



## Fundamentos

### Bootcamp: Engenharia de Dados

Prof. Dr. Neylson Crepalde

2020

## **Fundamentos**

Bootcamp: Engenharia de Dados

Prof. Dr. Neylson Crepalde

© Copyright do Instituto de Gestão e Tecnologia da Informação.

Todos os direitos reservados.

## Sumário

---

Capítulo 1.	Introdução .....	4
Capítulo 2.	Visão geral do pipeline de ciência de dados .....	6
Capítulo 3.	Extração de Dados.....	8
Capítulo 4.	Transformação de Dados.....	9
Capítulo 5.	Soluções de ETL.....	10
Capítulo 6.	Data Flow na Prática – Airflow .....	24
Capítulo 7.	Data Flow na Prática – Prefect.....	26
Capítulo 8.	Data Flow na Prática – Prefect.....	27
Referências.....		29

## Capítulo 1. Introdução

Dados em seu estado "cru" tem baixo potencial de agregar valor. Quando eles se encontram contextualizados, se transformam em **informação** e, quando processados, em **conhecimento**.

O conceito de Big Data representa a grande massa de dados que é produzida hoje no mundo. Este dado pode ser representado pelos seus três V's, a saber

- Volume;
- Velocidade;
- Variedade.

*Volume* representa o volume gigantesco com o qual dados são produzidos no mundo contemporâneo. *Velocidade* representa sua grande celeridade de produção e disseminação, enquanto *variedade* representa a multiplicidade de fontes e formatos com os quais ele pode aparecer. A mandala abaixo representa os três V's mencionados:

**Figura 1 – O que acontece em um minuto de internet em 2020.**



Fonte: <https://www.allaccess.com/merge/archive/31294/infographic-what-happens-in-an-internet-minute>.

Dados podem vir nos seguintes formatos:

- Dados em tabela;
- Fotos;
- Vídeos;
- Textos curtos (tweets, comentários);
- Textos longos;
- Transações bancárias;
- Conexões na rede;
- Dados georreferenciados.

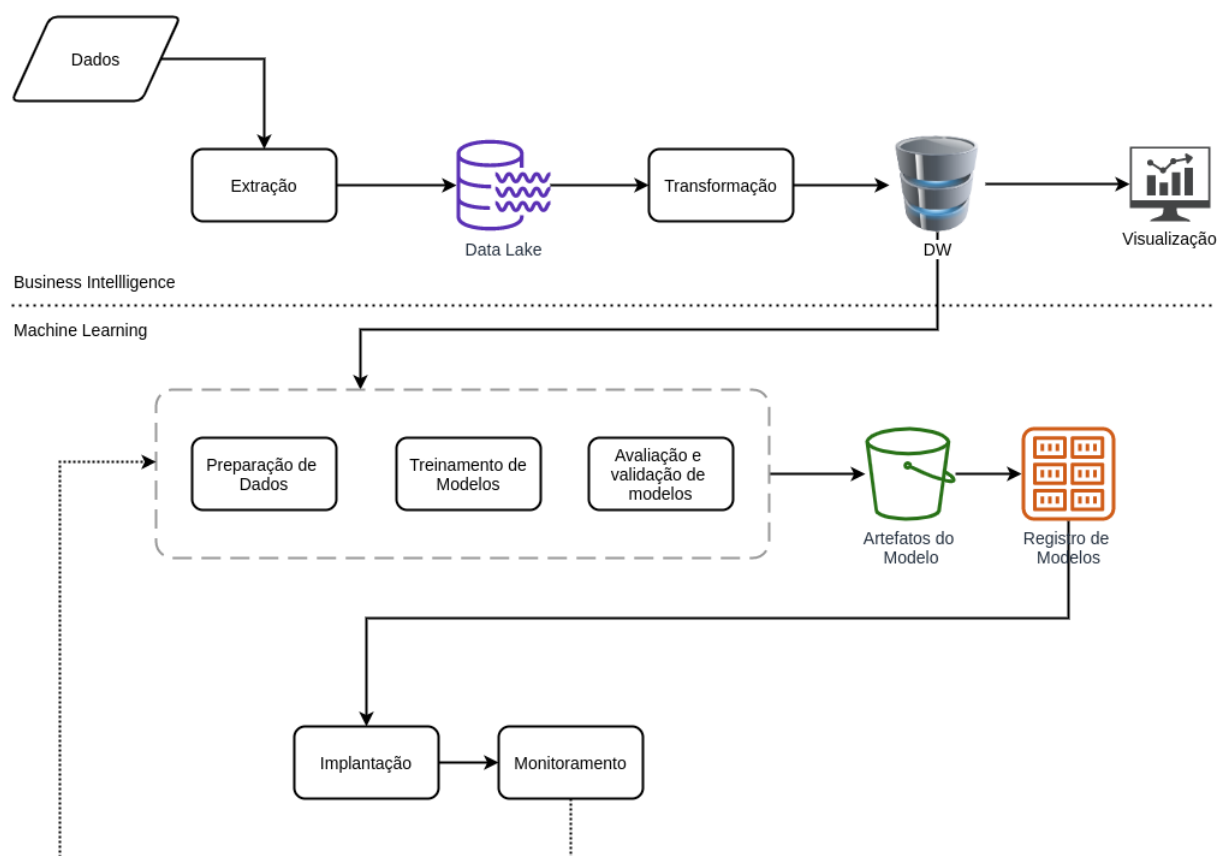
Dada essa grande variabilidade de dados, que ferramentas a pessoa Engenheira de Dados precisa conhecer e mobilizar para processar dados de fontes tão diversas?

- Repositório de dados capaz de armazenar formatos diversos (Data Lake);
- Extração, Transformação e Carga (ETL);
- Transformação pode ser entendida como técnicas de limpeza, organização, estruturação e enriquecimento de dados;
- Repositório de dados analíticos (Data Warehouse).

## Capítulo 2. Visão geral do pipeline de ciência de dados

O pipeline de Ciência de Dados pode ser representado pela figura abaixo:

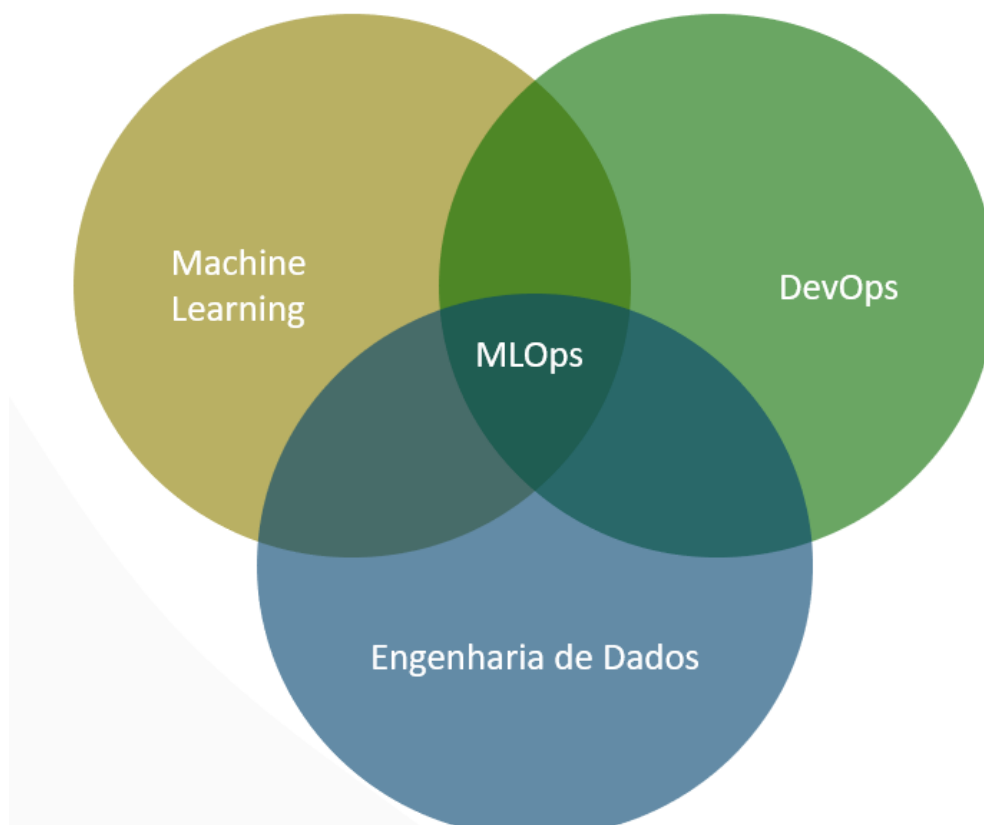
**Figura 2 – O Pipeline de Ciência de dados.**



**Fonte: Elaboração do autor.**

Dos esforços de automatização deste pipeline, nasceu a área conhecida como MLOps (*Machine Learning Operations*) a qual consiste da interseção do Machine Learning, da Engenharia de Dados e das técnicas de DevOps.

**Figura 3 – DataOps / MLOps.**



**Fonte: Elaboração do autor.**

### Capítulo 3. Extração de Dados

---

A extração de dados pode ser realizada a partir de:

- Requisições a API's;
- Crawlers para captura de dados on-line;
- Consulta a tabelas relacionais;
- Streaming de dado.

Para nossa atividade prática do Twitter, precisaremos criar uma conta na plataforma de desenvolvedor do Twitter. A plataforma pode ser acessada em <https://developer.twitter.com/en>. É necessário ter uma conta no Twitter para este cadastro. Após o cadastro e aprovação, é necessário criar um *app* para ter acesso às quatro chaves necessárias para requisições à API Stream do Twitter.

Para extração dos dados do Enade, podemos utilizar o seguinte link: [http://download.inep.gov.br/microdados/Enade\\_Microdados/microdados\\_enade\\_2019.zip](http://download.inep.gov.br/microdados/Enade_Microdados/microdados_enade_2019.zip).



## Capítulo 4. Transformação de Dados

---

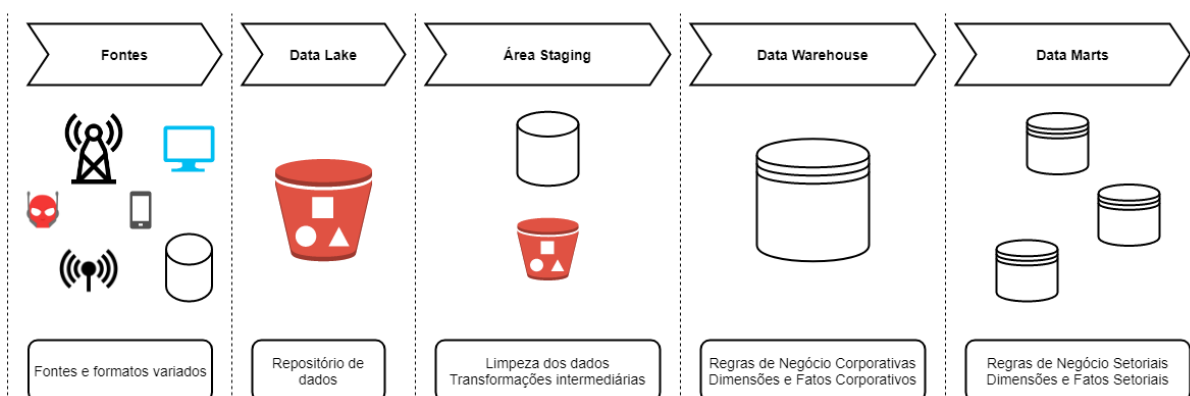
A etapa de transformação de dados pode compreender, principalmente, atividades de limpeza, organização, estruturação e enriquecimento dos dados. Esta costuma ser a etapa nos projetos de *analytics* que exige a maior parte do tempo de dedicação.

## Capítulo 5. Soluções de ETL

Neste capítulo, vamos ter um *overview* geral sobre as ferramentas de ETL, discutir os requisitos técnicos necessários para utilização e implementação, bem como as vantagens e desvantagens de cada tipo de ferramenta.

Primeiro, revisemos o pipeline de ETL:

**Figura 4 – Pipeline de ETL.**



Fonte: Adaptado de <https://www.igti.com.br/blog/o-que-e-etl-bi/>.

Do que precisamos quando pensamos nosso pipeline de ETL?

- Automatização;
- Consistência de execução de Jobs;
- Encadeamento sequencial ou paralelizado de tarefas;
- Possibilidade de conectar a diversas fontes tanto para extração quanto para entrega;
- *Scheduler* (programar execuções);
- Logs de execução;
- *Fail Safe* (possibilidade de recuperação em caso de falha);

- **Notificação** em caso de falha.

Boas ferramentas de ETL precisam nos dar a capacidade de automatizar o fluxo de dados, executá-los de maneira programada, consistente, segura contra falhas, agnóstica e com governança.

Podemos classificar as ferramentas de ETL em dois grandes subtipos, a saber, as ferramentas *Drag and Drop* e as soluções com código.

As ferramentas *Drag and Drop* possuem uma interface de usuário na qual o analista vai “montando” a sua pipeline arrastando as caixas que representam etapas do processo. Esse tipo de solução pode trazer maior facilidade de uso, visto que não é necessário implementar quase nenhum código, e pode ser amplamente utilizada por parceiros nas áreas de negócio sem formação em TI.

Soluções com código, por sua vez, permitem o desenvolvimento do pipeline através de frameworks específicos, bibliotecas, utilizando alguma linguagem de programação. As soluções com código tendem a ser mais maleáveis e customizáveis, são implantadas e escalam com maior facilidade e são extensíveis. Esta última *feature* é especialmente interessante, visto que podemos implementar extensões que vão ao encontro de necessidades mais personalizadas dos nossos times.

Algumas das soluções mais utilizadas no mercado são:

- Pentaho;
- Apache Nifi;
- Knime;
- Talend;
- KubeFlow;
- Apache Airflow;
- Prefect;

- Pachyderm;
- Argo;
- Luigi;
- Metaflow;
- Apache Kafka.

Entre as ferramentas do tipo *Drag and Drop*, podemos citar o Pentaho e o Apache Nifi.

O **Pentaho** é uma ferramenta para integração de dados. Ele possui uma interface de usuário amigável e fácil de usar, bem como conectores com diversas fontes de dados. Pentaho pode ser uma ótima ferramenta de ETL a ser utilizada por equipes com pouca experiência com programação ou que se encontram dentro de alguma área de negócio. Sua interface é amigável e intuitiva e ele possui excelente controle de logs e métricas de execução.

**Figura 5 – Exemplo da Interface do Pentaho.**

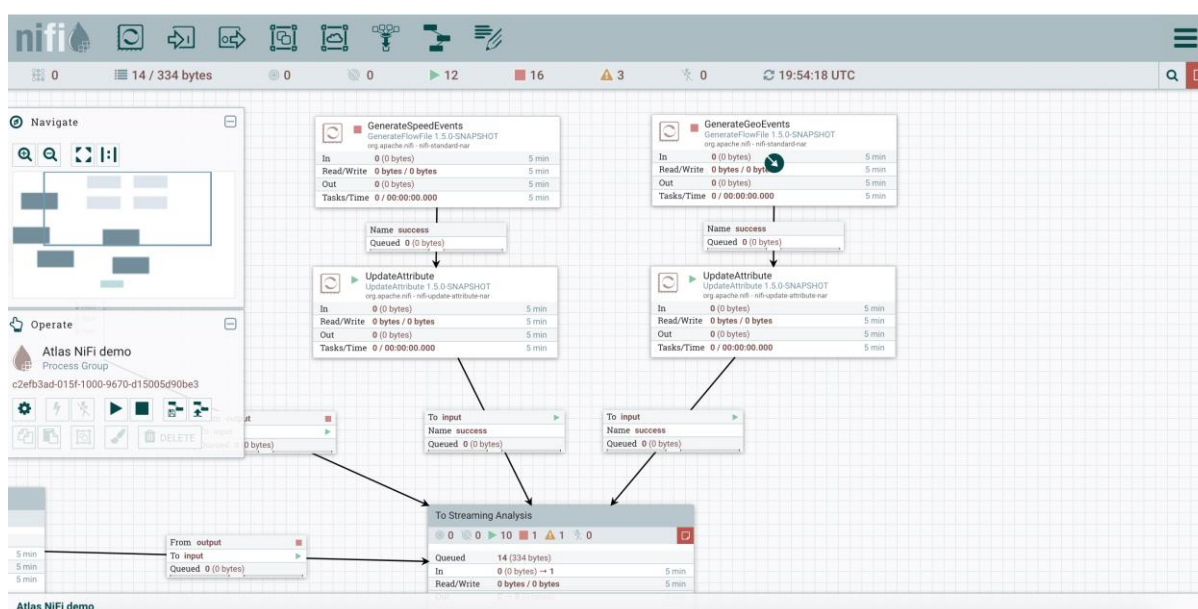


**Fonte:** <https://medium.com/soma-labs/pentaho-data-integration-a7b754fae960>.

O **Apache Nifi** é uma ferramenta de automatização de pipelines de dados. Ele se propõe a endereçar alguns problemas comuns no mercado relacionados ao ETL, a saber, falhas de sistema, limitações insuficientes de dados (dados grandes demais, pequenos demais, rápidos demais, lentos demais, etc.), mudanças de comportamento nos dados, *compliance*/segurança e melhorias contínuas que só acontecem em ambiente de produção.

Embora não seja tão simples de instalar, o Apache Nifi pode ser uma excelente ferramenta de ETL, sobretudo pela sua extensa gama de “*processors*” e sua interface gráfica intuitiva e simples de usar.

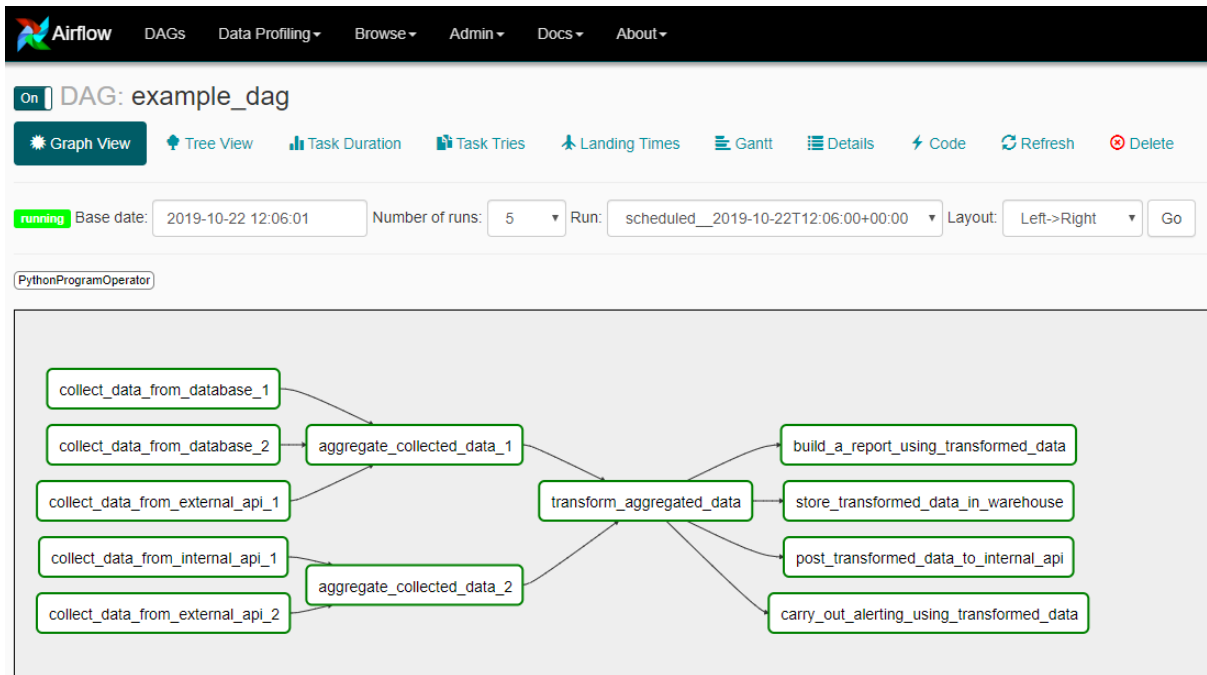
**Figura 6 – Exemplo da Interface do Apache Nifi.**



Fonte: <https://blog.cloudera.com/hdf-3-1-blog-series-part-6-introducing-nifi-atlas-integration/>.

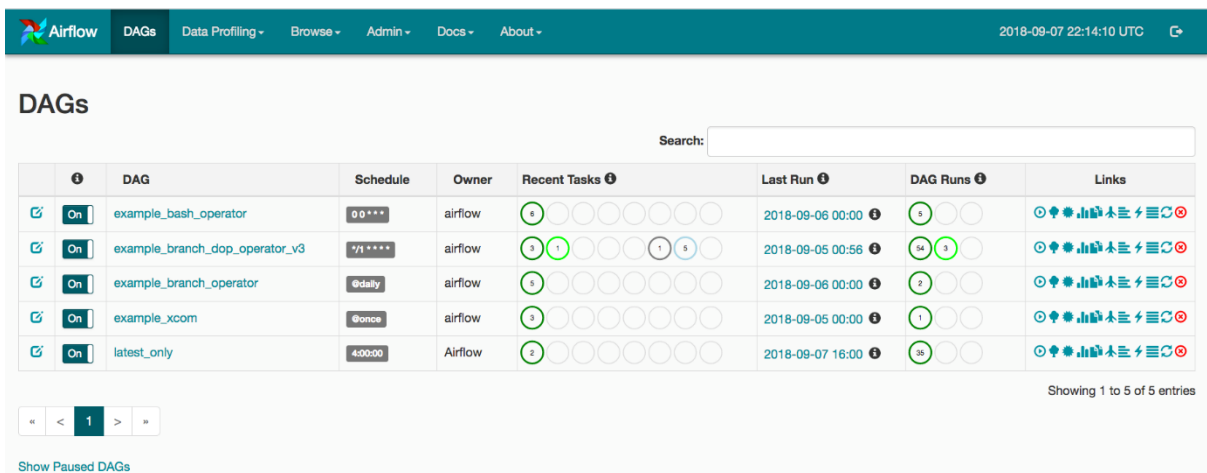
O **Apache Airflow** é uma plataforma para criar, *scheduler* (programar execuções) e monitorar *workflows*. Trata-se de um orquestrador. O projeto Airflow foi iniciado por Maxime Beauchemin quando trabalhava no AirBNB, em 2014. Em 2015, a primeira versão do software foi lançada. Em 2016, o projeto entrou para o programa de incubação da Fundação Apache e em 2019, foi anunciado como um projeto “Top Level”.

**Figura 7 – Exemplo da Interface de Controle de Flows do Apache Airflow.**



Fonte: <https://towardsdatascience.com/building-a-production-level-etl-pipeline-platform-using-apache-airflow-a4cf34203fbd>.

**Figura 8 – Exemplo da Interface do Apache Airflow.**



DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
example_bash_operator	00:00:00	airflow	5	2018-09-06 00:00	5	Links
example_branch_dop_operator_v3	*/1 * * * *	airflow	5	2018-09-05 00:58	54	Links
example_branch_operator	@daily	airflow	5	2018-09-06 00:00	2	Links
example_xcom	@once	airflow	5	2018-09-05 00:00	1	Links
latest_only	4:00:00	Airflow	2	2018-09-07 16:00	35	Links

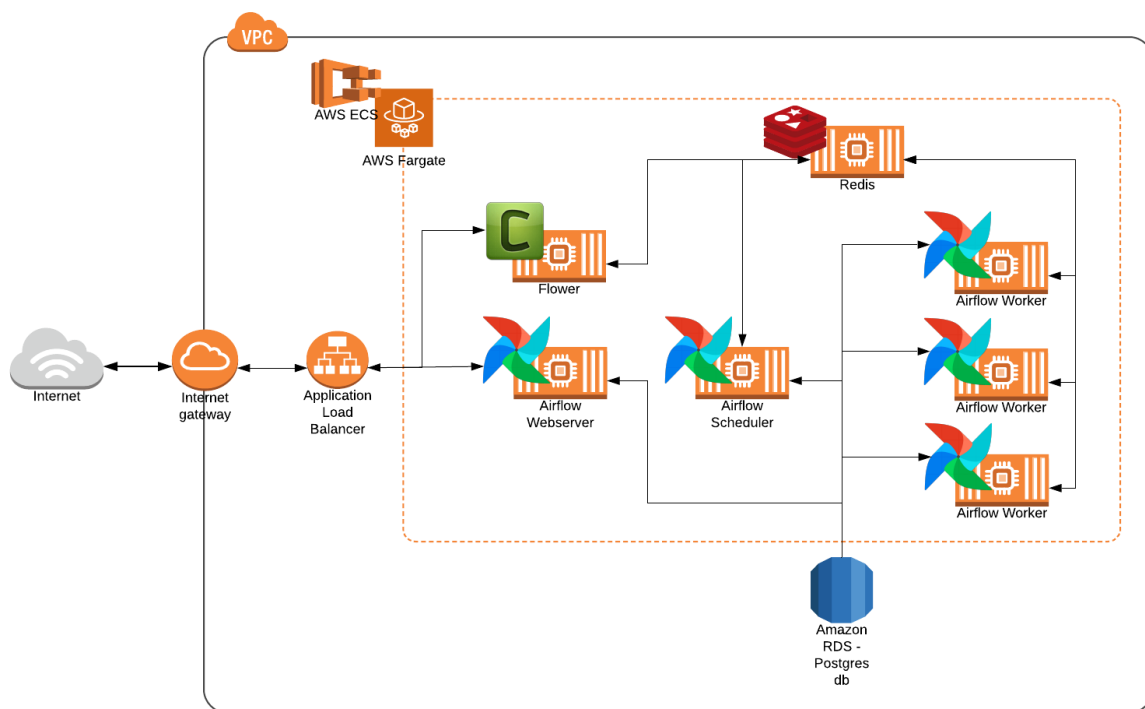
Fonte: <https://airflow.apache.org/docs/1.10.3/ui.html>.

Apache Airflow tem como princípios ser:

- **Escalável** – possui arquitetura modular que utiliza mensageria para orquestrar os workers;
- **Dinâmico** – as pipelines são definidas em Python permitindo geração dinâmica;
- **Extensível** – permite fácil desenvolvimento de operadores e bibliotecas compatíveis;
- **Elegante** – pipelines *lean* e explícitas.

Um exemplo de arquitetura do Airflow está representado na Figura 9 abaixo.

**Figura 9 – Arquitetura do Airflow usando Celery.**



Fonte: <https://towardsdatascience.com/how-to-deploy-apache-airflow-with-celery-on-aws-ce2518dbf631>.

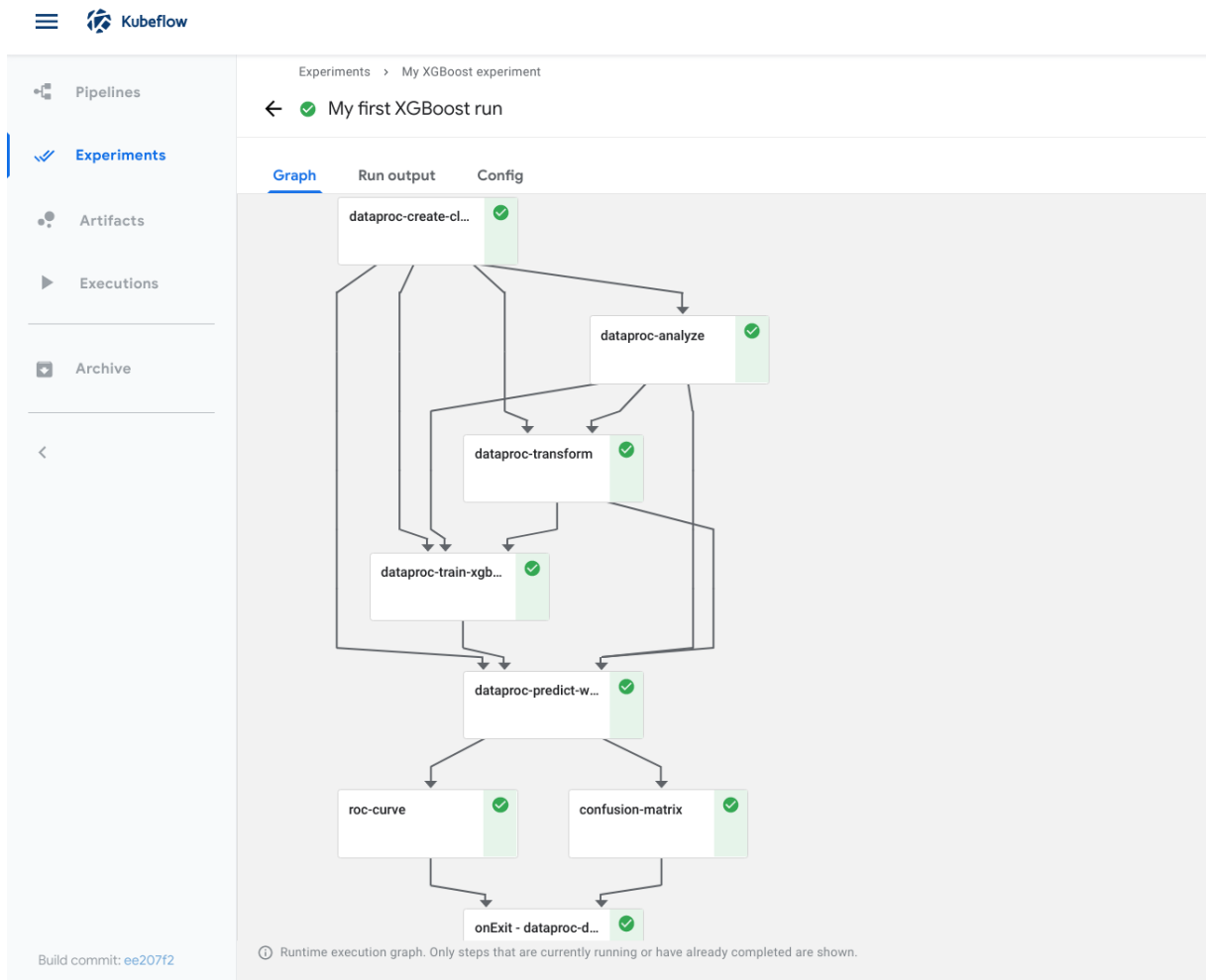
Embora o desenho arquitetural esteja apontando para uma implantação utilizando ambiente AWS, a solução é adaptável a qualquer cenário onde seja possível trabalhar com containers. O Webserver atua como interface do usuário e proporciona um meio de interação com a solução verificando os flows registrados, ativando o scheduler, acompanhando execuções, definindo conexões e parâmetros secretos como variáveis de ambiente, dentre outras atividades. O scheduler é o ponto responsável por orquestrar de fato as tarefas do Flow. Neste caso, ele usa Celery, um serviço assíncrono de enfileiramento de tarefas. Celery utiliza o banco de dados Redis para comunicação com os Workers (containers de execução) e possui sua própria interface de usuário para monitoramento do cluster chamado Flower. Airflow usa ainda um banco de dados relacional (neste caso, PostgreSQL) para registro de metadados.

AirFlow é uma das ferramentas mais utilizada pelos times de Engenharia de Dados no mundo profissional para orquestrar os processos de ETL, bem como pipelines de BI e Machine Learning. É uma ferramenta fácil de usar, escalável, e apresenta excelentes resultados em ambiente de produção.

O **KubeFlow** é uma ferramenta que se dedica a pipelines de Machine Learning escaláveis usando tecnologia Kubernetes. Foi iniciado pelo Google em um projeto vinculado à biblioteca *Tensor Flow*. Ele oferece grande parte do pipeline de Ciência de Dados e encoraja o desenho de soluções de Ciência de Dados, utilizando uma arquitetura de microsserviços (desacoplada).



**Figura 10 – Exemplo da Interface do KubeFlow Pipelines.**



**Fonte:** <https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/>.

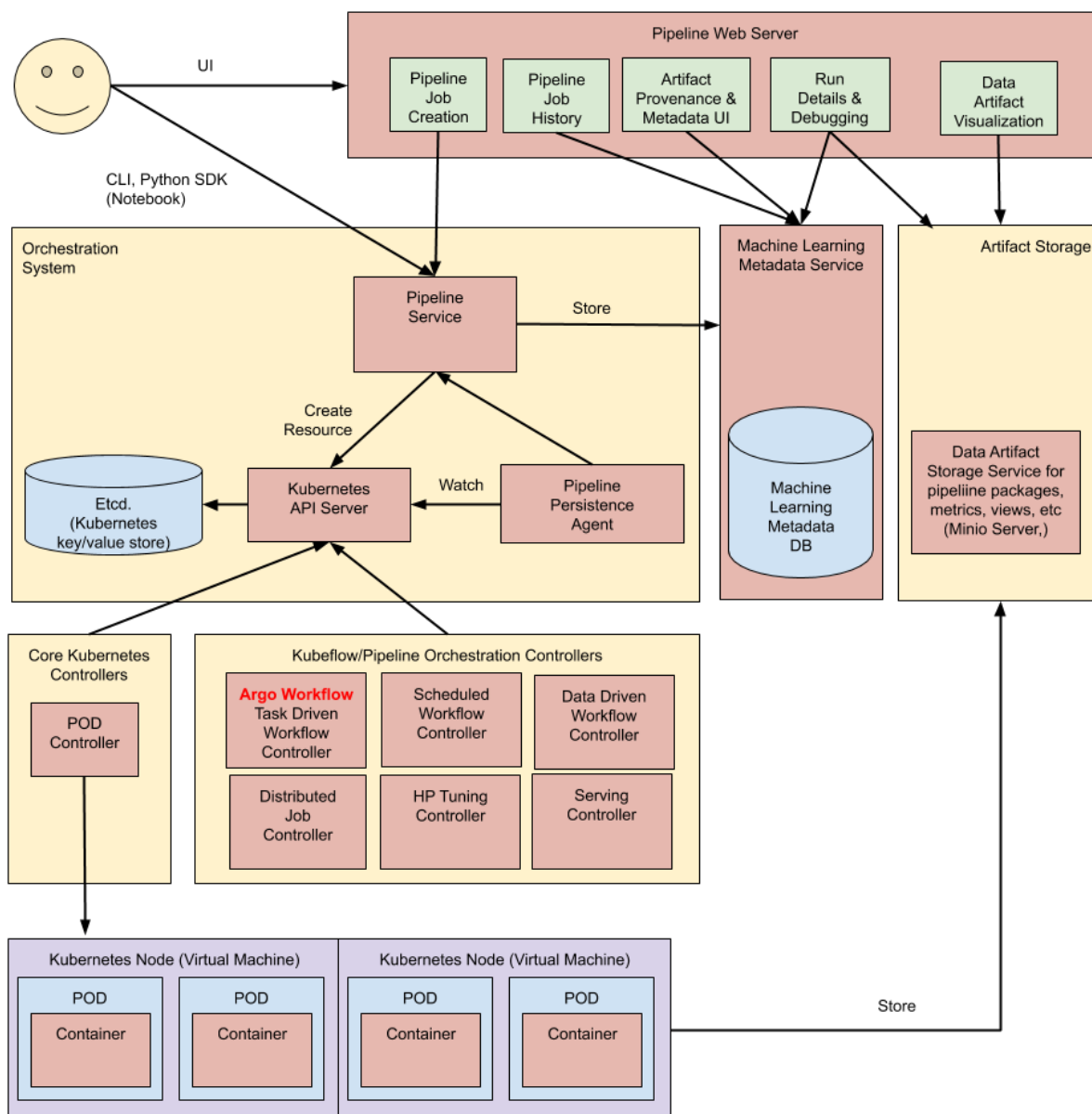
O KubeFlow possui vários componentes, entre eles:

- Dashboard;
- Metadados (logs e monitoramento de execuções);
- Um servidor de Jupyter Notebooks;
- Fairing (biblioteca para automatização de treino e deploy de Machine Learning);
- Feature Store;

- Suporte a diversos frameworks de Ciência de dados e Machine Learning;
- Tuning de Hiperparâmetros;
- KF Pipelines;
- KF Serving;
- Dentre outros.

O KubeFlow utiliza Kubernetes e, portanto, tem um desenho escalável, que permite alta disponibilidade e um alto grau de automatização. Ele agrega todas as partes de uma equipe de analytics desde a captura e estruturação dos dados passando pelo treino de modelos e experimentações até o deploy e monitoramento de soluções em ambiente de produção. A solução foi projetada especificamente para facilitar a implantação de soluções em produção com governança e bom monitoramento. Um desenho de arquitetura do KubeFlow é apresentado na Figura 11 a seguir.

**Figura 11 – Arquitetura do KubeFlow.**



Fonte: Site oficial do KubeFlow:

<https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/>.

Todos os componentes de KubeFlow são executados dentro de POD's (estrutura mínima de computação do Kubernetes). A primeira camada superior apresenta o Webserver dos pipelines. Nessa interface de usuário, podemos interagir com o KubeFlow realizando a criação de Jobs (a partir de pipelines previamente definidas com código) e acessar metadados de execuções, logs, artefatos e

visualização de dados. A segunda camada apresenta o sistema de orquestração (no nível do K8s), a gestão de metadados de Machine Learning e o Storage de artefatos. A terceira camada apresenta um sistema de orquestração no nível de execução dos pipelines. Aqui estão o controller dos POD's e o controller de execução de tarefas dos pipelines (que utiliza o framework Argo). Na quarta camada estão representados os POD's onde a computação das tarefas será de fato executada.

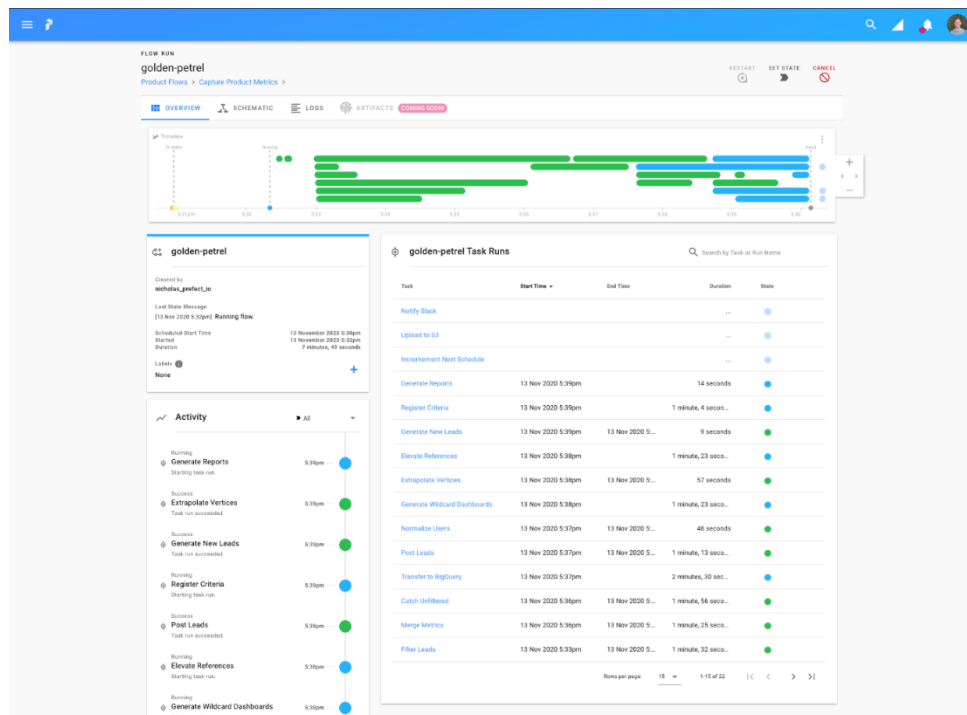
KubeFlow é uma ferramenta que engloba quase todo o pipeline de Ciência de Dados. Pode ser utilizado tanto por Engenheiros de Dados, quanto por Cientistas de Dados e Engenheiros de Machine Learning.

KubeFlow utiliza o estado da arte em tecnologia de deploy de soluções escaláveis, com alta disponibilidade e automatizadas, o Kubernetes.

Equipes com maior experiência com Kubernetes, tendem a ter melhores resultados com a ferramenta.

O **Prefect** se propõe a ser “o jeito mais fácil para automatização de dataflow”. Ele possui uma lógica muito parecida com o Airflow – criação, schedule e monitoramento de pipelines. O grande diferencial – além da biblioteca “Core” para orquestração de pipelines open source – está no fato de que Prefect oferece uma cloud própria com vários níveis de assinatura (inclusive gratuito) para servir de backend para as execuções de flows.

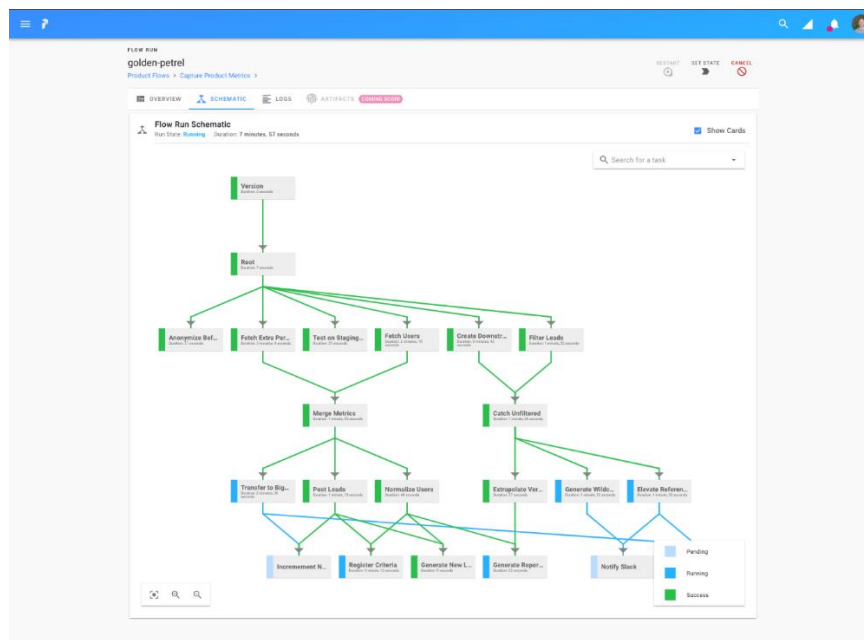
**Figura 12 – Exemplo da Interface do Prefect.**



Fonte: Documentação oficial do Prefect:

<https://docs.prefect.io/orchestration/server/architecture.html>.

**Figura 13 – Exemplo da Interface de Orquestração de Pipelines do Prefect**

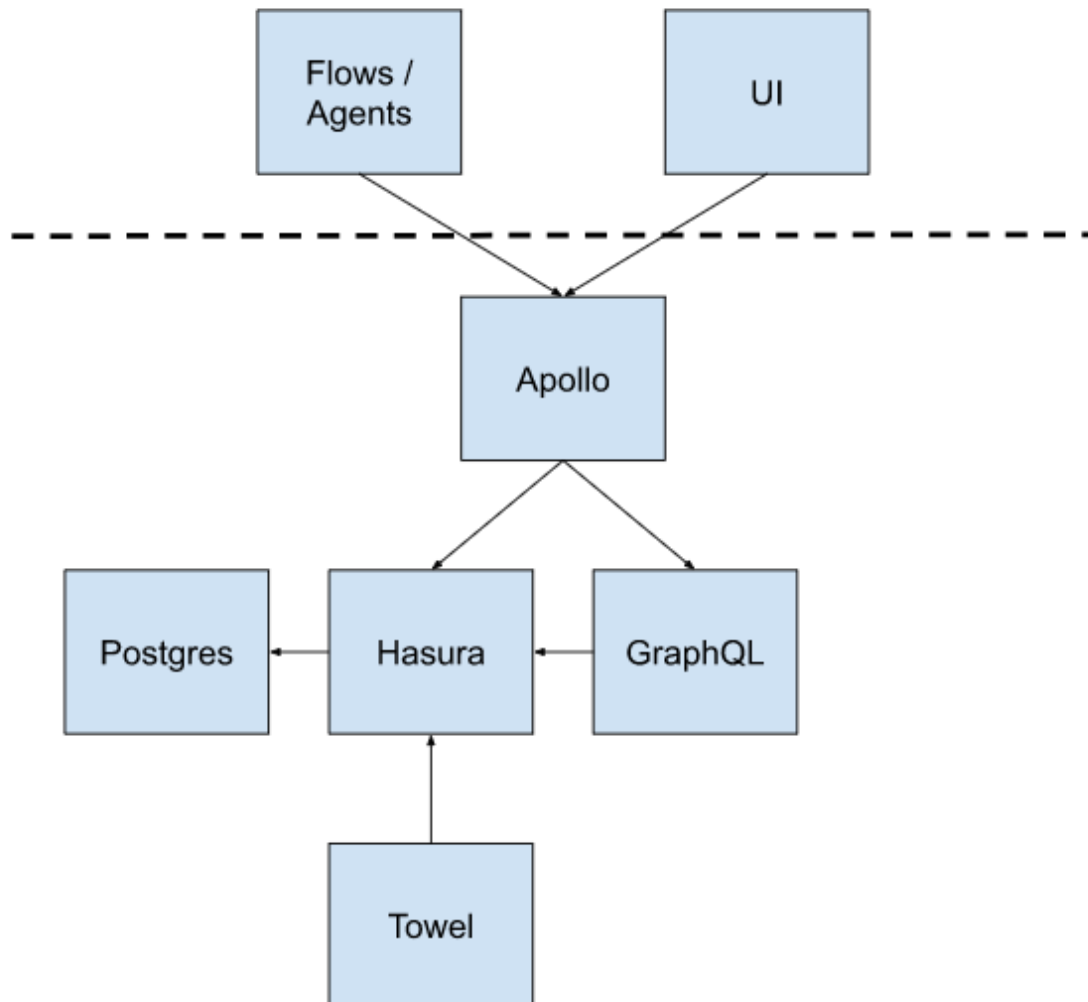


Fonte: Documentação oficial do Prefect:

<https://docs.prefect.io/orchestration/server/architecture.html>.

A arquitetura do Prefect está representada na Figura 14 abaixo.

**Figura 14 – Arquitetura do Prefect.**



**Fonte: Documentação oficial do Prefect:**

<https://docs.prefect.io/orchestration/server/architecture.html>.

Ela possui os seguintes componentes:

- UI: Interface do usuário;
- Apollo: Endpoint para interação com o servidor;
- PostgreSQL: camada de persistência dos dados;

- Hasura: GraphQL API para consultas aos metadados no PostgreSQL;
- GraphQL: As regras de negócio do servidor;
- Towel: Utilitários:
  - Scheduler: Agendamento de Execuções;
  - Zombie Killer: marca tarefas como falha;
  - Lazarus: reagenda execuções que mantém um estado não usual por um período.

Prefect é uma ferramenta de Orquestração de Pipelines de dados moderna, robusta, simples de usar e possui um ótimo diferencial, a saber, uma cloud própria que pode ser usada de backend para as execuções dos Flows.

Prefect tem sido amplamente utilizado por times de dados ao redor do mundo. Ela possui integração com Docker e Kubernetes para deploy em ambientes diferentes.

## Capítulo 6. Data Flow na Prática – Airflow

Neste capítulo, vamos exercitar a implementação de um Data Flow usando Airflow. A ferramenta funciona em sistemas Linux e pode ser instalada e configurada facilmente a partir dos passos na figura abaixo:

**Figura 15 – Passos para instalação e configuração do Airflow.**

```
# airflow needs a home, ~/airflow is the default,
# but you can lay foundation somewhere else if you prefer
# (optional)
export AIRFLOW_HOME=~/airflow

# install from pypi using pip
pip install apache-airflow

# initialize the database
airflow initdb

# start the web server, default port is 8080
airflow webserver -p 8080

# start the scheduler
airflow scheduler

# visit localhost:8080 in the browser and enable the example dag in the home page
```

**Fonte: Documentação oficial do Airflow:**

<https://airflow.apache.org/docs/stable/start.html>.

Airflow funciona apenas em sistemas Linux. Caso o sistema operacional disponível para execução da atividade seja Windows, há duas alternativas:

1. Utilizar uma máquina virtual com distribuição Linux instalada (preferencialmente Ubuntu pela sua ampla utilização e documentação);
2. Utilizar o WSL (Windows Subsystem for Linux) para executar softwares Linux nativamente em Windows.



Para a utilização de máquinas virtuais, recomendamos a utilização do Virtual Box. O download pode ser realizado a partir do link: <https://www.virtualbox.org>. Uma imagem do Ubuntu para instalação na máquina virtual pode ser acessada no link: <https://ubuntu.com/download/desktop>.

Para utilizar o WSL, um tutorial completo de instalação oficial da Microsoft está disponível no link: <https://docs.microsoft.com/pt-br/windows/wsl/install-win10>.

### **Datasets necessários para a realização das atividades:**

- Enade:  
[http://download.inep.gov.br/microdados/Enade\\_Microdados/microdados\\_enade\\_2019.zip](http://download.inep.gov.br/microdados/Enade_Microdados/microdados_enade_2019.zip).
- Titanic:  
[https://raw.githubusercontent.com/A3Data/hermione/master/hermione/file\\_text/train.csv](https://raw.githubusercontent.com/A3Data/hermione/master/hermione/file_text/train.csv).

## Capítulo 7. Data Flow na Prática – Prefect

---

Neste capítulo, vamos exercitar a implementação de um Data Flow usando Airflow. A ferramenta pode ser facilmente instalada utilizando o comando *pip install prefect*. Em seguida, para um melhor aproveitamento, é recomendado realizar um cadastro para utilizar o nível gratuito da nuvem disponibilizada pelo Prefect como backend de orquestração do Flow. O cadastro pode ser realizado em: <https://www.prefect.io>.

Para utilização do framework, é necessário cadastrar 2 tokens de acesso (um token pessoal e um token para execução de flows). Instruções detalhadas sobre o processo de autenticação para utilização da nuvem Prefect podem ser encontradas neste link: <https://docs.prefect.io/orchestration/tutorial/configure.html>.

## Capítulo 8. Data Flow na Prática – Prefect

---

Parabéns! Você completou a primeira disciplina do Bootcamp de Engenharia de Dados!

Atenção para o desenvolvimento do **Desafio**.

Até aqui, vimos:

- Big Data, dados, fontes de dados;
- ETL;
- Técnicas de extração – Crawlers;
- Transformação – Limpeza, organização e estruturação de dados;
- Orquestração de Pipelines de Dados;
- Drag and Drop;
- Com código;
- Ferramentas “on premisses” e na nuvem.

A partir deste ponto, outras ferramentas sobre as quais devemos orientar nossos estudos são:

- Ferramentas nativas de provedores de serviços em nuvem – (AWS StepFunctions, Google Data Flow, Azure Logic Apps);
- Aprofundar nas diversas possibilidades de armazenamento (SQL, NoSQL, Grafos).

Os próximos passos para dar continuidade em seus estudos de Engenharia de dados:

- Data Lake e Data Warehouse;

- Ferramentas para processamento distribuído (Hadoop, Spark, etc.);
- Tecnologias de Containers e Kubernetes (K8s);
- Desenhos de arquitetura de soluções.

Ao Engenheiro de Dados, é extremamente importante pensar todo o ciclo de vida dos dados, desde sua extração até seu consumo por equipes de analistas de negócios ou por equipes de ciências de dados que irão trabalhar no desenvolvimento de modelos preditivos e explicativos para orientação da tomada de decisão.

Além disso, é muito bom que o Engenheiro de Dados possua sólidos conhecimentos em ambiente de nuvem (visto o contexto de Big Data sobre o qual normalmente iremos atuar em projetos profissionais) e em arquitetura de soluções. Ele será o principal apoio do time de DevOps para implantação dos pipelines (ou até mesmo irá implantá-los “em primeira pessoa”).

## Referências

---

AIRFLOW. *Quick Start*. Apache Airflow. Disponível em: <<https://airflow.apache.org/docs/stable/start.html>>. Acesso em: 18 nov. 2020.

FURLAN, Axel. *How to deploy Apache Airflow with Celery on AWS*. Towards Data Science, 2019. Disponível em: <<https://towardsdatascience.com/how-to-deploy-apache-airflow-with-celery-on-aws-ce2518dbf631>>. Acesso em: 18 nov. 2020.

GARRO, Fernanda. *O que é ETL e qual sua importância entre os processos de BI*. IGTI Blog, 2017. Disponível em: <<https://www.igti.com.br/blog/o-que-e-etl-bi/>>. Acesso em: 18 nov. 2020.

KUBEFLOW. *Overview of Kubeflow Pipelines*. Kubeflow Docs. Disponível em: <<https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/>>. Acesso em: 18 nov. 2020.

LEWIS, Lori. *Infographic: What Happens In An Internet Minute 2020*. All Access, 2020. Disponível em: <<https://www.allaccess.com/merge/archive/31294/infographic-what-happens-in-an-internet-minute>>. Acesso em: 18 nov. 2020.

LIU, Haimo. *Hortonworks DataFlow (HDF) 3.1 blog series part 5: Introducing Apache NiFi-Atlas integration*. Cloudera Blog, 2018. Disponível em: <<https://blog.cloudera.com/hdf-3-1-blog-series-part-6-introducing-nifi-atlas-integration/>>. Acesso em: 18 nov. 2020.

MACHADO, Adriano. *Pentaho Data Integration*. Medium soma-labs, 2018. Disponível em: <<https://medium.com/soma-labs/pentaho-data-integration-a7b754fae960>>. Acesso em: 18 nov. 2020.

MICROSOFT. *Guia de instalação do Subsistema Windows para Linux para Windows 10*. Disponível em: <<https://docs.microsoft.com/pt-br/windows/wsl/install-win10>>. Acesso em: 18 nov. 2020.

PREFECT. *Configure Your Environment*. Prefect Docs. Disponível em: <<https://docs.prefect.io/orchestration/tutorial/configure.html>>. Acesso em: 18 nov. 2020.

PREFECT. Disponível em: <<https://www.prefect.io>>. Acesso em: 18 nov. 2020.

PYDI, Aakash. *Building a Production-Level ETL Pipeline Platform Using Apache Airflow*. Towards Data Science, 2019. Disponível em: <<https://towardsdatascience.com/building-a-production-level-etl-pipeline-platform-using-apache-airflow-a4cf34203fbd>>. Acesso em: 18 nov. 2020.

TWITTER. *Developers*. Disponível em: <<https://developer.twitter.com/en>>. Acesso em: 18 nov. 2020.

UBUNTU. *Download Ubuntu Desktop*. Disponível em: <<https://ubuntu.com/download/desktop>>. Acesso em: 18 nov. 2020.

VIRTUAL BOX. Disponível em: <<https://www.virtualbox.org>>. Acesso em: 18 nov. 2020.