2. Hands-On - Instalação do Helm Chart

Pre-Requisitos

- Criar um perfil no Minikube para subir um cluster do zero
  - minikube start --profile=testeJupyter

Instalação Objeto Default

Acessar o link https://artifacthub.io/packages/helm/jupyterhub/jupyterhub

- Clicar no canto direito em "Install"

- Copiar o comando de `Add Repository` e colar no seu terminal
- Executar `helm repo update` para garantir a versão mais recente
- Criar um arquivo default chamado `config.yaml` e editar com comentários basicos
- O comando abaixo instala um ambiente de notebook default com um dummy autenticator

```
helm upgrade --cleanup-on-fail \
  --install <helm-release-name> jupyterhub/jupyterhub \
  --namespace <k8s-namespace> \
  --create-namespace \
  --version=<chart-version> \
  --values config.yaml
```

- Sem o minikube tunnel configurado, Kubernetes ira mostrar o ip externo como PENDING
  - `minikube tunnel` executa como um processo, criando uma rota de rede do HOST da máquina para o service CIDR do cluster minikube utilizando o endereço IP como

gateway.
- O comando tunnel expõe o IP externo diretamente para qualquer programa executando no sistema operacional do host.

Customização Simples do Manifesto

- Editar o arquivo config.yaml com os parametros a serem modificados
- Executar:

```
helm upgrade --cleanup-on-fail \
  <helm-release-name> jupyterhub/jupyterhub \
  --namespace <k8s-namespace> \
  --version=<chart-version> \
  --values config.yaml
```

Customizações de ambiente do Usuário

Seguir documentação: https://z2jh.jupyter.org/en/stable/jupyterhub/customizing/user-environment.html

Por padrão cada usuário possui 10gb de espaço no disco que irá persistir entre restarts do servidor em um PV via PVC. Esse disco é montado no diretório home (home/jovyan). Tudo que for escrito nessa pasta será persistido, tudo que for escrito fora será resetado durante os restarts.

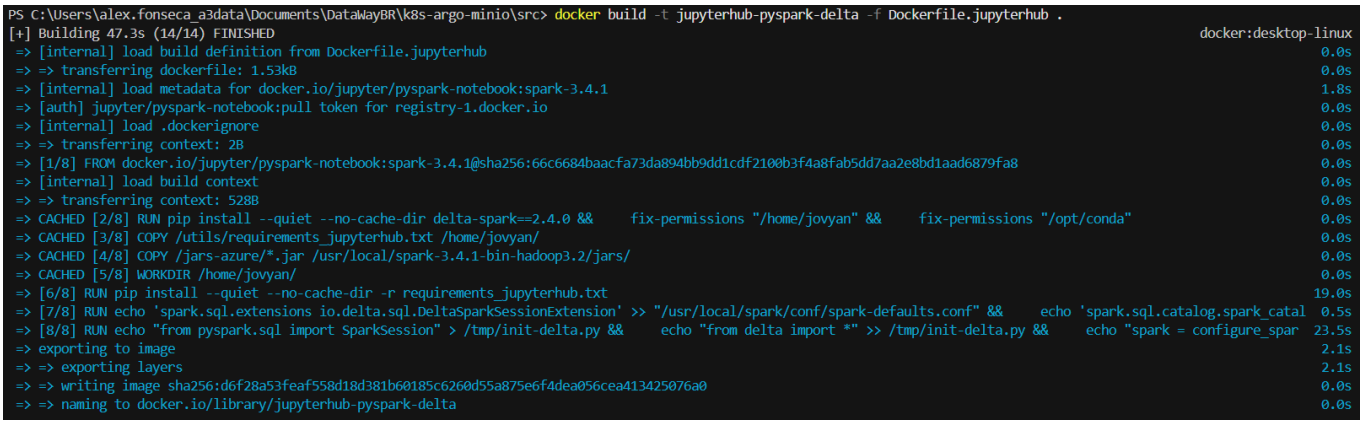
Por padrão, o jupyterhub nasce com uma classe de autenticação Dummy, o que isso significa?? Significa que qualquer usuário e qualquer senha pode entrar no ambiente (não recomendado para produção). O jupyter possui suporte a diversas classes de autenticação como:

- github
- google
- azure active directory

Tópicos importantes para ensinar:

Dockerfile customizado para Jupyter com PySpark

- docker login
- docker build -t jupyterhub-pyspark-delta -f Dockerfile.jupyterhub .
- docker tag jupyterhub-pyspark-delta:latest alexno9/jupyterhub-pyspark-delta:v01
- docker push alexno9/jupyterhub-pyspark-delta:v01



O dockerfile abaixo é necessário devido as dependencias do jupyterhub. Não podemos utilizar a imagem do modulo 1 devido a incompatibilidade de sistemas

```
FROM jupyter/pyspark-notebook:spark-3.4.1
# Metadados do imagem criada
LABEL project="Plataforma de Dados no Kubernetes" \
      mainteiners="DataWay BR" \
      version="1.0" \
      description="Jupyter Notebook com PySpark + Delta e Azure" \
      data.creation="2024-11-10"

ARG DELTA_CORE_VERSION="2.4.0"
RUN pip install --quiet --no-cache-dir delta-spark==${DELTA_CORE_VERSION} && \
     fix-permissions "${HOME}" && \
     fix-permissions "${CONDA_DIR}"

USER root

# Copia as libs
COPY requirements.txt /home/jovyan/
# Copia os jars de conexão ABFSS para a pasta do Spark
COPY /jars-azure/*.jar /usr/local/spark-3.4.1-bin-hadoop3.2/jars/

# Vai para a pasta
WORKDIR /home/jovyan/

# Atualiza as dependencias do pip e instala o pyspark
RUN pip install --quiet --no-cache-dir -r requirements.txt
RUN echo 'spark.sql.extensions io.delta.sql.DeltaSparkSessionExtension' >>
"${SPARK_HOME}/conf/spark-defaults.conf" && \
    echo 'spark.sql.catalog.spark_catalog
org.apache.spark.sql.delta.catalog.DeltaCatalog' >> "${SPARK_HOME}/conf/spark-
defaults.conf"

# Teste de build do spark com o container
```

```
USER ${NB_UID}

RUN echo "from pyspark.sql import SparkSession" > /tmp/init-delta.py && \
    echo "from delta import *" >> /tmp/init-delta.py && \
    echo "spark =
configure_spark_with_delta_pip(SparkSession.builder).getOrCreate()" >>
/tmp/init-delta.py && \
    python /tmp/init-delta.py && \
    rm /tmp/init-delta.py
```

Caso queira executar como Jupyter Notebook

```
# Executar como jupyter notebook
= docker run -it
             -p 8888:8888
             -e GRANT_SUDO=yes
             --user root
             -v jupytervolume:/home/jovyan/work
             jupyter/pyspark-notebook:latest
```

Trechos modificar docker customizado

```
singleuser:
  image:
    # You should replace the "latest" tag with a fixed version from:
    # https://hub.docker.com/r/jupyter/datascience-notebook/tags/
    # Inspect the Dockerfile at:
    # https://github.com/jupyter/docker-stacks/tree/HEAD/datascience-
notebook/Dockerfile
    name: jupyter/datascience-notebook
    tag: latest
  # `cmd: null` allows the custom CMD of the Jupyter docker-stacks to be used
  # which performs further customization on startup.
  cmd: null
```

Trecho para criar Autenticação de Acesso com o GitHub

- Necessário configurar um usuário admin + usuários do github para liberar no jupyterhub
- Utilização de secrets não funcionam por limitação do yaml
- Criação de aplicação Oauth dentro do Github
- Tomar cuidado com a url:porta add no callback url do oauth git

```yaml
hub:
  config:
    Authenticator:
      admin_users:
        - dwadmin
      allowed_users:
        - Alexno9
    GitHubOAuthenticator:
      client_id: <id>
      client_secret: <secret>
      oauth_callback_url: <url>
    JupyterHub:
      authenticator_class: github
```

Customização de recursos do Usuário

Tópicos importantes para ensinar:

Configuração de CPU e Memória

```yaml
singleuser:
  memory:
    limit: 1G
    guarantee: 1G
  cpu:
    limit: .5
    guarantee: .5
```

Configuração de StorageClass Específico

```yaml
singleuser:
  storage:
    dynamic:
      storageClass: <storageclass-name>
```

Configuração de Size do Storage por Usuário

O valor default de requisição de storage é `10Gi` volume por usuário. Recomendável utilizar o IEC binary prefixes (Ki, Mi, Gi, etc) para especificar quanto de storage voce precisa:

- `2Gi` (IEC binary prefix) => `(2 * 1024 * 1024 * 1024)` bytes
- `2G` (SI decimal prefix) => `(2 * 1000 * 1000 * 1000)` bytes.

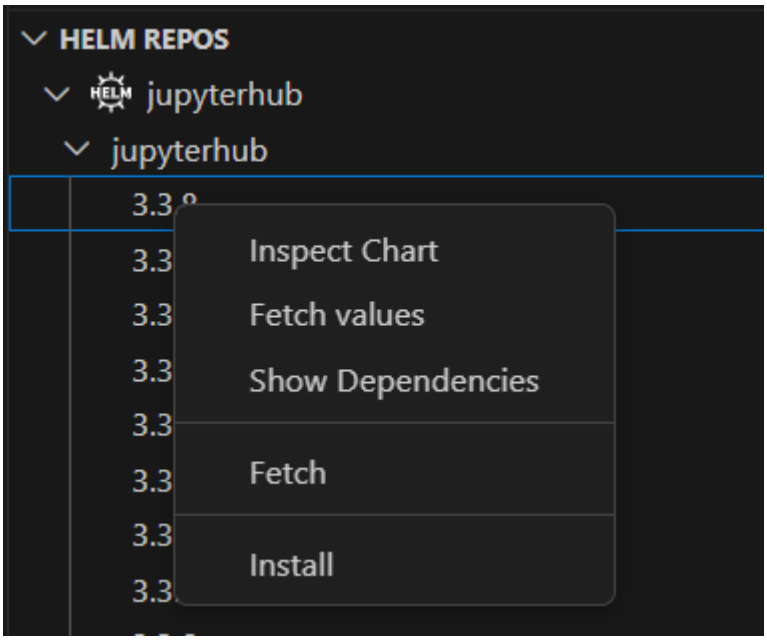
```
singleuser:
  storage:
    capacity: 2Gi
```

Caso queira desabilitar a persistência de dados no storage

```
singleuser:
  storage:
    type: none
```

Troca do Helm Manual p/ ArgoCD

1. Baixar os arquivos do repositório Helm a partir da versão mais recente (3.3.8). Clicar na opção Fetch (irá baixar todo o conteúdo para a sua máquina).



2. Transpor toda a nossa configuração realizada no arquivo config.yaml para o arquivo values.yaml baixado.

```
singleuser:
  image:
    # You should replace the "latest" tag with a fixed version from:
    # https://hub.docker.com/r/jupyter/datascience-notebook/tags/
    # Inspect the Dockerfile at:
    # https://github.com/jupyter/docker-stacks/tree/HEAD/datascience-
notebook/Dockerfile
    name: alexno9/jupyterhub-pyspark-delta
    tag: v01
```

```yaml
  # `cmd: null` allows the custom CMD of the Jupyter docker-stacks to be used
  # which performs further customization on startup.
  cmd: null
  # storage: capacity define a quantidade maxima de disco que será utilizada
por cada usuário
  storage:
    capacity: 2Gi
# Sessão para todo o hub do Jupyter
hub:
  config:
    Authenticator:
      admin_users:
        - dwadmin
      allowed_users:
        - Alexno9
    GitHubOAuthenticator:
      client_id: Ov23liFtVGtAfB4W6LCs
      client_secret: aac7520211aa1a8225fc534e65ff3356c639c247
      oauth_callback_url: http://localhost/hub/oauth_callback
      #client_id:
      #  valueFrom:
      #    secretKeyRef:
      #      name: github-client  # Nome da secret
      #      key: client_id       # Chave da secret
      #client_secret:
      #  valueFrom:
      #    secretKeyRef:
      #      name: github-client  # Nome da secret
      #      key: client_secret   # Chave da secret
    JupyterHub:
      authenticator_class: github
```

4. Criar aplicação para o ArgoCD com apontamento para a pasta dos arquivos