# Automated Continuous Deployment Pipeline for Django DRF on AWS

This document outlines the requirements and step-by-step directions for deploying a Dockerized Django Rest Framework application using an automated Continuous Deployment (CD) pipeline. The deployment will be in the AWS Tokyo region (`ap-northeast-1`) with availability zones `a` and `c`, leveraging AWS services such as ECS, ECR, and ELB.

## 1. AWS Resources

The following AWS resources will be required for this project:

1. 1. Elastic Container Registry (ECR): For storing your Docker images.
2. 2. Elastic Load Balancer (ELB): To distribute traffic across instances/services in different availability zones.
3. 3. Elastic Container Service (ECS) with Fargate: For serverless container deployment.
4. 4. Target Group: To route traffic from the load balancer to ECS tasks.
5. 5. Virtual Private Cloud (VPC): A secure networking layer with public and private subnets in availability zones `ap-northeast-1a` and `ap-northeast-1c`.
6. 6. Amazon Relational Database Service (RDS) (Optional): For a managed PostgreSQL database.
7. 7. Amazon S3: For serving static and media files.

## 2. Deployment Pipeline

1. 1. Code Repository: Use GitHub to store and manage your code.
2. 2. GitHub Actions: Automate the CD pipeline to build, test, and push images to ECR, and trigger ECS service updates.

## 3. Security and Permissions

1. 1. IAM Roles: Grant ECS tasks, GitHub Actions, and ECR appropriate permissions.
2. 2. Secrets Management: Use AWS Secrets Manager or GitHub Secrets for storing sensitive credentials.

## 4. Networking

1. 1. VPC with Subnets: Create a VPC with public and private subnets in availability zones `ap-northeast-1a` and `ap-northeast-1c`.

* - Assign appropriate routing tables and internet gateway for public access.

1. 2. Security Groups: Configure security groups for ECS tasks and RDS instances to allow only necessary traffic.

## 5. Logging and Monitoring

1. 1. Amazon CloudWatch: Monitor ECS logs, application metrics, and errors.
2. 2. ALB Access Logs: Enable logging for the Application Load Balancer.

## 6. Deployment and Testing Tools

1. 1. Docker: For building and testing images locally.
2. 2. GitHub Actions Workflow: For deploying changes automatically upon a code push.

## 7. Configuration Files

1. 1. GitHub Actions Workflow (`ci.yml`): Automate build, test, and deployment tasks.
2. 2. Dockerfile and docker-compose.yml: Already provided in your project.
3. 3. ECS Task Definition: Define how containers are deployed (CPU, memory, networking, etc.).

## 8. Miscellaneous

1. 1. Domain Name (Optional): Use AWS Route 53 to manage a custom domain for your API.
2. 2. SSL Certificate (Optional but Recommended): Use AWS Certificate Manager for HTTPS.