~~Serving & training optimisation~~
ML Pipeline

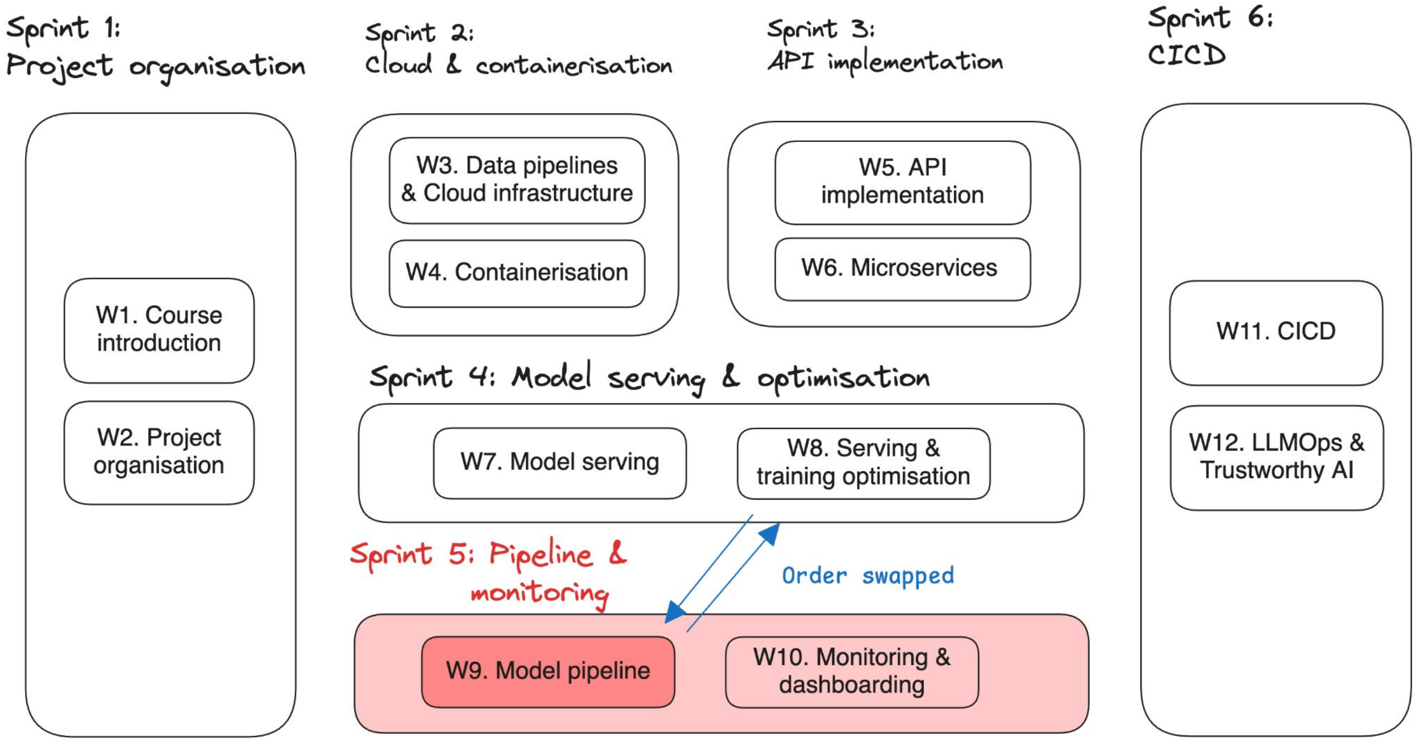
Sprint 5 - Week 9

INFO 9023 - Machine Learning Systems Design

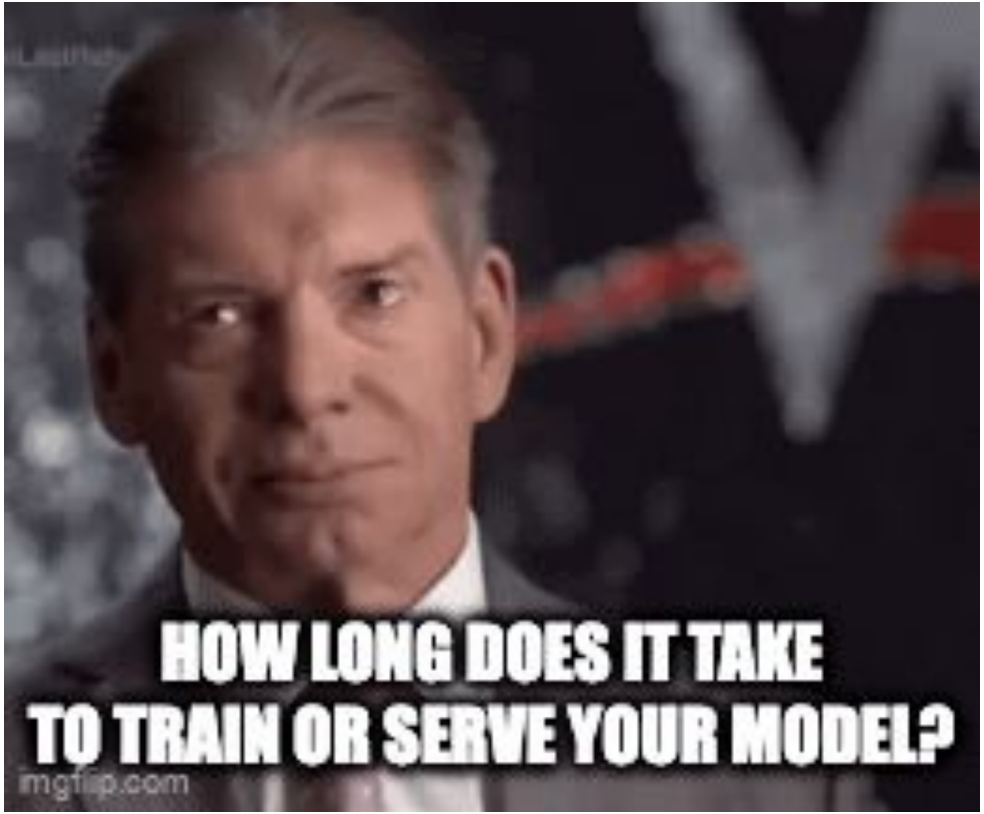
Thomas Vrancken (t.vrancken@uliege.be)
Matthias Pirlet (matthias.pirlet@uliege.be)

2025 Spring

Status on our overall course roadmap



LIÈGE université


HOW LONG DOES IT TAKE
TO TRAIN OR SERVE YOUR MODEL?
imgflip.com

Agenda

What will we talk about today

Lecture

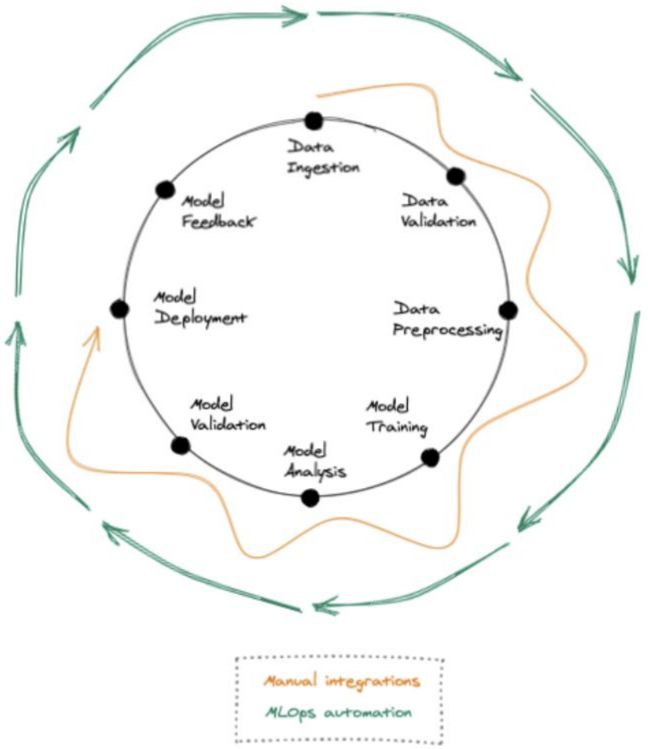
1. ML model pipeline
2. ML platforms & orchestrators

Directed Work

3. Vertex Pipeline

ML model pipeline

Why do we need ML pipelines?



The general idea is to not treat **ML workflows** as a one-off, but to treat them in a **reliable** and **reproducible** way.

LIÈGE université

Why do we need ML pipelines?

The Spotify experience

The Winding Road to Better Machine Learning Infrastructure Through Tensorflow Extended and Kubeflow

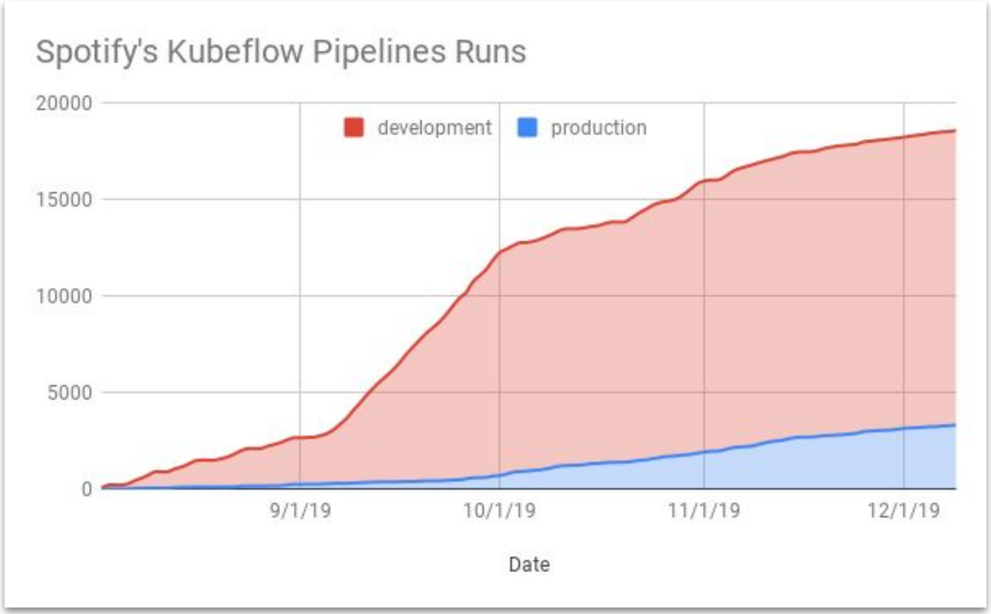
Posted on December 13, 2019 by josh baer and samuelngahane

"As we built these new Machine Learning systems, we started to hit a point where **our engineers spent more of their time maintaining data and backend systems in support of the ML-specific code** than iterating on the model itself. We realized we needed to standardize best practices and build tooling to bridge the gaps between data, backend, and ML: we needed a Machine Learning platform. "

https://labs.spotify.com/2019/12/13/the-winding-road-to-better-machine-learning-infrastructure-through-tensorflow-extended-and-kubeflow/

Why do we need ML pipelines?

Spotify accomplished an astonishing 7x for running machine learning pipelines

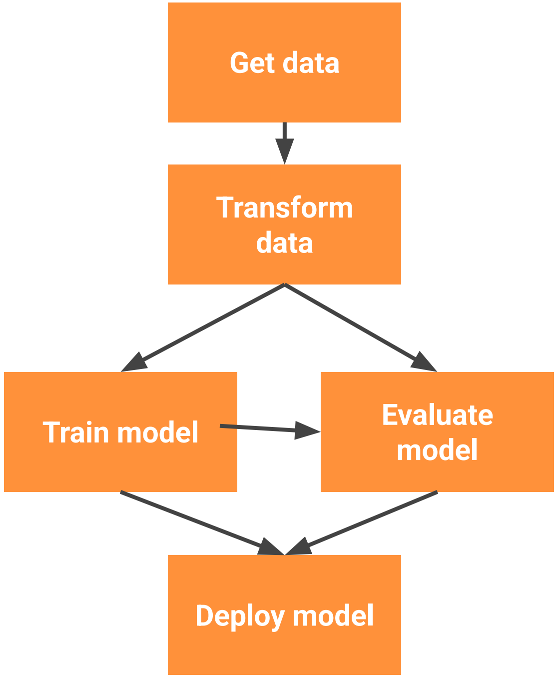


https://labs.spotify.com/2019/12/13/the-winding-road-to-better-machine-learning-infrastructure-through-tensorflow-extended-and-kubeflow/

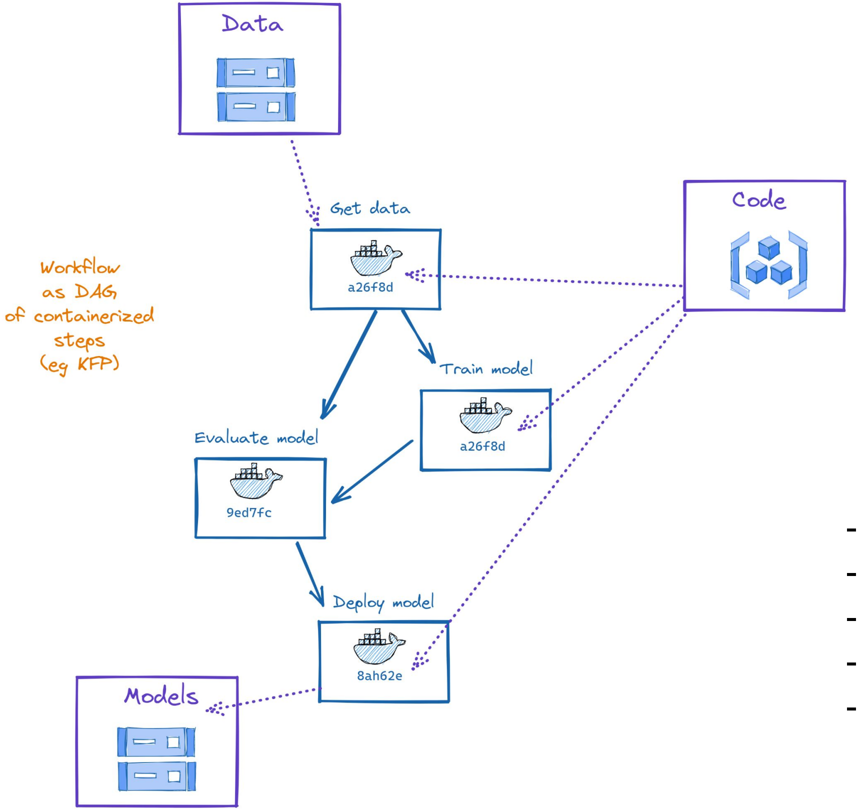
Represent your entire ML workflow as a DAG

Think about your workflow in terms of Directed Acyclic Graph (**DAGs**):

- What needs to be done sequentially vs in parallel
- Does the step process something itself or call an external service?
- Process first, implementation second

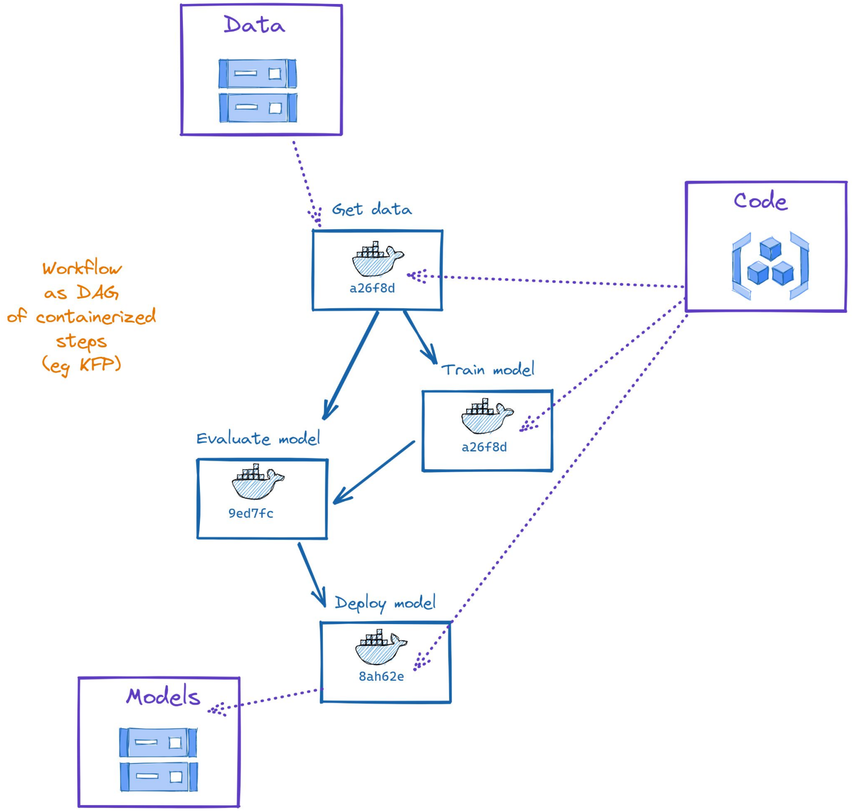


Represent your entire ML workflow as a DAG



- Kubeflow pipelines
- Sagemaker pipelines
- Vertex pipelines
- Valohai
- MLFlow

Definition of a ML Pipeline



"A machine learning pipeline is a series of interconnected data processing and modeling steps designed to automate, standardize and streamline the process of building, training, evaluating and deploying machine learning models."

https://www.ibm.com/topics/machine-learning-pipeline

ML (pipeline) platforms

Amazon SageMaker

Vertex AI

Azure Machine Learning

Kubeflow

ML (pipeline) platforms

Amaz
SageM

ertex AI

flow

Azure Machine

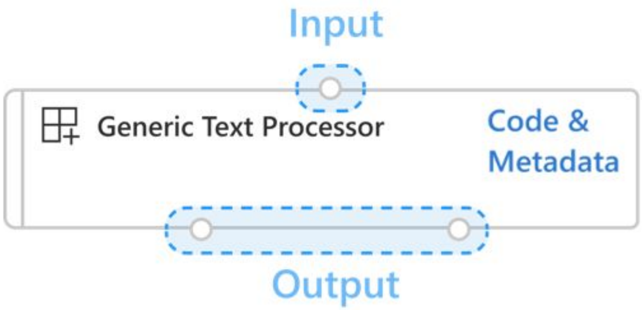
What are components of an ML pipeline?

Components are the building blocks of an ML pipeline. They are the nodes of the ML pipeline DAG.

They contain:

- Metadata
- Interface (input/output specifications)
- Command, Code & Environment.

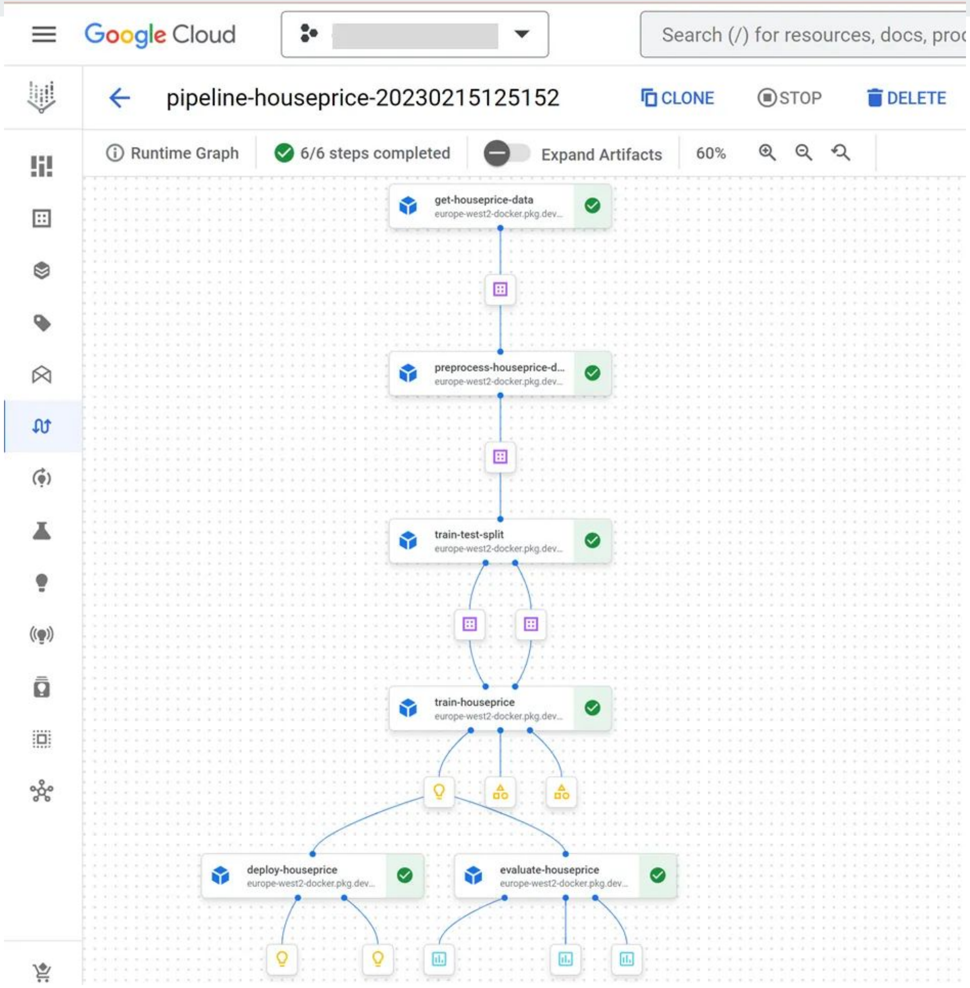
Each component will be made of a script ran in a separated **Docker image**.



https://learn.microsoft.com/en-us/azure/machine-learning/concept-component?view=azureml-api-2

Example ML Pipeline

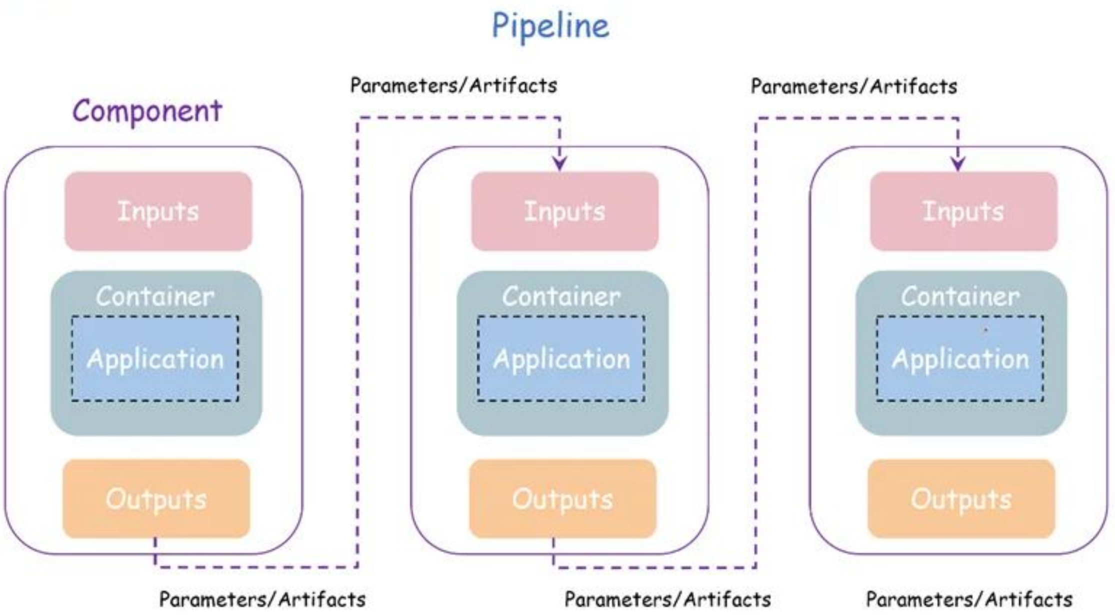
Predict house prices with GCP Vertex AI



https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

How to pass data between components?

Data staging

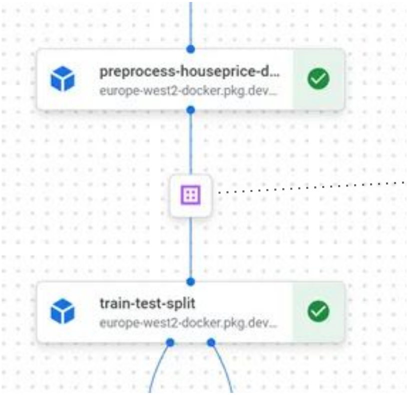


https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-1-61185bc24f6a

How to pass data between components?

Data staging

Data needs to be **staged** between components, stored into a Cloud service at the end of a component so it can be taken up by the next component.



Google Cloud Storage

Vertex pipelines integrate with Google Cloud Storage out of the box.

How to pass data between components?

Data staging

```python
from kfp.v2 import dsl, compiler
from kfp.v2.dsl import component, Input, Output, Dataset

@component
def generate_data(output_data: Output[Dataset]):
    import pandas as pd
    df = pd.DataFrame({'numbers': [1, 2, 3, 4, 5]})
    df.to_csv(output_data.path, index=False)

@component
def process_data(input_data: Input[Dataset], output_data: Output[Dataset]):
    import pandas as pd
    df = pd.read_csv(input_data.path)
    df['squared'] = df['numbers'] ** 2
    df.to_csv(output_data.path, index=False)
```

```
gs://your-bucket/pipeline-root/runs/{run-id}/generate_data/output_data/
gs://your-bucket/pipeline-root/runs/{run-id}/process_data/output_data/
```

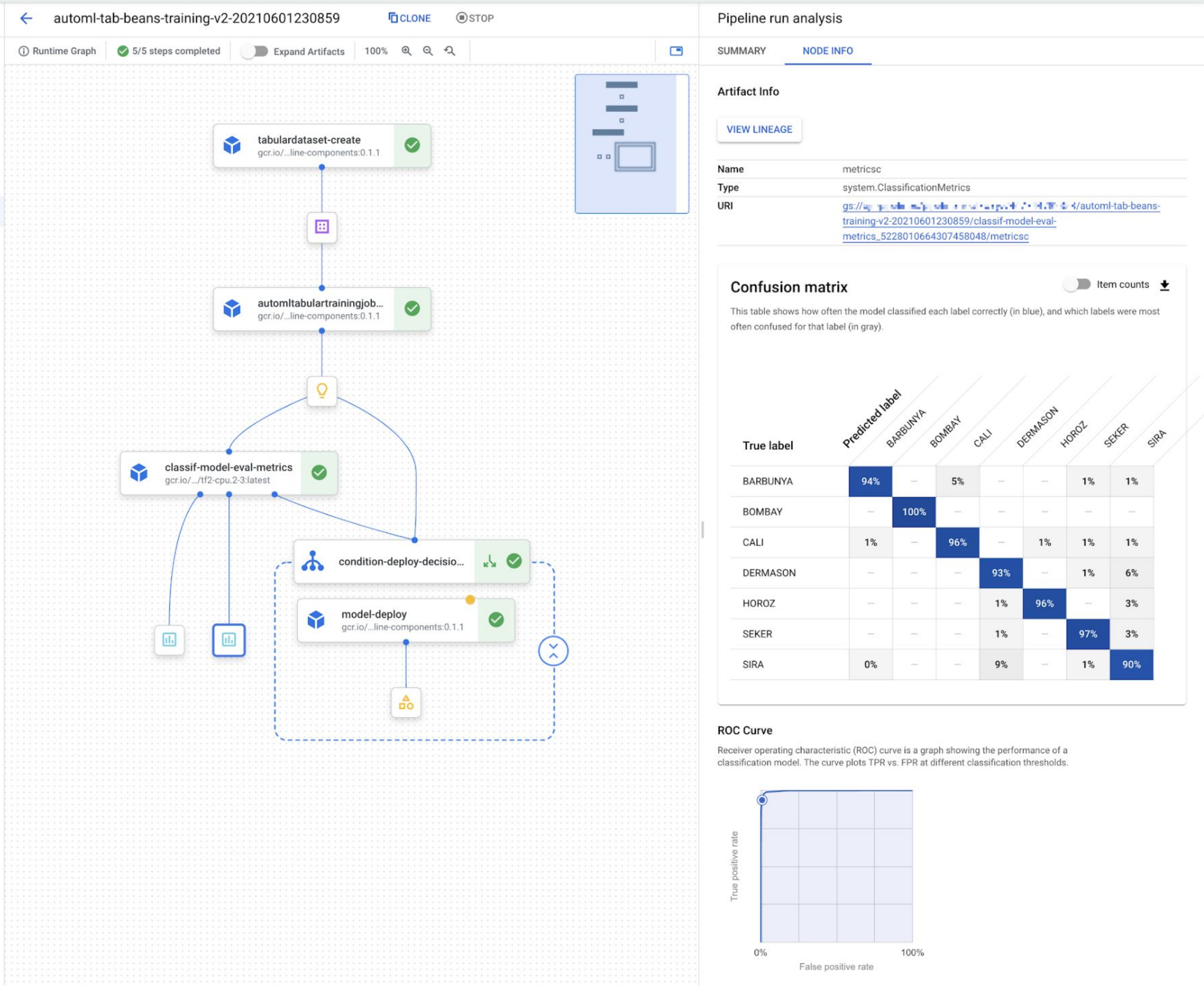
Creates the following GCS objects

Same example but using BigQuery for staging

```python
from kfp.v2 import dsl, compiler
from kfp.v2.dsl import component, Input, Output, Dataset, Model

BQ_DATASET = "your_bq_dataset"
BQ_SOURCE_TABLE = f"{PROJECT_ID}.{BQ_DATASET}.source_table"
BQ_TARGET_TABLE = f"{PROJECT_ID}.{BQ_DATASET}.processed_table"

@component
def create_bigquery_table():
    """Creates a BigQuery table and inserts sample data."""
    from google.cloud import bigquery

    client = bigquery.Client()
    schema = [
        bigquery.SchemaField("id", "INTEGER"),
        bigquery.SchemaField("value", "FLOAT"),
    ]

    table = bigquery.Table(BQ_SOURCE_TABLE, schema=schema)
    client.create_table(table, exists_ok=True)

    rows = [{"id": i, "value": float(i)} for i in range(1, 6)]

@component
def process_bigquery_data(bq_source: str, bq_target: str):
    """Reads from BigQuery, processes the data, and writes back to BigQuery."""
    from google.cloud import bigquery

    client = bigquery.Client()
    query = f"""
    SELECT id, value, value * value AS squared_value
    FROM `{bq_source}`
    """
    df = client.query(query).to_dataframe()

    job_config = bigquery.LoadJobConfig(write_disposition="WRITE_TRUNCATE")
    client.load_table_from_dataframe(df, bq_target, job_config=job_config)
```

LIÈGE université

Vertex lets you easily visualise artifacts - such as evaluation metrics



https://cloud.google.com/blog/topics/developers-practitioners/use-vertex-pipelines-build-automl-classification-end-end-workflow

LIÈGE université

Example ML Pipeline

Predict house prices with GCP Vertex AI

Based on [Kubeflow Pipelines](#) (KFP)

```python
# IMPORT REQUIRED LIBRARIES
from kfp.v2 import dsl
from kfp.v2.dsl import (Artifact,
                        Dataset,
                        Input,
                        Model,
                        Output,
                        Metrics,
                        Markdown,
                        HTML,
                        component,
                        OutputPath,
                        InputPath)
from kfp.v2 import compiler
from google.cloud.aiplatform import pipeline_jobs
```


https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

Example ML Pipeline

Predict house prices with GCP Vertex AI

Define **components**.

Can be separate **Docker containers** or **python functions**.

Can create a pipeline in one single notebook

```
@component(
    base_image=BASE_IMAGE,
    output_component_file="get_data.yaml"
)

def get_houseprice_data(
    filepath: str,
    dataset_train: Output[Dataset],
):

    import pandas as pd

    df_train = pd.read_csv(filepath + '/train.csv')

    df_train.to_csv(dataset_train.path, index=False)
```

Data Ingestion component

LIÈGE université

https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

Example ML Pipeline

Predict house prices with GCP Vertex AI

```python
@component(
    base_image=BASE_IMAGE,
    output_component_file="preprocessing.yaml"
)

def preprocess_houseprice_data(
    train_df: Input[Dataset],
    dataset_train_preprocessed: Output[Dataset],
):

    import pandas as pd
    from src.data_preprocessing.preprocessing import data_preprocessing_pipeline

    train_df = pd.read_csv(train_df.path)

    # data_preprocessing_pipeline creates a copy of the df, removes id col, converts to
    # subtracts YearSold from temporal features and cosine transforms cyclic features.
    train_df_preprocessed = data_preprocessing_pipeline(train_df)

    train_df_preprocessed.to_csv(dataset_train_preprocessed.path, index=False)
```

Data Preprocessing component

```python
@component(
    base_image=BASE_IMAGE,
    output_component_file="train_test_split.yaml",
)

def train_test_split(dataset_in: Input[Dataset],
                     dataset_train: Output[Dataset],
                     dataset_test: Output[Dataset],
                     test_size: float = 0.2):

    import pandas as pd
    from sklearn.model_selection import train_test_split

    df = pd.read_csv(dataset_in.path)
    df_train, df_test = train_test_split(df, test_size=test_size, random_state=42)

    df_train.to_csv(dataset_train.path, index=False)
    df_test.to_csv(dataset_test.path, index=False)
```

Train test split component

https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

Example ML Pipeline

Predict house prices with GCP Vertex AI

...

```python
import pandas as pd
import pickle
import shap
from src.modelling.train import HousePriceModel
from src.utils.utils import get_image_data

TARGET = 'SalePrice'

# Read train and test data
train_data = pd.read_csv(dataset_train.path)
test_data = pd.read_csv(dataset_test.path)

# Instantiate the model class
house_price_model = HousePriceModel(test_data.copy(),   #we perform hyperparameter t
                                    target=TARGET,
                                    n_kfold_splits=3,
                                    n_trials=100,
                                    random_state=42)

# Create X_train and y_train
X_train = train_data.drop(TARGET, axis=1)
y_train = train_data[TARGET]
```

...

Model training component

```python
@component(
    base_image=BASE_IMAGE,
    output_component_file="model_evaluation.yaml"
)
def evaluate_houseprice(
    houseprice_model: Input[Model],
    metrics_baseline: Output[Metrics],
    metrics_train: Output[Metrics],
    metrics_test: Output[Metrics]):

    import pickle

    file_name = houseprice_model.path
    with open(file_name, 'rb') as file:
        model_data = pickle.load(file)

    scores = model_data["scores_dict"]

    def log_metrics(scores, metric):
        for metric_name, val in scores.items():
            metric.log_metric(metric_name, float(val))

    log_metrics(scores["baseline_scores"], metrics_baseline)
    log_metrics(scores["train_scores"], metrics_train)
    log_metrics(scores["test_scores"], metrics_test)
```

Model evaluation component

https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

Example ML Pipeline

Predict house prices with GCP Vertex AI

```python
from google.cloud import aiplatform as vertex_ai
from pathlib import Path

# Checks existing Vertex AI Enpoint or creates Endpoint if it is not exist.
def create_endpoint ():
    endpoints = vertex_ai.Endpoint.list(
    filter='display_name="{}"'.format(model_endpoint),
    order_by='create_time desc',
    project=gcp_project,
    location=gcp_region,
    )
    if len(endpoints) > 0:
        endpoint = endpoints[0] # most recently created
    else:
        endpoint = vertex_ai.Endpoint.create(
            display_name=model_endpoint,
            project=gcp_project,
            location=gcp_region
    )
    return endpoint
```

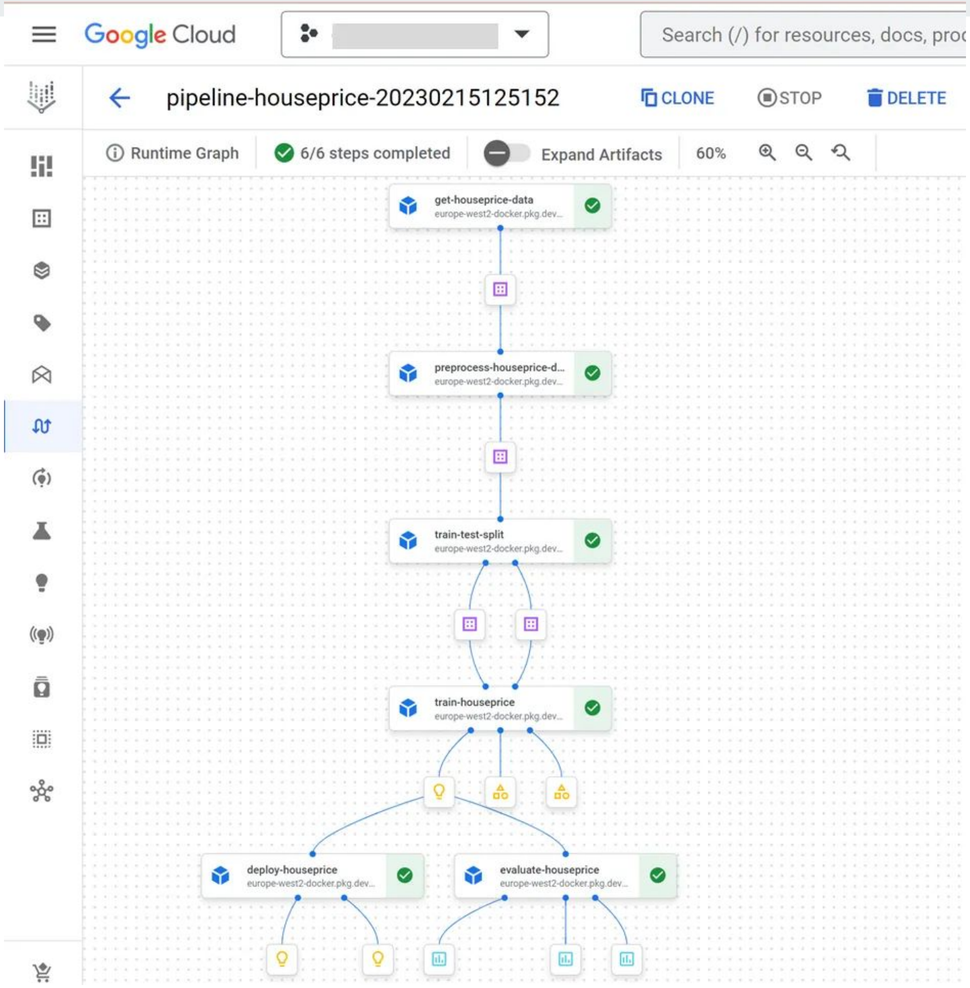
```python
# Uploads trained model to Vertex AI Model Registry or creates new model version in
def upload_model ():
    listed_model = vertex_ai.Model.list(
    filter='display_name="{}"'.format(display_name),
    project=gcp_project,
    location=gcp_region,
    )
    if len(listed_model) > 0:
        model_version = listed_model[0] # most recently created
        model_upload = vertex_ai.Model.upload(
                display_name=display_name,
                parent_model=model_version.resource_name,
                artifact_uri=str(Path(model.path).parent),
                serving_container_image_uri=serving_container_image_uri,
                location=gcp_region,
                serving_container_predict_route="/predict",
                serving_container_health_route="/health"
        )
```

Model deployment component

https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

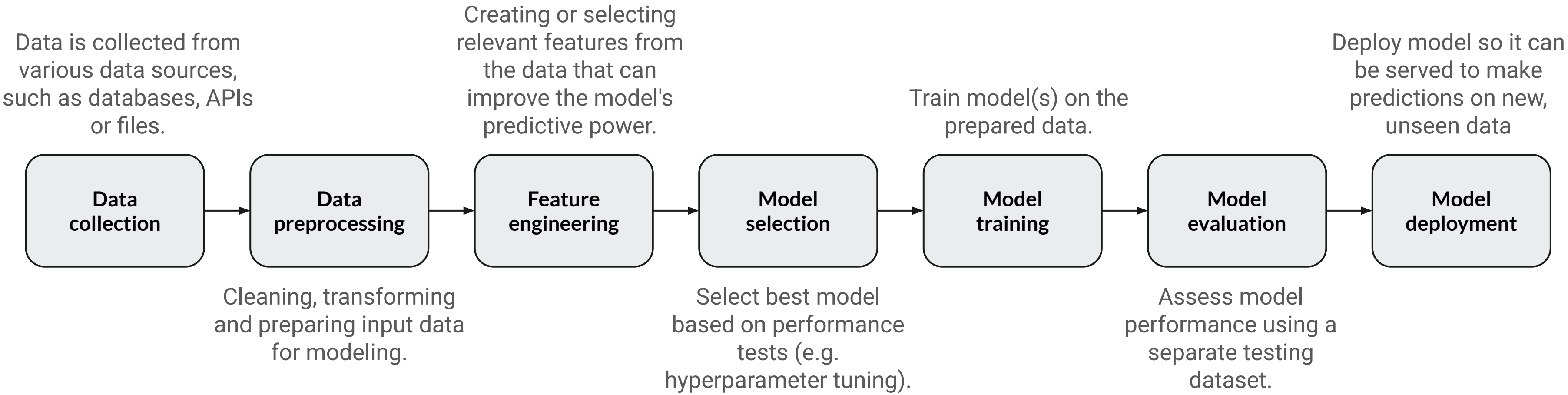
Example ML Pipeline

Predict house prices with GCP Vertex AI



https://medium.com/@wardarahim25/step-by-step-guide-to-creating-and-deploying-custom-ml-pipelines-with-gcp-vertex-ai-part-2-3be6e314bc48

Typical components of an ML pipeline



https://www.ibm.com/topics/machine-learning-pipeline

Avoiding the endless POC loop…

- Let’s change the model architecture
- Let’s try with another model
- Let’s gather more labeled data
- Ah wait, it was the wrong data, here is the new data, can we get results tomorrow?
- What were the accuracies again with the other model?
- Oops not sure what notebook was executed to get that model
- Could we try out the model to get feedback?
- …

*The key to a **successful POC outcome** is to **plan for production** during the proof of concept.*

Benefits of an ML Pipeline

- Modularization: Breaks ML workflows into reusable components, making development faster and easier to manage.
- Reproducibility: Ensures experiments and results can be consistently recreated by tracking configurations and data.
- Efficiency: Automates repetitive tasks like data preprocessing and model training, saving time and effort.
- Scalability: Seamlessly handles large datasets and complex models by leveraging cloud infrastructure.
- Experimentation: Enables quick testing of different models and hyperparameters without manual reconfiguration.
- Deployment: Streamlines moving models from development to production with automated workflows.
- Collaboration: Facilitates teamwork by structuring workflows and sharing components across teams.
- Version control and documentation: Tracks changes in datasets, models, and configurations, ensuring transparency and traceability.

https://www.ibm.com/topics/machine-learning-pipeline

Triggers

Timed

- **Schedule**: Pipeline runs repetitively in relation to the creation time of the Recurring Run. Set the unit (minutes, hours, etc.) and the scalar that goes with it

Event based

- **Manual**: Launch the pipeline manually
- **Triggered**: As part of another service (e.g. if users upload a new batch of training in a specific GCS bucket)

```
# ┌───────────── minute (0–59)
# │ ┌───────────── hour (0–23)
# │ │ ┌───────────── day of the month (1–31)
# │ │ │ ┌───────────── month (1–12)
# │ │ │ │ ┌───────────── day of the week (0–6) (Sunday to Saturday;
# │ │ │ │ │                                   7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

LIÈGE université

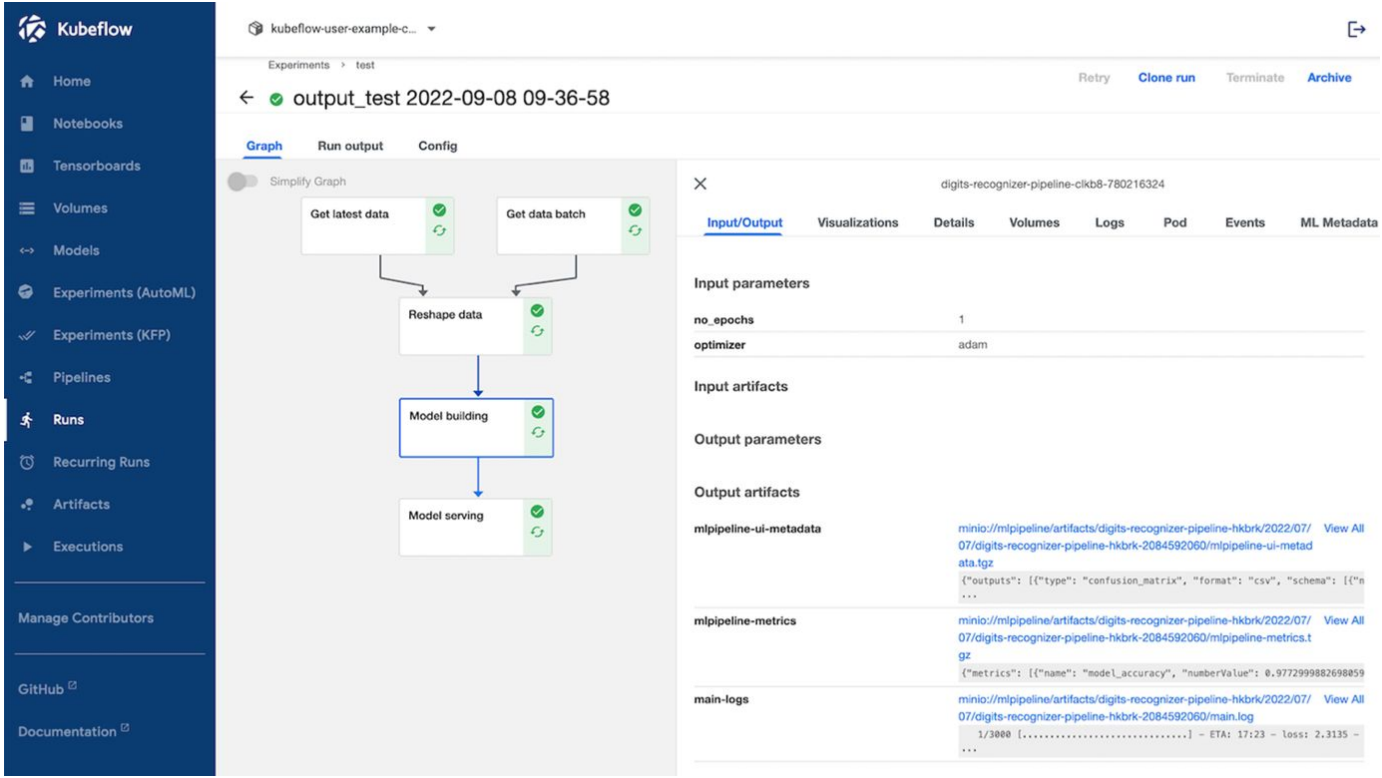
https://en.wikipedia.org/wiki/Cron

Microservices vs ML Pipeline

	Microservices	**ML Pipeline**
What is it?	One full application made of different services that are calling each other to serve the overall purpose of the application.	Ran a single workflow made of components that run sequentially.
Is made of…	Services (~= APIs)	Components. Single process that is ran once (e.g. data preparation, model training, evaluation and deployment).
Utilisation	The microservices stay up.	One time run (triggered/scheduled).
Purpose	All over software engineering.	ML.

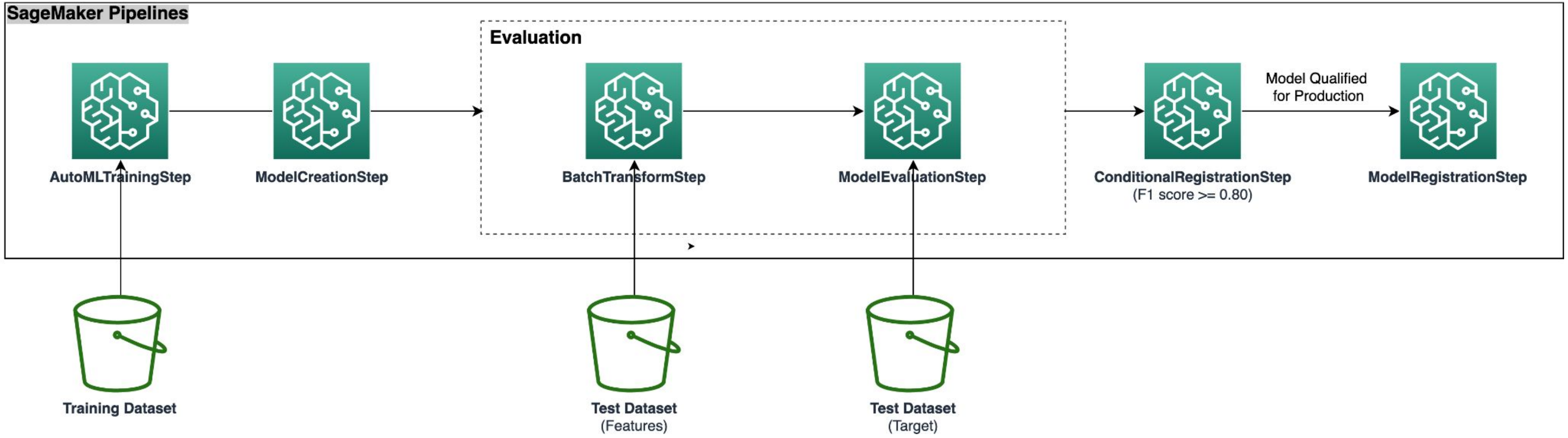
LIÈGE université

Example ML Pipeline (Kubeflow pipeline KFP)



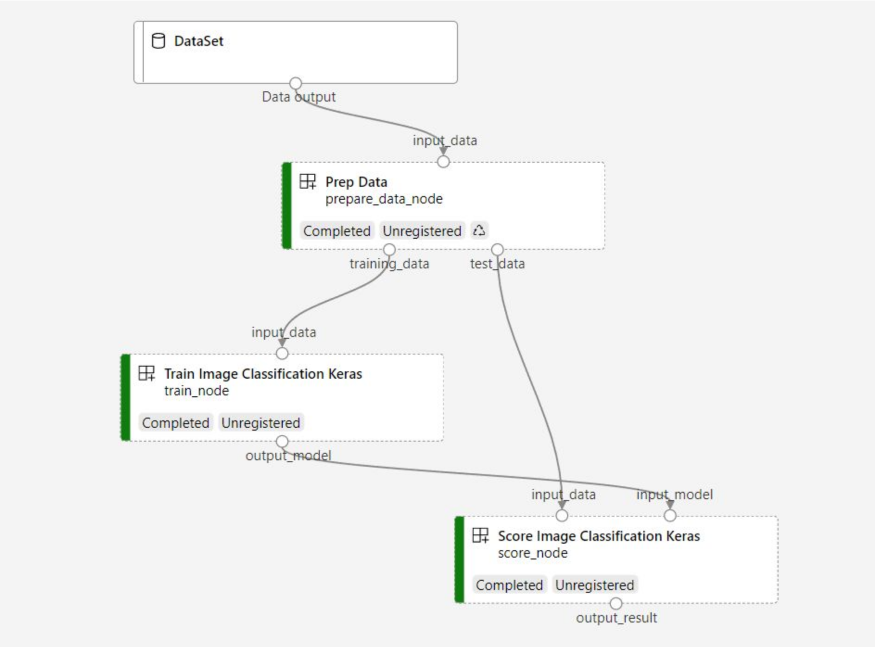
https://learn.microsoft.com/en-us/azure/machine-learning/how-to-create-component-pipeline-python?view=azureml-api-2

Example ML Pipeline (Sagemaker)



https://github.com/aws/amazon-sagemaker-examples/blob/main/sagemaker-pipelines/tabular/automl-step/sagemaker_autopilot_pipelines_native_auto_ml_step.ipynb

Example ML Pipeline (Azure ML)



https://learn.microsoft.com/en-us/azure/machine-learning/how-to-create-component-pipeline-python?view=azureml-api-2

ML Platforms

ML (pipeline) platforms

Amazon SageMaker

Vertex AI

Azure Machine Learning

Kubeflow

LIÈGE université

Sagemaker

Amazon SageMaker is an automated platform and comprehensive suite of tools that simplifies the development, training and deployment of machine learning (ML) models. It reduces the complexity of model development by providing a web-based interface for creating ML pipelines and pre-built algorithms.

Key features:

1. Notebooks
2. Training Job
3. Model registry
4. Prediction
5. Pipelines

https://www.missioncloud.com/blog/what-is-sagemaker-and-how-does-it-help-mlops

Sagemaker: Notebooks

Machine learning (ML) compute instance running the Jupyterlab. Use Jupyter notebooks in your notebook instance to prepare and process data, write code to train models, deploy models to SageMaker hosting, and test or validate your models.

Advantages:

- No use of data locally
- Select desired (GPUs)
- Better memory
- Manage access
- Collaborate as a team

Equivalent

Vertex Workbench

Azure Notebooks

https://docs.aws.amazon.com/sagemaker/latest/dg/gs-setup-working-env.html

Sagemaker: Training Job

Managed way of launching a model training

1. Either be fully specify the ML algorithm, hyperparameters and input data location.
2. Or customize training job by selecting specific Sagemaker instance types and adding software libraries.

Advantages:

- Access more compute (GPUs)
- Access more memory
- Consistency
- Automation

https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html
https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-supported-frameworks.html

SageMaker AI-supported frameworks and algorithms
TensorFlow
PyTorch
MXNet
XGBoost
SageMaker AI generic estimator

Equivalent

Vertex Training

Azure ML Training

Sagemaker: Model registry

SageMaker Model Registry is a service for packaging model artifacts with deployment information. That information includes your deployment code, what type of container to use and what type of instance to deploy. The Model Registry decreases time to deployment, as you already have the necessary information about how the model should be deployed.

Advantages:

- Consistency
- Automation
- Scalability
- Time to deployment

Equivalent

Vertex AI Model Registry

Azure ML Model Registry

https://www.missioncloud.com/blog/what-is-sagemaker-and-how-does-it-help-mlops

Sagemaker: Prediction

After you've trained and deployed your model in SageMaker, you can use it to generate predictions based on new data. There are two main ways to do this.

- Endpoint deployment: Deploy your model to an endpoint, allowing users and applications to send API requests to get model predictions in real time.
- Batch predictions: Generate predictions on large amounts of data without needing an immediate response..

Advantages:

- Consistency
- Automation
- Scalability
- Time to deployment

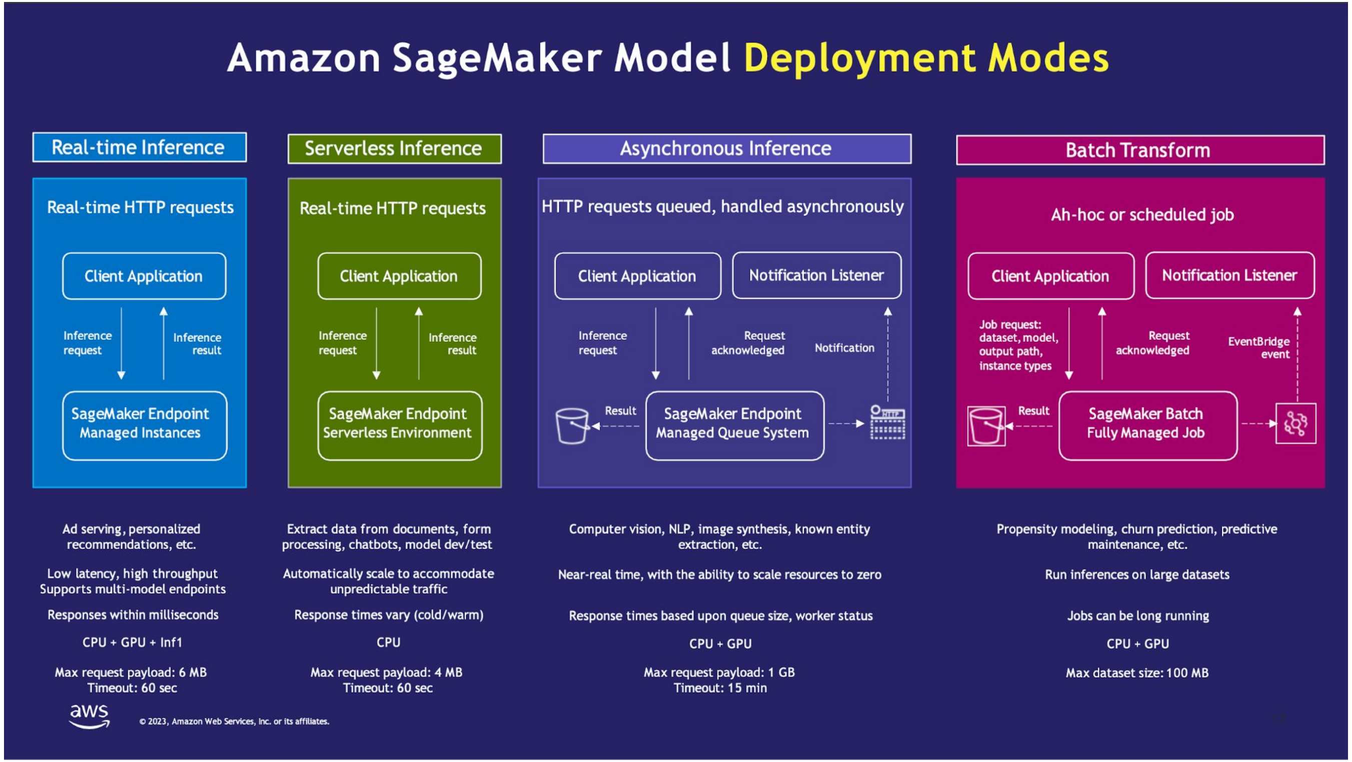
Equivalent

Vertex AI Prediction

Azure ML Prediction

https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Sagemaker: Prediction



https://docs.aws.amazon.com/sagemaker/latest/dg/deploy-model-get-started.html

Sagemaker: Pipeline

SageMaker Pipelines is a tool for building, deploying and managing end-to-end ML workflows. Users can create automated workflows that cover data preparation, model training and deployment — all from a single interface. Because Pipeline logs everything, you can easily track and re-create models.

Advantages:

- Resource consumption
- Consistency
- Automation
- Scalability
- Time to deployment

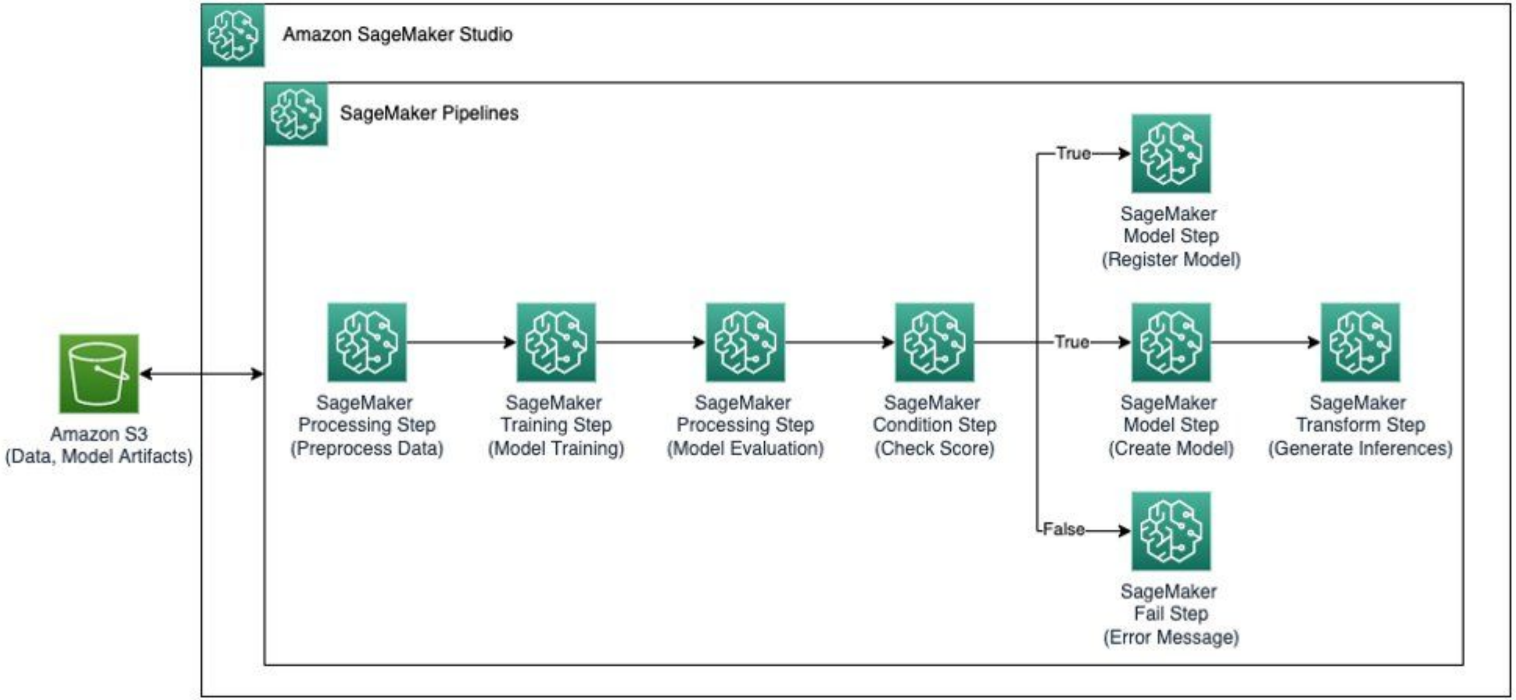
Equivalent

Vertex AI Pipeline

Azure ML Pipeline

https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Sagemaker: Pipeline (example)



https://mkai.org/new-features-for-amazon-sagemaker-pipelines-and-the-amazon-sagemaker-sdk/

Directed work: Vertex Pipeline

Wrap-up

Lecture summary

(split over next week)

Topic	Concepts	To know for…	
		Project	Exam
ML model pipeline	• What it is • Standard steps of ML pipelines		Yes
ML platforms & orchestrators	• Sagemaker offerings (workbench, training, registry. Predictions & pipelines)	Possibly (Vertex equivalent)	
Vertex Pipeline	• Build a vertex pipeline	Possibly	

LIÈGE université

Project objective for sprint 5

#	Week	Work package	Requirement
5.1	W09	Run your model training as a job in the Cloud. You can implement this in different ways: ● Containerise your training script and run it on a VM in the Cloud (e.g. on EC2 or on Cloud Run, example,) ● Use a managed service such as Vertex Training or Sagemaker Training **Attention**: This can incur Cloud **costs**. Make sure to use a platform where you have credits and not burn through them. You can ask for support from the teaching staff in that regard.	Optional
5.2	W10	Build a simple user interface or dashboard to show your results and deploy it on the Cloud.	Optional

That’s it for today!