### **Deployment Documentation for Java Application with Helm Chart and GitHub Actions**

### **Overview**

This document outlines the architecture, workflow, and processes involved in deploying a Java-based application using a robust CI/CD pipeline. The application is containerized and deployed to Kubernetes (EKS) using Helm charts, with automation managed through GitHub Actions.

### **Application Components**

1. **Java Application:**
   * Built using Maven with a pom.xml configuration.
   * Packaged as a Docker image for deployment.
2. **Dockerfile:**
   * Defines the containerization process for the application.
   * Ensures the application can run consistently in various environments.
3. **Helm Chart:**
   * Used for deploying the application onto Kubernetes.
   * Contains templates and configuration files to manage Kubernetes resources such as Deployments, Services, and ConfigMaps.
4. **Deployment Script (deploy.sh):**
   * Automates the Helm chart rebuilding and deployment process.

### **CI/CD Pipeline**

The deployment process is automated using GitHub Actions, which provides a seamless mechanism to build, test, and deploy the application. Below is an outline of the workflow:

1. **Build :**
   * The Java application is built using Maven.
   * Unit tests and integration tests (if applicable) are run to validate the application.
2. **Containerization:**
   * The Dockerfile is used to create a Docker image of the application.
   * The image is tagged with a unique identifier, such as the Git commit hash or build number.
3. **Push to ECR:**
   * The built Docker image is pushed to an Amazon Elastic Container Registry (ECR) repository for secure storage.
4. **Deploy to EKS:**
   * The pipeline connects to the Amazon Elastic Kubernetes Service (EKS) cluster.
   * The Helm chart is deployed or updated, applying the new image and configurations to the Kubernetes environment.