



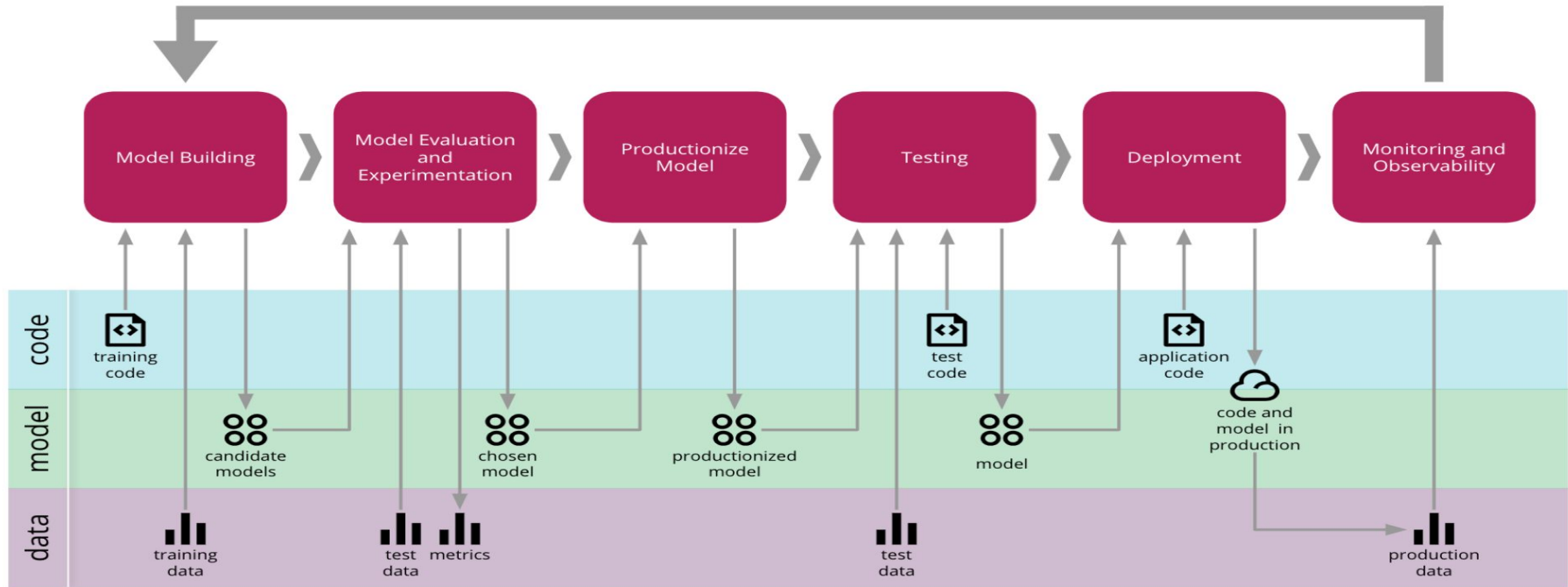
Prof. Jean Carlos Alves

DataOps e Implantação de Sistemas de Machine Learning



PUC Minas

Fluxo CD para Machine Learning



Treinamento/Deploy(Kubeflow)

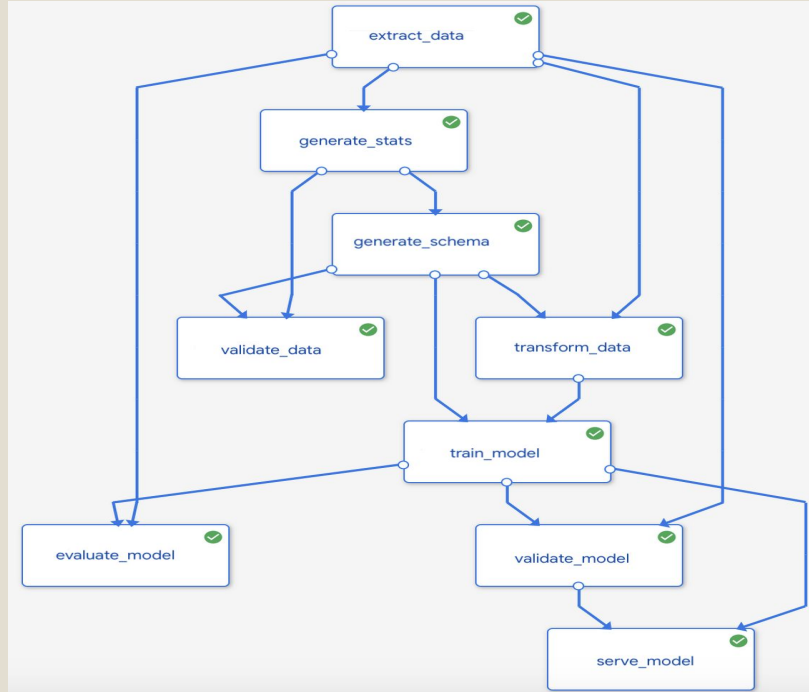
- O projeto Kubeflow é dedicado a tornar as implantações de fluxos de trabalho de aprendizado de máquina (ML) no Kubernetes simples, portáteis e escalonáveis. Nosso objetivo não é recriar outros serviços, mas fornecer uma maneira direta de implantar os melhores sistemas de código aberto para ML em diversas infraestruturas. Em qualquer lugar que você esteja executando o Kubernetes, deve ser capaz de executar o Kubeflow.

Treinamento/Deploy(Kubeflow)

- Framework código aberto
- Automatizar fluxos de ML
- Suporta vários frameworks ML
- Utiliza contêiner
- Portabilidade

<https://www.kubeflow.org/docs/started/getting-started/>

Treinamento/Deploy(Kubeflow)



<https://cloud.google.com/solutions/machine-learning/architecture-for-mlops-using-tfx-kubeflow-pipelines-and-cloud-build>

Kubeflow - Pipeline

The screenshot shows the Kubeflow Pipelines web interface. On the left is a sidebar with navigation links: Pipelines (active), Experiments, Runs, Artifacts, Executions, Documentation, and Github Repo. The main area is titled 'Pipelines' and contains a table of pipeline definitions. At the top right of the main area are buttons for '+ Upload pipeline', 'Refresh', and 'Delete'. Below the title is a search bar labeled 'Filter pipelines'. The table has columns for 'Pipeline name', 'Description', and 'Uploaded on'. The last row, '[Demo] XGBoost - Iterative model training', is highlighted with a red rectangular box. The 'source code' link for this pipeline is also highlighted.

<input type="checkbox"/>	Pipeline name	Description	Uploaded on
<input type="checkbox"/>	[Tutorial] DSL - Control structures	source code Shows how to use conditional execution and exit handlers. This pipeline will randomly fail to demonstrate that t...	20/02/2021, 16:44:33
<input type="checkbox"/>	[Tutorial] Data passing in python compo...	source code Shows how to pass data between python components.	20/02/2021, 16:44:32
<input type="checkbox"/>	[Demo] TFX - Iris classification pipeline	source code . Example pipeline that classifies Iris flower subspecies and how to use native Keras within TFX.	20/02/2021, 16:44:31
<input type="checkbox"/>	[Demo] TFX - Taxi tip prediction model t...	source code GCP Permission requirements . Example pipeline that does classification with model analysis based on a public ...	20/02/2021, 16:44:30
<input type="checkbox"/>	[Demo] XGBoost - Iterative model training	source code This sample demonstrates iterative training using a train-eval-check recursive loop. The main pipeline trains the ...	20/02/2021, 16:44:29

Rows per page: 10

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

Kubeflow - Pipeline

The screenshot shows the 'Start a run' page in the Kubeflow Pipeline UI. On the left is a sidebar with navigation links: Pipelines, Experiments, Runs (highlighted with a blue bar and icon), Artifacts, Executions, Documentation, and Github Repo. The main content area is titled 'Experiments > XGBoost experiment' and 'Start a run'. It contains several form fields: 'Pipeline*' with a dropdown showing '[Demo] XGBoost - Iterative model training' and a 'Choose' link; 'Pipeline version*' with a dropdown showing '[Demo] XGBoost - Iterative model training' and a 'Choose' link; 'Run name*' with a text input containing 'XGBoost run'; 'Description (optional)' with a text input; 'Experiment*' with a dropdown showing 'XGBoost experiment' and a 'Choose' link; and 'Service Account (Optional)' with a text input. Below these is the 'Run Type' section with radio buttons for 'One-off' (selected) and 'Recurring'. The 'Run parameters' section states 'This pipeline has no parameters' and includes 'Start' and 'Skip this step' buttons.

Experiments > XGBoost experiment

Start a run

Run details

Pipeline*
[Demo] XGBoost - Iterative model training [Choose](#)

Pipeline version*
[Demo] XGBoost - Iterative model training [Choose](#)

Run name*
XGBoost run

Description (optional)

This run will be associated with the following experiment

Experiment*
XGBoost experiment [Choose](#)

This run will use the following Kubernetes service account. [?](#)

Service Account (Optional)

Run Type

☒ One-off ☐ Recurring

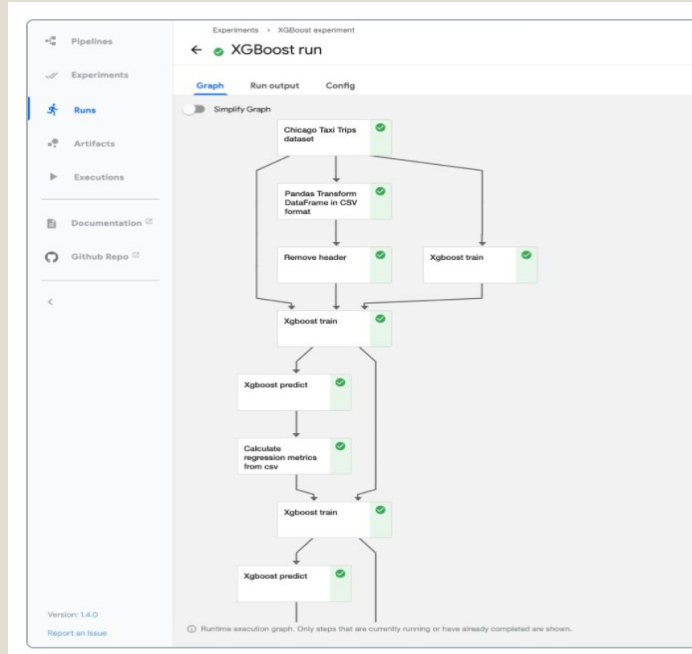
Run parameters

This pipeline has no parameters

[Start](#) [Skip this step](#)

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

Kubeflow - Pipeline



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

Airflow



Escalável

O Airflow tem uma arquitetura modular e usa uma fila de mensagens para orquestrar um número arbitrário de trabalhadores. O Airflow está pronto para escalar até o infinito.



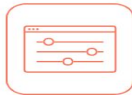
Dinâmico

Os pipelines do Airflow são definidos em Python, permitindo a geração dinâmica do pipeline. Isso permite escrever código que instancia pipelines dinamicamente.



Extensível

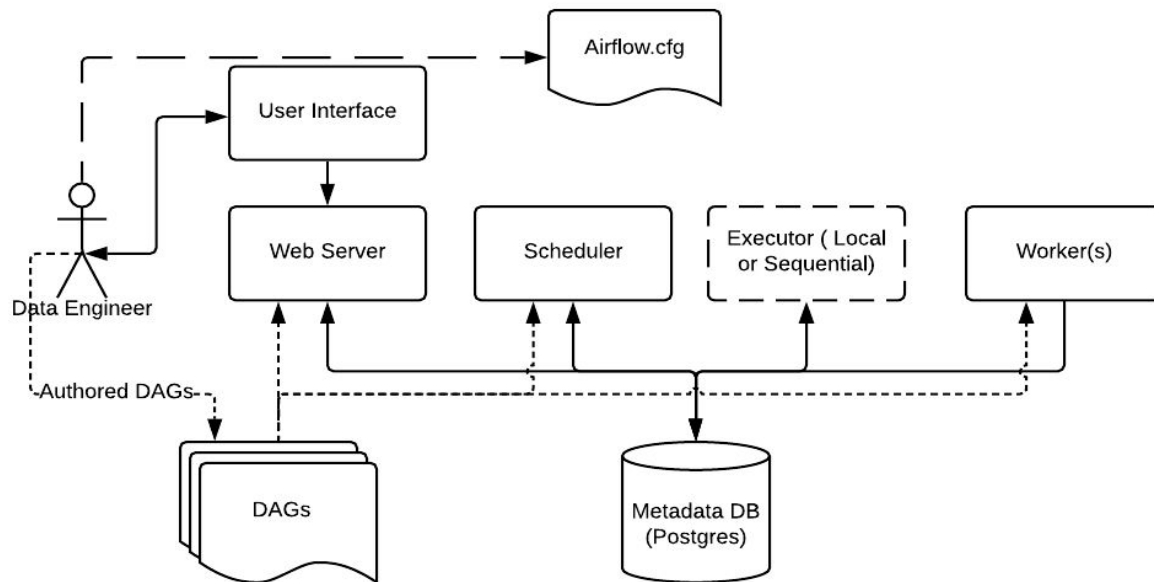
Defina facilmente seus próprios operadores e estenda as bibliotecas para se adequar ao nível de abstração adequado ao seu ambiente.



Elegante

Os pipelines do Airflow são enxutos e explícitos. A parametrização é construída em seu núcleo usando o poderoso mecanismo de modelagem Jinja.

Airflow



Airflow

Existem alguns componentes a serem observados:

- **Banco de dados de metadados** : o Airflow usa um banco de dados SQL para armazenar metadados sobre os pipelines de dados em execução. No diagrama acima, isso é representado como Postgres, que é extremamente popular com o Airflow. Bancos de dados alternativos compatíveis com Airflow incluem MySQL.
- **Servidor da Web e Agendador** : o servidor da web do Airflow e o Agendador são processos separados executados (neste caso) na máquina local e interagem com o banco de dados mencionado acima.
- O **Executor** é mostrado separadamente acima, uma vez que é comumente discutido no Airflow e na documentação, mas na realidade NÃO é um processo separado, mas executado dentro do Scheduler.
- O **(s) Worker (s)** são processos separados que também interagem com os outros componentes da arquitetura do Airflow e o repositório de metadados.
- `airflow.cfg` é o arquivo de configuração do Airflow que é acessado pelo Web Server, Scheduler e Workers.
- **DAGs** referem-se aos arquivos DAG que contêm código Python, representando os pipelines de dados a serem executados pelo Airflow. A localização desses arquivos é especificada no arquivo de configuração do Airflow, mas eles precisam estar acessíveis para o servidor da Web, o agendador e os trabalhadores.

Airflow - Características

- Python
- Monitoramento
- Integrações
- Facilidade
- Opensource

Airflow - Operadores

- **BashOperator** - executa um comando bash
- **PythonOperator** - chama uma função Python arbitrária
- **EmailOperator** - enviar um e-mail
- **SimpleHttpOperator** - enviar uma solicitação HTTP

Bibliografia

- <https://medium.com/data-ops/dataops-is-not-just-devops-for-data-6e03083157b7>
- <https://www.aitrends.com/machine-learning/mlops-not-just-ml-business-new-competitive-frontier/>
- <https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning?hl=pt-br>
- <https://www.aitrends.com/machine-learning/mlops-not-just-ml-business-new-competitive-frontier/>
- <https://martinfowler.com/articles/cd4ml.html>
- <https://hopsworks.readthedocs.io/en/1.1/featurestore/featurestore.html>
- <https://mlflow.org/docs/latest/tutorials-and-examples/tutorial.html>
- <https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html>
- <https://spark.apache.org>
- <https://airflow.apache.org/>
- <https://hopsworks.readthedocs.io/en/1.1/featurestore/featurestore.html>
- <https://feast.dev/>
- <https://aws.amazon.com/pt/sagemaker/feature-store/>



OBRIGADO



Jean Carlos Alves