The following steps are followed to design the required pipeline

1.Creating the Node.js App:

mkdir hello-world-nodejs

cd hello-world-nodejs

npm init -y

npm install express

2. Create app.js file:

3. Creating a docker file

```
FROM node:14

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

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

4.Creating a terraform file

```
provider "aws" {
  region = "us-east-1"  # Specify your desired region
}

resource "aws_ecs_cluster" "hello_world_cluster" {
  name = "hello-world-cluster"
}

resource "aws_ecs_task_definition" "hello_world_task" {
  family                = "hello-world-task"
  network_mode          = "bridge"  # Using bridge network mode for EC2
  requires_compatibilities = ["EC2"]
  cpu                   = "256"
  memory                = "512"

  container_definitions = jsonencode([
   {
     name      = "hello-world-app"
     image     = "50a514f72022"
     essential = true
     portMappings = [
      {
        containerPort = 3000
        hostPort      = 3000
      }
     ]
```

```
  }
 ])
}

resource "aws_launch_configuration" "hello_world_lc" {
 name            = "hello-world-lc"
 image_id        = "ami-04b70fa74e45c3917"  # Specify the AMI ID for your EC2
 instance_type   = "t3.micro"
 iam_instance_profile = "ecsInstanceProfile"  # Correct IAM instance profile

 lifecycle {
  create_before_destroy = true
 }

 user_data = <<-EOF
        #!/bin/bash
        echo ECS_CLUSTER=${aws_ecs_cluster.hello_world_cluster.name} >>
/etc/ecs/ecs.config
        EOF
}

resource "aws_autoscaling_group" "hello_world_asg" {
 desired_capacity    = 1
 max_size        = 1
 min_size        = 1
 launch_configuration = aws_launch_configuration.hello_world_lc.name
 vpc_zone_identifier  = [""]
```

```
  tag {
    key                 = "Name"
    value               = "hello-world-ecs-instance"
    propagate_at_launch = true
  }
}

resource "aws_ecs_service" "hello_world_service" {
  name            = "hello-world-service"
  cluster         = aws_ecs_cluster.hello_world_cluster.id
  task_definition = aws_ecs_task_definition.hello_world_task.arn
  desired_count   = 1
  launch_type     = "EC2"

  network_configuration {
    subnets         = [""]
    security_groups = [""]
  }

  depends_on = [aws_autoscaling_group.hello_world_asg]
}
```

6. Creating a role and setting instance profile

7.creating a JSON file named ‘trust-policy.json’

8.Creating a GitHub actions workflow

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'

    - name: Install dependencies
      run: npm install
```

```
    - name: Run tests
      run: npm test # Ensure you have tests defined

    - name: Build Docker image
      run: docker build -t YOUR_DOCKER_IMAGE .

    - name: Log in to Amazon ECR
      id: login-ecr
      uses: aws-actions/amazon-ecr-login@v1

    - name: Push Docker image to ECR
      run: |
        docker tag YOUR_DOCKER_IMAGE:latest AWS_ACCOUNT_ID.dkr.ecr.REGION.amazonaws.com/YOUR_ECR_REPO:latest
        docker push AWS_ACCOUNT_ID.dkr.ecr.REGION.amazonaws.com/YOUR_ECR_REPO:latest

  deploy:
    needs: build
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Configure AWS credentials
```

```
    uses: aws-actions/configure-aws-credentials@v1
    with:
      aws-access-key-id: ${{ secrets.AWS_ACCESS_KEY_ID }}
      aws-secret-access-key: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
      aws-region: us-east-1

  - name: Deploy to ECS
    run: |
      echo "Deploying to ECS"
      aws ecs update-service --cluster hello-world-cluster --service hello-world-service --force-new-deployment
```