Step 1: Set Up a VPC Using Terraform

Prerequisites:

- Terraform installed on your machine
- AWS CLI configured with appropriate permissions
- Basic knowledge of Terraform and AWS networking concepts

Steps:

1. **Create a Project Directory:**

   ```
   sh
   Copy code
   mkdir terraform-vpc
   cd terraform-vpc
   ```

2. **Write the Terraform Configuration:** Create a file named `main.tf` and add the following code to define the VPC, subnets, and necessary resources.

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

   resource "aws_vpc" "main" {
     cidr_block = "10.0.0.0/16"
   }

   resource "aws_subnet" "subnet1" {
     vpc_id            = aws_vpc.main.id
     cidr_block        = "10.0.1.0/24"
     availability_zone = "us-east-1a"
   }

   resource "aws_subnet" "subnet2" {
     vpc_id            = aws_vpc.main.id
     cidr_block        = "10.0.2.0/24"
     availability_zone = "us-east-1b"
   }

   resource "aws_internet_gateway" "igw" {
     vpc_id = aws_vpc.main.id
   }

   resource "aws_route_table" "route_table" {
     vpc_id = aws_vpc.main.id

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

   resource "aws_route_table_association" "a" {
     subnet_id      = aws_subnet.subnet1.id
   ```

```
  route_table_id = aws_route_table.route_table.id
}

resource "aws_route_table_association" "b" {
  subnet_id      = aws_subnet.subnet2.id
  route_table_id = aws_route_table.route_table.id
}

output "vpc_id" {
  value = aws_vpc.main.id
}
```

3. **Initialize and Apply Terraform Configuration:**

   ```
   sh
   Copy code
   terraform init
   terraform apply
   ```

   Confirm the apply with `yes`.

Step 2: Create a Jenkins Pipeline

Prerequisites:

- Jenkins installed and running
- Jenkins user with necessary permissions
- Jenkins plugins: Pipeline, Git, AWS, Docker

Steps:

1. **Set Up Jenkins Credentials:**
   - Go to Jenkins Dashboard -> Manage Jenkins -> Manage Credentials -> Add Credentials.
   - Add AWS credentials (Access Key ID and Secret Access Key).
   - Add any other necessary credentials (e.g., GitHub token, Docker Hub credentials).
2. **Create a New Pipeline Job:**
   - Go to Jenkins Dashboard -> New Item -> Pipeline -> OK.
   - Configure the pipeline:
     - In the pipeline script, use the following example to integrate with AWS and Docker:

       ```
       groovy
       Copy code
       pipeline {
         agent any

         environment {
           AWS_ACCESS_KEY_ID = credentials('aws-access-key-id')
           AWS_SECRET_ACCESS_KEY = credentials('aws-secret-access-
       key')
         }

         stages {
       ```

```
    stage('Checkout') {
      steps {
        git 'https://github.com/your-repo/hello-world-api.git'
      }
    }
    stage('Build Docker Image') {
      steps {
        script {
          dockerImage = docker.build("your-dockerhub-
username/hello-world-api")
        }
      }
    }
    stage('Push Docker Image') {
      steps {
        script {

docker.withRegistry('https://registry.hub.docker.com',
'dockerhub-credentials') {
            dockerImage.push("latest")
          }
        }
      }
    }
    stage('Deploy to AWS Fargate') {
      steps {
        script {
          sh 'aws ecs update-service --cluster your-cluster --
service your-service --force-new-deployment'
        }
      }
    }
  }
}
```

3. **Run the Pipeline:**
   - Click on `Build Now` to run the pipeline.

Step 3: Deploy a Hello World API on AWS Fargate Using Jenkins

Prerequisites:

- AWS ECS cluster and service set up
- Docker installed and configured
- Jenkins pipeline from the previous step

Steps:

1. **Create a Dockerfile for the Hello World API:**
   - Ensure your `Dockerfile` is in the root of your GitHub repository.

```
dockerfile
Copy code
FROM node:14

WORKDIR /app
```

```
COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8080

CMD ["node", "index.js"]
```

2. **Configure ECS Service for Fargate:**
   - Ensure your ECS service is set up to use Fargate and can pull images from Docker Hub.
3. **Update Jenkins Pipeline:**
   - Ensure the Jenkins pipeline has the necessary steps to build, push, and deploy the Docker image to ECS Fargate (as shown in Step 2).
4. **Run the Pipeline:**
   - Ensure your Jenkins pipeline runs successfully, building the Docker image, pushing it to Docker Hub, and updating the ECS service.

Recap:

- **Set Up a VPC Using Terraform:** You created a VPC, subnets, an internet gateway, and route tables using Terraform.
- **Create a Jenkins Pipeline:** You configured a Jenkins pipeline to checkout code, build a Docker image, push it to Docker Hub, and deploy to AWS ECS.
- **Deploy a Hello World API on AWS Fargate Using Jenkins:** You ensured your Hello World API is containerized with Docker, pushed to Docker Hub, and deployed to ECS Fargate via the Jenkins pipeline.