**Create Terraform module structure**

**Step-1. VPC, ECR, and ECS with the required variables.tf,**

**outputs.tf,**

**main.tf files**

**for each module.**

**Step -2 .Paste code of respective module in their .tf file**

**Step -3 Run below command**

**terraform init**

**terraform plan**

**Directory Structure**

**terraform-project/**

**│── main.tf**

**│── variables.tf**

**│── outputs.tf**

**│── modules/**

**│ ├── vpc/**

**│ │ ├── main.tf**

**│ │ ├── variables.tf**

**│ │ ├── outputs.tf**

**│ ├── ecr/**

**│ │ ├── main.tf**

**│ │ ├── variables.tf**

**│ │ ├── outputs.tf**

**│ ├── ecs/**

**│ │ ├── main.tf**

**│ │ ├── variables.tf**

**│ │ ├── outputs.tf**

**VPC and Networking Documentation**

**1. VPC**

**Resource:** aws\_vpc

**Description:** Creates a custom VPC with DNS support and hostnames enabled.

**Code Explanation:**

resource "aws\_vpc" "Task" {

cidr\_block = var.cidr\_block

enable\_dns\_support = true

enable\_dns\_hostnames = true

tags = {

Name = var.vpc\_name

}

}

**Explanation:**

* **cidr\_block:** Defines the IP range for the VPC.
* **enable\_dns\_support:** Enables DNS resolution inside the VPC.
* **enable\_dns\_hostnames:** Allows instances to receive public DNS hostnames.
* **Tags:** Assigns a name to the VPC.

**2. Internet Gateway**

**Resource:** aws\_internet\_gateway

**Description:** Provides internet access for resources within the VPC.

**Code Explanation:**

resource "aws\_internet\_gateway" "Task" {

vpc\_id = aws\_vpc.Task.id

tags = {

Name = var.vpc\_name

}

}

**Explanation:**

* **vpc\_id:** Associates the internet gateway with the created VPC.
* **tags:** Assigns a name to the internet gateway.

**3. Public Subnets**

**Subnet 1**

**Resource:** aws\_subnet

**Description:** Creates a public subnet within the VPC.

**Code Explanation:**

resource "aws\_subnet" "sub-1" {

vpc\_id = aws\_vpc.Task.id

cidr\_block = var.subnet1\_cidr\_block

availability\_zone = "ap-south-1a"

map\_public\_ip\_on\_launch = true

tags = {

Name = "sub-1"

}

}

**Explanation:**

* **vpc\_id:** Associates the subnet with the VPC.
* **cidr\_block:** Defines the IP range for the subnet.
* **availability\_zone:** Specifies the availability zone.
* **map\_public\_ip\_on\_launch:** Enables automatic public IP assignment.
* **tags:** Assigns a name to the subnet.

**Subnet 2**

**Resource:** aws\_subnet

**Description:** Creates another public subnet for high availability.

**Code Explanation:**

resource "aws\_subnet" "sub-2" {

vpc\_id = aws\_vpc.Task.id

cidr\_block = var.subnet2\_cidr\_block

availability\_zone = "ap-south-1c"

map\_public\_ip\_on\_launch = true

tags = {

Name = "sub-2"

}

}

**Explanation:**

* **vpc\_id:** Associates the subnet with the VPC.
* **cidr\_block:** Defines the IP range for the subnet.
* **availability\_zone:** Specifies the second availability zone.
* **map\_public\_ip\_on\_launch:** Enables automatic public IP assignment.
* **tags:** Assigns a name to the subnet.

**4. Route Table**

**Resource:** aws\_route\_table

**Description:** Defines a route table to route traffic through the internet gateway.

**Code Explanation:**

resource "aws\_route\_table" "table-1" {

vpc\_id = aws\_vpc.Task.id

tags = {

Name = "table-1"

}

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.Task.id

}

}

**Explanation:**

* **vpc\_id:** Associates the route table with the VPC.
* **tags:** Assigns a name to the route table.
* **route:** Defines a default route (0.0.0.0/0) through the internet gateway.

**5. Route Table Associations**

**Resources:** aws\_route\_table\_association

**Description:** Associates public subnets with the route table to enable internet access.

**Code Explanation:**

resource "aws\_route\_table\_association" "Task-association-1" {

subnet\_id = aws\_subnet.sub-1.id

route\_table\_id

resource "aws\_route\_table\_association" "Task-association-1" {

subnet\_id = aws\_subnet.sub-1.id

route\_table\_id = aws\_route\_table.table-1.id

}

resource "aws\_route\_table\_association" "Task-association-2" {

subnet\_id = aws\_subnet.sub-2.id

route\_table\_id = aws\_route\_table.table-1.id

}

**subnet\_id:** Specifies the subnet to associate.

**route\_table\_id**: Associates the subnet with the route table.

***6.Variables in variables.tf***

Add unique name in all variables :

variable "cidr\_block" {}

variable "subnet1\_cidr\_block" {}

variable "subnet2\_cidr\_block" {}

variable "availability\_zones" { type = list(string)}

variable "vpc\_name" {}

**Elastic Container Registry (ECR) Documentation**

**7. Elastic Container Registry (ECR)**

* **Resource: aws\_ecr\_repository**
* **Description: Creates a private ECR repository to store container images.**
* **Code Explanation:**

resource "aws\_ecr\_repository" "ecr\_repository" {

name = var.repository\_name

tags = {

name = var.repository\_name

}

}

* + **name**: Defines the name of the repository.
  + **tags**: Adds metadata to the repository.

**8. Pushing Docker Images to ECR**

To authenticate and push an image to your repository, follow these steps in terminal where aws cli is configured

To Check aws cli configured

aws configure

Add AWS Access Key ID and AWS Secret Access Key if not present

1. **Authenticate Docker with AWS ECR:**

aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin <account-id>.dkr.ecr.ap-south-1.amazonaws.com

This command retrieves an authentication token and logs you into your ECR repository.

1. **pull the Docker image:**

docker pull nginx

This command builds a Docker image from your local directory.

1. **Tag the image:**

docker tag nginx:latest <account-id>.dkr.ecr.ap-south-1.amazonaws.com/task-repo-project:latest

This step assigns a unique tag to your image so it can be recognized in ECR.

1. **Push the image to the repository:**

docker push <account-id>.dkr.ecr.ap-south-1.amazonaws.com/task-repo-project:latest

This command uploads your image to the ECR repository, making it available for deployment.

