.id]  # Correct reference
  subnet_id    = aws_subnet.subnet2.id
  user_data   = file("userdata1.sh") #  Make sure file exists
}
```



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic
☐		i-0561f0f4f995c089d	Stopped	t2.micro	–	View alarms +	ap-south-1b	–	–	–
☐		i-0f3a5a030e5bfa883	Stopped	t2.micro	–	View alarms +	ap-south-1a	–	–	–

1. Create the S3 Bucket

```
resource "aws_s3_bucket" "firstbucket" {
  bucket = "rimsha-terraform-bucket-20250718" #
}
```

> This resource provisions a new S3 bucket in your AWS account with a globally unique name.

2. Set Ownership Controls to allow ACL usage (required for public access via ACL)

```
resource "aws_s3_bucket_ownership_controls" "example" {
  bucket = aws_s3_bucket.firstbucket.id

  rule {
    object_ownership = "ObjectWriter"
  }
}
```

> This configures ownership rules to allow the use of ACLs. Required for setting public read access.

3. Allow Public Access by disabling public access blocks

```
resource "aws_s3_bucket_public_access_block" "example" {
  bucket = aws_s3_bucket.firstbucket.id

  block_public_acls       = false
  ignore_public_acls      = false
  block_public_policy     = false
  restrict_public_buckets = false
}
```

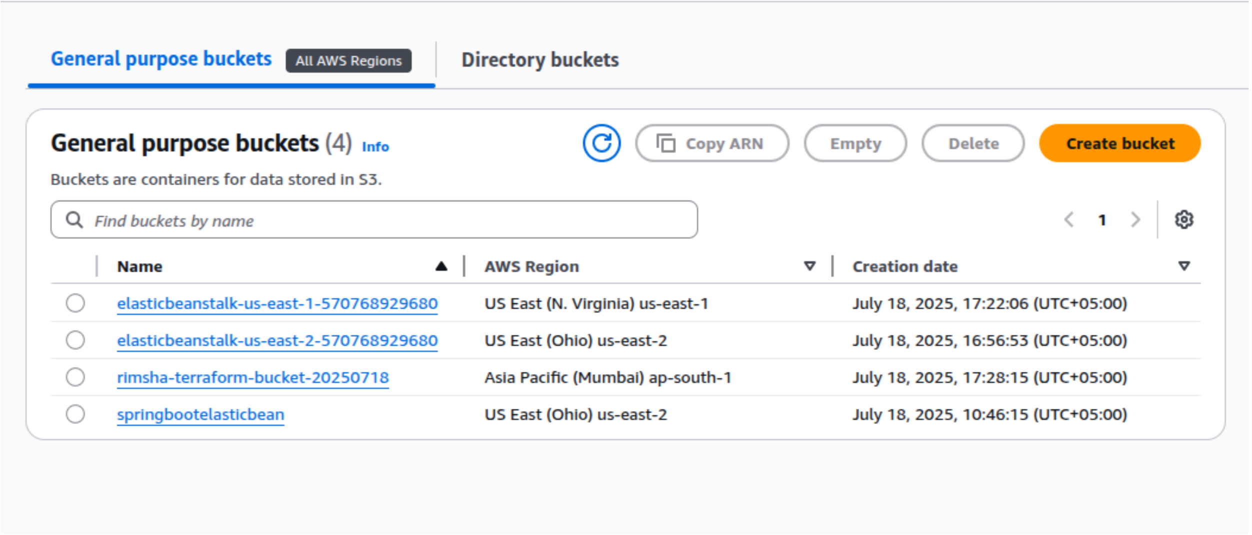
> Disables the blocking mechanisms that AWS enables by default to restrict public access.

4. Set ACL to make the bucket publicly readable

```
resource "aws_s3_bucket_acl" "example" {
  bucket = aws_s3_bucket.firstbucket.id
  acl    = "public-read"

  depends_on = [
    aws_s3_bucket_ownership_controls.example,
    aws_s3_bucket_public_access_block.example
  ]
}
```

> Applies an Access Control List (ACL) to allow read access to the public. Depends on public access settings.



General purpose buckets (4)

Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
elasticbeanstalk-us-east-1-570768929680	US East (N. Virginia) us-east-1	July 18, 2025, 17:22:06 (UTC+05:00)
elasticbeanstalk-us-east-2-570768929680	US East (Ohio) us-east-2	July 18, 2025, 16:56:53 (UTC+05:00)
rimsha-terraform-bucket-20250718	Asia Pacific (Mumbai) ap-south-1	July 18, 2025, 17:28:15 (UTC+05:00)
springbootelasticbean	US East (Ohio) us-east-2	July 18, 2025, 10:46:15 (UTC+05:00)

Optional: Upload a sample file (index.html) to test public access

```
resource "aws_s3_object" "index_file" {
  bucket = aws_s3_bucket.firstbucket.id
  key    = "index.html"
  source = "./index.html"
  content_type = "text/html"
  acl    = "public-read"
}
```

> Uploads a static HTML file to the bucket and makes it publicly accessible.

Terraform Commands to Run

Step 1: Initialize Terraform

```
terraform init
```

> Initializes the Terraform project and downloads provider plugins.

Step 2: Preview the Plan

```
terraform plan
```

> Previews what Terraform will do (create, destroy, update).

Step 3: Apply the Infrastructure

```
terraform apply
```

> Actually provisions the resources in your AWS account.

Cleaning Up Resources
To avoid unnecessary AWS charges:

```
terraform destroy
```

> Tears down all resources defined in your Terraform project.