



INSTITUTO DE GESTÃO E TECNOLOGIA  
DA INFORMAÇÃO

---

## **Pipelines de Dados**

---

Neylson Crepalde

2022

## **Pipeline de Dados**

### **Bootcamp: Engenheiro(a) de Dados**

Neylson Crepalde

© Copyright do Instituto de Gestão e Tecnologia da Informação.

Todos os direitos reservados.

## Sumário

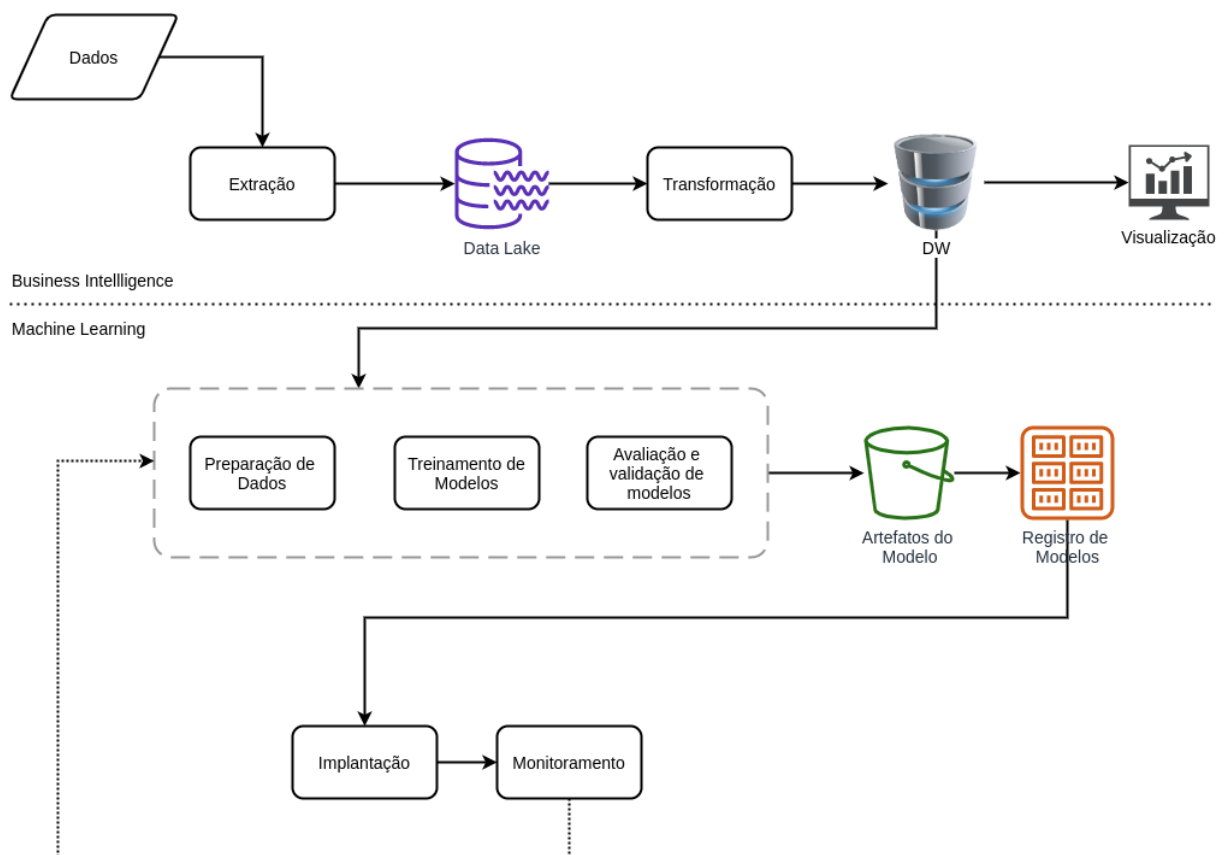
---

Capítulo 1.	Visão geral do pipeline de ciência de dados.....	4
Capítulo 2.	Extração de Dados .....	6
Capítulo 3.	Transformação de Dados .....	7
Capítulo 4.	Soluções de ETL.....	8
Capítulo 5.	Data Flow na Prática – Airflow.....	25
Capítulo 6.	Data Flow na Prática – Prefect.....	27
Capítulo 7.	Data Flow em <i>Near Real Time</i> – Kafka.....	28
Capítulo 8.	Encerramento e próximos passos.....	31
Referências.....		33

## Capítulo 1. Visão geral do pipeline de ciência de dados

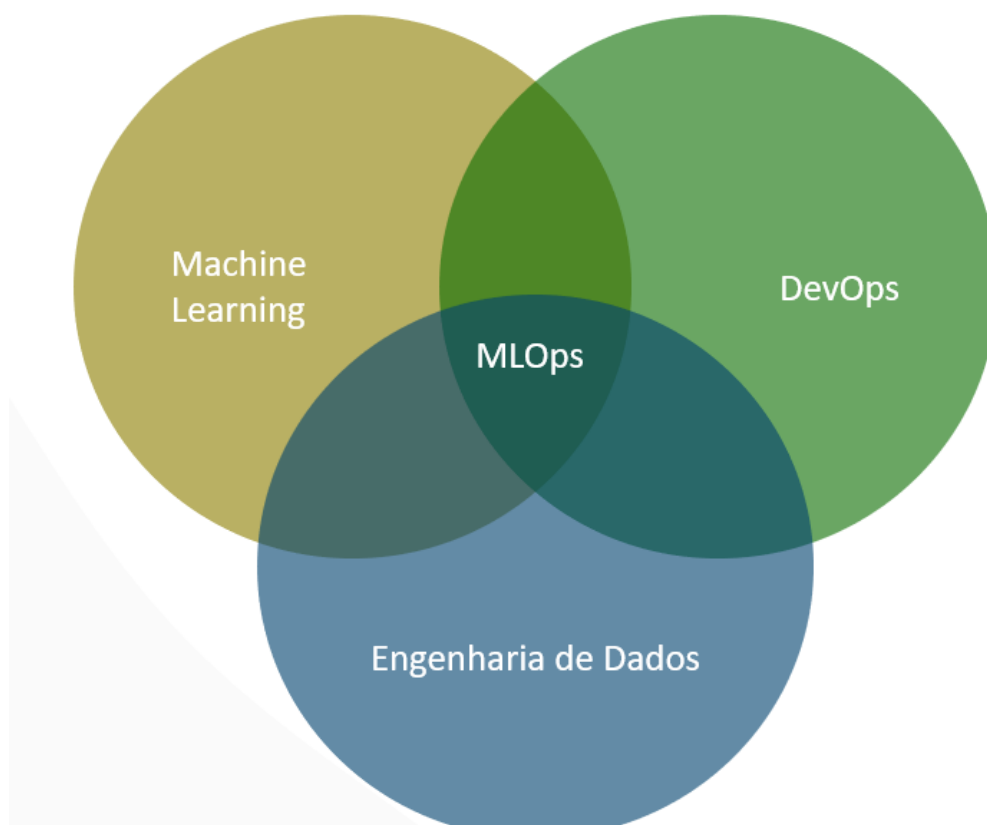
O pipeline de Ciência de Dados pode ser representado pela figura abaixo:

**Figura 1 – O Pipeline de Ciência de dados.**



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

Figura 2 – DataOps / MLOps.



## Capítulo 2. 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, precisaremos criar uma conta na plataforma de desenvolvedor do Twitter que pode ser acessada em: <https://developer.twitter.com/en>. É necessário ter uma conta no Twitter para esse 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 3. Transformação de Dados

---

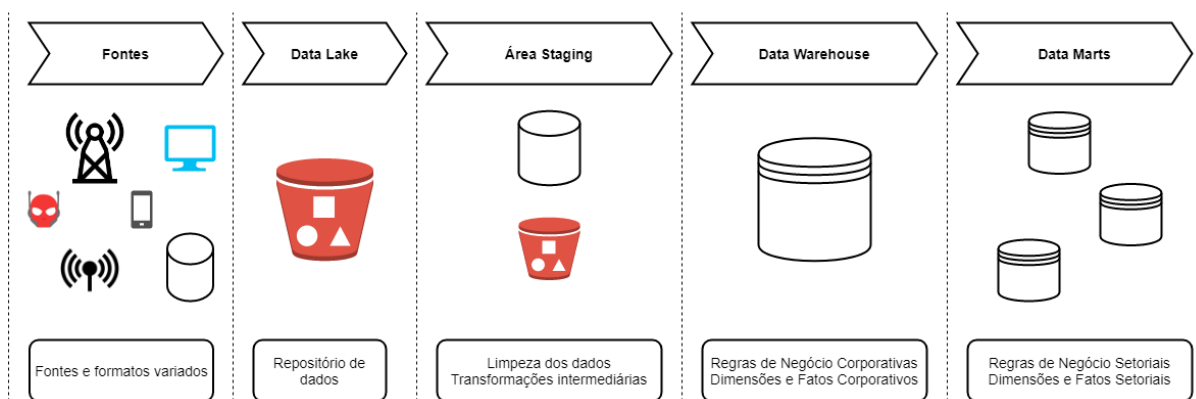
A etapa de transformação de dados pode compreender, principalmente, atividades de limpeza, organização, estruturação e enriquecimento dos dados. Esta costuma ser a etapa nos projetos de *analytics* que exige a maior parte do tempo de dedicação.

## Capítulo 4. 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 3 – Pipeline de ETL.**



Fonte: Adaptado de IGTI Blog, 2017. <sup>1</sup>

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;

<sup>1</sup> Disponível em: <https://www.igti.com.br/blog/o-que-e-etl-bi/>.



- *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, 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:

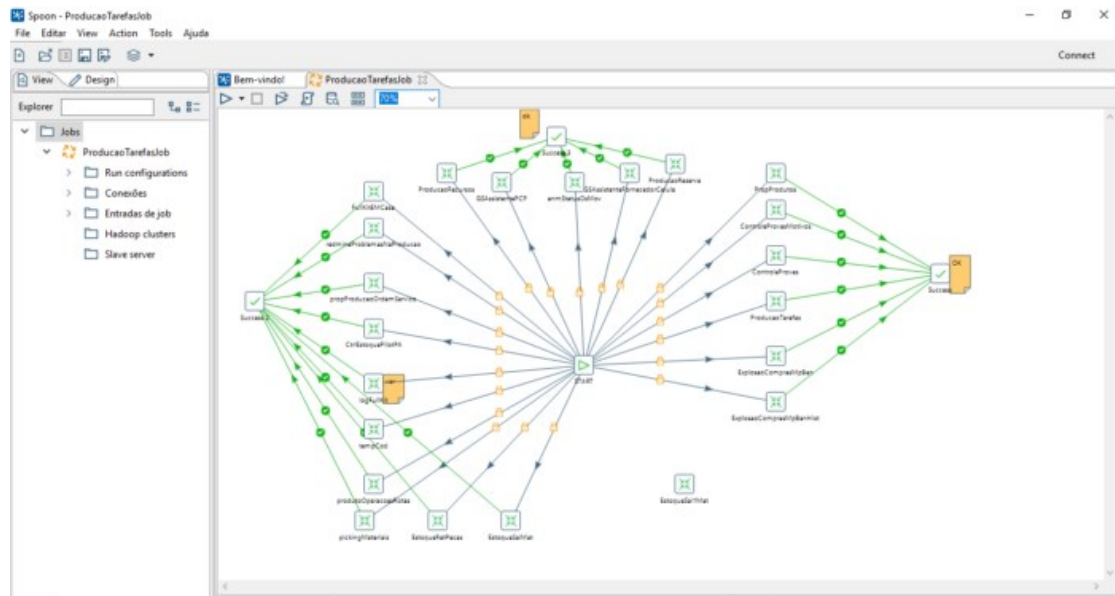
- Pentaho;
- Apache Nifi;

- 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 4 – Exemplo da Interface do Pentaho.



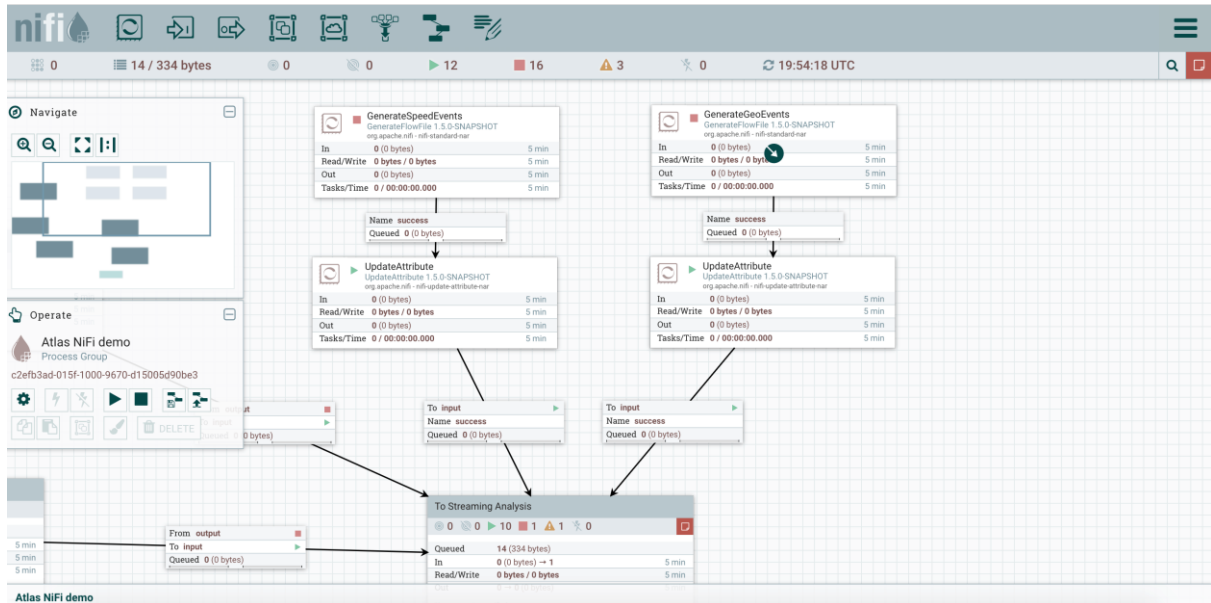
Fonte: Soma-labs, 2018<sup>2</sup>.

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.

<sup>2</sup> Disponível em: <https://medium.com/soma-labs/pentaho-data-integration-a7b754fae960>.

**Figura 5 – Exemplo da Interface do Apache Nifi.**

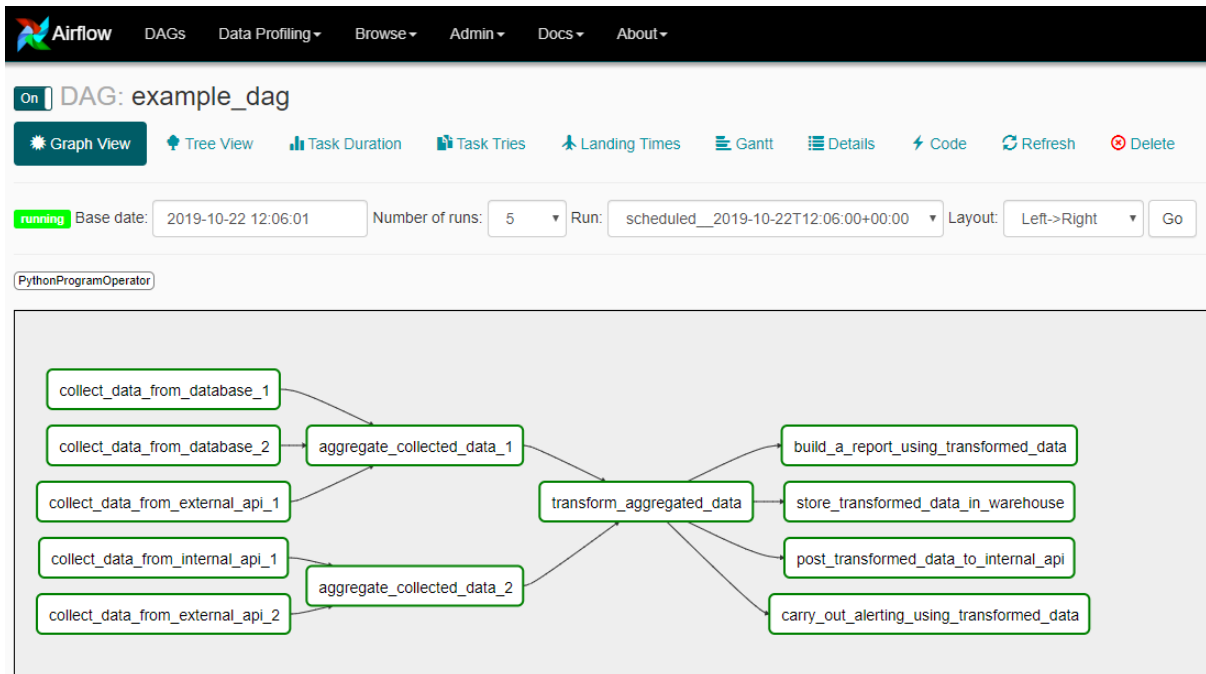


Fonte: Cloudera blog, 2018.<sup>3</sup>

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”.

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

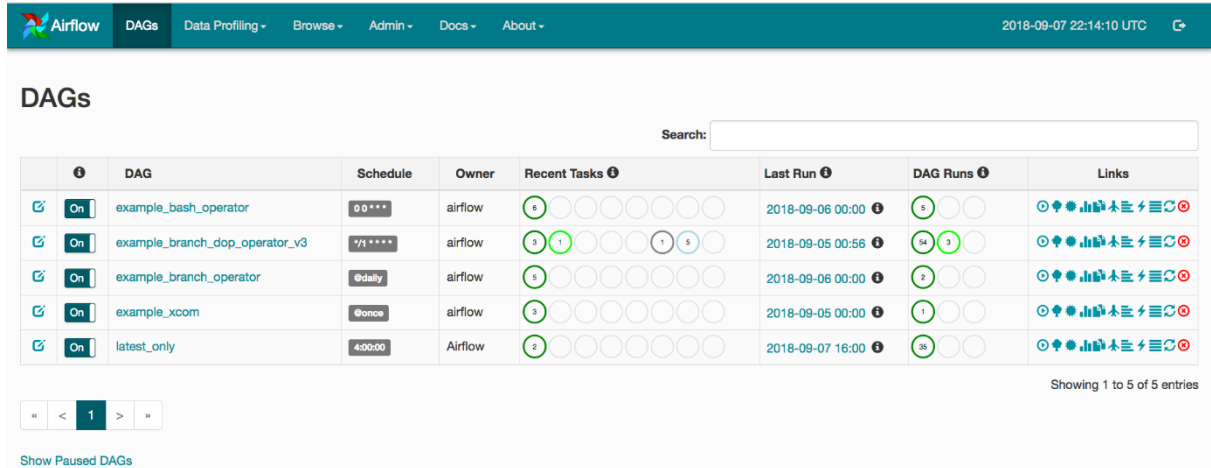
**Figura 6 – Exemplo da Interface de Controle de Flows do Apache Airflow.**



**Fonte: Towards Data Science, 2019.<sup>4</sup>**

<sup>4</sup> Disponível em: <https://towardsdatascience.com/building-a-production-level-etl-pipeline-platform-using-apache-airflow-a4cf34203fbd>.

Figura 7 – Exemplo da Interface do Apache Airflow.



	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
On	example_bash_operator	00***	airflow	6	2018-09-06 00:00	1	<a href="#">DAG View</a> <a href="#">Graph View</a> <a href="#">Code</a> <a href="#">Logs</a> <a href="#">Refresh</a> <a href="#">Pause</a> <a href="#">Unpause</a> <a href="#">Delete</a>
On	example_branch_dop_operator_v3	*1***	airflow	3	2018-09-05 00:56	4	<a href="#">DAG View</a> <a href="#">Graph View</a> <a href="#">Code</a> <a href="#">Logs</a> <a href="#">Refresh</a> <a href="#">Pause</a> <a href="#">Unpause</a> <a href="#">Delete</a>
On	example_branch_operator	@daily	airflow	5	2018-09-06 00:00	1	<a href="#">DAG View</a> <a href="#">Graph View</a> <a href="#">Code</a> <a href="#">Logs</a> <a href="#">Refresh</a> <a href="#">Pause</a> <a href="#">Unpause</a> <a href="#">Delete</a>
On	example_xcom	@once	airflow	3	2018-09-05 00:00	1	<a href="#">DAG View</a> <a href="#">Graph View</a> <a href="#">Code</a> <a href="#">Logs</a> <a href="#">Refresh</a> <a href="#">Pause</a> <a href="#">Unpause</a> <a href="#">Delete</a>
On	latest_only	4:00:00	Airflow	2	2018-09-07 16:00	39	<a href="#">DAG View</a> <a href="#">Graph View</a> <a href="#">Code</a> <a href="#">Logs</a> <a href="#">Refresh</a> <a href="#">Pause</a> <a href="#">Unpause</a> <a href="#">Delete</a>

Showing 1 to 5 of 5 entries

Show Paused DAGs

Fonte: Airflow Apache 1.10.3<sup>5</sup>.

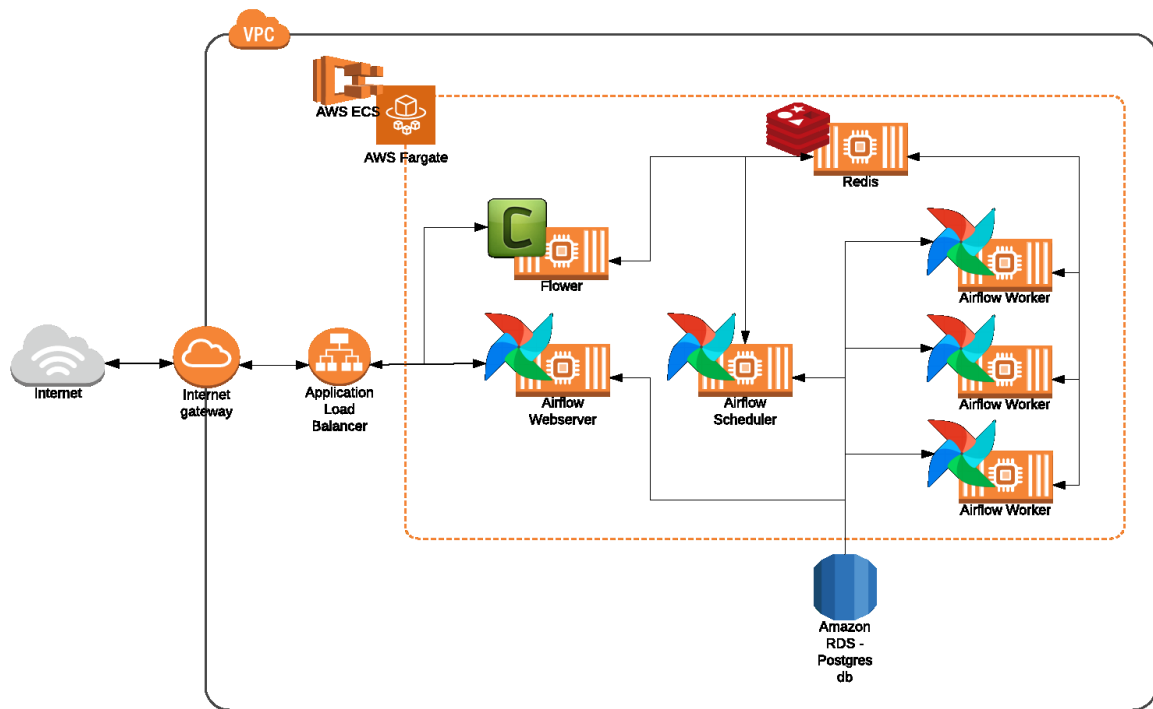
Apache Airflow tem como princípios ser:

- **Escalável** – possui arquitetura modular que utiliza mensageria para orquestra 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.

<sup>5</sup> Disponível em: <https://airflow.apache.org/docs/1.10.3/ui.html>.

Um exemplo de arquitetura do Airflow está representado na Figura 8 abaixo.

**Figura 8 – Arquitetura do Airflow usando Celery.**



**Fonte: Towards Data Science, 2020. <sup>6</sup>**

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

<sup>6</sup> Disponível em: <https://towardsdatascience.com/how-to-deploy-apache-airflow-with-celery-on-aws-ce2518dbf631>.

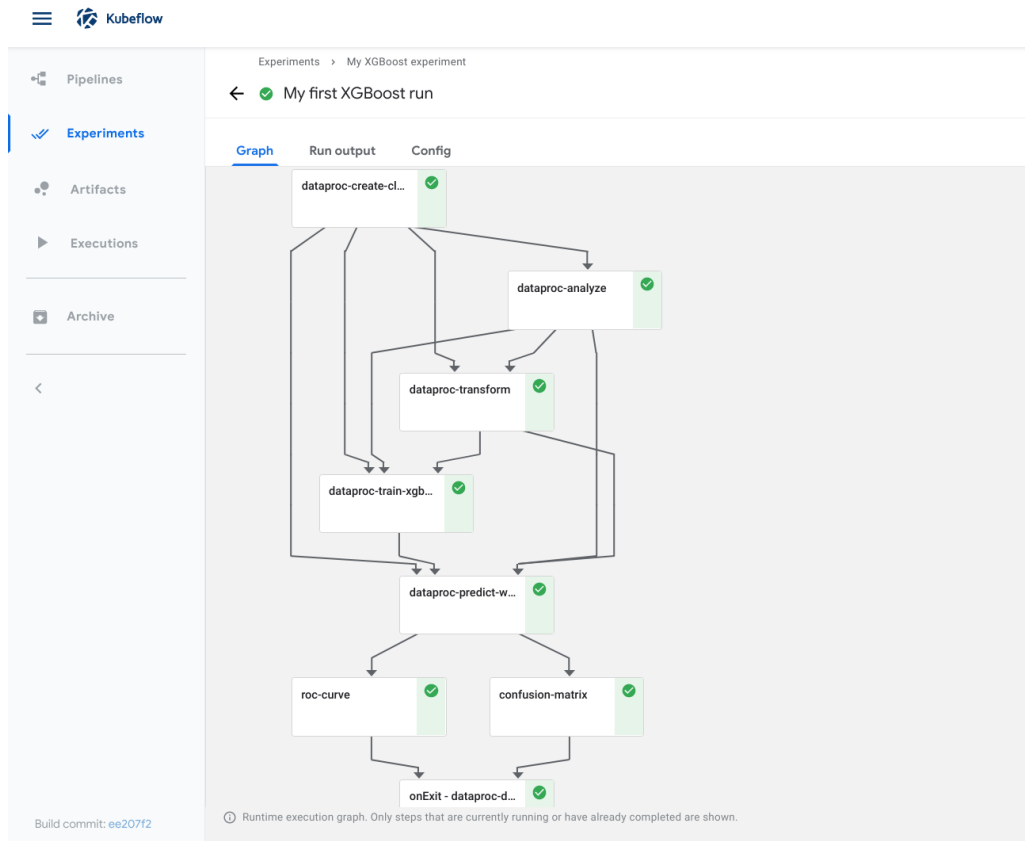
orquestrar de fato as tarefas do Flow. Neste caso, ele usa Celery que é um serviço assíncrono de enfileiramento de tarefas. Celery utiliza o banco de dados Redis para comunicação com os Workers (containers de execução) e possui sua própria interface de usuário para monitoramento do cluster chamado Flower. Airflow usa ainda um banco de dados relacional (neste caso, PostgreSQL) para registro de metadados.

AirFlow é uma das ferramentas mais utilizadas 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 9 – Exemplo da Interface do KubeFlow Pipelines.**



**Fonte: Kubeflow, 2021.<sup>7</sup>**

O KubeFlow possui vários componentes, entre eles:

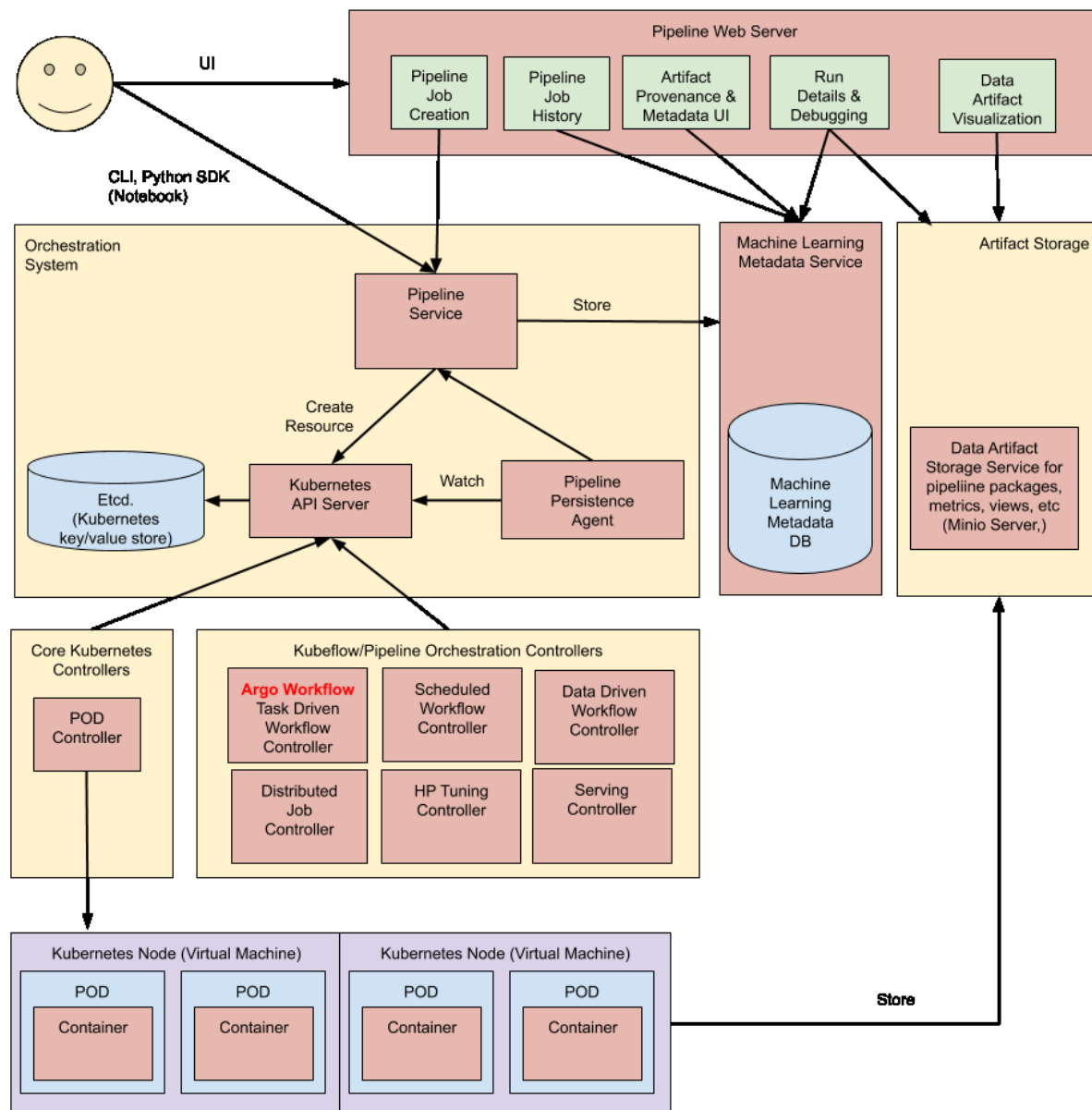
- Dashboard;
- Metadados (logs e monitoramento de execuções);
- Um servidor de Jupyter Notebooks;
- Fairing (biblioteca para automatização de treino e deploy de Machine Learning);

<sup>7</sup> Disponível em: <https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/>.

- 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 10 abaixo.

Figura 10 – Arquitetura do KubeFlow.



Fonte: KubeFlow. 2021.<sup>8</sup>

<sup>8</sup> Disponível em: <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.

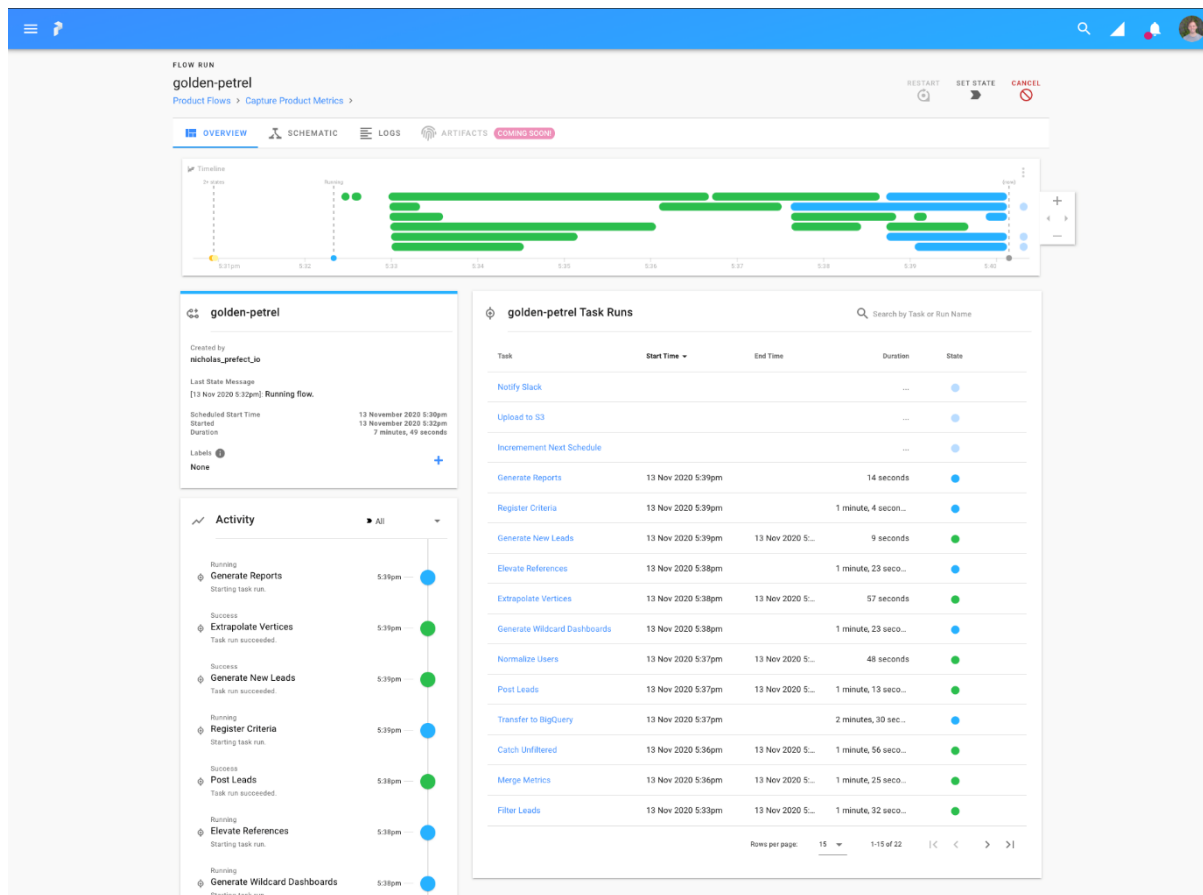
A segunda camada apresenta o sistema de orquestração (no nível do K8s), a gestão de metadados de Machine Learning e o Storage de artefatos. A terceira camada apresenta um sistema de orquestração no nível de execução dos pipelines. Aqui estão o controller dos POD's e o controller de execução de tarefas 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.

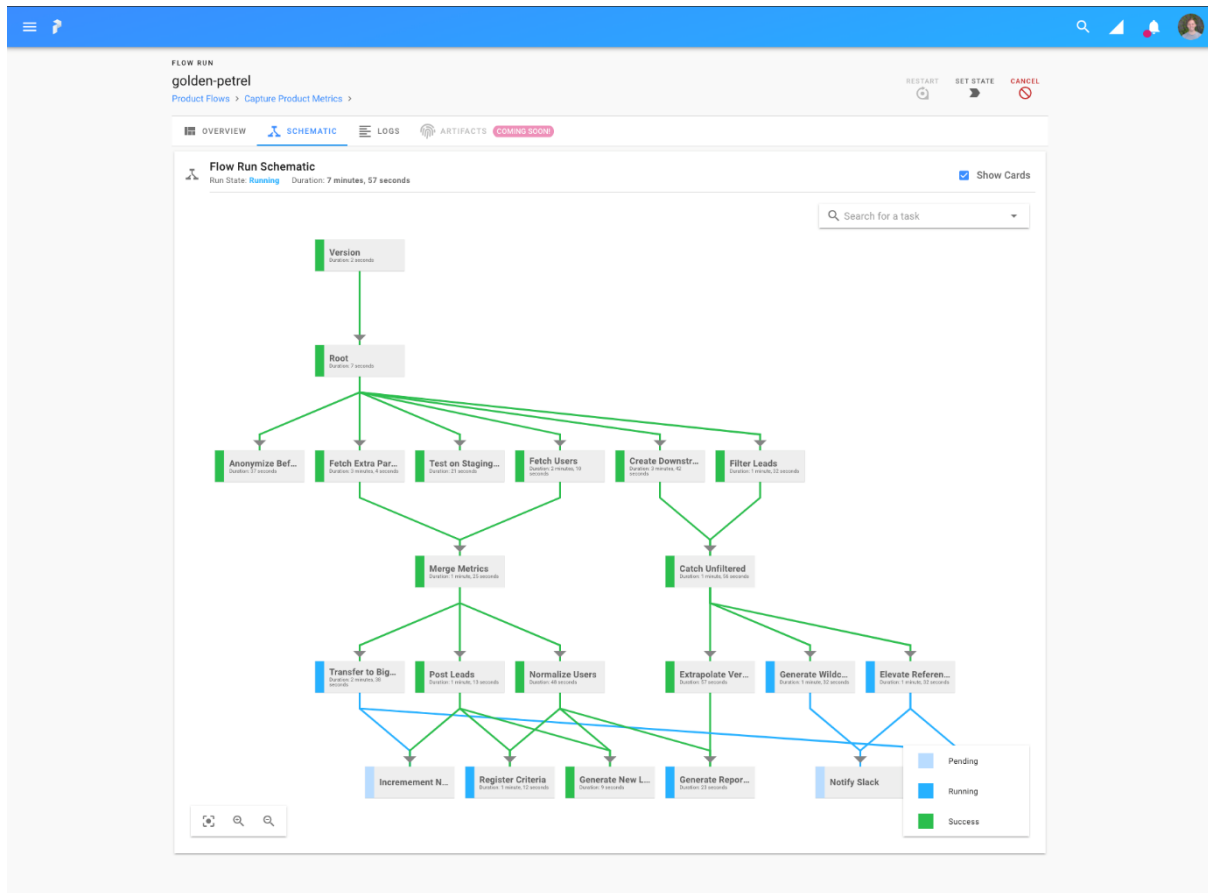
Figura 11 – Exemplo da Interface do Prefect.



Fonte: Prefect Docs, 2021.<sup>9</sup>

<sup>9</sup> Disponível em: <https://docs.prefect.io/orchestration/server/architecture.html>.

Figura 12 – Exemplo da Interface de Orquestração de Pipelines do Prefect.

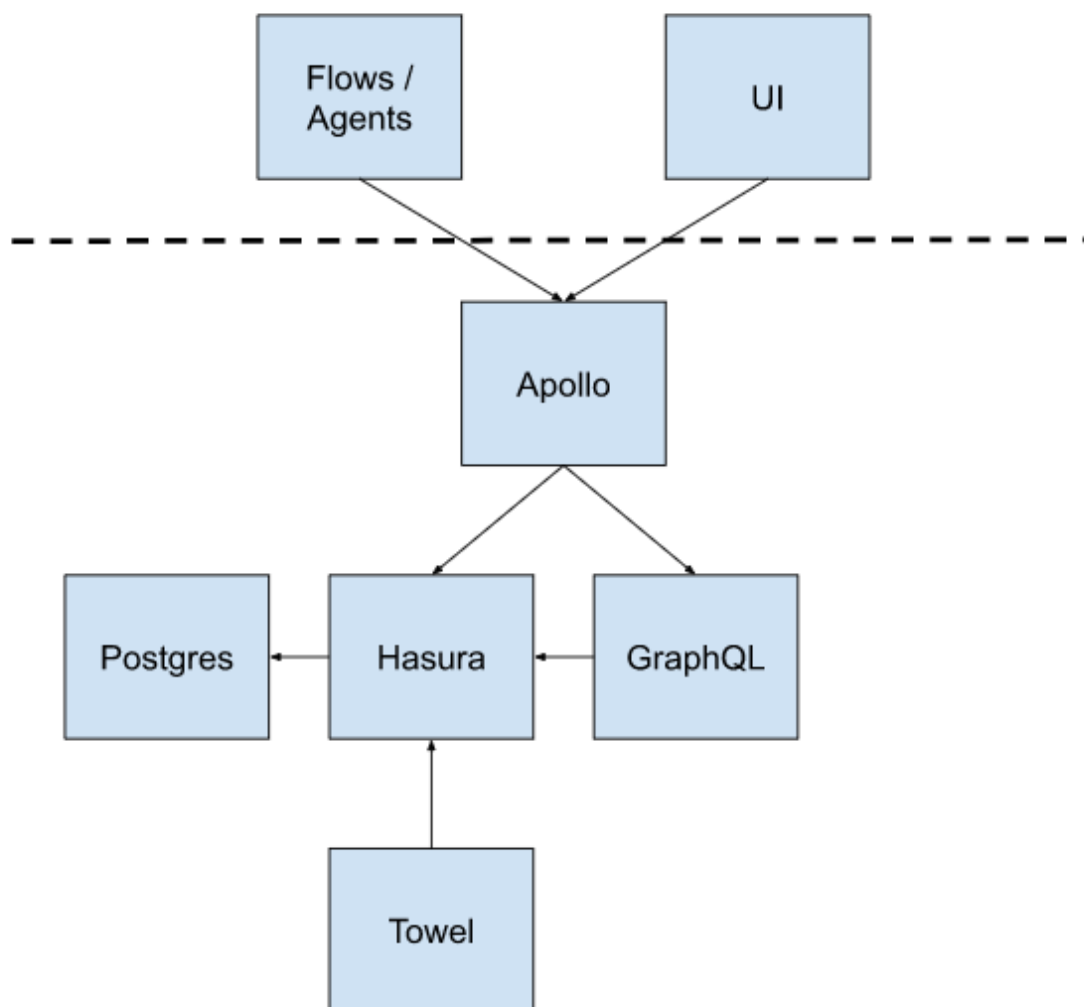


Fonte: Prefect Docs, 2021.<sup>10</sup>

A arquitetura do Prefect está representada na Figura 13 abaixo.

<sup>10</sup> Disponível em: <https://docs.prefect.io/orchestration/server/architecture.html>.

**Figura 13 – Arquitetura do Prefect.**



**Fonte: Prefect Docs, 2021.** <sup>11</sup>

Ela possui os seguintes componentes:

<sup>11</sup> Disponível em: <https://docs.prefect.io/orchestration/server/architecture.html>.

- 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. Ele possui integração com Docker e Kubernetes para deploy em ambientes diferentes.



## Capítulo 5. 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 14 – 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 Apache Airflow 2.1.0<sup>12</sup>**

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

---

<sup>12</sup> Disponível em: <https://airflow.apache.org/docs/apache-airflow/stable/start/index.html>.

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 6. 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 7. Data Flow em *Near Real Time* – Kafka

---

Eventualmente, há alguns casos em o negócio tem a necessidade de entrega e atualização de dados em um tempo muitíssimo curto, em alguns minutos ou até mesmo segundos. Nesses casos, os exemplos estudados nos capítulos anteriores não nos atendem.

Ao invés disso precisamos de outro tipo de ferramentas que nos permitam implementar um pipeline de dados *Near Real Time* (quase em tempo real). A ferramenta mais conhecida no mercado para realizar esta tarefa é o Apache Kafka.

O **Apache Kafka** é uma plataforma de *streaming* de dados do tipo *Pub/Sub*. De acordo com a documentação oficial do Kafka,

streaming de eventos é a prática de capturar dados em tempo real de fontes de eventos como bancos de dados, sensores, dispositivos móveis, serviços em nuvem e aplicativos de software na forma de fluxos de eventos; armazenar esses fluxos de eventos de forma duradoura para recuperação posterior; manipular, processar e reagir aos fluxos de eventos em tempo real e também retrospectivamente; e encaminhar os fluxos de eventos para diferentes tecnologias de destino, conforme necessário. O streaming de eventos, portanto, garante um fluxo contínuo e interpretação dos dados para que as informações certas estejam no lugar certo, na hora certa. (Documentação oficial do Kafka<sup>13</sup>, tradução do autor)

Para processar dados em fluxo contínuo, o Kafka implementa a seguinte solução:

---

<sup>13</sup> Disponível em: <https://kafka.apache.org/intro>.

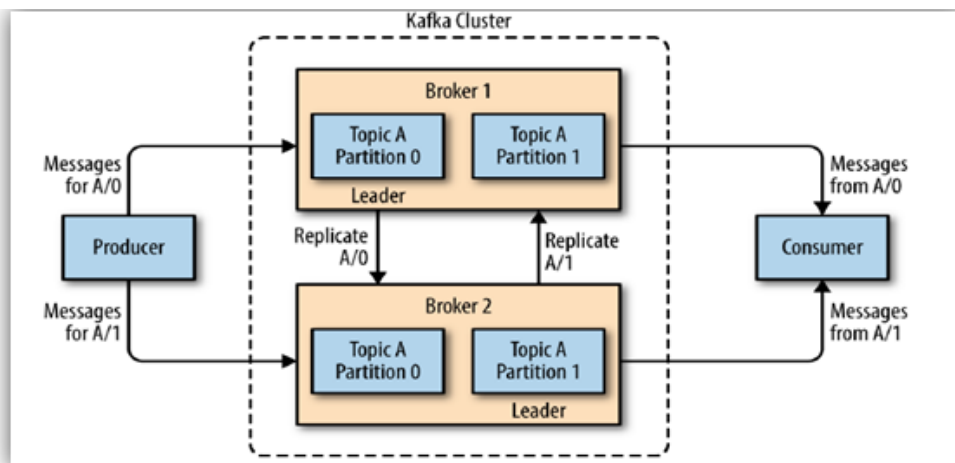
1. **Publicar** (escrever) e **inscrever-se** (ler) em streams de eventos,
2. **Armazenar** os eventos por um período determinado,
3. **Processar** os streams de eventos no momento em que eles acontecem ou após.

Um **evento** pode ser considerado uma unidade mínima de informação. Ele também é comumente chamado de mensagem e possui três partes, uma **chave**, um **valor** e um **registro de tempo**. Abaixo, um exemplo:

- Chave (key): “Neylson”
- Valor (value): “Fez um pagamento de R\$50,00 para Fernanda”
- Registro de tempo (timestamp): “04 de julho de 2021 às 15:32”

Os eventos são registrados dentro do Kafka em **tópicos**. Os tópicos são a estrutura onde os **publishers** escrevem as mensagens e de onde os **consumers** realizam a leitura. Abaixo, um desenho da arquitetura do Kafka:

**Figura 15 – Arquitetura do Kafka.**



Fonte: Eduardo Hattori, 2020. <sup>14</sup>

<sup>14</sup> Disponível em: <https://eduardohattoriif.medium.com/nutshell-apache-kafka-4f6e7ef3cff2>.

Para a realização da atividade proposta nas aulas, vamos precisar de:

- Docker instalado em sua máquina;
- docker-compose também instalado.

O passo a passo para a realização da atividade pode ser acessado neste link:

<https://github.com/neylsoncrepalde/kafka-exercise>. Divirta-se!

## Capítulo 8. Encerramento e próximos passos

---

Parabéns! Você completou a quarta 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

---

APACHE SOFTWARE FOUNDATION. **Apache Airflow**. Version 2.1.0. Maryland, 2021. Disponível em: <https://airflow.apache.org/docs/stable/start.html>. Acesso em: 07 mar. 2021.

CANONICAL LTD. **Download page of Ubuntu Desktop**. Version 20.04.2.0 LTS. Austin, 2021. Disponível em: <https://ubuntu.com/download/desktop>. Acesso em: 07 mar. 2021.

CREPALDE, Neylson; MEDEIROS, Mariana. Kafka Exercise. **GitHub**, abril. 2021. Disponível em: <https://github.com/neylsoncrepalde/kafka-exercise>. Acesso em: 07 mar. 2021.

FURLAN, Axel. How to deploy Apache Airflow with Celery on AWS – The complete guide. **Towards Data Science**, [Toronto], 19 de fevereiro de 2019. Disponível em: <https://towardsdatascience.com/how-to-deploy-apache-airflow-with-celery-on-aws-ce2518dbf631>. Acesso em: 07 mar. 2021.

GARRO, Fernanda. O que é ETL e qual sua importância entre os processos de BI. **IGTI Blog**, Belo Horizonte, 9 de agosto de 2017. Disponível em: <https://www.igti.com.br/blog/o-que-e-etl-bi/>. Acesso em: 22 jan. 2021.

HATTORI, Eduardo. Nutshell Apache Kafka. **Medium**, 20 dez. 2020. Disponível em: <https://eduardohattori.medium.com/nutshell-apache-kafka-4f6e7ef3cff2>. Acesso em: 07 mar. 2021.

KUBEFLOW. **Overview of Kubeflow Pipelines** - Kubeflow Docs. [S.l.], 2021. Disponível em: <https://www.kubeflow.org/docs/pipelines/overview/pipelines-overview/>. Acesso em: 07 mar. 2021.

LEWIS, Lori. Infographic: What Happens In An Internet Minute 2020. **All Access**, Malibu, 10 de março de 2020. Disponível em: <https://www.allaccess.com/merge/archive/31294/infographic-what-happens-in-an-internet-minute>. Acesso em: 07 mar. 2021.

LIU, Haimo. Hortonworks DataFlow (HDF) 3.1 blog series part 5: Introducing Apache NiFi-Atlas integration. **Cloudera Blog**, Palo Alto, 20 de fevereiro de 2018. Disponível em: <https://blog.cloudera.com/hdf-3-1-blog-series-part-6-introducing-nifi-atlas-integration/>. Acesso em: 07 mar. 2021.

MACHADO, Adriano. Pentaho Data Integration. **Medium soma-labs**, [S./], 04 de outubro de 2018. Disponível em: <https://medium.com/soma-labs/pentaho-data-integration-a7b754fae960>. Acesso em: 07 mar. 2021.

MICROSOFT. **Guia de instalação do Subsistema Windows para Linux para Windows 10**. [Washington], 2021. Disponível em: <https://docs.microsoft.com/pt-br/windows/wsl/install-win10>. Acesso em: 07 mar. 2021.

ORACLE. **Download page of VirtualBox**. Version 6.1, Austin, 2021. Disponível em: <https://www.virtualbox.org>. Acesso em: 07 mar. 2021.

PREFECT. **Prefect Plataform Page**. [S./], 2020. Disponível em: <https://www.prefect.io>. Acesso em: 07 mar. 2021.

PYDI, Aakash. Building a Production-Level ETL Pipeline Platform Using Apache Airflow. **Towards Data Science**, [Toronto], 26 de outubro de 2019. Disponível em: <https://towardsdatascience.com/building-a-production-level-etl-pipeline-platform-using-apache-airflow-a4cf34203fbd>. Acesso em: 07 mar. 2021.

TWITTER. **Developer Plataform**. San Francisco, 2021. Disponível em: <https://developer.twitter.com/en>. Acesso em: 07 mar. 2021.