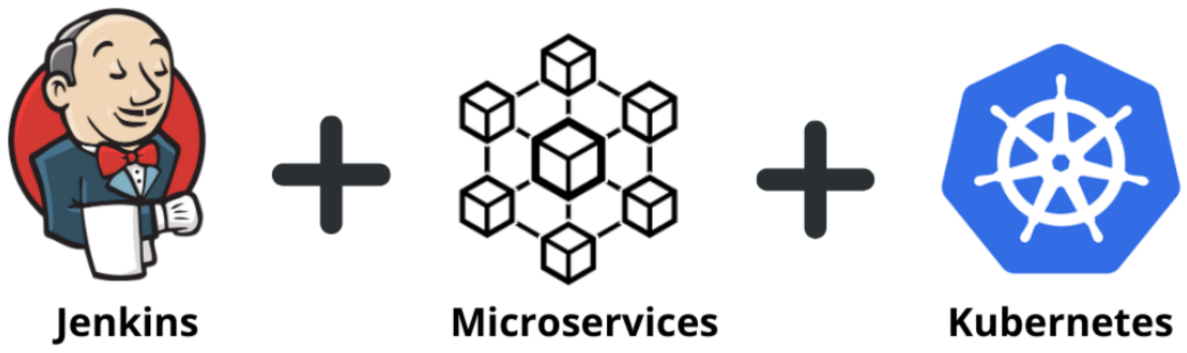
## My Approach

CI/CD Pipeline for Event Driven Microservices via ECS on AWS

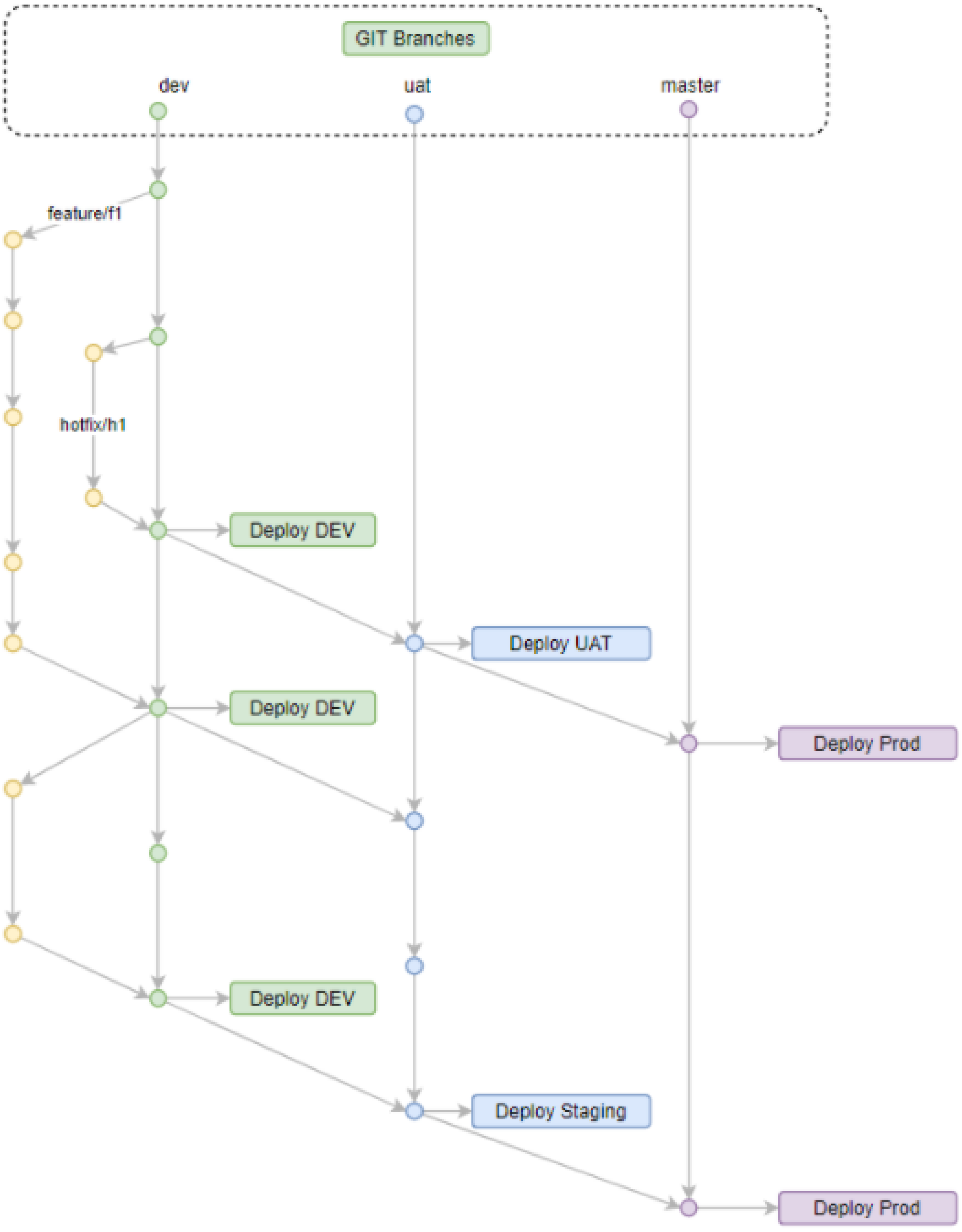


## Pre Requisites

- Jenkins
- AWS Account
- AWS CLI
- Helm
- Docker
- Jenkins Plugins
- A few AMI Roles

## Branching Strategy

3 mainline branches mapped to 3 environments running in the Kubernetes cluster. Each environment is running in it's Kubernetes namespace



## BUILD

In a typical microservices-based architecture, each microservice can be developed in a different programming language or framework. This puts Docker to it’s best use case. All the build instructions for a particular microservice stay in Dockerfile placed at the root of each repositor

## Deployment

The deployments are packaged as helm charts that define all aspects of a microservice like — container images, placement constraints, networking, storage, configurations, etc.

These helm charts are stored in a separate git repository, which includes per service helm chart as well as other resources like common configmaps and secrets, ingress and different routing configuration, etc. These are fetched on Jenkins server as a dedicated Jenkins job with webhook configured to fetch updates automatically.

Overall Architecture

## Pipeline Steps

- Source
- Build and Test
- Artifact
- Deploy
- Cleanup ( Crucial part of the pipeline )
- Notify Jenkinsfile can be found here

## Jenkinsfile and its Reusability

Jenkinsfile can be found here – https://github.com/hani0523/PSI_CI-CD/blob/main/Jenkinsfile

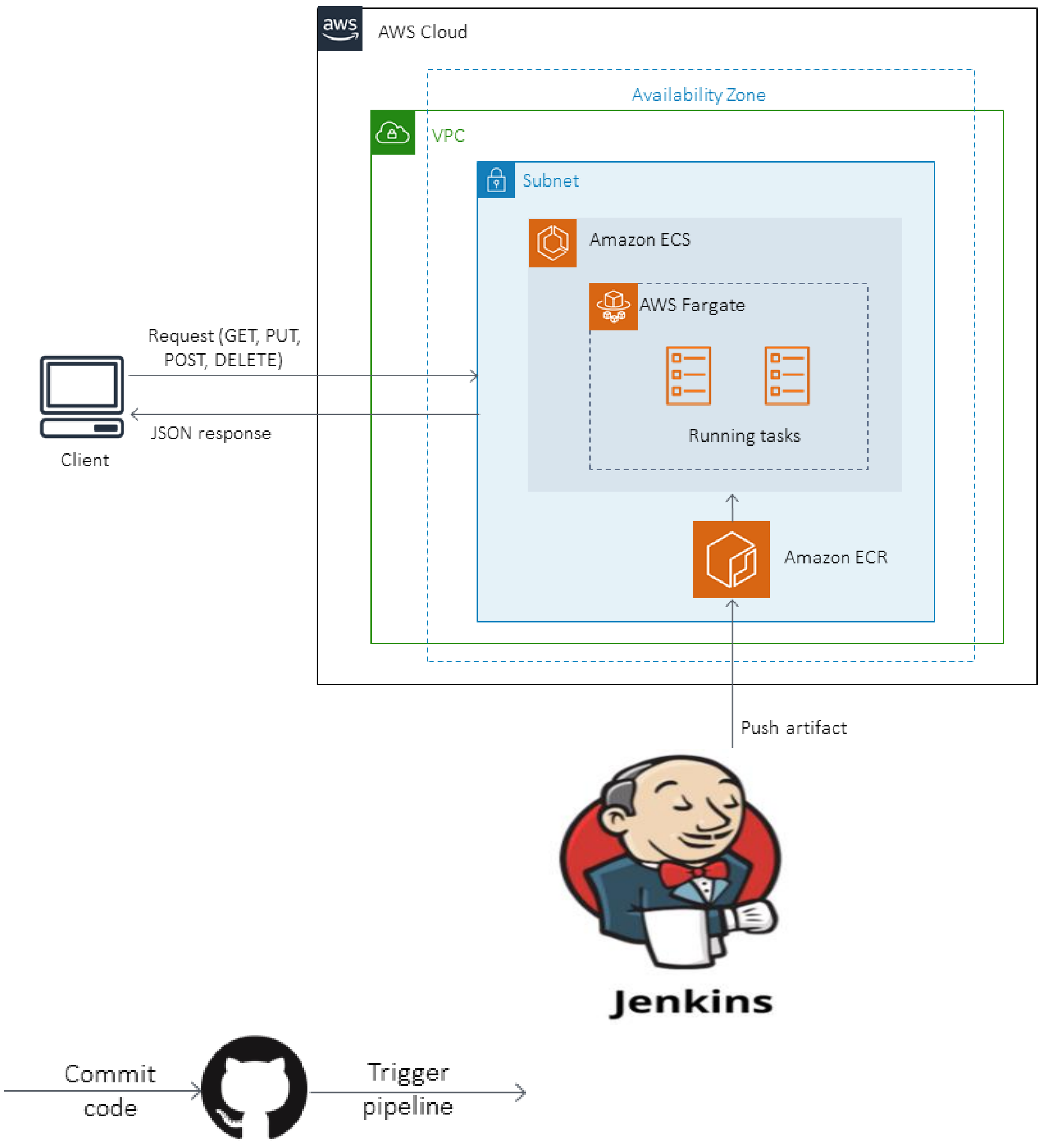
This Jenkinsfile is independent of microservice or environment. You can have a single Jenkins file for all microservices or can use separate Jenkins files stored in git for every microservice.

I will configure a Multibranch Pipeline Jenkins job to use the above-mentioned pipeline file

## Overall Architecture After Deployment

### Source technology stack

- Microservices running on Amazon EKS/ECS
- Code repository in Amazon ECR
- AWS Fargate: A compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters



## AWS Services

### Tools

- **Amazon ECR** - Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with Amazon ECS to simplify your development-to-production workflow. Amazon ECR hosts your images in a highly available and scalable architecture so you can reliably deploy containers for your applications. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.

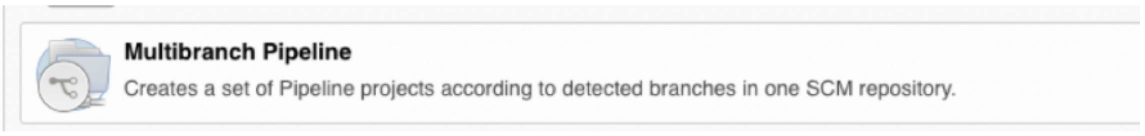
- **Amazon ECS** - Amazon Elastic Container Service (Amazon ECS) is a highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need for you to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines
- **AWS Fargate** - AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.
- **Docker** - Docker is a platform that lets you build, test, and deliver applications in packages called containers.

## Build Steps

Build steps can be found here - https://github.com/hani0523/PSI_CI-CD/blob/main/buildspec.yml

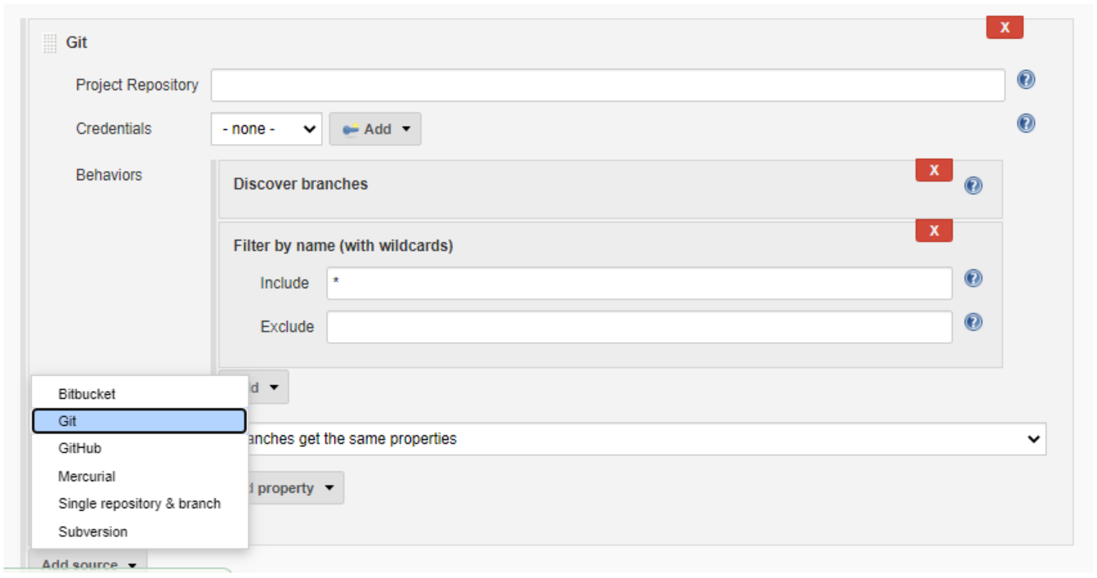
## Detailed Illustration of Jenkins Procedure

Creating Multibranch Pipeline


**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

In Branch Source, update your Git repository and credential, then you can keep the default behaviour for auto discover branches, or you can filter branches by name.

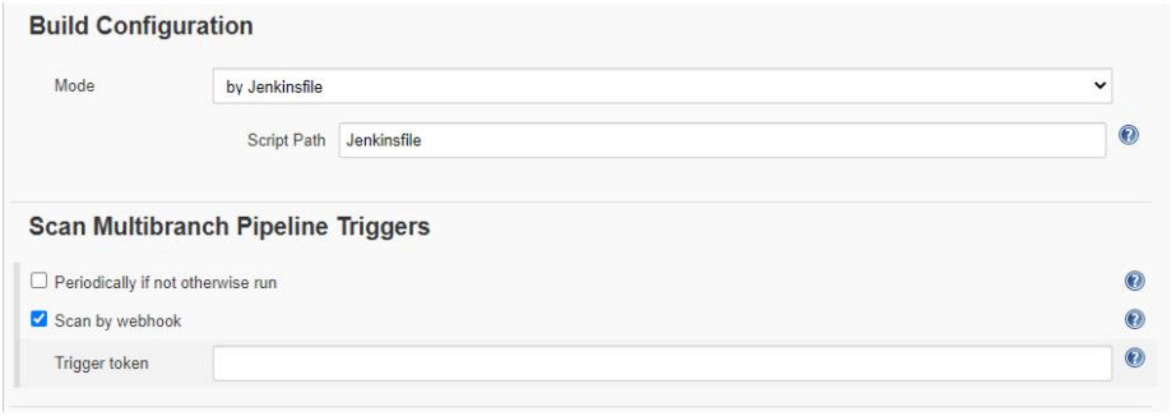

Git

Project Repository:

Credentials: - none -

Behaviors: Discover branches; Filter by name (with wildcards) — Include: *; Exclude:

In Build Configuration, use Jenkinsfile and update your file path.


**Build Configuration**

Mode: by Jenkinsfile

Script Path: Jenkinsfile

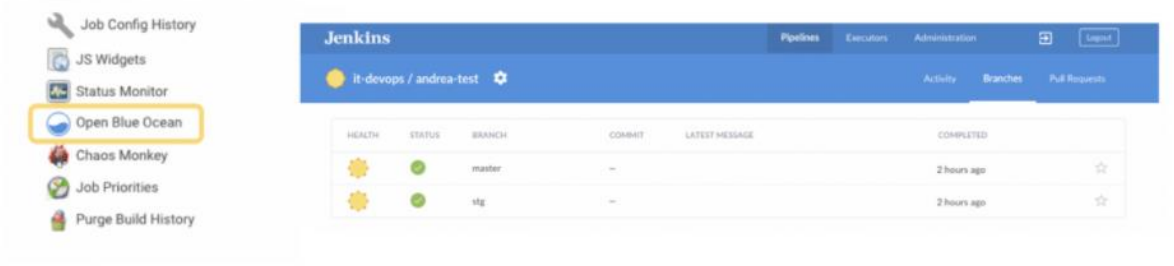
**Scan Multibranch Pipeline Triggers**

☐ Periodically if not otherwise run

☑ Scan by webhook

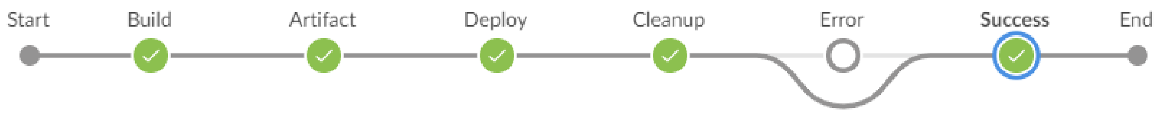
Trigger token:

Save your configuration and then Click Open Blue Ocean, it would auto-discover your Github branch and the pipeline you defined in Jenkinsfile



| HEALTH | STATUS | BRANCH | COMMIT | LATEST MESSAGE | COMPLETED |
|---|---|---|---|---|---|
| ☀ | ✓ | master | – | | 2 hours ago |
| ☀ | ✓ | stg | – | | 2 hours ago |

After making a commit to a specific branch, you should see the pipeline initiation of that branch, followed by the final deployment.


Start
Build
Artifact
Deploy
Cleanup
Error
Success
End