Trabalho Prático 2 - Computação em Nuvem

Aluno: Jean George Alves Evangelista

Matrícula: 2024661178

Introdução

Este documento apresenta um relatório do Trabalho Prático 2 da disciplina de Computação em Nuvem, cursada em 2024/02.Neste projeto, foi explorado o uso do Kubernetes e do ArgoCD para gerenciar o ciclo de vida de uma aplicação baseada em microserviços.

O objetivo principal foi implementar um sistema de recomendação de playlists utilizando um modelo gerado por algoritmos de mineração de dados, integrado a serviços REST e implementado com práticas modernas de CI/CD.

O projeto utiliza um pipeline completo que inclui geração do modelo, atualização automática com novas versões do dataset e detecção de alterações no modelo no back-end. Este relatório apresenta os casos de teste realizados, os resultados observados e as estratégias adotadas para garantir a continuidade do serviço.

Desenvolvimento

O desenvolvimento ocorreu seguindo a especificação à risca. Um script em Python, que utiliza a biblioteca FPGrowth, foi desenvolvido para gerar recomendações de músicas. Também foi desenvolvida uma aplicação em Flask, contendo um endpoint para calcular e retornar as recomendações. Ainda na aplicação Flask, uma página web foi desenvolvida utilizando HTML, CSS e JavaScript, para acessar a API criada.

Após o desenvolvimento, foi criada uma aplicação no ArgoCD:

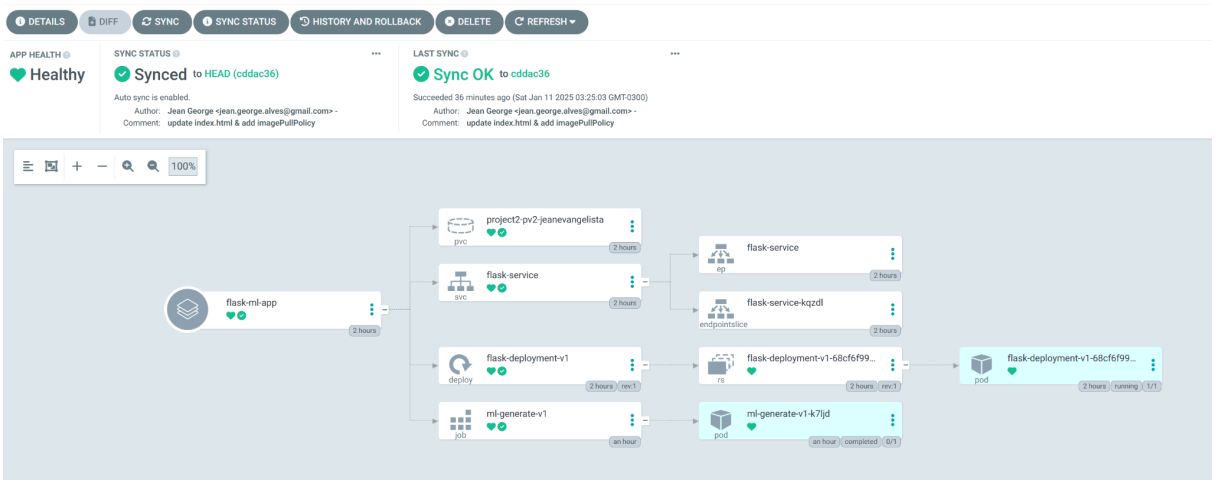

Figura 1 - ArgoCD

Testes

Foram realizados diversos testes para validar a funcionalidade do pipeline implementado com Kubernetes e ArgoCD.

Testes manuais

Os primeiros e mais intuitivos testes foram feitos manualmente, para verificar que tudo estava funcionando. Nesses testes foi feito encaminhamento de portas para acessar o serviço do Flask rodando na máquina virtual. Abaixo está o resultado da página web que consome a API de recomendações:

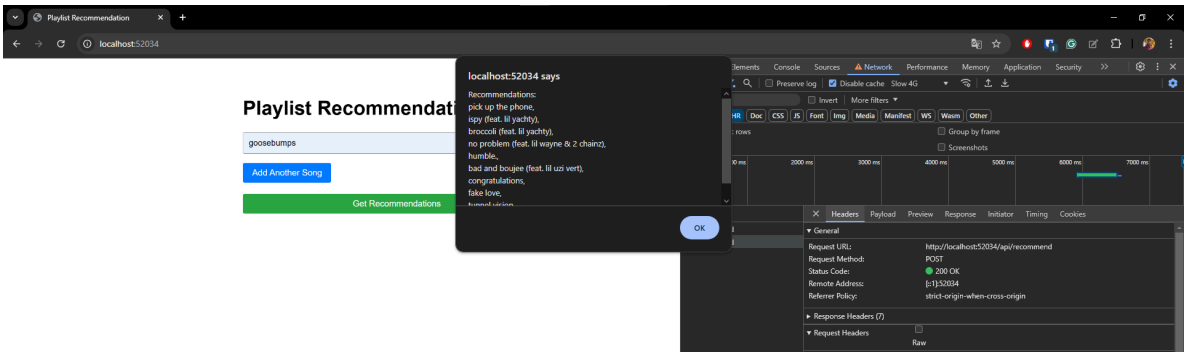

Figura 2 - Página web para testar a API e o serviço de recomendação

Atualização do Código do Modelo

Foi testada a alteração do código Python responsável pela geração do modelo (generate_rules.py). Após o push do código atualizado para o repositório Git monitorado pelo ArgoCD, foi verificado que o pipeline reconstruiu a imagem Docker do serviço de ML e reimplantou o job no Kubernetes. O modelo foi regenerado e salvo no volume persistente, garantindo que as atualizações fossem refletidas.

Alteração no Número de Réplicas

Ao alterar o número de réplicas, o ArgoCD aplicou as mudanças conforme esperado, sem interromper o serviço.

Atualização do Dataset

Ao substituir o dataset, um novo job de ML foi executado, regenerando o modelo e atualizando automaticamente as regras usadas no serviço de recomendação.

Resultados

Tempo de Implantação

- Atualização de Código: Aproximadamente 1 a 2 minutos para reconstruir a imagem e reimplantar o job.
- Alteração de Réplicas: Modificações aplicadas em menos de 1 minuto sem downtime.
- Atualização de Dataset: O modelo foi regenerado em cerca de 5 a 8 minutos.

Serviço Offline

Em nenhum momento o serviço ficou offline, após implantado com sucesso.

Obtendo Novos Datasets

Os novos datasets são armazenados no volume persistente compartilhado entre os containers de ML e front-end. A lógica do job de ML garante que o modelo seja gerado utilizando os arquivos mais recentes.

Conclusão

A integração de Kubernetes e ArgoCD provou ser eficaz para implementar práticas de CI/CD em um sistema de recomendação baseado em microserviços.

Os testes realizados demonstraram a robustez do pipeline para lidar com atualizações de código, mudanças no número de réplicas e novos datasets.

A aplicação manteve alta disponibilidade durante a maior parte dos testes, evidenciando o sucesso da abordagem.

Referências

Persistent Volumes. Kubernetes Documentation. Disponível em: https://kubernetes.io/docs/concepts/storage/persistent-volumes/. Acesso em: 11 jan. 2025.

ArgoCD. ArgoCD Documentation. Disponível em: https://www.thoughtco.com/what-is-the-interquartile-range-rule-3126244. Acesso em: 11 jan. 2025