Vinícius Braga Freire - 2020054889

TP2 - Discussão

1. Introdução

O código do projeto foi hosteado no repositório https://github.com/vinciusb/TP2-CN-DEPLOY, enquanto o dado foi hosteado no repositório https://github.com/vinciusb/TP2-CN-DATA.

No primeiro temos:

- **`.github/workflows`:**
  - **`tag-on-commit.yml`:** Este arquivo define um job do github actions (CI) que cria uma nova tag no projeto toda vez que um novo commit é recebido. Esta tag tem o formato "**`{MAJOR}.{MINOR}.{PATCH}`**". Essa tag é importante, pois é através da mais recente das tags que o gerador de modelo obtém a versão do modelo a ser gerado.
- **`model_generator`:** Todos arquivos relacionados ao container gerador de modelo. Aqui estão o dockerfile, arquivo python e entrypoint do container.
- **`recomendation_engine`:** Todos arquivos relacionados ao container do motor de recomendação. Aqui estão o dockerfile, arquivo python e entrypoint do container.
- **`recomendation-deployment.yaml`:** Arquivo contendo o deployment, service e o persistent volume claim do kubernetes.

No segundo temos alguns arquivos que podem ser possivelmente os dados para a geração do modelo. O que liga um repositório ao outro é uma variável de ambiente **`REPO_URL`** no container **`model-generator`** no arquivo **`recomendation-deployment.yaml`**. Esta variável recebe uma string com a URL dos dados nos quais o modelo deve ser construído.

2. Testes:

Para que um novo deployment seja executado, basta que um commit novo seja feito no primeiro repositório. Embora isto, realisticamente falando, haverão três maneiras de forçar um novo deployment no código:

1. Alterando o código do gerador de modelo ou do motor de recomendação. Assim, basta alterar, no arquivo de deployment, a tag da imagem do container (não possuímos jobs de CI que sobem uma nova imagem para o docker hub ao mudar o

código, logo novas versões de código e novas imagens devem ser geradas localmente e, posteriormente, alterada a tag do container no arquivo de deployment).

2. Alterando o dataset utilizado para criar o modelo. Este pode ser alterado, também, através do arquivo de deployment. Na definição do container **`model_generator`** há a variável **`REPO_URL`** com a URL para se baixar o dataset.
3. Alterando qualquer outra configuração do deployment, como por exemplo o número de réplicas ou a quantidade de recursos reservados para os containers.

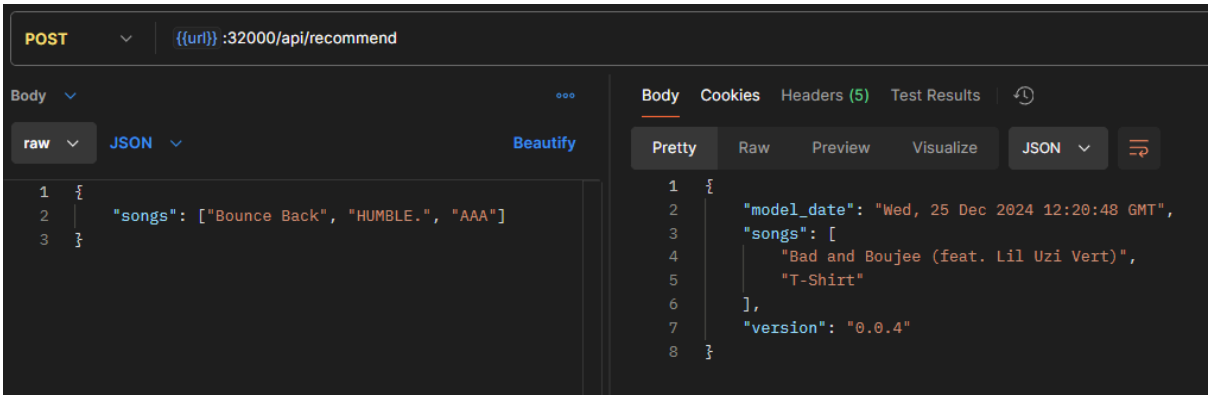
Assim, essas três ações anteriormente citadas serão utilizadas como teste de funcionamento do nosso CD.

Ao executar os testes anteriores, obtivemos as seguintes métricas para o tempo que um novo deployment demorou para ser detectado e efetuado pelo argoCD:

1. Atualizar o código (alterando a tag do container de geração do modelo): **3 minutos e 15 segundos.**
2. Atualizar o *dataset* do DS1 para DS2: **2 minutos e 49 segundos**.
3. Atualizar o deployment em si, reduzindo o número de réplicas: **4 minutos e 30 segundos.**

Ao executar os testes acima, a API não ficou indisponível em momento algum, pois o pod antigo era derrubado somente após o novo pode estar rodando normalmente.

A figura a seguir mostra um exemplo de chamada ao endpoint da API. Note que, fazendo a requisição do localhost, é necessário acessar a API pela porta 32000, que é o nodePort que o service do kubernetes aloca para expor o container à internet.



3. Outras questões

3.1- Como a API detecta mudanças no modelo?

A API em si não detecta mudanças no modelo. Sempre que há um novo deployment, o container **`model_generator`** que gera o modelo é executado antes do container da API, assim sempre que a API está de pé já existe um modelo novo já criado em uma pasta padrão compartilhada entre os dois containers.

Sempre que há uma mudança no código do gerador de modelos ou do dataset usado é gerado um novo deployment. Assim, a API sempre está portando o mais recente modelo gerado.

3.2- Como os containers obtêm o novo dataset quando forem gerar o novo modelo?

Na realidade o único container que utiliza do dataset é o **`model_generator`**, que obtém o dataset ao executar **`wget -P /tmp/data $REPO_URL`** no entrypoint do container. Como falado anteriormente, **`REPO_URL`** é definida no deployment do kubernetes. **`/tmp/data`** é um diretório temporário que existe dentro deste container, utilizado apenas para armazenar o dataset.