

2. Hands-On Orquestração

Deploy do Airflow via Helm Chart + ArgoCD

Criar Minikube Adequado p/ Teste

Vamos configurar um novo minikube com mais potência pra esse teste:

- `minikube start -p testeAirflow --driver=docker --cpus=4 --memory=6152`

Add Repo Oficial Helm Chart + Instalação Default

Vamos adicionar o repositório helm e ver um comando de instalação default

- `helm repo add apache-airflow https://airflow.apache.org/`
- `helm install my-airflow apache-airflow/airflow --version 1.15.0`

Estrutura do Airflow Utilizado como Guia:

- Versão: 2.9.3
- Versão Helm Chart: 1.15.0 (02/12/24)
- executor: Kubernetes Executor ou Celery Executor
- gitSync: Repo no GitHub com subpasta
 - Conexão: ssh com chave privada
- Banco Metadata: Postgres local
- Imagem Customizada: Não
- Logs Remotos: Sim
 - Dentro do Minio: Sim
- Configuração SMTP: Não (apenas guia)

Nos próximos passos abaixo vamos adequar o manifesto para instalar o Airflow de maneira dinamica via ArgoCD:

Configuração do Executor

Ideal para o Minikube: `LocalExecutor` ou `LocalKubernetesExecutor`

Ideal para clusters na nuvem: `KubernetesExecutor`

Adequar serviço SMTP da sua escolha

No values.yaml do Airflow, preencher os parâmetros abaixo caso queira receber notificações por email:

```
#- name: AIRFLOW__LOGGING__REMOTE_TASK_HANDLER_KWARGS
#   value: '{"delete_local_copy": true}'
#- name: AIRFLOW__SMTP__SMTP_HOST
#   value: 'smtp.office365.com'
#- name: AIRFLOW__SMTP__SMTP_PORT
#   value: '587'
#- name: AIRFLOW__SMTP__SMTP_STARTTLS
#   value: 'True'
#- name: AIRFLOW__SMTP__SMTP_SSL
#   value: 'False'
#- name: AIRFLOW__SMTP__SMTP_MAIL_FROM
#   value: ''
#- name: AIRFLOW__SMTP__SMTP_USER
#   value: ''
#- name: AIRFLOW__SMTP__SMTP_PASSWORD
#   value: ''
```

Adequar login e senha das aplicações

```
# Create initial user.
defaultUser:
  enabled: true
  role: Admin
  username: admin
  email: admin@example.com
  firstName: dataway
  lastName: br
  password: Za9W0bt
#=====
postgresql:
  enabled: true
  auth:
    enablePostgresUser: true
    postgresPassword: postgres
    username: ""
    password: ""
```

Criar secret a partir da chave SSH do Github (local das dags)

A secret deve ser com base na chave ssh privada (mesma utilizada pelo ArgoCD).
Transformar a chave ssh em base64 encoded:

- `openssl base64 -in .\gitAirflow -out gitAirflowencoded.txt`

```
apiVersion: v1
kind: Secret
metadata:
  name: airflow-ssh-secret
  namespace: orchestrator
data:
  # key needs to be gitSshKey
  gitSshKey: |
    <sua secret aqui>
```

Desabilitar Hooks e CustomEnvs para o ArgoCD

Link documentação: <https://airflow.apache.org/docs/helm-chart/stable/index.html#installing-the-chart-with-argo-cd-flux-rancher-or-terraform>

Editar na seção `migrateDatabaseJob` e `createUserJob`:

- `useHelmHooks: false`
- `applyCustomEnvs: false`

Sem essas opções somente a instalação manual via Helm vai funcionar, pelo ArgoCD entrará em loop infinito sem conseguir criar o init-container `wait-for-airflow-migrations`

Logs Remotos com Minio

Para salvar os logs de maneira histórica dentro do Storage e os logs dentro do Airflow acabarem sumindo, é necessário ativar o Log Remoto e salvar em arquivos `.log` dentro de um bucket.

Processos necessários:

- Acessar o console do MinIO
 - `minikube service dlkdataway-console -n storage -p testeAirflow`
- Endereço de Endpoint do MinIO
 - `{ "endpoint_url": "http://dlkdataway-hl.storage.svc.cluster.local:9000" }`
- Minio Keys:
 - Access Key: `erAPAmQGNIS3dAb16S0p`
 - Secret Key: `yJxoNbQMMWJsQ0wSJ1PNMYKn3vFRPq9MLy8CLHGk`
- Airflow Connection

Add Connection

Connection Id *	minio_s3_conn
Connection Type *	Amazon Web Services <small>Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.</small>
Description	Conexao para salvar os logs remotos das dags executadas dentro do bucket do Minio airflow-logs
AWS Access Key ID	erAPAmQGNIS3dAb16SOp
AWS Secret Access Key
	<code>{ "endpoint_url": "http://dlkdataway-h1.storage.svc.cluster.local:9000" }</code>

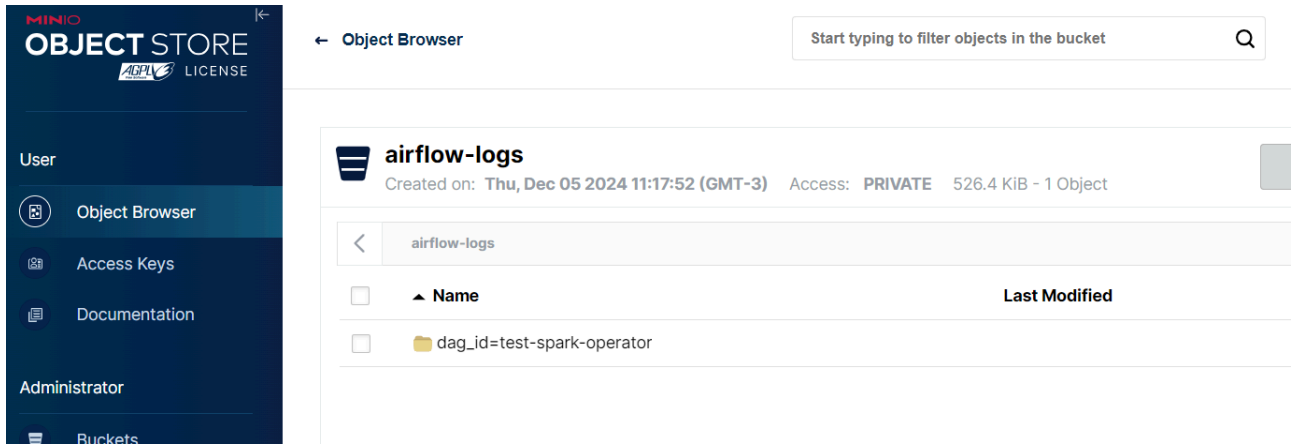
Edição dentro do values.yaml do Airflow

```
logging:
  # Opcoes para logs remotos dentro do Minio via S3
  #remote_logging: '{{- ternary "True" "False" .Values.elasticsearch.enabled
  }}'
  remote_logging: 'True'
  colored_console_log: 'False'
  # bucket dentro do minio
  remote_base_log_folder: 's3://airflow-logs'
  # nome da conexao do tipo amazon web services dentro do Airflow
  remote_log_conn_id: minio_s3_conn
  encrypt_s3_logs: False
```

Dentro do bucket do MinIO, é criada a hierarquia:

- dag_id=nome-dag
 - task_id=nome-task
 - arquivo.log
- O log é criado próximo de finalizar a execução da dag, onde o log é exportado pra o local

- Com o tempo, o Airflow vai eliminando os logs da interface.



Instalação via ArgoCD

Criação da aplicação na pasta do repositório:

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: airflow-dataawaybr
  namespace: argocd
spec:
  project: default
  source:
    repoURL: 'git@github.com:Alexno9/k8s-argo-minio.git'
    path: helm-chart/airflow
    targetRevision: main
    helm:
      valueFiles:
        - values.yaml
  destination:
    server: 'https://kubernetes.default.svc'
    namespace: orchestrator
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

Pós Instalação - Permissão entre Airflow & Spark Operator

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: airflow-dataawaybr-scheduler
```

```
namespace: orchestrator
subjects:
- kind: ServiceAccount
  name: airflow-datawaybr-scheduler
  namespace: orchestrator
roleRef:
  kind: ClusterRole
  name: cluster-admin
  apiGroup: rbac.authorization.k8s.io
```

Pós Instalação - Airflow Connection Kubernetes

- Navegar até `Connections`
- Criar uma nova no símbolo `+`
- Escolher a conexão do tipo `Kubernetes Cluster`
- Marcar a opção `in cluster configuration`
- No campo `namespace` , colocar o namespace do Spark Operator
- Salvar

Edit Connection

Connection Id *

kubernetes_minikube_conn

Connection Type *

Kubernetes Cluster Connection

Connection Type missing? Make su

Description

In cluster configuration



Kube config path

Kube config (JSON format)

Namespace

operator

Airflow UI

Airflow - DataWayBR






All1Active1Paused0

Running0Failed0

Filter DAGs by tag

Search DAGs

Auto-refresh

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
 test-spark-operator	datawaybr		None	2024-12-04, 20:47:45			 	...

« < 1 > »

Showing 1-1 of 1 DAGs

Version: v2.9.3
Git Version: .release:81845de9d95a733b4eb7826aaabe23ba9813eba3

DAG: test-spark-operator

05/12/2024 12:51:29

All Run Types

All Run States

Clear Filters

Press **shift** + **/** for Shortcuts

deferredfailedqueuedremovedrestartingrunningscheduled

test-spark-operator

Duration

00:15:38

00:07:49

00:00:00

job_spark_demo

DAG test-spark-operator

Details

Graph

Gantt

Code

Audit Log

Run Duration

Calendar

DAG Runs Summary

Total Runs Displayed

2

Total success

2

First Run Start

2024-12-04, 20:01:07 UTC

Last Run Start

2024-12-04, 20:47:46 UTC

Max Run Duration

00:15:38

Mean Run Duration

00:09:11

Min Run Duration

00:02:44

» DAG

test-spark-operator

Run

2024-12-05, 14:33:27 UTC

Task

job_spark_demo

⚙️ Details

📊 Graph

📅 Gantt

🔗 Code

🔍 Audit Log

📖 Logs

⚙️ XCom

🕒 Task Duration

(by attempts)

1

All Levels

All File Sources

```
airflow-datawaybr-scheduler-0.airflow-datawaybr-scheduler-orquestrator.svc.cluster.local
*** Found logs in s3:
    * s3://airflow-logs/dag_id-test-spark-operator/run_id-manual__2024-12-05T14:33:27.159910+00:00/task_id-job_spark_demo/attempt=1.log
[2024-12-05, 14:33:30 UTC] {local_task_job_runner.py:129} ▶ Pre task execution logs
[2024-12-05, 14:33:31 UTC] {spark_kubernetes.py:282} INFO - Creating sparkApplication.
[2024-12-05, 14:33:31 UTC] {base.py:84} INFO - Using connection ID 'kubernetes_minikube_conn' for task execution.
[2024-12-05, 14:33:31 UTC] {custom_object_launcher.py:301} WARNING - Spark job submitted but not yet started. job_id: job-spark-demo-gwhyiym3
[2024-12-05, 14:33:44 UTC] {custom_object_launcher.py:301} WARNING - Spark job submitted but not yet started. job_id: job-spark-demo-gwhyiym3
[2024-12-05, 14:33:57 UTC] {custom_object_launcher.py:301} WARNING - Spark job submitted but not yet started. job_id: job-spark-demo-gwhyiym3
2024-12-05T14:33:57.907871817Z
[2024-12-05, 14:34:07 UTC] {baseoperator.py:400} WARNING - SparkKubernetesOperator.execute cannot be called outside TaskInstance!
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] spark 14:33:57.81 INFO ==>
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] spark 14:33:57.82 INFO ==> Welcome to the Bitnami spark container
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] spark 14:33:57.83 INFO ==> Subscribe to project updates by watching https://github.com/bitnami/containers
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] spark 14:33:57.83 INFO ==> Submit issues and feature requests at https://github.com/bitnami/containers/issues
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] spark 14:33:57.83 INFO ==>
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] :: loading settings :: url = jar:file:/opt/bitnami/spark/jars/ivy-2.5.1.jar!/org/apache/ivy/core/settings/ivysettings.xml
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] Ivy Default Cache set to: /tmp/.ivy/cache
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] The jars for the packages stored in: /tmp/.ivy/jars
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] io.deltalabsdelta-core_2.12 added as a dependency
[2024-12-05, 14:34:07 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] :: resolving dependencies :: org.apache.spark#spark-submit-parent-54e52727-0611-4087-8339-7d5883922c75;1.0
[2024-12-05, 14:34:10 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] confs: [default]
[2024-12-05, 14:34:10 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] found io.deltalabsdelta-core_2.12;2.4.0 in central
[2024-12-05, 14:34:12 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] found io.deltalabsdelta-storage;2.4.0 in central
[2024-12-05, 14:34:13 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] found org.antlr4#antlr4-runtime;4.9.3 in central
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] downloading https://repo1.maven.org/maven2/io/delta/delta-core_2.12/2.4.0/delta-core_2.12-2.4.0.jar ...
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] [SUCCESSFUL ] io.deltalabsdelta-core_2.12;2.4.0!delta-core_2.12.jar (1380ms)
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] downloading https://repo1.maven.org/maven2/io/delta/delta-storage/2.4.0/delta-storage-2.4.0.jar ...
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] [SUCCESSFUL ] io.deltalabsdelta-storage;2.4.0!delta-storage.jar (275ms)
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] downloading https://repo1.maven.org/maven2/org/antlr/antlr4-runtime/4.9.3/antlr4-runtime-4.9.3.jar ...
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] [SUCCESSFUL ] org.antlr4#antlr4-runtime;4.9.3!antlr4-runtime.jar (304ms)
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] :: resolution report :: resolve 5379ms :: artifacts dl 1976ms
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] :: modules in use:
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] io.deltalabsdelta-core_2.12;2.4.0 from central in [default]
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] io.deltalabsdelta-storage;2.4.0 from central in [default]
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] org.antlr4#antlr4-runtime;4.9.3 from central in [default]
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] -----
[2024-12-05, 14:34:14 UTC] {pod_manager.py:472} INFO - [spark-kubernetes-driver] |                                     | modules | artifacts |
```

Deploy do Airflow com Logs no Azure Storage Account

Build da Imagem Docker com Airflow customizado

A imagem customizada uma padrão do Airflow para ter as libs python do seu projeto + arquivo de config local que permite salvar os logs dentro da Azure (AWS s3 já é nativo):

```
# Use uma versão específica do Airflow
FROM apache/airflow:2.7.1

# Copia as libs externas para o container
COPY requirements.txt /
COPY airflow_local_settings.py /home/airflow/.local/lib/python3.8/site-packages/airflow/config_templates/airflow_local_settings.py

# Instale pacotes necessários e dependências Python
USER root
RUN apt-get update
RUN apt-get install -y --no-install-recommends \
    python3-dev \
    build-essential \
    python3-pip
```

```
RUN rm -rf /var/lib/apt/lists/*
USER airflow
RUN pip install --no-cache-dir -r /requirements.txt
```

Com a imagem dentro do DockerHub, devemos cadastra-la no values.yaml do Airflow:

```
# Images
images:
  airflow:
    repository: alexno9/airflow-custom
    tag: v01
    # Specifying digest takes precedence over tag.
    digest: ~
    pullPolicy: IfNotPresent
```

Adequar local externo de armazenamento de logs

Preenchendo os valores abaixo, o Airflow substitui dentro do arquivo airflow.cfg.

Valores para salvar em blob storage da Azure:

```
logging:
  #remote_logging: '{{- ternary "True" "False" .Values.elasticsearch.enabled
  }}'
  base_log_folder: '/opt/airflow/logs'
  remote_logging: 'False'
  remote_base_log_folder: 'wasb://logs-
  airflow@staccdatabaybrlogs.blob.core.windows.net/logs'
  remote_log_conn_id: 'azure_blob_storage_logs'
  delete_local_copy: 'False'
  colored_console_log: 'False'
```

Comunicação de Todo Ambiente

Utilização de Cluster na Azure Cloud

Para aproveitarmos todo o potencial de todo o ambiente em conjunto, iremos construir as nossas aplicações dentro da nuvem Azure.

- Criação de Resource Group
- Criação de Storage Account
- Criação de Cluster AKS

MinIO

- Logar no ambiente: `minikube service dlkdataway-console -p testeJupyter -n storage`
- Access Key ID: erAPAmQGNIS3dAb16SOp
- Access Secret Key: yJxoNbQMMWJsQOwSJ1PNMYKn3vfRPq9MLy8CIHGk
- Utilizar os buckets criados para simular um código spark que leia e grave os dados delta nos buckets
- `kubectl create secret generic minio-s3-credentials -n operator --from-literal=AWS_ACCESS_KEY_ID=erAPAmQGNIS3dAb16SOp --from-literal=AWS_SECRET_ACCESS_KEY=yJxoNbQMMWJsQOwSJ1PNMYKn3vfRPq9MLy8CIHGk`

Airflow DAG

- Rodas a dag-minio-delta-tables para simular leitura e tratamento de dados no storage