Bing

DevOps Approach for OMS Implementation Program

Introduction

The Order Management System (OMS) is a critical component of any e-commerce business, handling everything from order entry and processing to inventory management and shipping. Implementing an OMS requires careful planning and coordination across multiple teams. This document outlines our approach to applying DevOps methodologies to the OMS implementation program, ensuring a smooth and efficient deployment process.

Objective

The primary objective of this document is to provide a comprehensive guide on the application of DevOps methodologies for the implementation of the OMS program. The aim is to ensure that DevOps tasks are coordinated effectively with the overarching program plan. This involves understanding the requirements pertaining to the creation of the environment, pipeline, and repository. By aligning these elements with the overall program plan, we aim to ensure a smooth and efficient implementation of the OMS program using DevOps practices. This approach not only improves the speed and quality of software development but also fosters better collaboration between the development and operations teams.

Key Components

Environment Creation

Setting up the necessary infrastructure is a critical aspect of the OMS implementation. We'll be using **AWS Cloud** and will make use of two AWS regions, primary is **us-east-1**; and secondary **us-west-2** which is the disaster recovery region. We'll make use of two Availability Zones (AZs) in these regions for deploying the resources. We'll be having 2 private subnets and 2 public subnets.

Infrastructure as code (IaC) tools such as **alo-terraform** will be employed to define and provision servers, databases, and other required resources. This ensures consistency and reproducibility across different environments. By using IaC, we can automate the process of setting up the infrastructure, reducing the risk of human error and increasing efficiency.

Pipeline Creation

A robust CI/CD pipeline is essential for automating the software development lifecycle. We'll use **GitHub Actions** and create **5 CI/CD pipelines** for the deployment of these services to AWS ECS cluster with Fargate in an automated fashion.

We'll be building the Docker image within the CI/CD pipeline, once built we use **Trivy image scan** for vulnerability check. Trivy is a popular open-source security scanner that is reliable, fast, and easy to use. It is used to find vulnerabilities & IaC misconfigurations, SBOM discovery, Cloud scanning, Kubernetes security risks, and more.

We'll be using **JaCoCo** for code coverage in the CI/CD pipeline. JaCoCo, which stands for Java Code Coverage, is an open-source code coverage tool for the Java programming language. It is used to measure and report how much of your Java code is executed during the execution of your tests or your business code in environments.

Automated testing suites, including unit tests and integration tests, will be integrated into the pipeline to ensure code quality and reliability. By automating these processes, we can ensure that any changes to the codebase are immediately tested and validated, reducing the risk of introducing bugs into the production environment.

Repository Creation

Version control is fundamental to collaboration and code management. We'll be using **Git**, hosted on platforms like GitHub or GitLab, will be used to create a centralized repository for OMS code. We'll be using Git 3 branching strategy with **develop**, **main**, **release**.

Feature branching strategies will be implemented to facilitate collaboration, and pull requests will serve as a mechanism for code review and merging. By using a version control system, we can track changes to the codebase over time, making it easier to identify and fix bugs. It also allows for better collaboration among team members, as they can work on different features simultaneously without interfering with each other's work.

Collaboration with Development Team

The DevOps team will closely collaborate with the development team throughout the program. DevOps Engineers will provide the required setup for proper code deployment in a secure environment.

We need to integrate **NewRelic** for each Java microservice, APM monitoring + logs in the New Relic are needed. As a part of the CI/CD pipeline, NewRelic key is injected in a secure manner using secrets and sed command.

Continuous monitoring using tools like Prometheus or AWS CloudWatch will be implemented to detect and address any issues promptly. Regular feedback loops and retrospective meetings will foster continuous improvement in both development and operations processes.

We'll make use of **Makefile** for the deployment, separate GitHub Actions workflow will be created for dev and prod.

Conclusion

By aligning these elements with the overall program plan, we aim to ensure a smooth and efficient implementation of the OMS program using DevOps practices. This approach not only improves the speed and quality of software development but also fosters better collaboration between the development and operations teams. Through continuous integration, continuous delivery, and infrastructure as code, we can ensure that our OMS implementation is robust, scalable, and efficient. With this DevOps approach, we are confident that we can successfully implement the OMS program and deliver a high-quality product to our customers.