Agora que criamos nosso ambiente virtual e o configuramos, podemos partir para a instalação do *Airflow*.

O Airflow é uma ferramenta utilizada para criar, agendar e monitorar os pipelines de dados de uma forma programática, utilizando a linguagem Python. Se assemelha a um aplicativo ou uma biblioteca, por isso, às vezes, podemos encontrar dificuldades para instalá-lo. Neste curso usaremos o pacote *pip* pra realizar a instalação desta ferramenta. Vamos lá?

Primeiramente, vamos acessar a pasta "airflowalura" no nosso terminal. Para isso, rode o comando cd Documents/airflowalura. Agora ativaremos o ambiente virtual com source venv/bin/activate. Estamos dentro do ambiente e podemos iniciar a instalação do Airflow!

cd Documents/airflowalura **COPIAR CÓDIGO**source venv/bin/activate **COPIAR CÓDIGO**

O comando pip install apache-airflow é responsável por instalar a ferramenta, mas, para desenvolver o nosso projeto, também precisaremos de alguns pacotes do próprio Airflow que darão suporte a outros programas. Podemos instalá-los juntos, portanto, adicionaremos [postgres, celery, redis] ao comando de instalação.

pip install apache-airflow[postgres, celery, redis

**COPIAR CÓDIGO**

Ainda nesse comando, precisamos identificar a versão do Airflow com o trecho ==2.3.2.

pip install apache-airflow[postgres, celery, redis]==2.3.2

**COPIAR CÓDIGO**

Por fim, passamos o arquivo de restrição *Constraint file*, que serve para especificar todas as versões das dependências que o Airflow precisa para funcionar. Para adicioná-lo, utilizamos --constraint " " e entre as aspas passamos a url do arquivo de restrição que fica no GitHub do Airflow: https://raw.githubusercontent.com/apache/airflow/constraint-2.3.2/constraints-3.9.txt.

pip install apache-airflow[postgres,celery,redis]==2.3.2 --constraint "https://raw.githubusercontent.com/apache/airflow/constraints-2.3.2/constraints-3.9.txt"**COPIAR CÓDIGO**

Observe que na url constam as versões do Airflow e do Python. É importante nos atentarmos a estes detalhes porque esses arquivos de restrição mudam de acordo com a versão utilizada. Agora basta executar o comando.

Após a instalação do Airflow, vamos iniciá-lo localmente. Mas antes, temos que setar a variável de ambiente *Airflow Home*. Essa variável é importante porque é por meio dela que o sistema consegue localizar onde a ferramenta está instalada e consegue rodá-la localmente. Sendo assim, todas as vezes que formos executar o Airflow, precisaremos, antes, importar esta variável. Para importá-la utilizamos o seguinte comando:

**export** AIRFLOW\_HOME=~/Documents/airflowalura **COPIAR CÓDIGO**

Agora, passe o comando airflow standalone para rodar localmente o Airflow na nossa máquina. Este comando inicia o banco de dados padrão da ferramenta e cria um usuário e uma senha para que possamos acessá-la, além de ativar seus principais serviços.

airflow standalone

**COPIAR CÓDIGO**

Pronto! Observe que o log de execução do comando nos retorna uma série de informações, entre elas "Airflow is ready" (ou "Airflow está pronto", em português) seguido do usuário e senha que nos permitem acessar a interface do Airflow por meio do navegador.

Caso você não tenha conseguido localizar essa informação, ela ficará salva na pasta de instalação do Airflow. Para isso, acesse a pasta "airflowalura" e observe que há um arquivo chamado "standalone\_admin\_password". Clique nele e copie a senha.

No navegador, acesse "localhost8080". Abrirá uma guia onde você deve passar seu *username* ("nome de usuário" em português) e colar a senha. Pronto! Acessamos a interface do Airflow!

Note dois avisos em amarelo no topo da tela. O primeiro informa que estamos utilizando um banco de dados *SQLite* que não pode ser usado em produção, apenas em testes e desenvolvimento. O segundo indica que o executor *Sequential* também não pode ser usado em produção. Esses avisos surgem porque ainda não mexemos em nenhuma configuração do Airflow, então estão sendo usadas as configurações padrões que só suportam testes e desenvolvimento.

Logo mais, aprenderemos a trabalhar com esses ajustes a fim de tornar nossa instalação pronta para produção e, para que possamos visualizar o que acontece à medida que alteramos essas configurações, precisaremos ter um DAG executando algumas *taks* ("tarefas" em português), assim conseguimos observar como as configurações afetam a dinâmica de execução dessas tarefas.

Nos próximos vídeos começaremos a desenvolver o DAG. Te vejo lá!

Vamos começar com as alterações na configuração do Airflow para que possamos utilizar o executor Local.

Para começar, temos que acessar o arquivo de configuração do Airflow que fica dentro da pasta de instalação "airflowalura". Abra a pasta em questão no *VS Code* e, na barra lateral esquerda, onde constam os arquivos internos a ela, procure por "airflow.cfg". Dê dois cliques para abri-lo e perceba que possui diversas variáveis de configuração.

Pressione "Ctrl + F" para abrir a barra de pesquisa do Airflow e busque por "executor". Essa pesquisa irá destacar a ocorrência do termo pesquisado no corpo deste arquivo. Perceba que, entre as ocorrências, há uma variável com este nome definida como "SequentialExecutor". Podemos apagar essa definição e substituir por "LocalExecutor" para que passemos a utilizar o executor local. Tecle "Ctrl + S" para salvar essa alteração.

Alteramos o executor do Airflow, mas não é suficiente para que possamos utilizá-lo porque, por padrão, essa ferramenta utiliza o banco de dados *SQLite*, que só permite uma conexão por vez. Como queremos que nossas tarefas sejam executadas em paralelo, precisamos de um banco que suporte mais de uma conexão simultaneamente, então utilizaremos o *Postgres*, sugerido pela própria documentação do Airflow.

Para isso, abra o terminal da máquina e execute o comando sudo apt update para atualizar os pacotes.

sudo apt **updateCOPIAR CÓDIGO**

Em seguida, digite sua senha e aguarde o carregamento das atualizações. Ao finalizar, execute o comando sudo apt upgrade.

sudo apt upgrade**COPIAR CÓDIGO**

Pronto! Os pacotes foram atualizados e, agora, podemos rodar o comando de instalação do *Postgres*:

sudo apt install postgresql postgresql-contrib**COPIAR CÓDIGO**

Após a execução do comando anterior, tecle "y" para confirmar e aguarde a conclusão do carregamento.

Com o *Postgres* instalado, vamos acessar o usuário para conseguir criar o banco de dados a ser utilizado. Para isso, utilizamos o comando sudo -i -u postgres.

sudo -**i** -u postgres**COPIAR CÓDIGO**

Dentro do usuário, criaremos um banco de dados chamado "airflow\_db" utilizando o código createdb airflow\_db.

createdb airflow\_db**COPIAR CÓDIGO**

Agora podemos acessá-lo com psql airflow\_db.

psql airflow\_db**COPIAR CÓDIGO**

Em seguida, criaremos um usuário e senha que o Airflow utilizará para acessar este banco. O comando para esta criação é:

**CREATE** **USER** airflow\_user **WITH** PASSWORD 'airflow\_pass'**COPIAR CÓDIGO**

Criamos um usuário chamado "airflow\_user" e uma senha "airflow\_pass". Agora, precisamos garantir que esse usuário tenha acesso a todos os privilégios necessários para alterar o banco de dados. Para isso, utilizaremos GRANT ALL PRIVILEGES ON DATABASE airflow\_db TO airflow\_user, que concede todos os privilégios de "airflow\_db" ao usuário que criamos, "airflow\_user". Isso permitirá que o usuário adicione e exclua dados do banco, além de realizar conexões.

GRANT ALL PRIVILEGES ON DATABASE airflow\_db **TO** airflow\_user**COPIAR CÓDIGO**

Agora que criamos e configuramos nosso banco, podemos usar o comando exit, duas vezes, para sair do banco de dados e do usuário.

exit**COPIAR CÓDIGO**exit**COPIAR CÓDIGO**

Hora de conectar esse banco ao Airflow! No arquivo de configuração no *VS Code*, pesquisaremos por "sql\_alchemy", que nos indicará a variável "sql\_alchemy\_conn". Perceba que ela armazena uma url, porque o Airflow utiliza o framework *SQLAlchemy* para fazer a conexão com o banco de dados.

Para conectar ao banco de dados que queremos, substituiremos a url por outra, estruturada a partir do que nos é informado na documentação do Airflow. A estrutura consiste, basicamente, em colocar o usuário "airflow\_user", a senha "airflow\_pass" e o nome do banco de dados "airflow\_db". Ficará assim:

sql\_alchemy\_conn = postgresql+psycopg2://airflow\_user:airflow\_pass@localhost/airflow\_db**COPIAR CÓDIGO**

Tecle "Ctrl + S" para salvar.

Com a variável configurada, nos resta inicializar o banco de dados no Airflow. Para isso, retorne ao terminal da máquina, acesse a pasta "airflowalura" com o comando cd Documents/airflowalura, ative o ambiente (source venv/bin/activate) e exporte a variável de ambiente Airflow\_Home com export AIRFLOW\_HOME=~/Documents/airflowalura. Por fim, inicialize o banco de dados no Airflow com airflow db init.

cd Documents/airflowalura**COPIAR CÓDIGO**source venv/bin/activate**COPIAR CÓDIGO**AIRFLOW\_HOME=~/Documents/airflowalura**COPIAR CÓDIGO**airflow db **initCOPIAR CÓDIGO**

Após a inicialização, retorne ao VS Code e exclua a pasta "stocks". Quando o Airflow executar o nosso DAG, ela deverá ser criada novamente.

De volta ao terminal, execute o comando airflow standalone para que possamos acessar a interface do Airflow.

airflow standalone**COPIAR CÓDIGO**

Após a execução, acesse "localhost8080" no navegador e entre com usuário e senha.

Você pode encontrar esta senha no arquivo "standalone\_admin\_password.txt", dentro da pasta "airflowalura".

Na interface do Airflow, vamos procurar por "get\_stocks\_dag" e abri-lo. Em seguida, podemos ativá-lo no símbolo superior esquerdo. Ative, também, o "Auto-refresh" para que recarregue automaticamente.

Perceba que "Total runnig" varia de valores, mas em certo momento corresponde ao número 4, o que significa que há 4 DAG runs em execução. No *VS Code,* dentro de "airflowalura", podemos observar que a pasta "stocks" foi criada com as subpastas correspondentes às ações, onde constam os arquivos csv. As tarefas estão sendo executadas!

Além da configuração do executor, existem vários outros ajustes importantes para que possamos trabalhar com produção de tarefas em paralelo. Veremos algumas dessas configurações nos próximos vídeos!