FIWARE Global Summit

From Data to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Building real-time inference services based on ML with FIWARE

José Andrés Muñoz Arcentales, Javier Conde Díaz, Joaquín Salvachúa

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

FIWARE
Global
Summit

ML-OPS

From Data
to Value

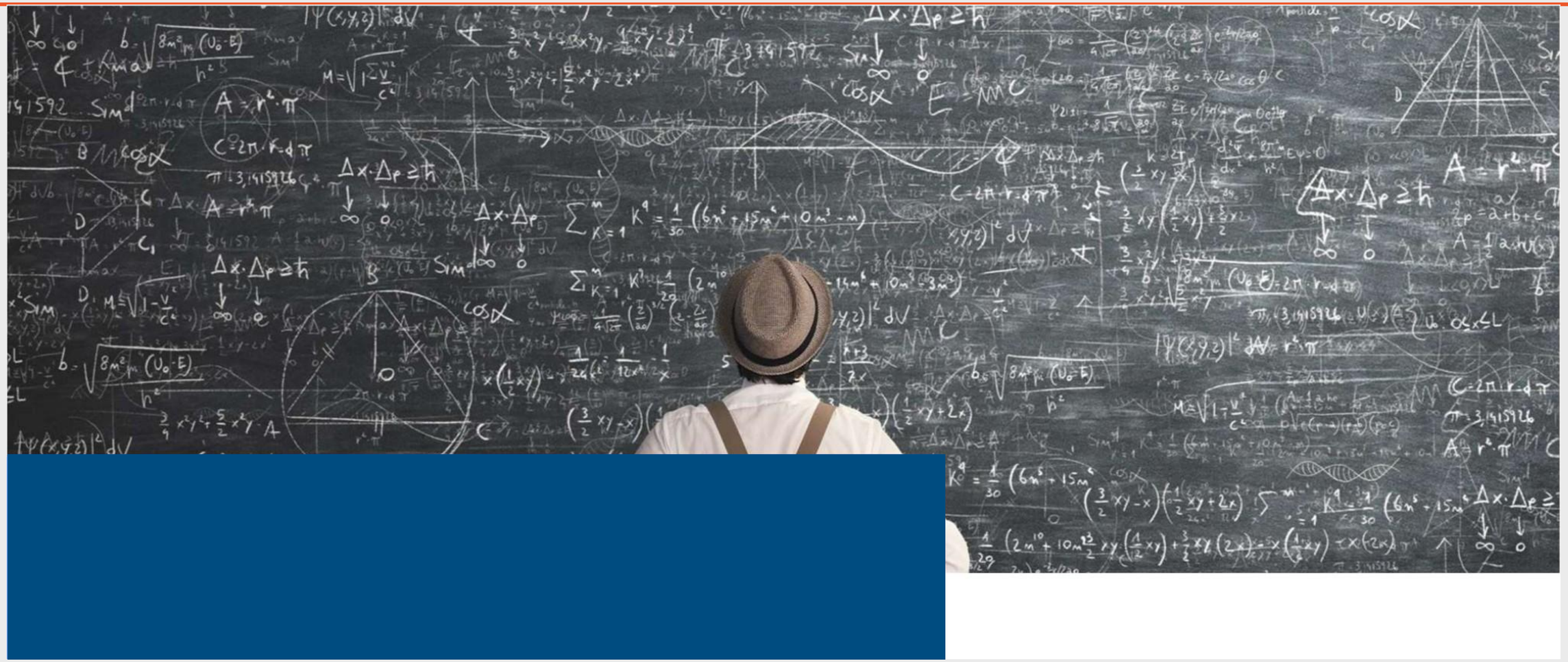
OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

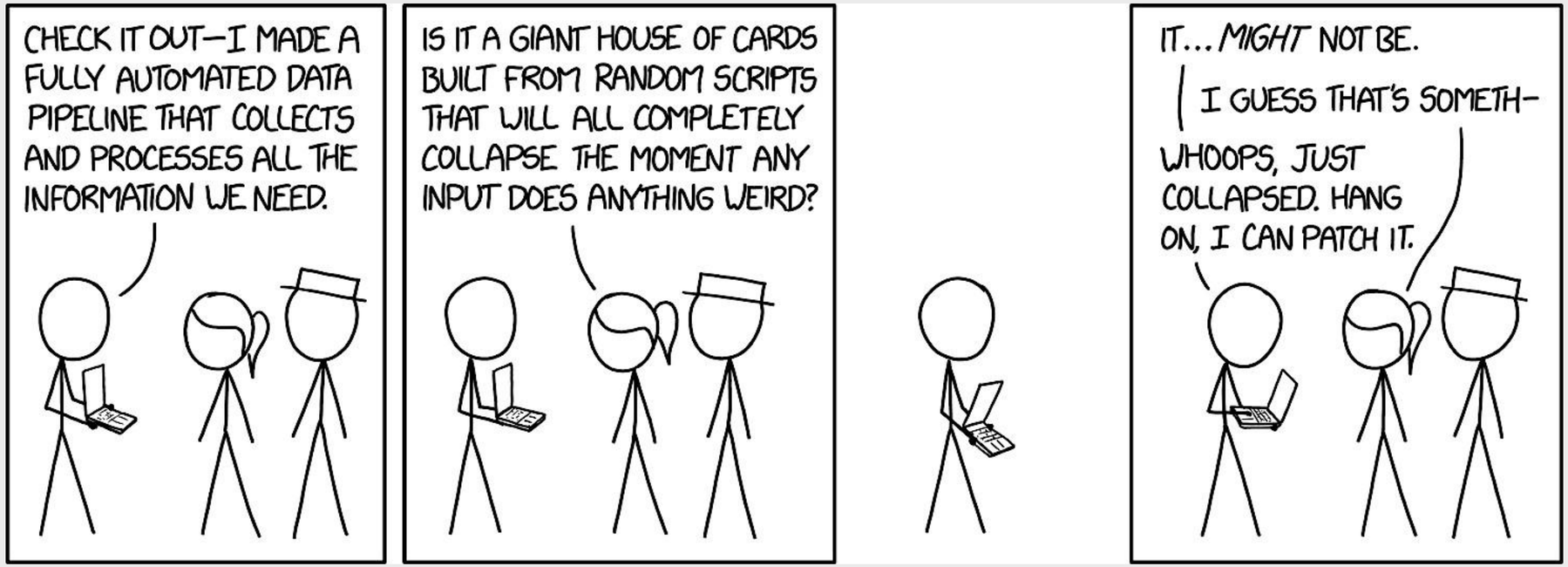
Joaquín Salvachúa

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

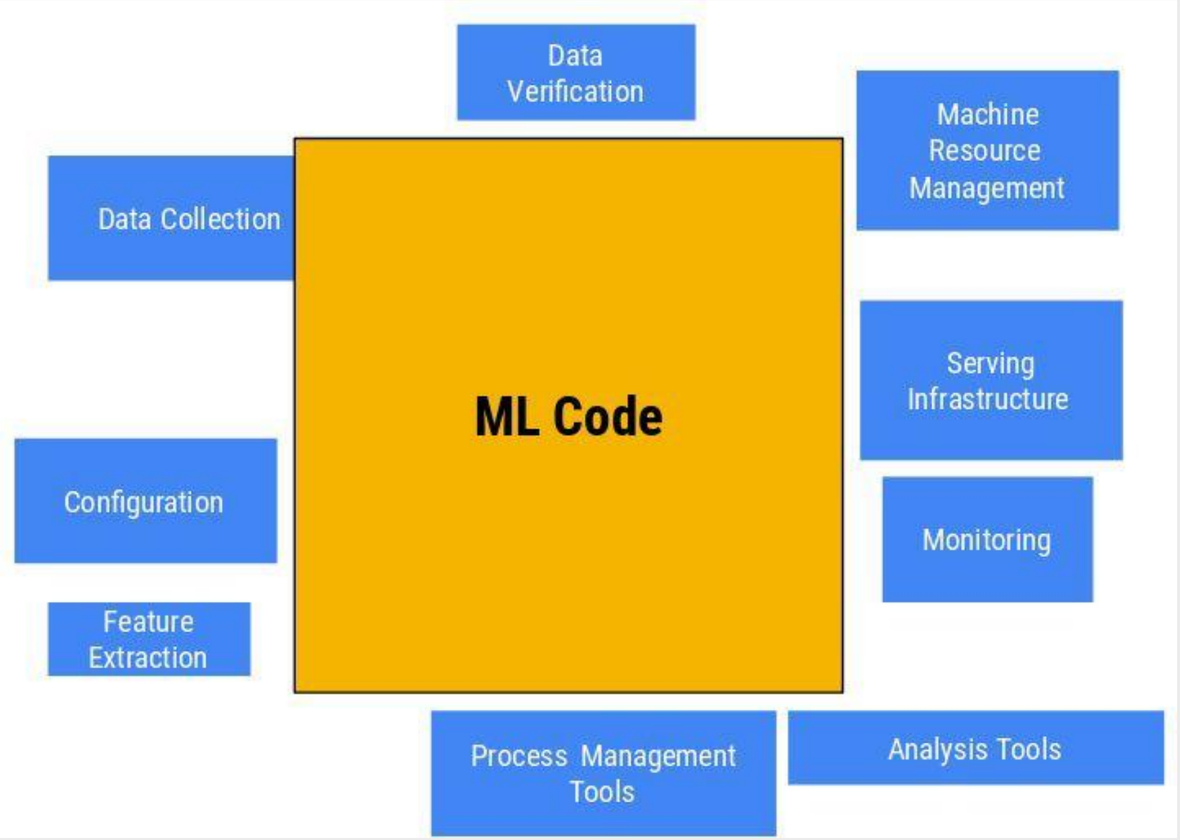
ML is hard!




CHECK IT OUT—I MADE A FULLY AUTOMATED DATA PIPELINE THAT COLLECTS AND PROCESSES ALL THE INFORMATION WE NEED.
IS IT A GIANT HOUSE OF CARDS BUILT FROM RANDOM SCRIPTS THAT WILL ALL COMPLETELY COLLAPSE THE MOMENT ANY INPUT DOES ANYTHING WEIRD?
IT... *MIGHT* NOT BE.
I GUESS THAT'S SOMETH-
WHOOPS, JUST COLLAPSED. HANG ON, I CAN PATCH IT.

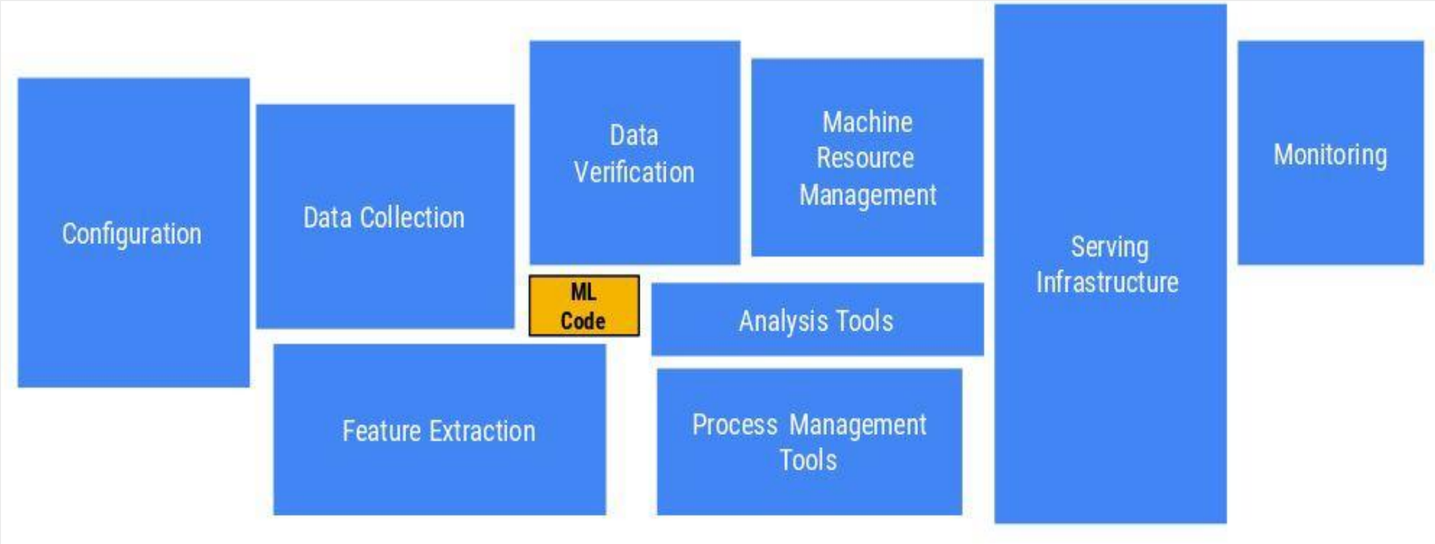
Current ML workflow

What you think



Current ML workflow

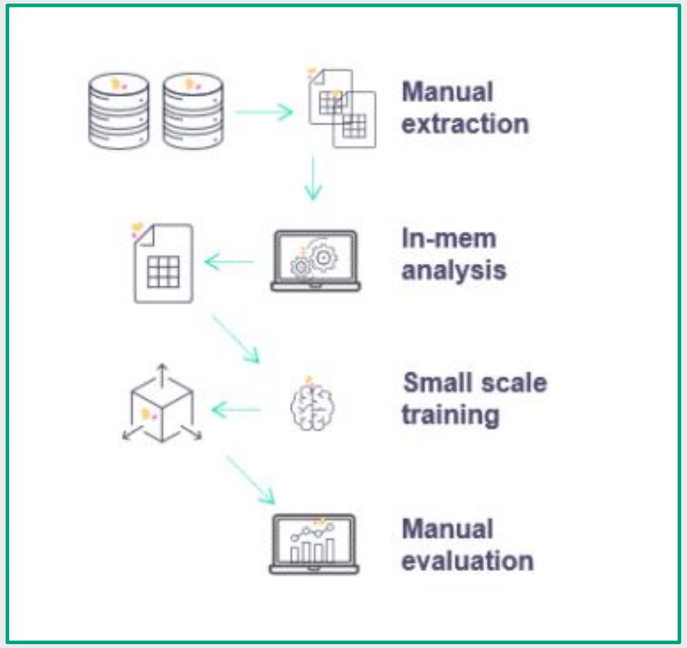
The reality



Source: https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

Introduction

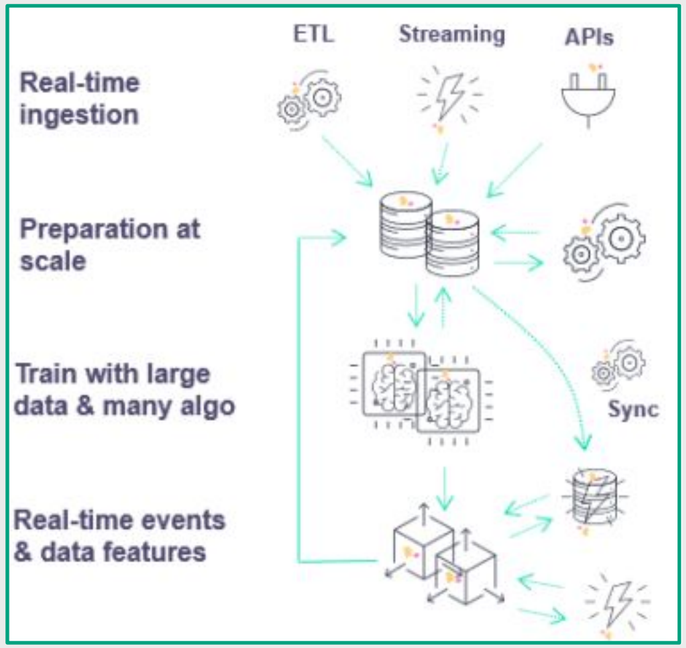
Research Environment



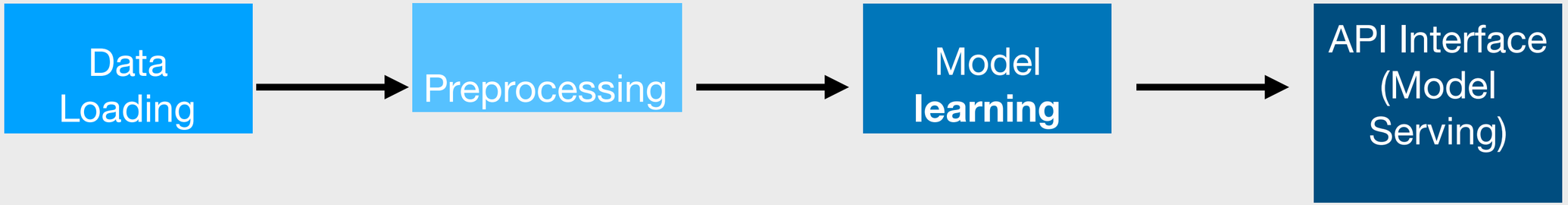
Build from Scratch

with a large team

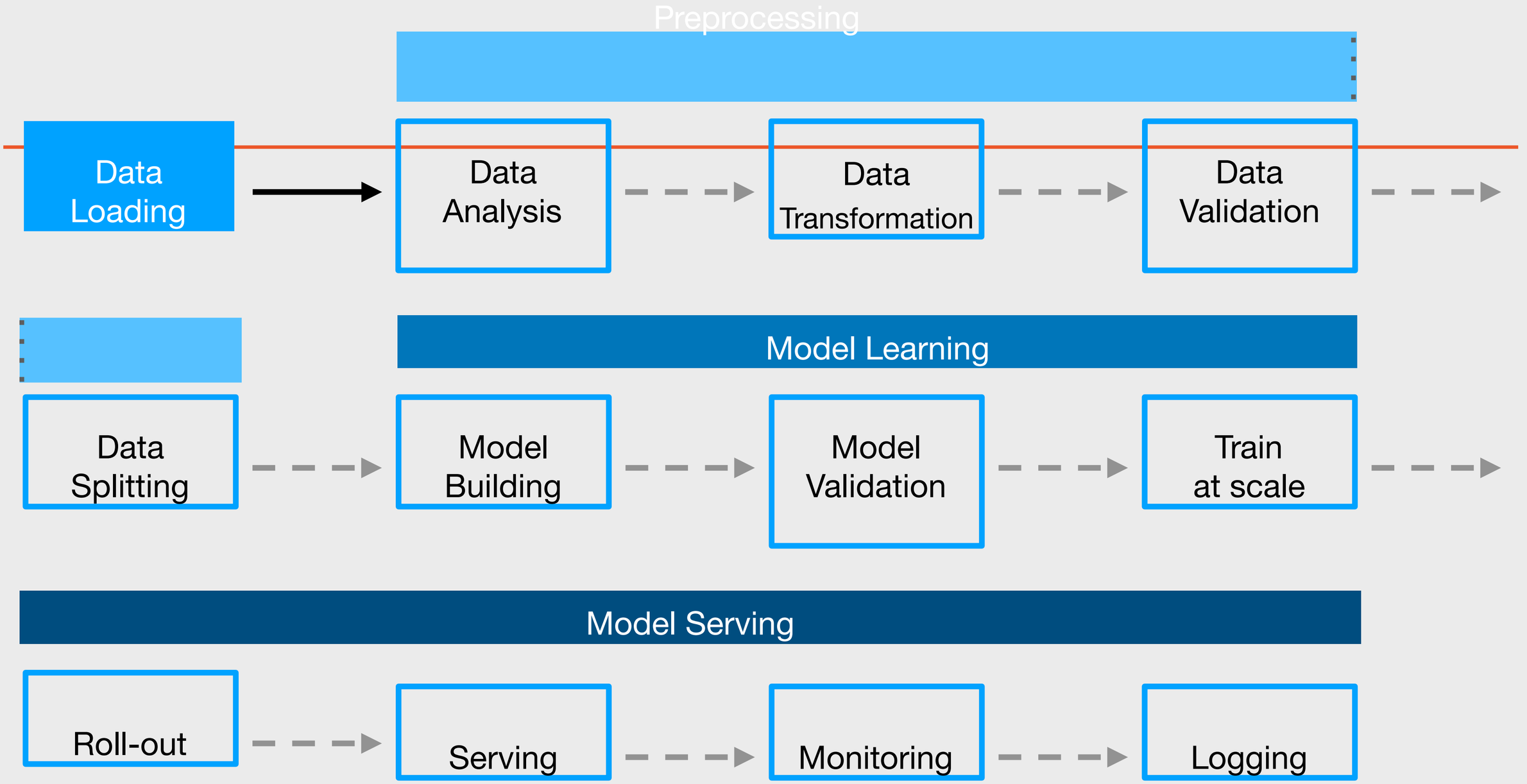
Production Pipeline



“toy” ML Pipeline



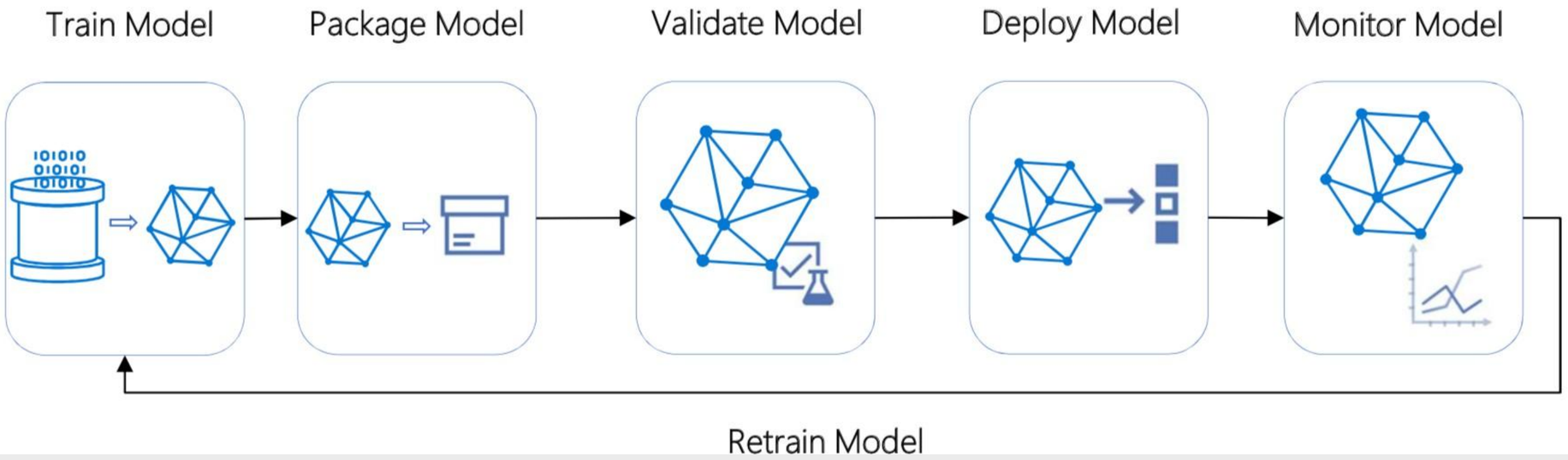
Adapted from “What about tests in Machine Learning projects?” Sarah Diot-Girard - EuroSciPy 2019



Adapted from "Managing Machine Learning in Production with Kubeflow and MLOps" David Aronchick - KubeCon EU 2019

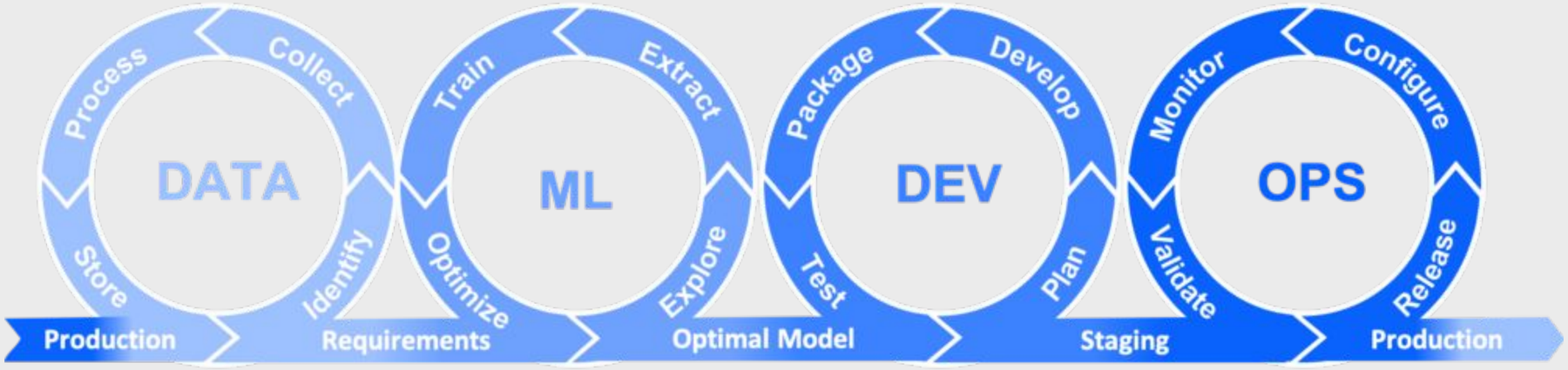
What is the E2E ML lifecycle?

- **Develop & train model** with reusable ML pipelines
- **Package model** using containers to capture runtime dependencies for inference
- **Validate model behavior –** functionally, in terms of responsiveness, in terms of regulatory compliance
- **Deploy model -** to cloud & edge, for use in real-time / streaming / batch processing
- **Monitor model** behavior & business value, know **when to replace / deprecate a stale model**

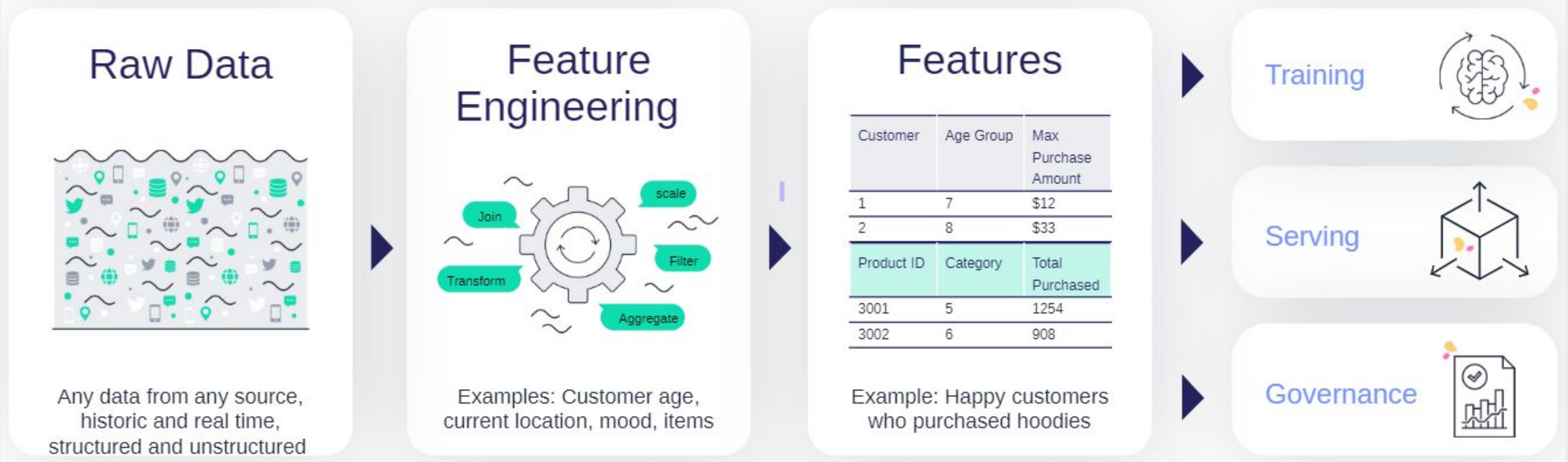


Adapted from "Managing Machine Learning in Production with Kubeflow and MLOps" David Aronchick - KubeCon EU 2019

Introduction



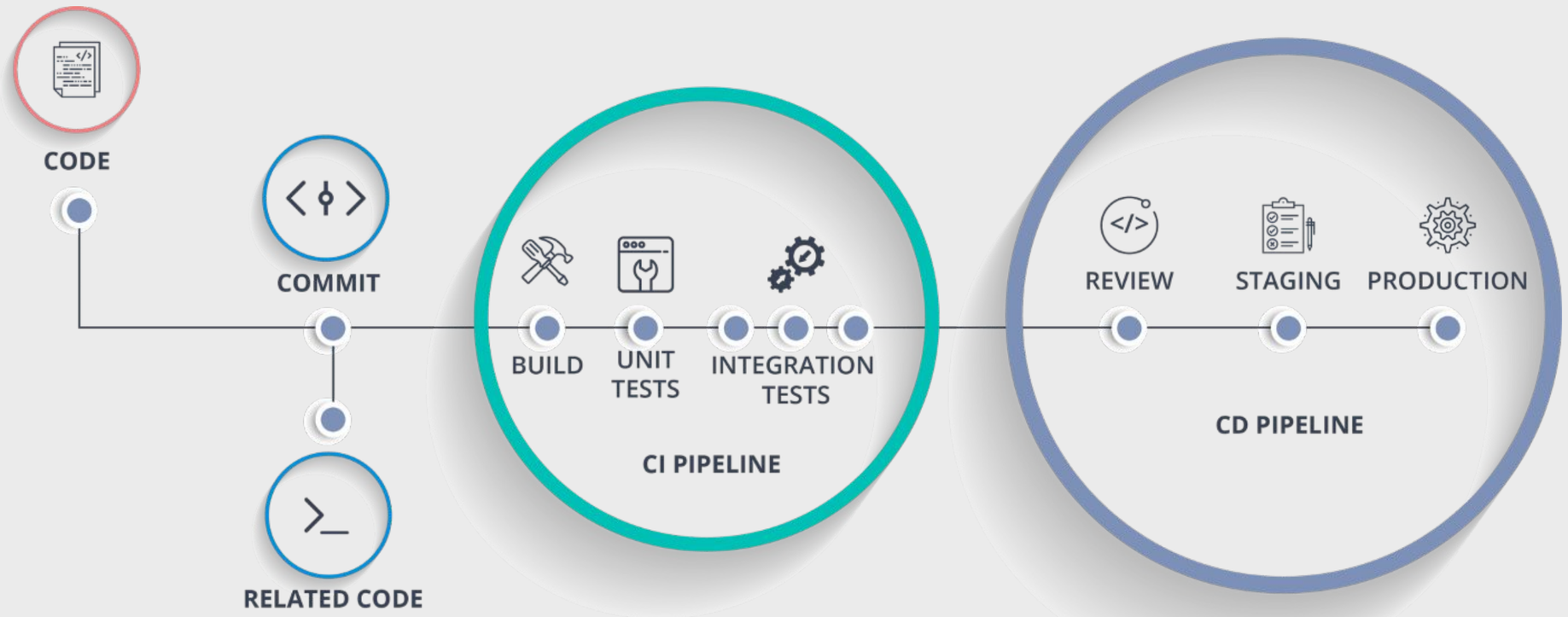
Introduction



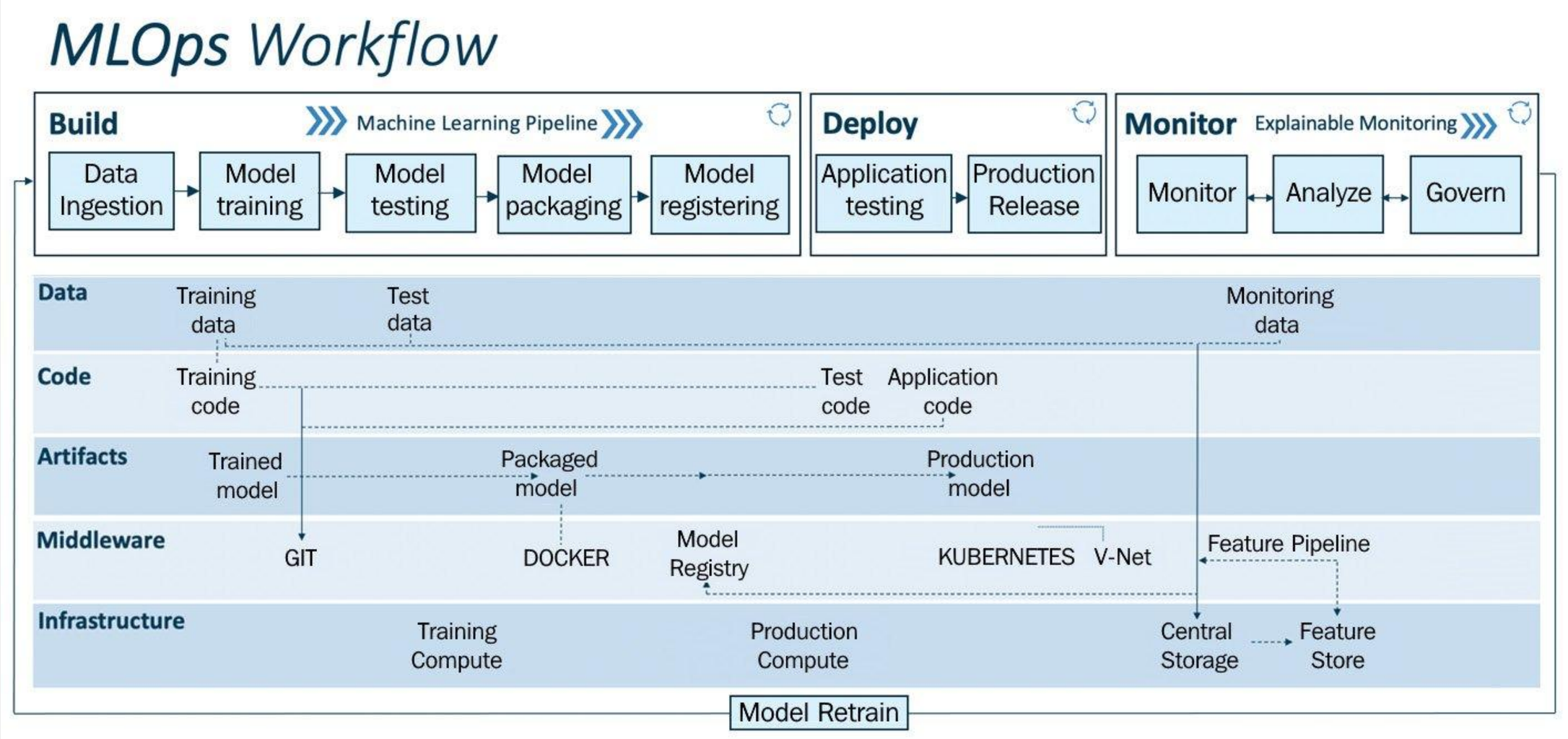
MLOps Challenges

- **Process** and **model** vast amount of data
  - In **real time** and **batch**
- Data acquisition from **heterogeneous sources**
- Communication protocols **interoperability**
- **Security** and **privacy**
  - Authentication, authorization and encryption

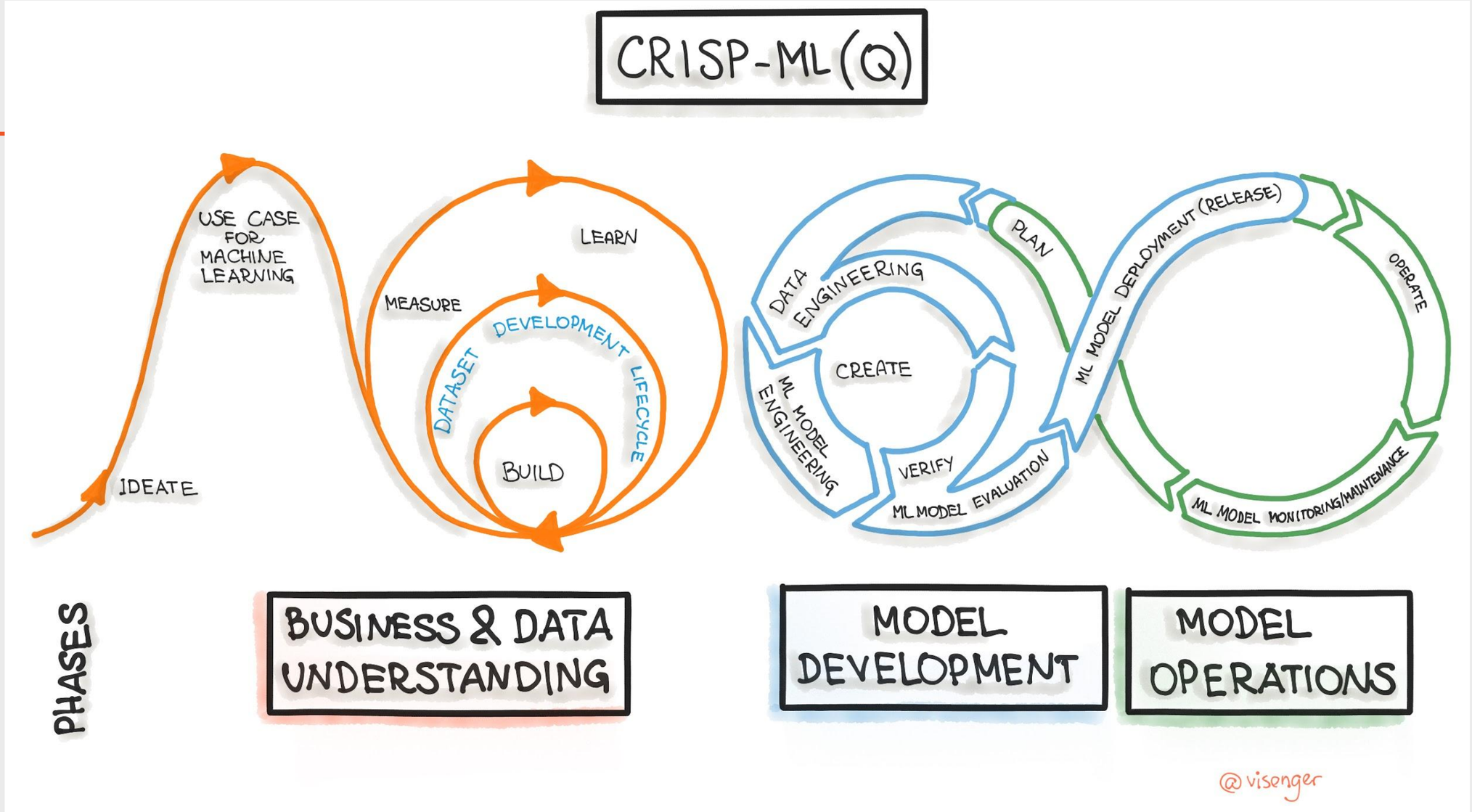
Introduction



Introduction to all the steps



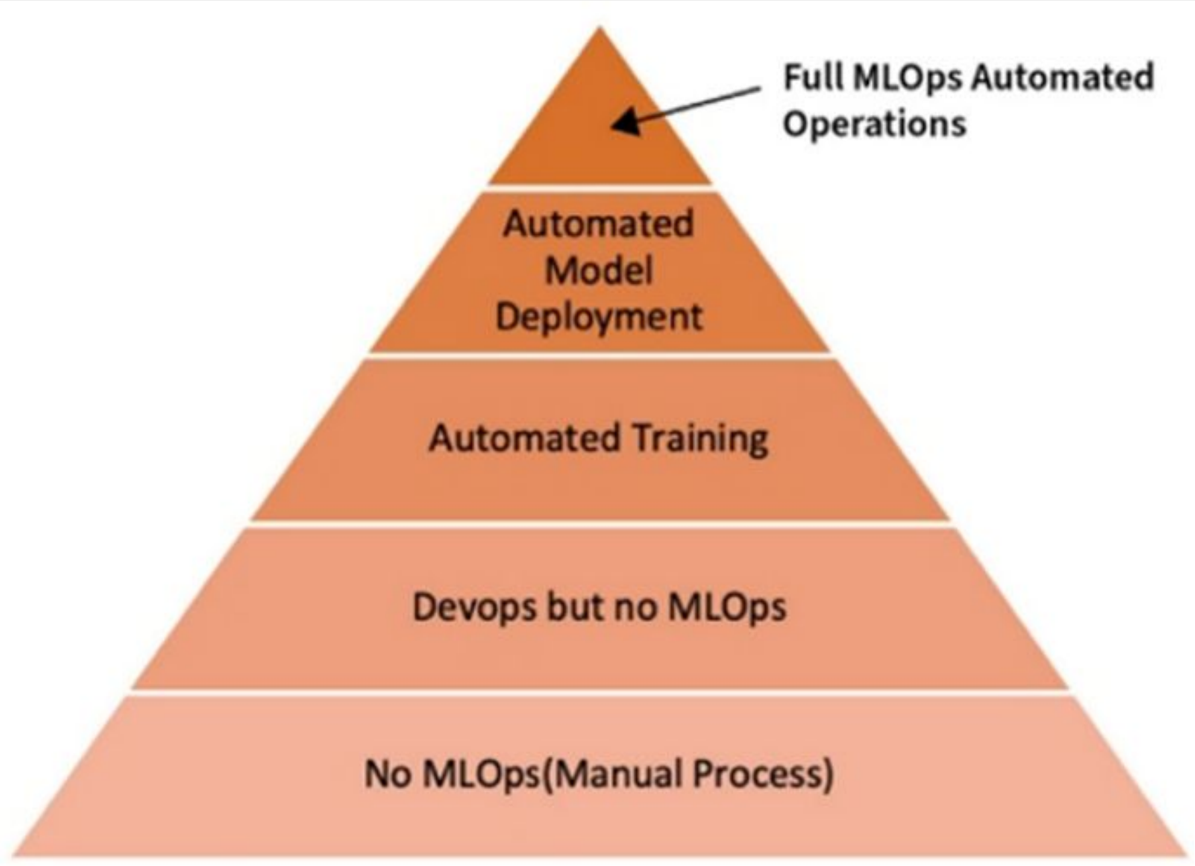
Crisp-ML



https://arxiv.org/abs/2003.05155

https://arxiv.org/abs/2003.05155

Maturity levels of MLOps



https://learn.microsoft.com/en-us/azure/architecture/example-scenario/mlops/mlops-maturity-model
https://blog.onesaitplatform.com/en/2022/04/12/mlops-maturity-levels-and-open-source-tools/

MLOps Challenges

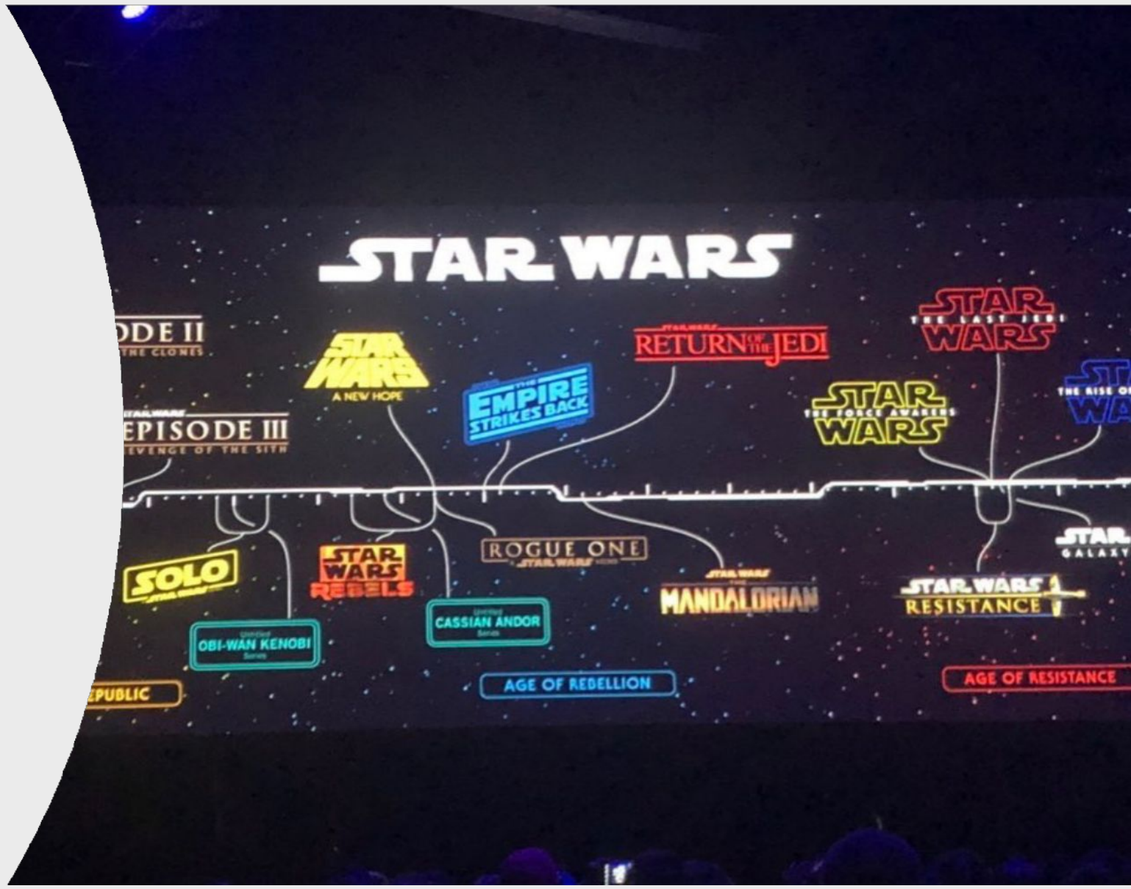
Need for different tools that help along all the ML and data analysis workflow.

Fully integrated environment with NGSI-LD

Some tools:

- **Apache Airflow**
  - Tool to create workflows (similar to NIFI interface) for different tools.
  - Google support.

- **MLFlow**
  - Created by the same authors for Apache Spark.
  - Takes care of the full lifecycle for ML models.



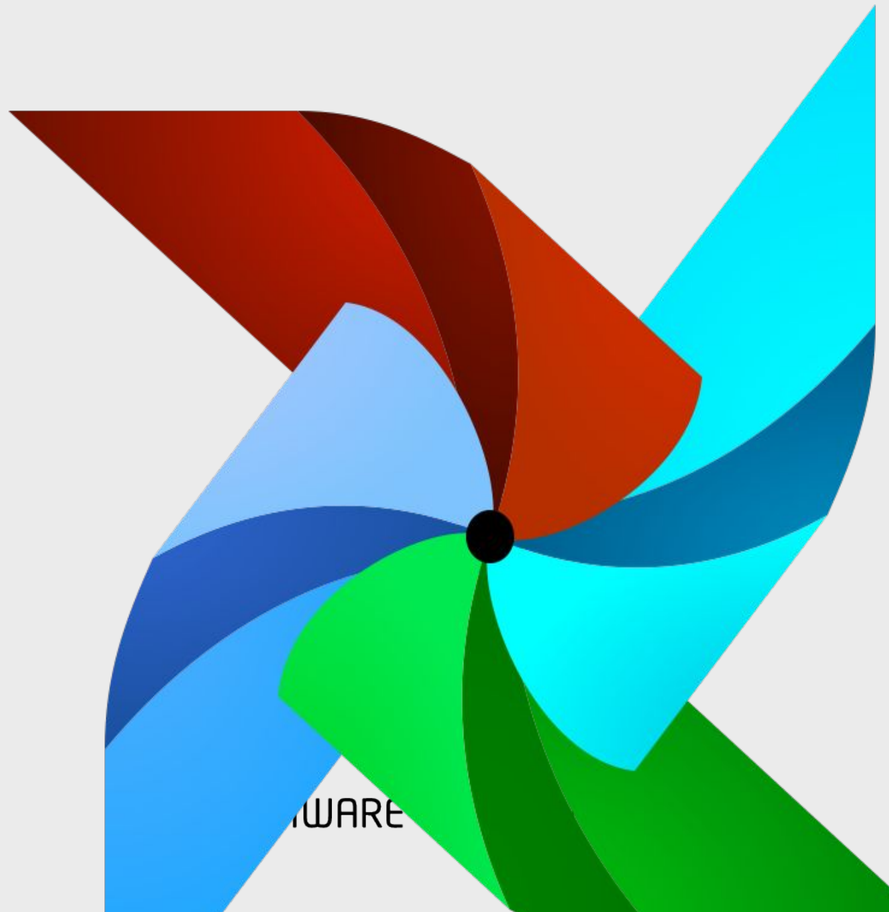
Platforms

Prefect and Apache linkis

Alternatives (Orchestrators)

For DS & ML

Apache Airflow



Origin

- Created by Airbnb in October 2014.
- Airflow is the industry standard open-source tool for programmatic workflow orchestration.
- Try to solve the company's increasingly complex workflows.
- Written in Python, and workflows are created via Python scripts.
- Is designed under the principle of "configuration as code".

- Is open source from the start :
  - https://github.com/apache/airflow
- Complements other workflow tools:
  - Luigi (https://github.com/spotify/luigi) (created at Spotify)
  - Apache Oozie (https://oozie.apache.org/)
  - Azkaban (https://azkaban.github.io) (created at Linkedin)

Use cases

- ETL pipelines
- Machine learning pipelines
- PredicXve data pipelines: fraud detecXon, scoring/ ranking, classificaXon, recommender system, etc.
- General job scheduling: DB back-ups
- Anything… automate the garage door?

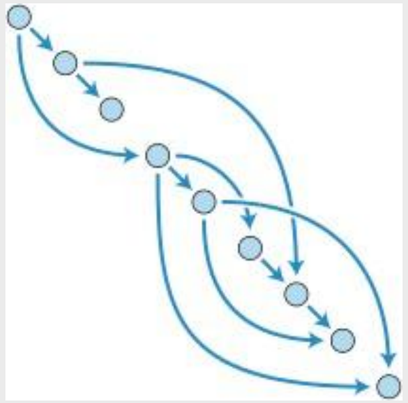
Airflow uses ***Operators*** as the fundamental unit of abstracXon to define tasks, and uses a ***DAG*** to define workflows using a set of operators

MAD · NOV 23-24 · 2018

DAGS

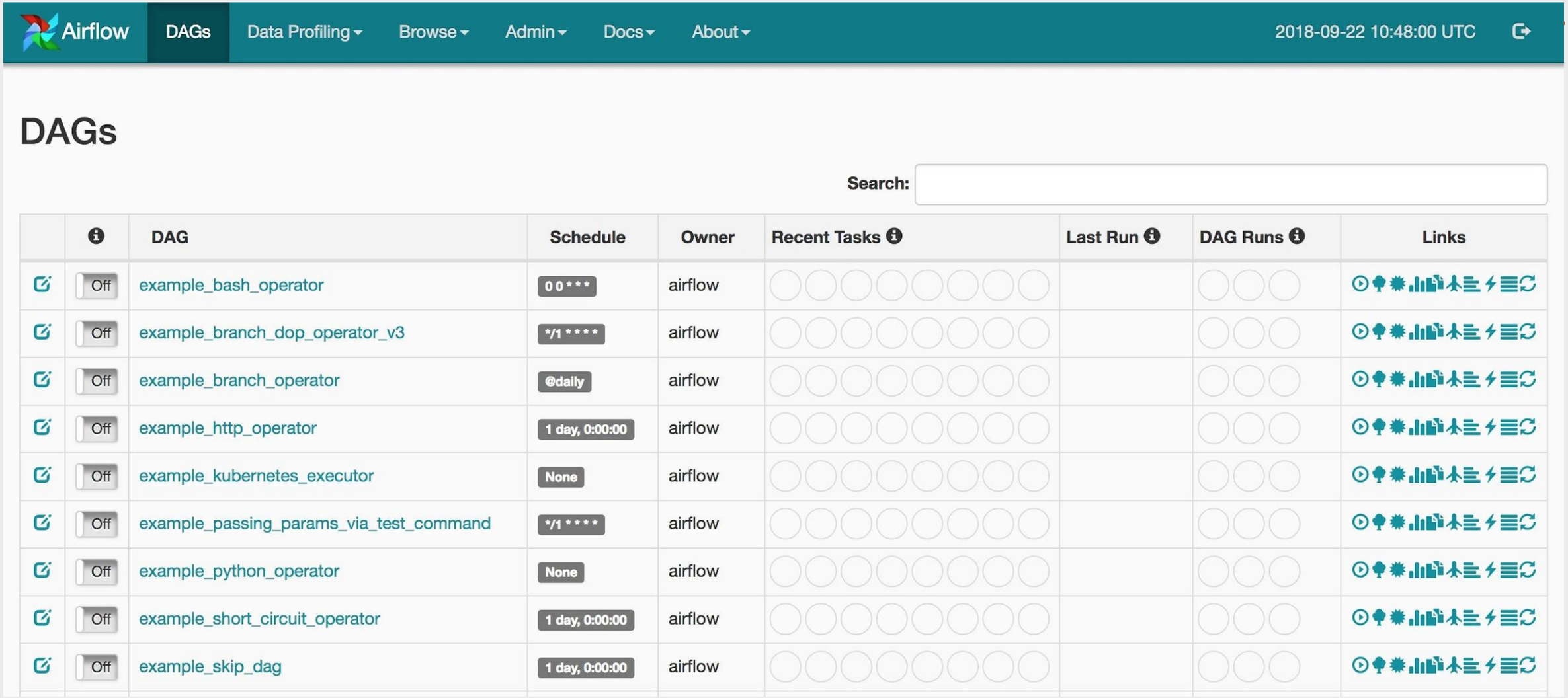
- At the heart of the tool is the concept of a DAG (Directed Acyclic Graph). A DAG is a series of tasks that you want to run as part of your workflow. This might include something like extracting data via a SQL query, performing some calculations with Python and then loading the transformed data into a new table. In Airflow each of these steps would be written as individual tasks in a DAG.
- Airflow enables you to also specify the relationship between the tasks, any dependencies (e.g. data having loaded in a table before a task is run) and the order in which the tasks should be run.
- A DAG is written in Python and saved as a .py file. The DAG_ID is used extensively by the tool to orchestrate the running of the DAG's.

What is a DAG

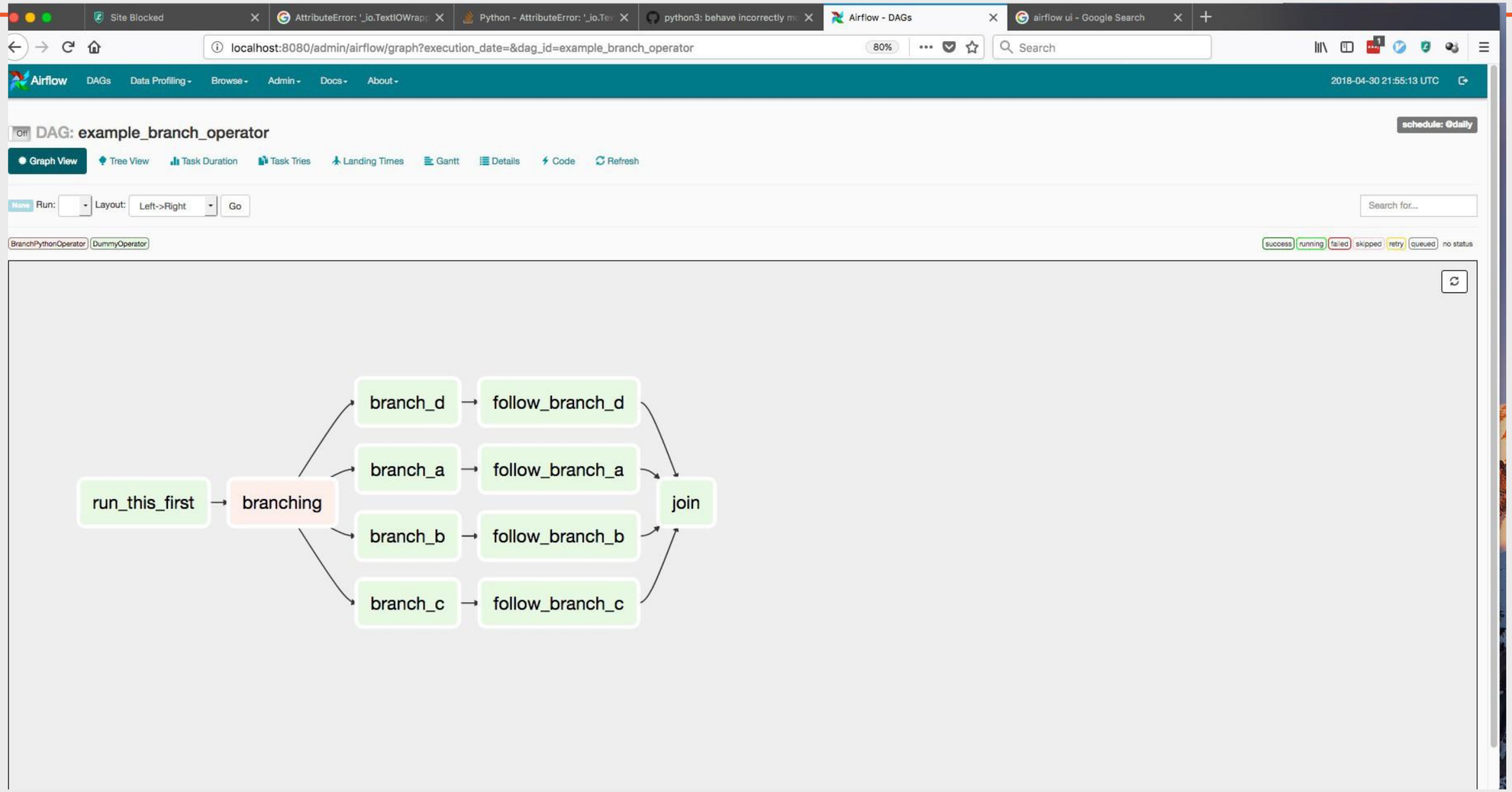


- **D**irected **A**cyclic **G**raph
- Represents a workflow: set of tasks with a dependency structure
- Each node represents some form of

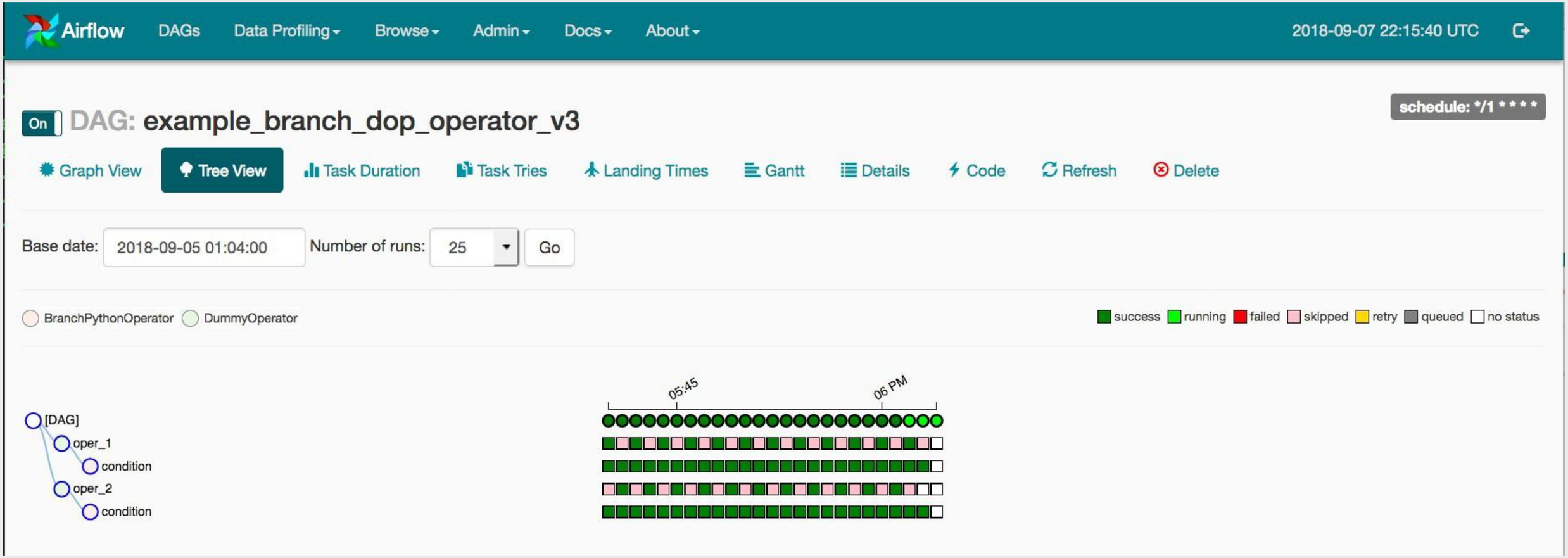
UI Screens – DAGs List



UI Screens – DAGs View



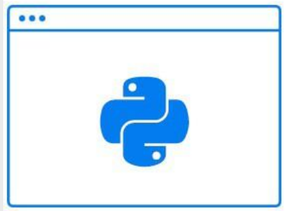
UI Screens – DAGs Runs, Tree View



DAG Code Example

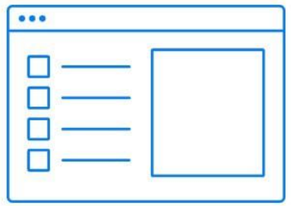
```
import uuid
from datetime import datetime from
airflow import DAG
from airflow.utils.trigger_rule import TriggerRule
from airflow.operators.postgres_operator import PostgresOperator


dag_params = {
        'dag_id': 'PostgresOperator_dag',
        'start_date': datetime(2019, 10, 7),
        'schedule_interval': None
}

with DAG(**dag_params) as dag:

        create_table = PostgresOperator(

                    task_id='create_table',
                sql='''CREATE TABLE new_table(
                        custom_id integer NOT NULL, timestamp TIMESTAMP NOT NULL, user_id VARCHAR (50) NOT NULL
                        );''',
        )

            insert_row = PostgresOperator(
                    task_id='insert_row',
                sql='INSERT INTO new_table VALUES(%s, %s, %s)',
                trigger_rule=TriggerRule.ALL_DONE,
                parameters=(uuid.uuid4().int % 123456789, datetime.now(), uuid.uuid4().hex[:10])
        )

        create_table >> insert_row
```

Main Features



Pure Python

No more command-line or XML black-magic! Use standard Python features to create your workflows, including date time formats for scheduling and loops to dynamically generate tasks. This allows you to maintain full flexibility when building your workflows.



Easy to Use

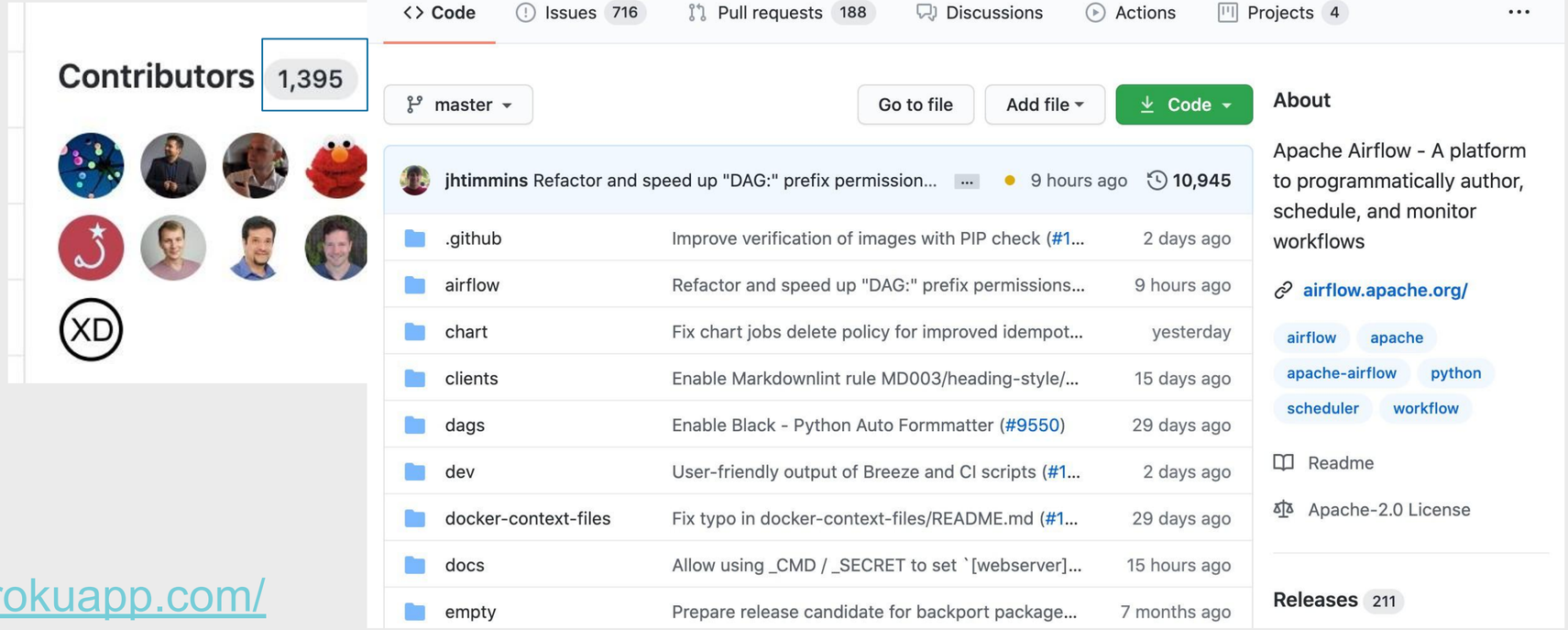
Anyone with Python knowledge can deploy a workflow. Apache Airflow does not limit the scope of your pipelines; you can use it to build ML models, transfer data, manage your infrastructure, and more.

Open Source

Wherever you want to share your improvement you can do this by opening a PR. It's simple as that, no barriers, no prolonged procedures. Airflow has many active users who willingly share their experiences. Have any questions? Check out our buzzing slack.

Apache Airflow Community

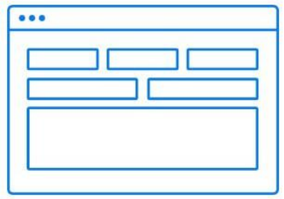
https://github.com/apache/airflow



Official community Slack:
https://apache-airflow-slack.herokuapp.com/

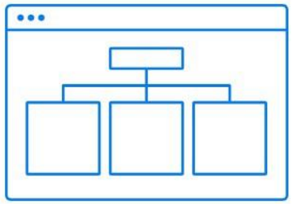
List of committers (maintainers):
https://people.apache.org/committers-by-project.html#airflow (about 40 people)

Main Features



Useful UI

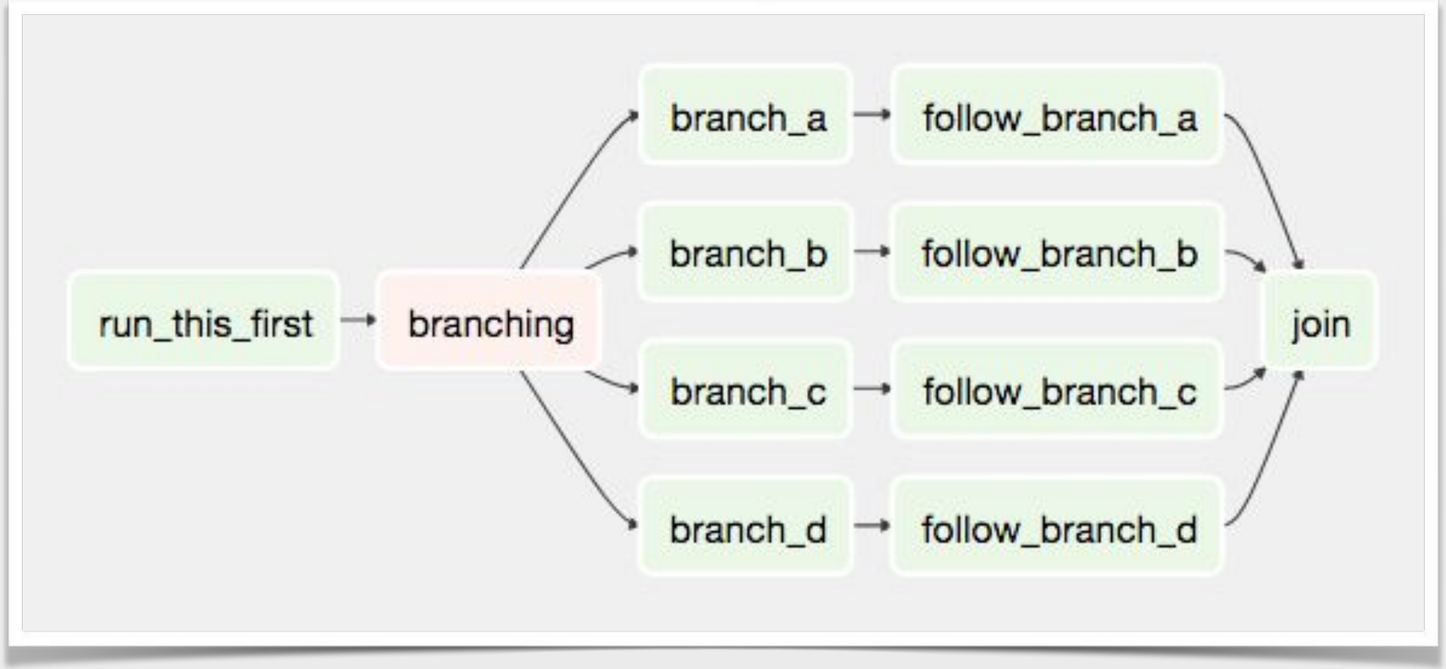
Monitor, schedule and manage your workflows via a robust and modern web application. No need to learn old, cron-like interfaces. You always have full insight into the status and logs of completed and ongoing tasks.



Robust Integrations

Airflow provides many plug-and-play operators that are ready to execute your tasks on Google Cloud Platform, Amazon Web Services, Microsoft Azure and many other third-party services. This makes Airflow easy to apply to current infrastructure and extend to next-gen technologies.

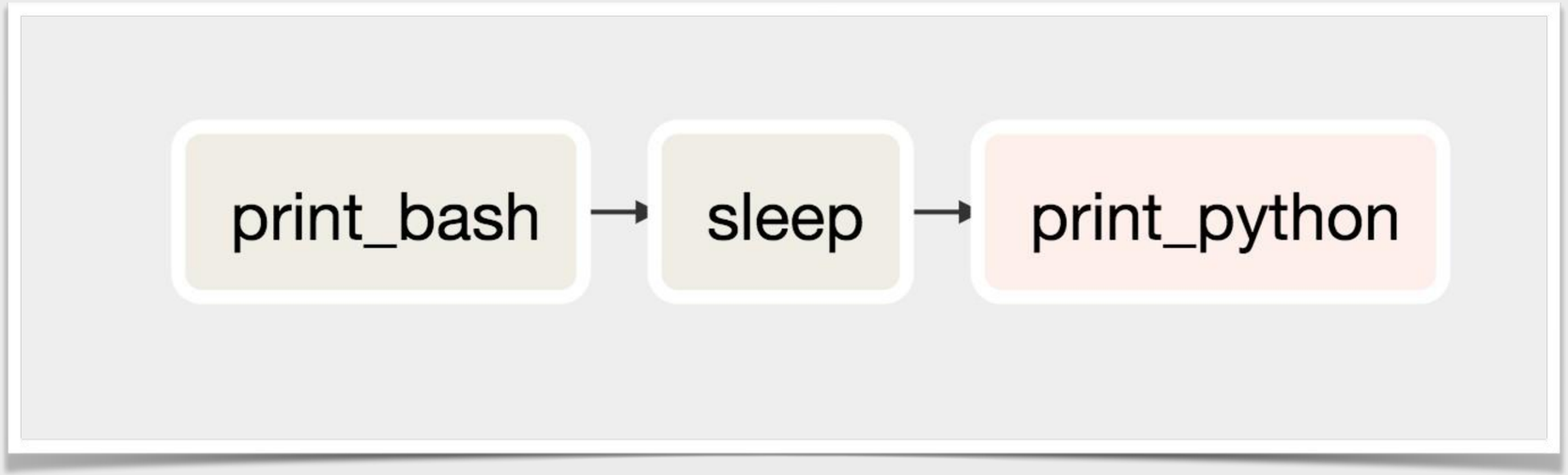
What does it look like?



err...

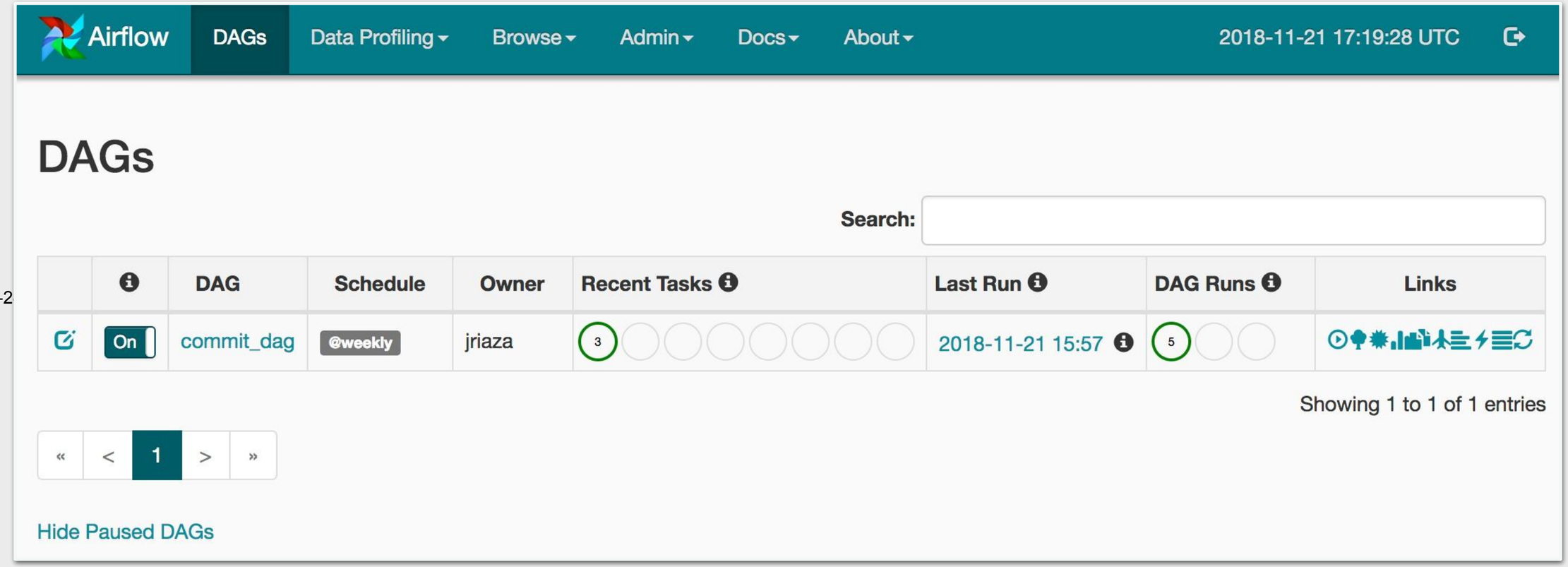

log_cx_ticket_user_issues
staging_cx_tickets_summary_2
staging_cx_tickets_summary_3
log_cx_ticket_reservation_code_changes
log_cx_ticket_assignments
log_cx_ticket_inbox_id_changes
log_cx_ticket_priority_changes
wf_zendesk_ticket_comments
dq_zendesk_audit_history
wf_reservation2s_summary
zendesk_audit_history
nps
wf_medallia_nps_results
log_cx_ticket_status_changes
log_cx_ticket_tag_changes
reservations_summary_for_cx_summary
wf_reservation2s
zendesk_ticket_solvers
wf_zendesk_tickets
wf_cs_satisfaction_surveys
fct_cx_ticket_channel
fct_cx_last_solve
fct_cx_ticket_comments
users_summary_for_cx_summary
staging_cx_tickets_summary_1
fct_cx_ticket_solves
cx_nps_summary
genesys_bash
genesys_hive
cx_tickets_summary
contact_rate_metrics
agent_performance_for_lantern
cx_dashboard_producerboard
fct_cx_agent_closes
cx_dashboard_user_issues
wf_lantern_sessions
dim_cx_solves
lead_dash
email_done
fct_cx_ticket_opens
dim_cx_roster
varsha_dash
fct_cx_ticket_closes

How it’s made

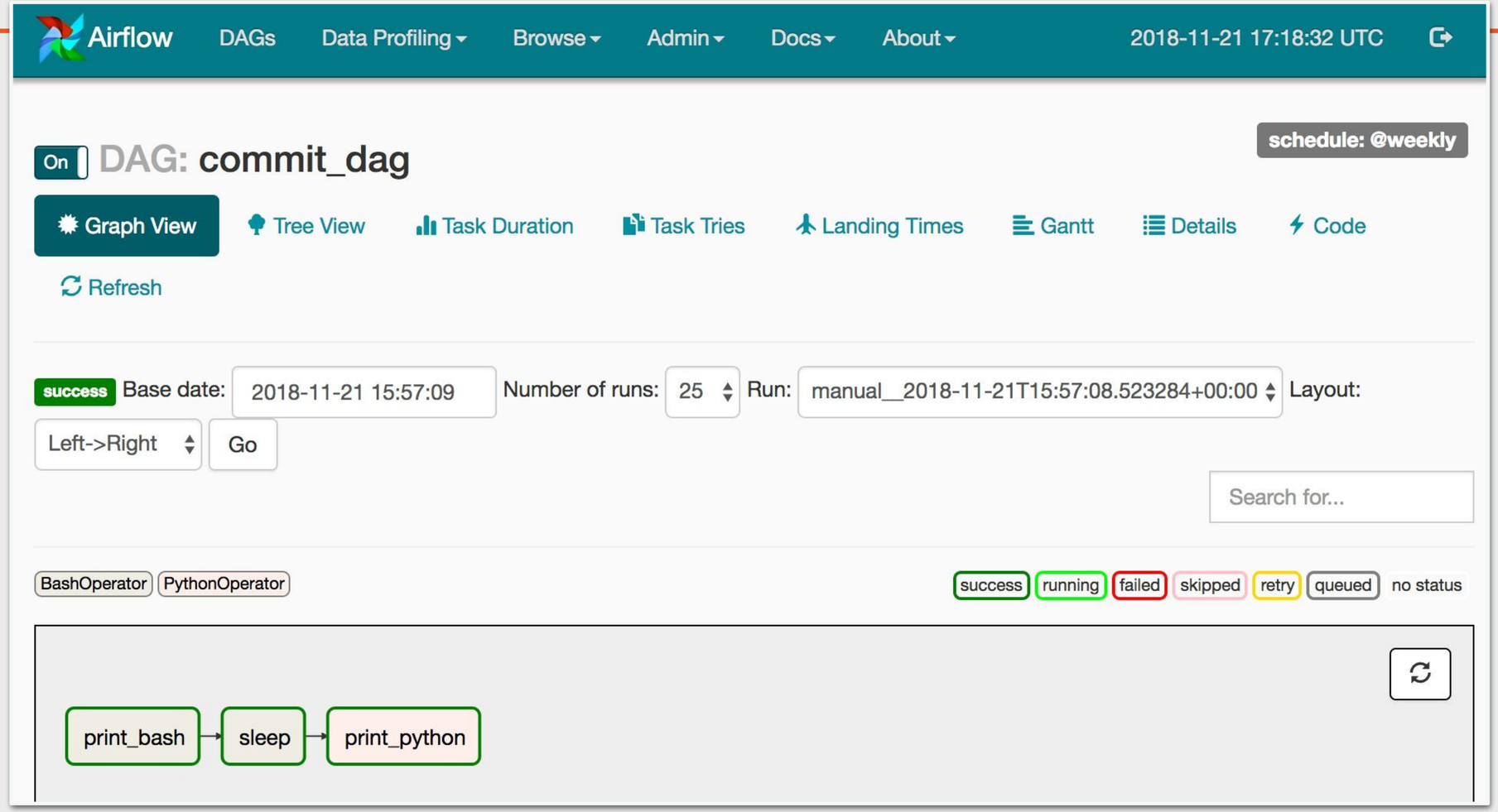


```
import datetime
from airflow import DAG
from airflow.operators.bash_operator import BashOperator from
airflow.operators.python_operator import PythonOperator

def print_commit(): print('    commit
        conf')

with DAG('commit_dag', schedule_interval='@weekly') as dag: print_bash =
        BashOperator(
                task_id='print_bash', bash_command='echo " sleep =      commit conf"')
        BashOperator(
                task_id='sleep', bash_command='sleep 5')
        print_python = PythonOperator(
                task_id='print_python', python_callable=print_commit)

        print_bash >> sleep >> print_python
```

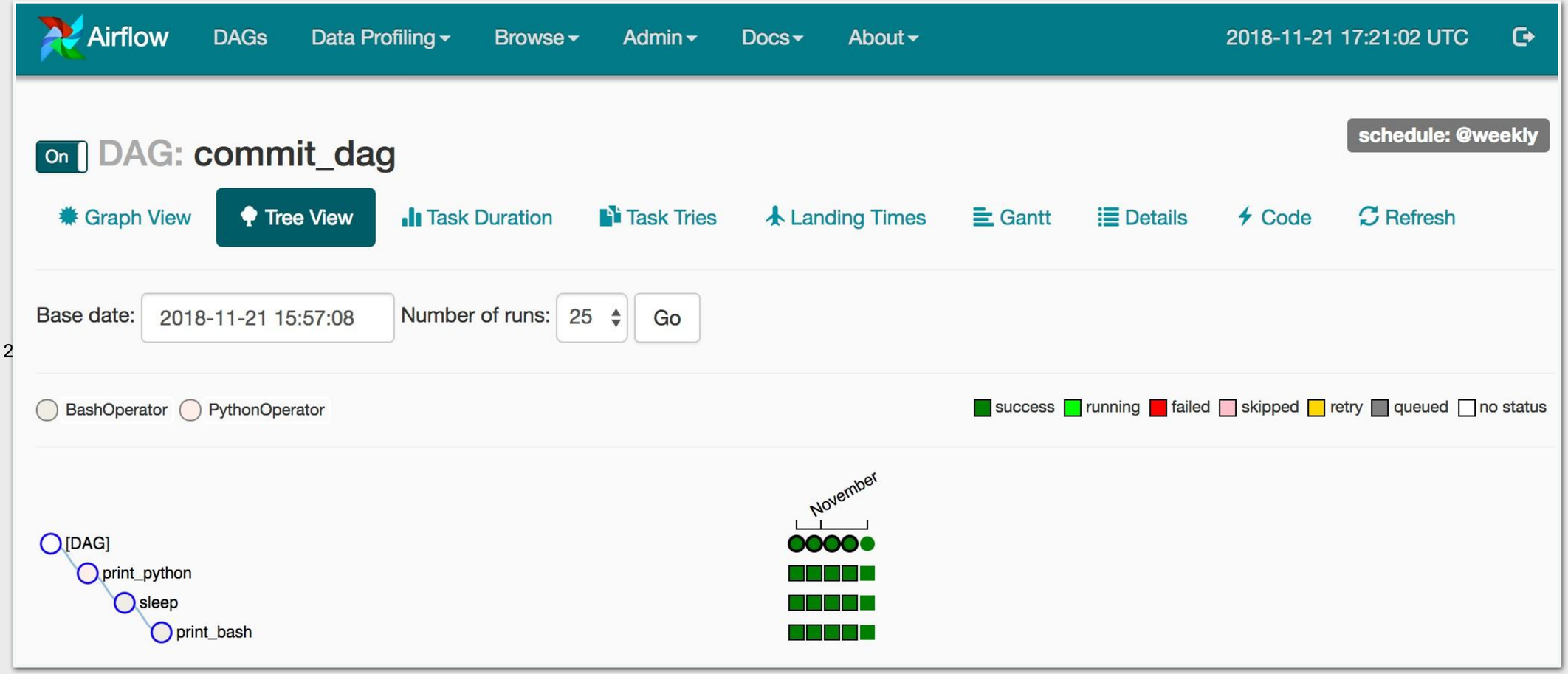
Airflow UI


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2018-11-21 17:19:28 UTC
DAGs
Search:
DAG
Schedule
Owner
Recent Tasks
Last Run
DAG Runs
Links
On
commit_dag
@weekly
jriaza
3
2018-11-21 15:57
5
Showing 1 to 1 of 1 entries
«
<
1
>
»
Hide Paused DAGs

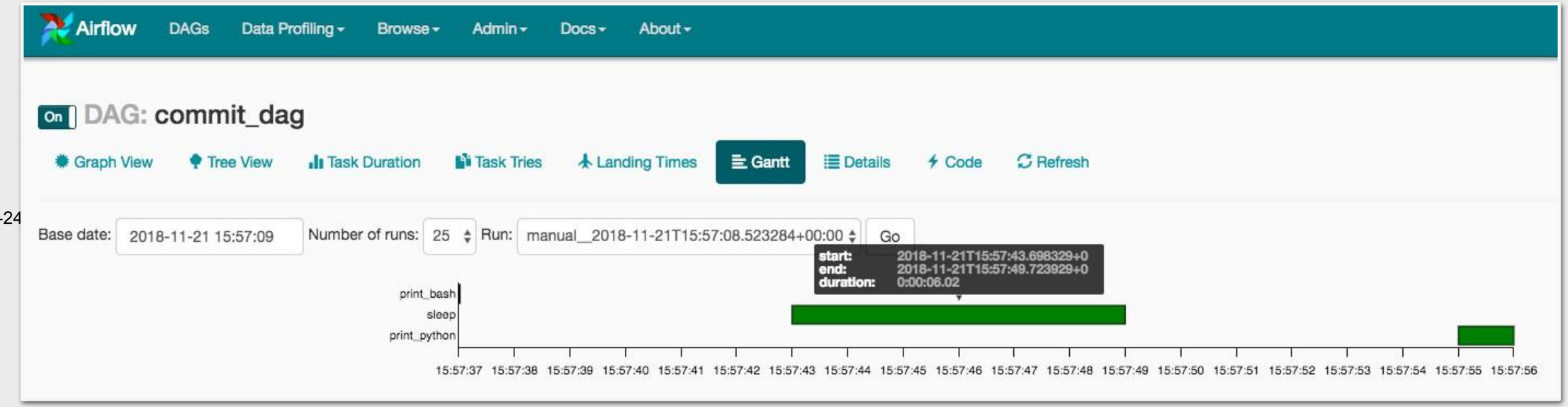
Airflow UI


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2018-11-21 17:18:32 UTC
On
DAG: commit_dag
schedule: @weekly
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
success
Base date:
2018-11-21 15:57:09
Number of runs:
25
Run:
manual__2018-11-21T15:57:08.523284+00:00
Layout:
Left->Right
Go
Search for...
BashOperator
PythonOperator
success
running
failed
skipped
retry
queued
no status
print_bash
sleep
print_python

Airflow UI


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2018-11-21 17:21:02 UTC
On
DAG: commit_dag
schedule: @weekly
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Base date:
2018-11-21 15:57:08
Number of runs:
25
Go
BashOperator
PythonOperator
success
running
failed
skipped
retry
queued
no status
November
[DAG]
print_python
sleep
print_bash

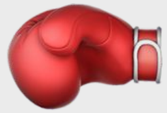
Airflow UI


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
On
DAG: commit_dag
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Base date: 2018-11-21 15:57:09
Number of runs: 25
Run: manual__2018-11-21T15:57:08.523284+00:00
Go
start: 2018-11-21T15:57:43.698329+0
end: 2018-11-21T15:57:49.723929+0
duration: 0:00:06.02
print_bash
sleep
print_python
15:57:37 15:57:38 15:57:39 15:57:40 15:57:41 15:57:42 15:57:43 15:57:44 15:57:45 15:57:46 15:57:47 15:57:48 15:57:49 15:57:50 15:57:51 15:57:52 15:57:53 15:57:54 15:57:55 15:57:56

Operators

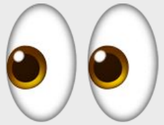
- **Action**: perform an aclon locally or make a call to an external system to perform another actioon
- **Transfer**: move data from one system to another
- **Sensor**: wait for and detect some condition in a source system

Action operators



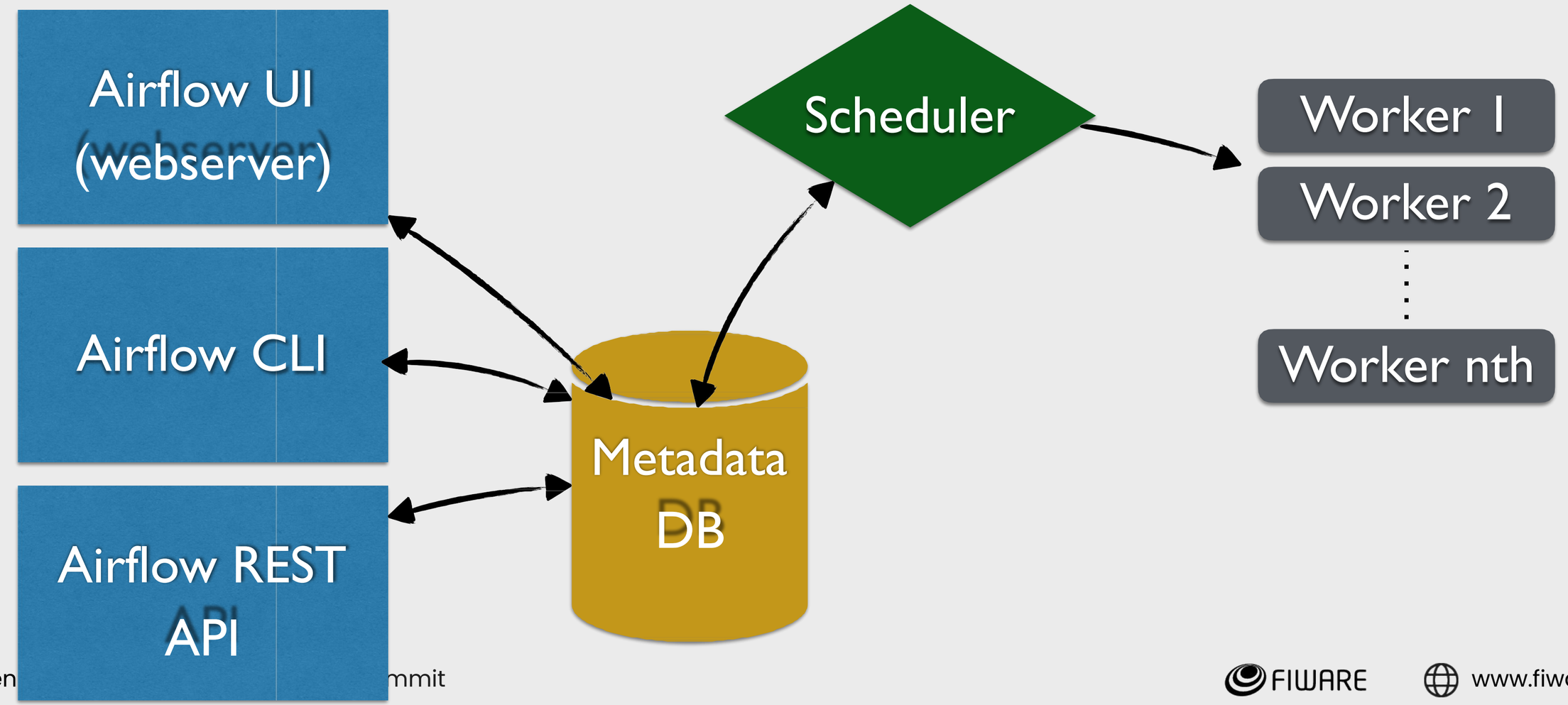
- Perform an action such as executing a Python function or submiting a Spark Job
- **Built-in** BashOperator, PythonOperator, DockerOperator, EmailOperator, …

Sensor operators

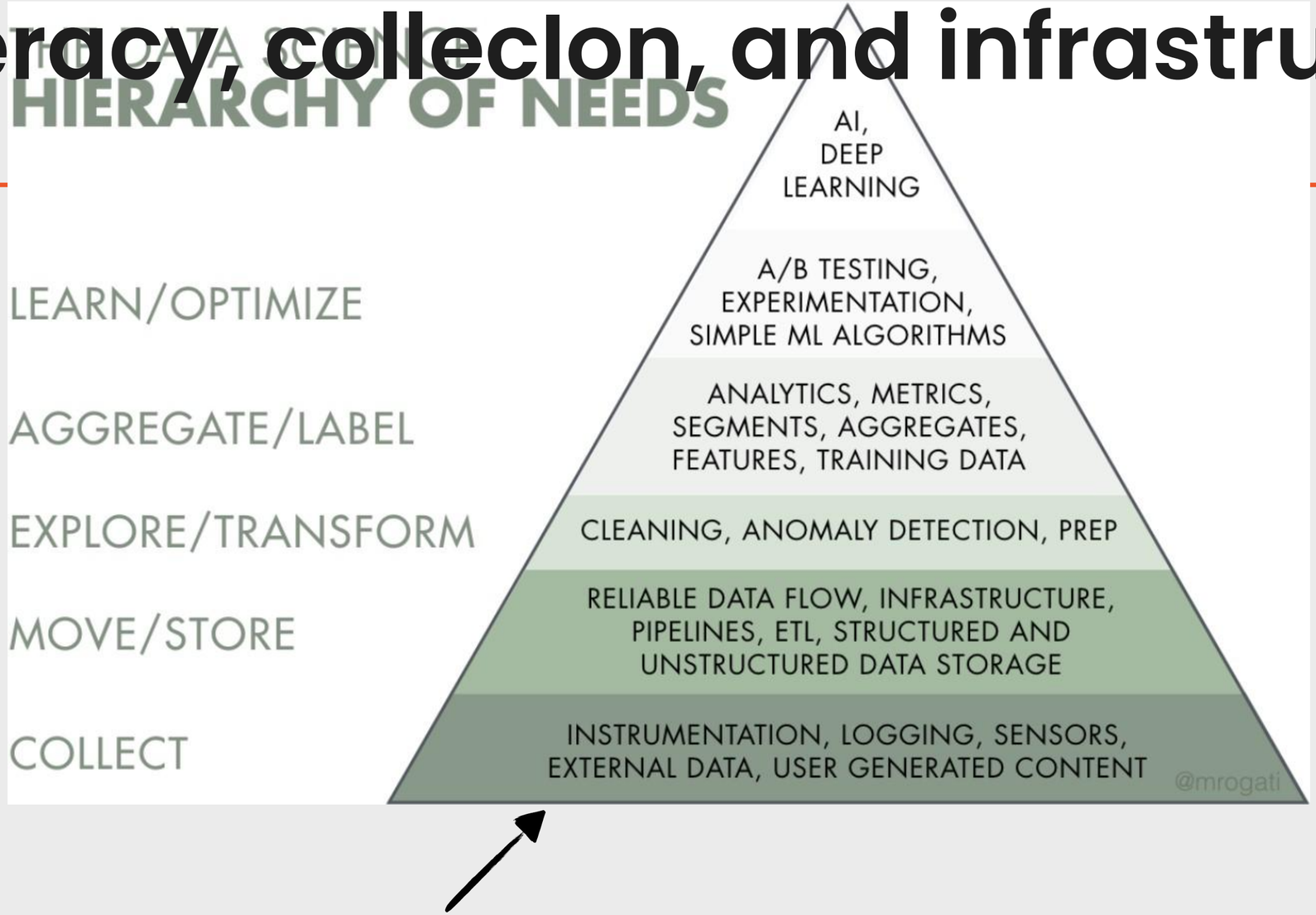


- Triggers downstream tasks in the dependency graph when a certain criteria is met. For example, checking for a certain file has become available on S3 before using it downstream
- **Built-in** HivePartitionSensor, HdfsSensor, S3KeySensor,

The big picture



Data literacy, collection, and infrastructure



Airflow deployment

MAD · NOV 23-24 · 2018

https://github.com/idealista/airflow-role

Astronomer

- https://www.astronomer.io

- Tool to create Airflow deployments easily.

  curl -sSL https://install.astronomer.io | sudo bash

  astro dev init #(genera scaffold)

  astro dev start

  Web interface en puerto 8080

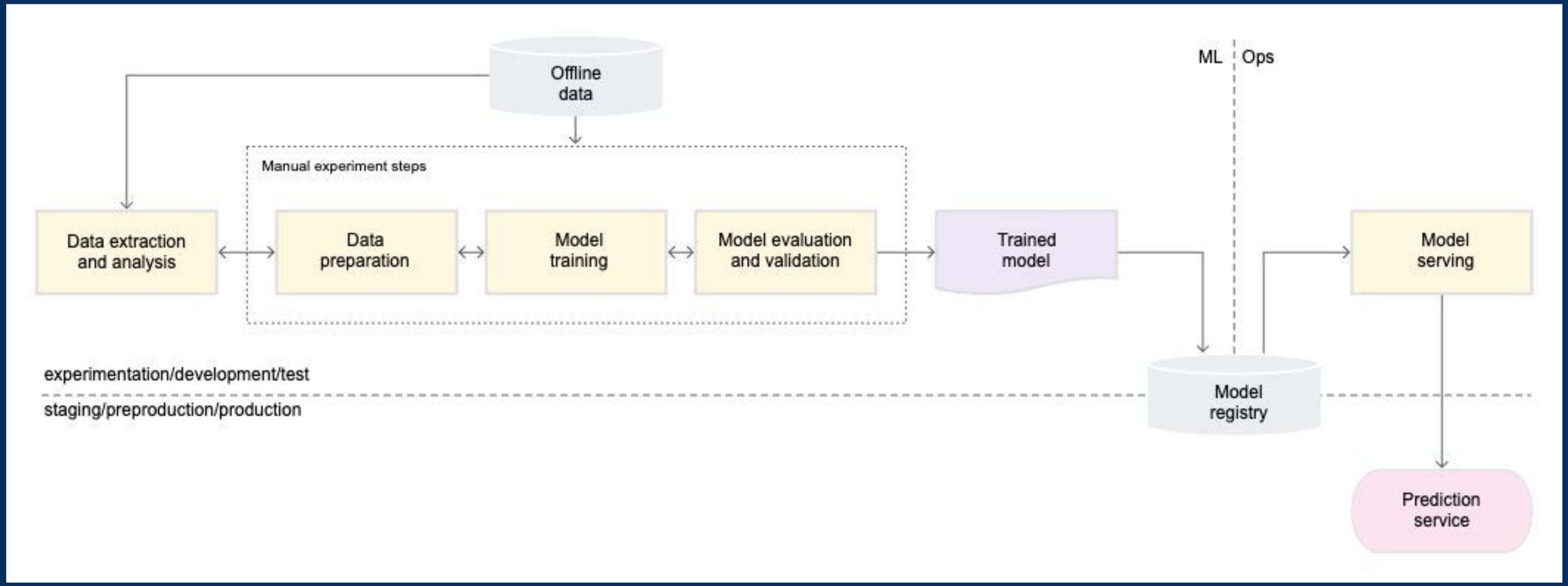
The Maturity of the MLOPS Process

- *MLOps* is an ML engineering culture and practice that aims at unifying ML system development (Dev) and ML system operation (Ops).

level 0	**Manual process**
level 1	**ML pipeline automation**
level 2	**CI/CD pipeline automation**

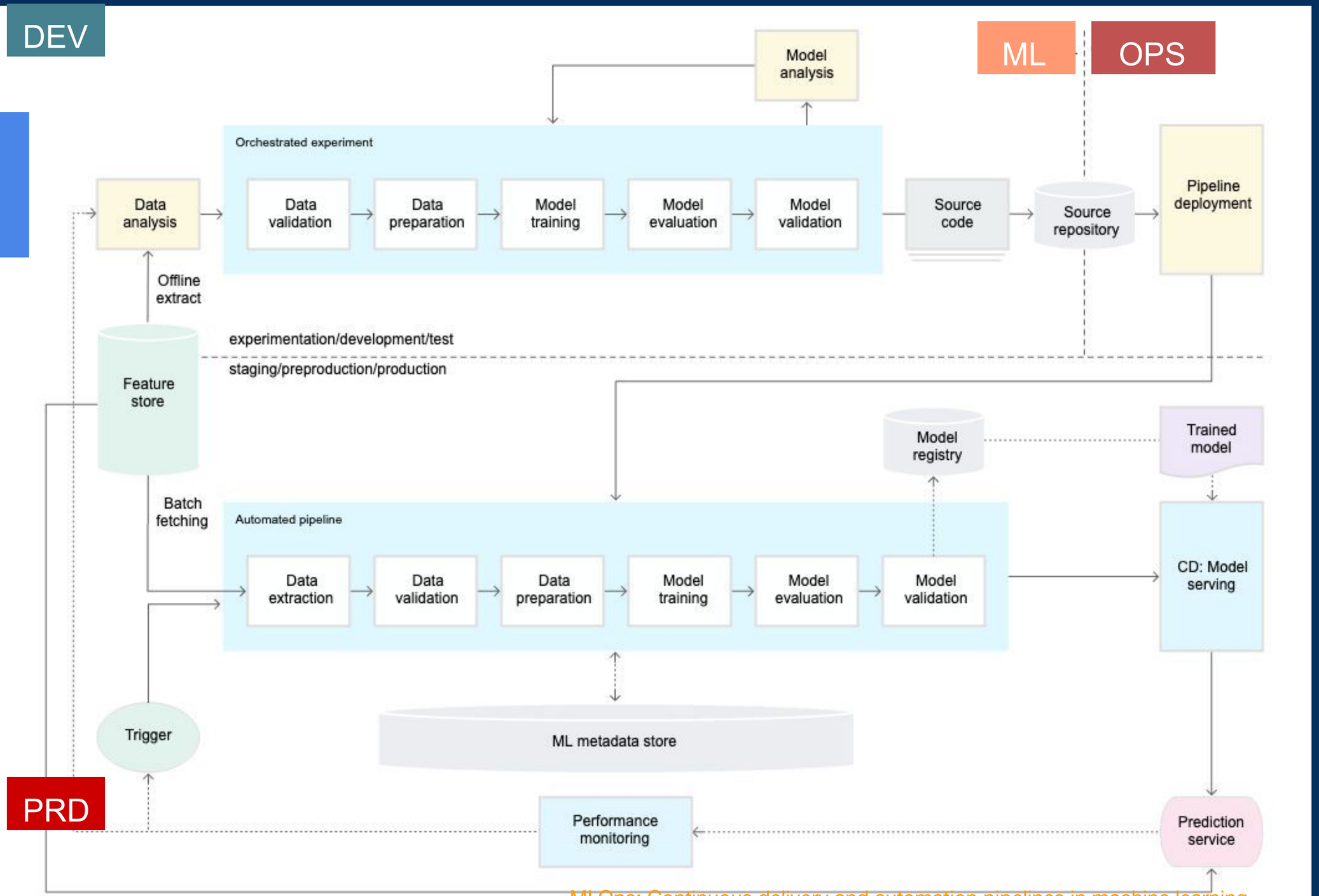
- Practicing MLOps means that you advocate for automation and monitoring at all steps of ML system construction, including integration, testing, releasing, deployment and infrastructure management.

MLOps level 0: Manual process



MLOps: Continuous delivery and automation pipelines in machine learning

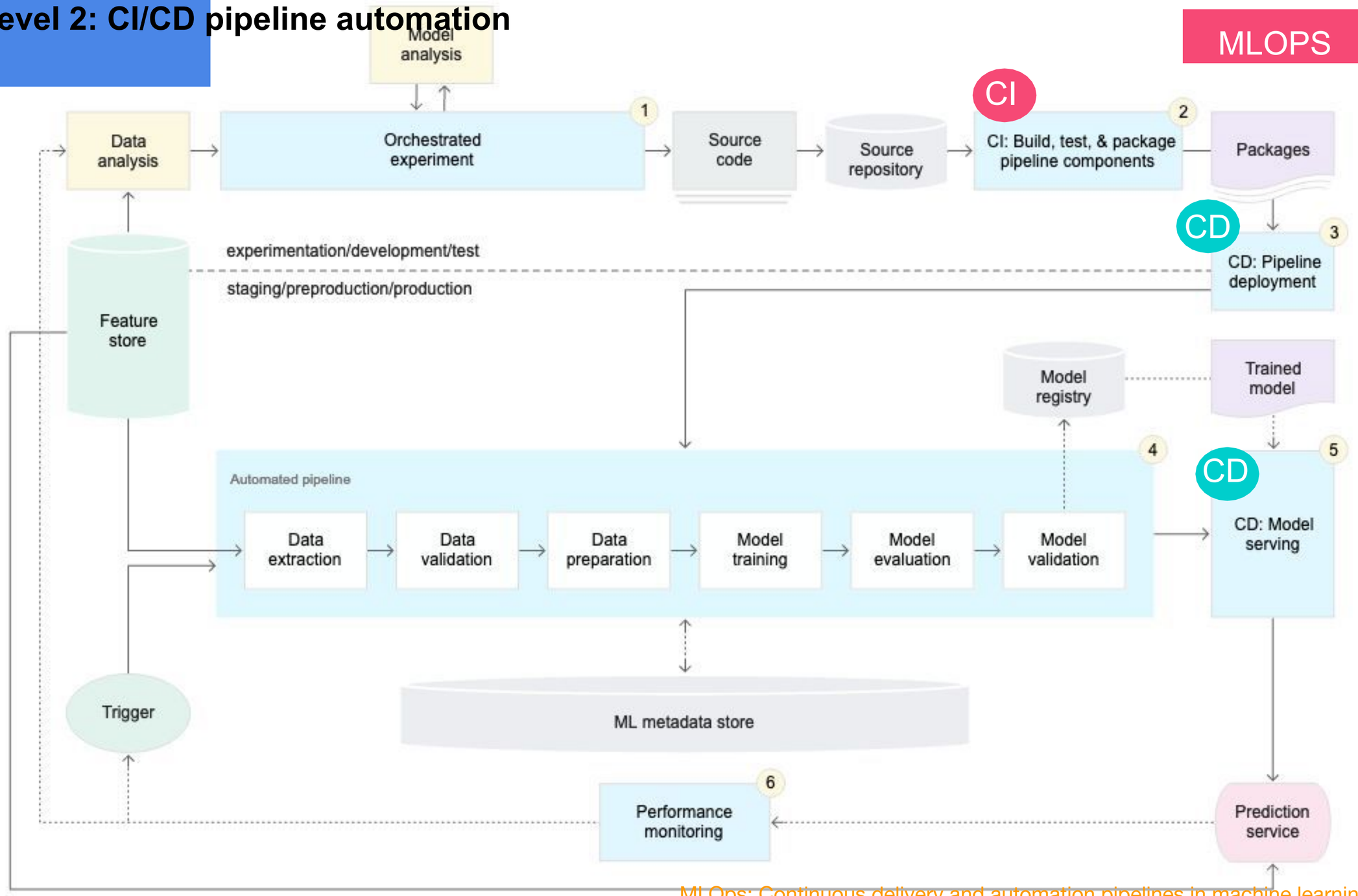
MLOps level 1: ML pipeline automation



MLOps: Continuous delivery and automation pipelines in machine learning

MLOps level 2: CI/CD pipeline automation

MLOPS



MLOps: Continuous delivery and automation pipelines in machine learning

Experiment Tracking

Model development & Post-Deployment

- **Prove value of experiment**
  - Need baseline to show and compare
- **Collaborate**
  - Need to refer and access models and artifacts from other members
- **Reproduce work**
  - Need same parameters and model of ex-run

What we should log/track in ML

Log day-to-day work in ML life cycle

- Hyper parameters
- Training/modeling performances
- Model
  - Type
  - Building environment
  - Modeling version
- and so on

parameters

- Convolutional filter
- Kernel_size
- Max pooling
- Dropout
- Dense
- Batch_size
- Epochs
- ….

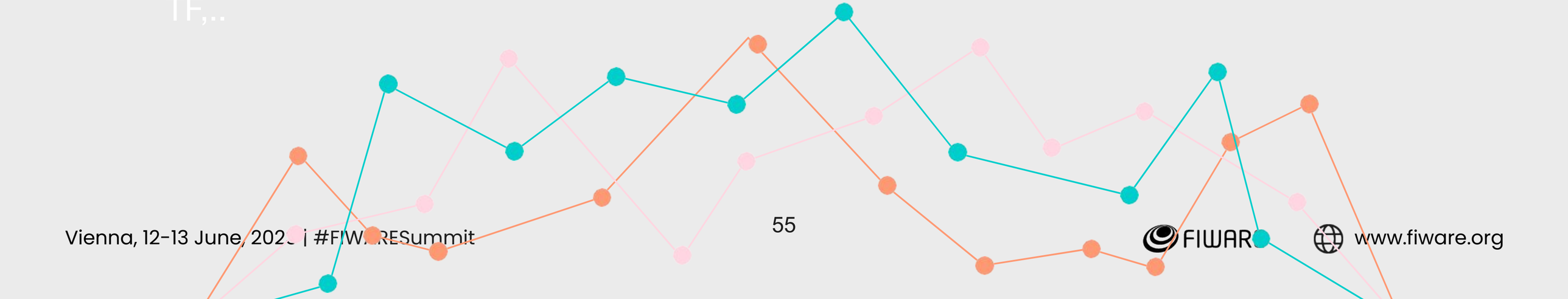
Evaluation metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RSME)
- R-squared (r2)
- ...

https://github.com/mlflow

mlflow

- Since 2018 from DataBricks (Main contributor)
- An open platform for the machine learning lifecycle
- Python Library; runs locally and on the cloud
- Built-in UI for experiment visualization
- Logging integrations for major frameworks: scikit-learn, PyTorch, TF,..



mlflow Components

MLflow is an open source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow currently offers four components:

MLflow Tracking

Record and query experiments: code, data, config, and results

Read more

MLflow Projects

Package data science code in a format to reproduce runs on any platform

Read more

MLflow Models

Deploy machine learning models in diverse serving environments

Read more

Model Registry

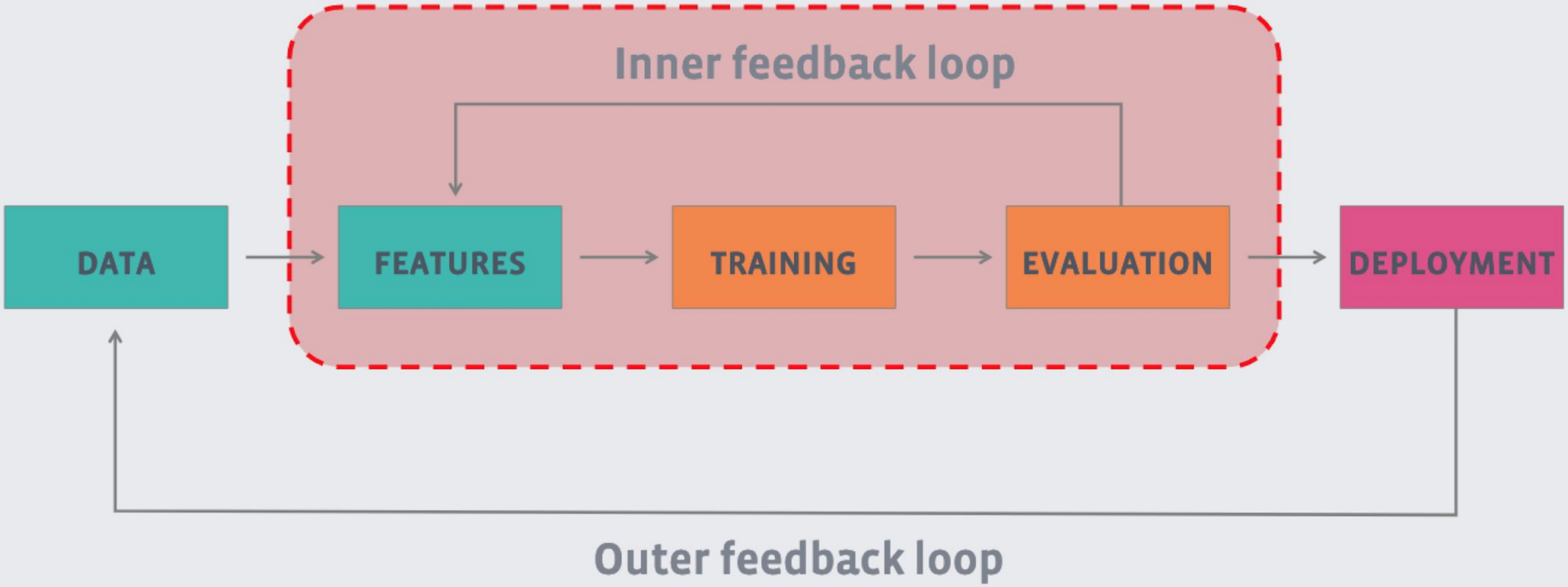
Store, annotate, discover, and manage models in a central repository

Read more

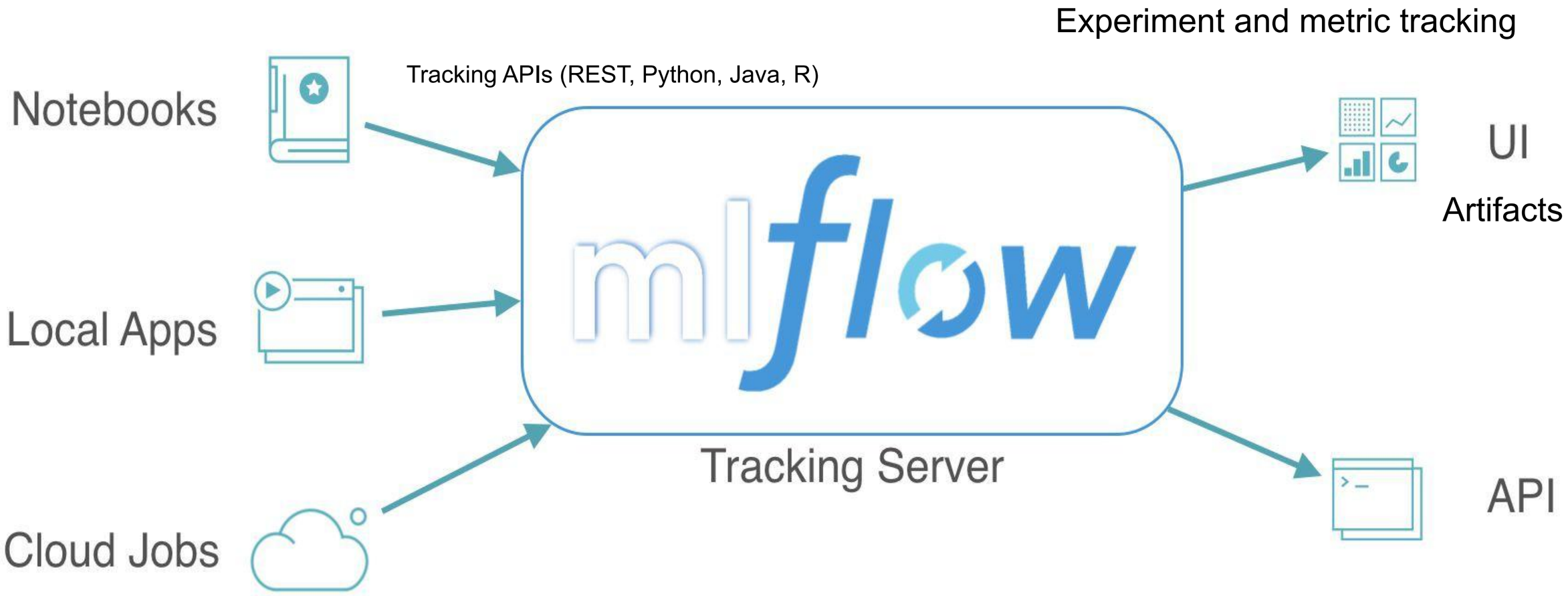
Main focus of this sharing!

MLflow Components

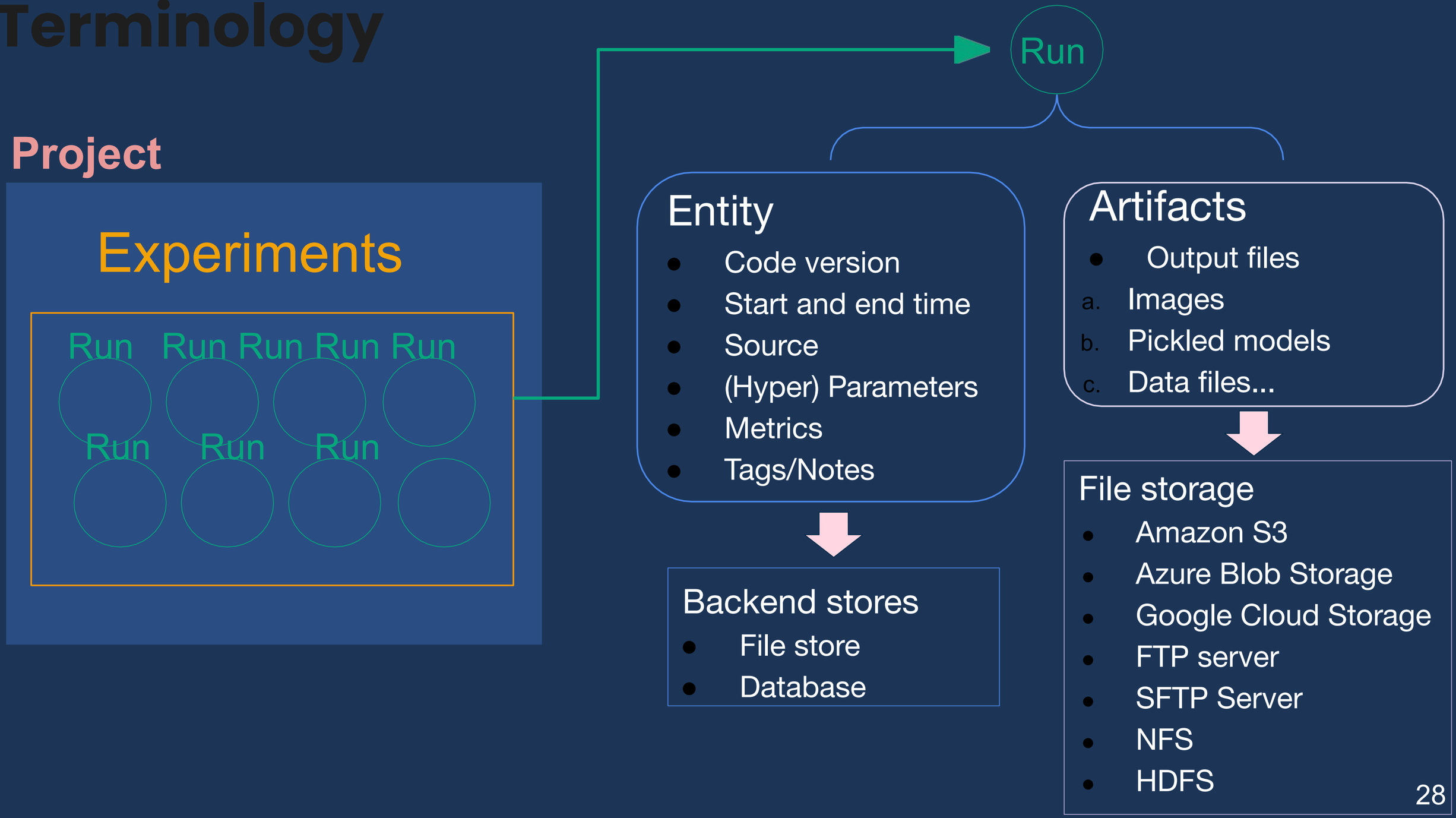
Experiment tracking



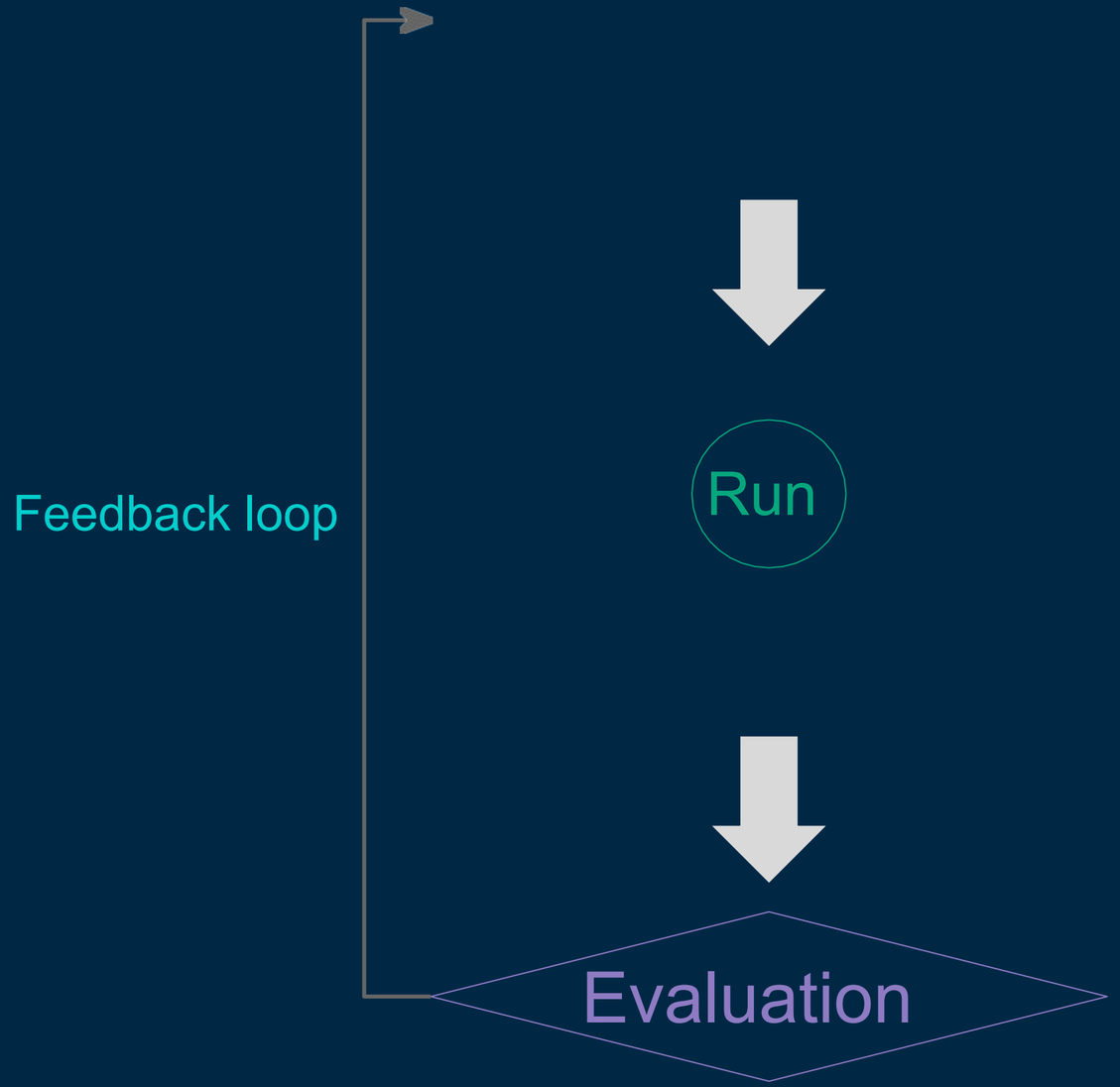
Experiment/Production pipeline



Terminology



Experiments



Setup & Initialize

- Data preparation
- Setup MLflow experiment

Train & Inference

- Start a Run
- Log (Hyper)parameter
- Log metrics
- Log artifact
- Log model

Post-analysis

- Compare runs/ Tuning (parameters/models)

MLflow Tracking for ML Development

Tracking.API

- start_Run()
- log_param()
- log_metric()
- log_artifact()
- end_Run()

Output

- Parameters
- Metrics
- Output file
  - Artifact
- Code version
- ...

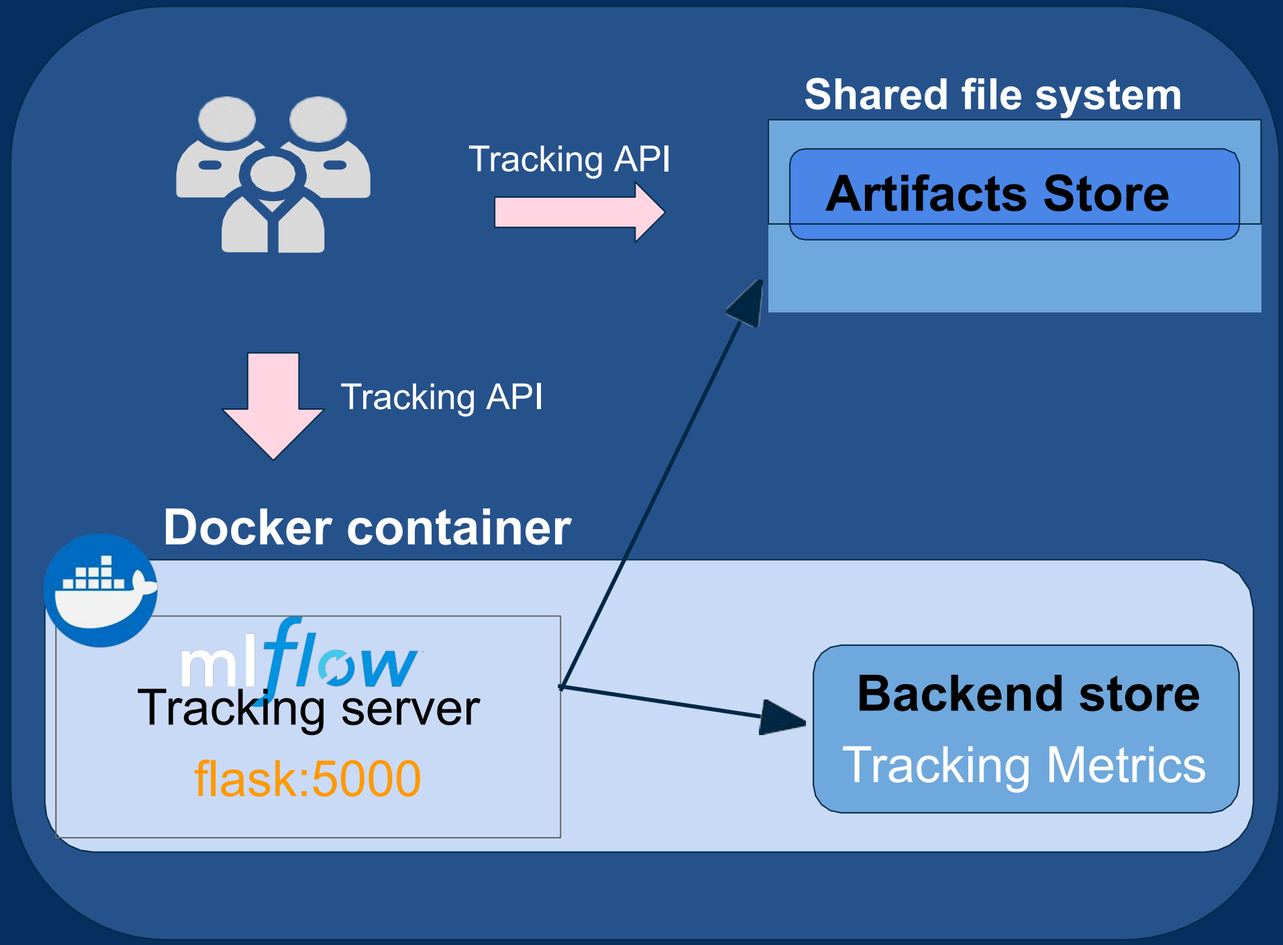
- log_model

- Model

Example Architecture

```
mlflow server \
    --backend-store-uri $FILE_STORE \
    --default-artifact-root $ARTIFACT_STORE \
    --host $SERVER_HOST \
    --port $SERVER_PORT
```

```
docker run -d -p 5000:5000 \
        -v /tmp/artifactStore:/tmp/mlflow/artifactStore \
        --name mlflow-tracking-server \
        suci/mlflow-tracking
```



https://github.com/sucitw/mlflow_tracking

Tracking- Experiments

Setup & Initialize

```
# Setup & Initialize MLflow experiment
 experiment_name = "PyconTW 2020 Demo "
 tracking_server = "http://localhost:5000"

 mlflow.set_tracking_uri(tracking_server)
 mlflow.set_experiment(experiment_name)
```

```
#System Env setting

 #backend-store-uri
    $FILE_STORE
 #default-artifact-root
    $ARTIFACT_STORE
 #host
    $SERVER_HOST
 #port
    $SERVER_PORT
```

Tracking- Run

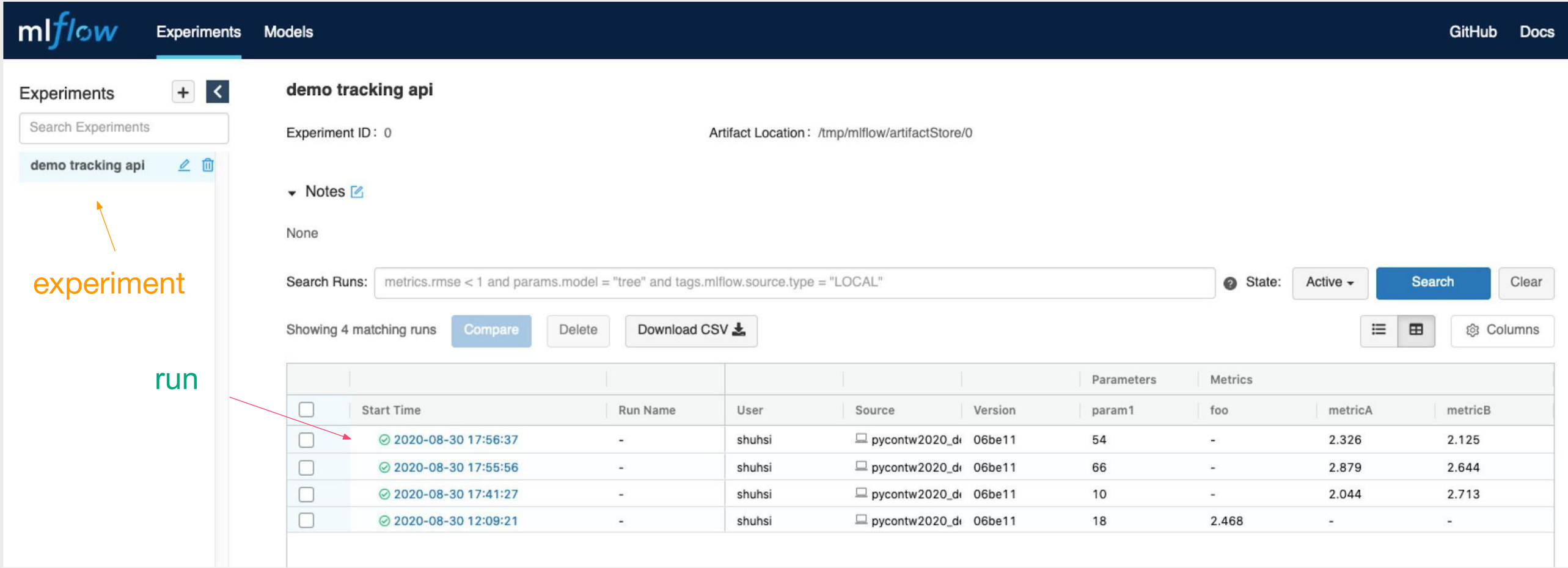
Train & Inference

```
with mlflow.start_run() as run:

    # Log a parameter (key-value pair)
    log_param("param1", randint(0, 100))

    # Log a metric; metrics can be updated throughout the run
    log_metric("metricsA", random())
    log_metric("metricsA", random() + 1)
    log_metric("metricsA", random() + 2)
    log_metric("metricsB", random() + 2)

    # Log an artifact (output file)
    with open("outputs/test.txt", "w") as f:
        f.write("hello world! Run id:{}".format(type(mlflow.active_run().info)))

    log_artifacts("outputs")
```

Log (Hyper)parameter

Log metrics

Log artifact



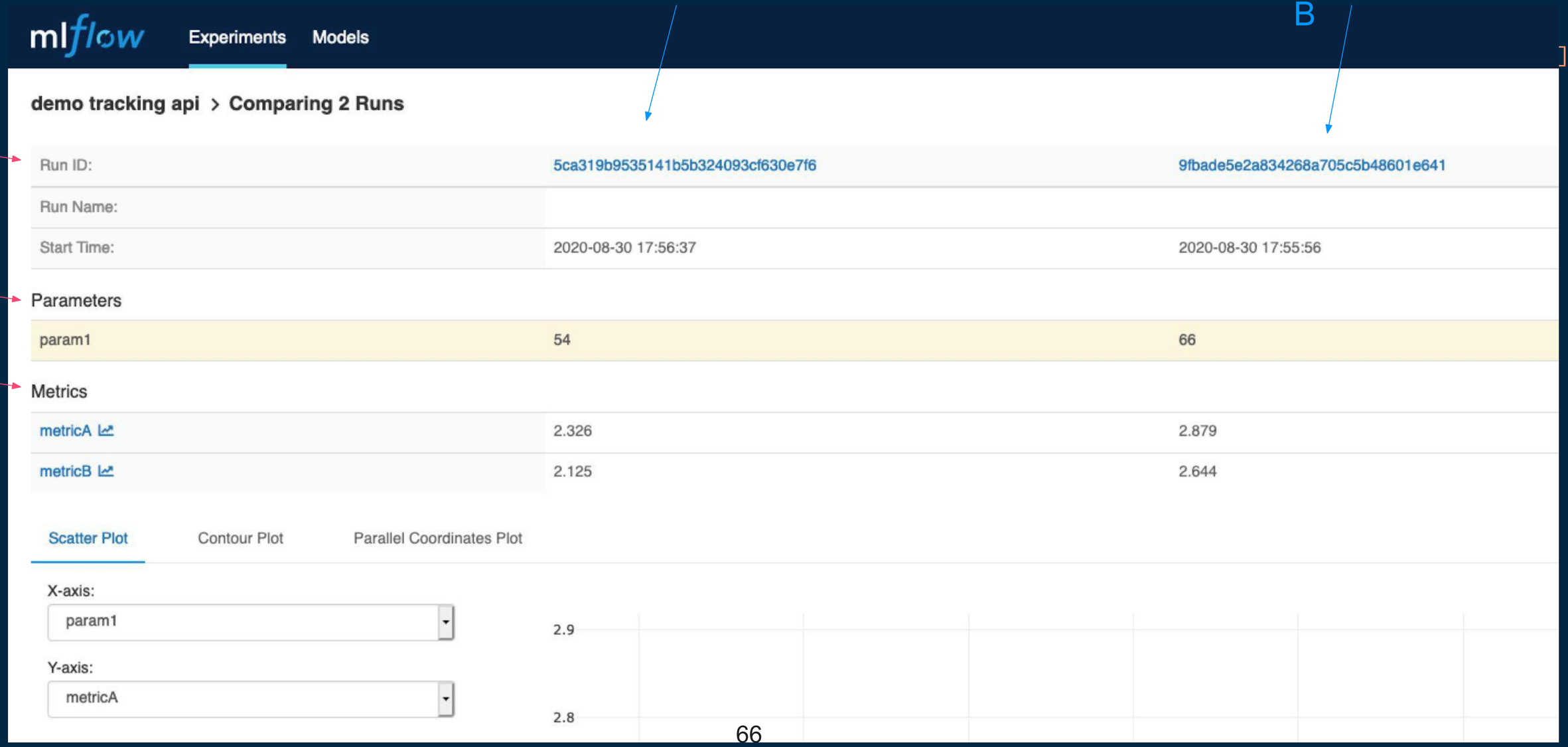
experiment

run

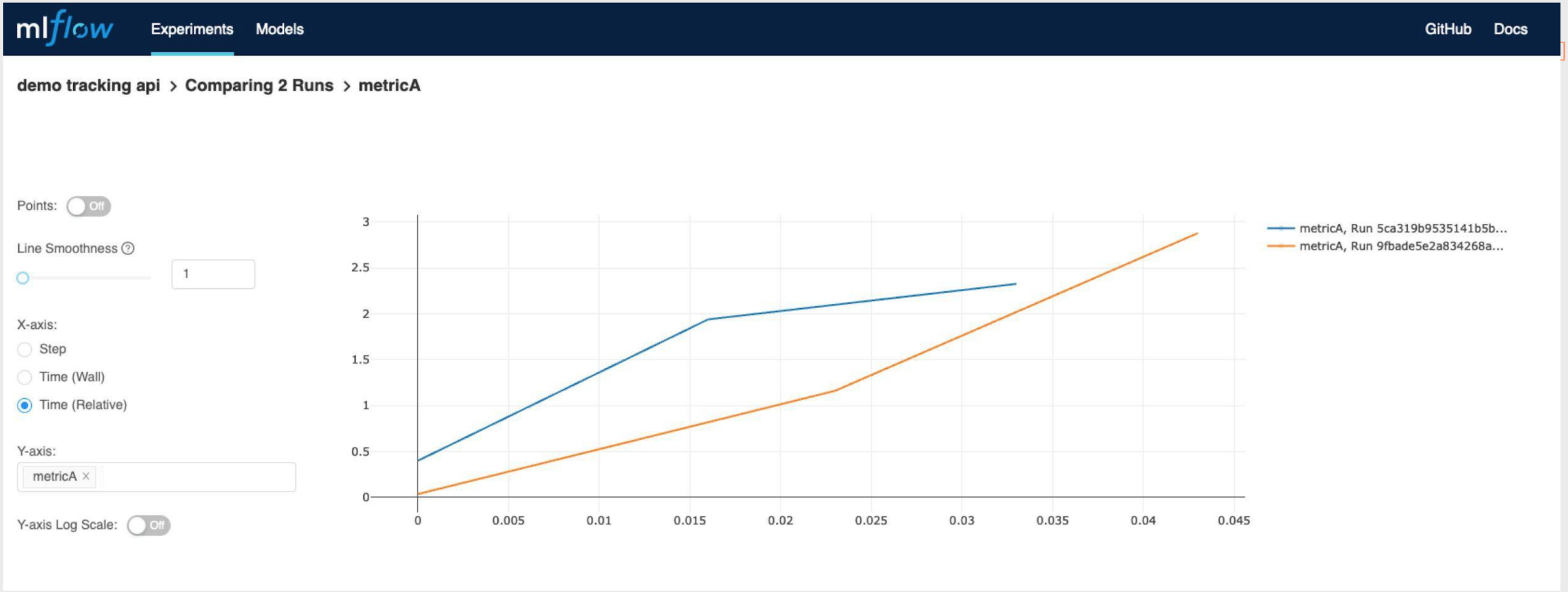
Compare Two Runs

Run A

Run B



Compare Two Runs



Model

Log->Load->deploy

```
mlflow.<model-type>.log_model(model, ...)

mlflow.<model-type>.load_model(modelpath)

mlflow.<model-type>.deploy()
```

```
mlflow.sklearn.log_model(lr, "model")
```

Built-In Model Flavors

`<model-type>`

- Python Function (python_function)
- R Function (crate)
- H_2O (h2o)
- Keras (keras)
- MLeap (mleap)
- PyTorch (pytorch)
- Scikit-learn (sklearn)
- Spark MLlib (spark)
- TensorFlow (tensorflow)
- ONNX (onnx)
- MXNet Gluon (gluon)
- XGBoost (xgboost)
- LightGBM (lightgbm)
- Spacy(spaCy)
- Fastai(fastai)

https://github.com/sueitw/mlflow_tracking/blob/master/pytw2020_demo_SK_model.py

Artifacts

model

MLmodel

conda.yaml

model.pkl

Full Path: /tmp/mlflow/artifactStore/4/557a222e5c8e454689a112cd…

Size: 150B

```yaml
channels:
- defaults
- conda-forge
dependencies:
- python=3.7.1
- scikit-learn=0.23.2
- pip
- pip:
  - mlflow
  - cloudpickle==1.5.0
name: mlflow-env
```

https://github.com/sueit/mlflow_tracking/blob/master/pytw2020_demo_SK_model.py

Auto-Logging - TF

Manually logging

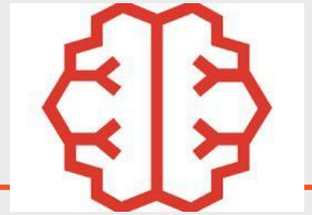
```
import mlflow

mlflow.log_param("layers", layers)
model = train_model()
mlflow.log_metric("mse", model.mse())
mlflow.log_artifact("plot", plot(model))
mlflow.tensorflow.log_model(model)
```

With autologging

```
import mlflow

mlflow.tensorflow.autolog()
model = train_model()
```

Capture TensorBoard metrics

https://cs.stanford.edu/people/matei/papers/2020/deem_mlflow.pdf



RedisAI

MLflow Tracking
MLflow Project
MLflow Models
MLflow Model registry
MLflow Deployment

Experiment tracking
Model deployment/serving

Model governance

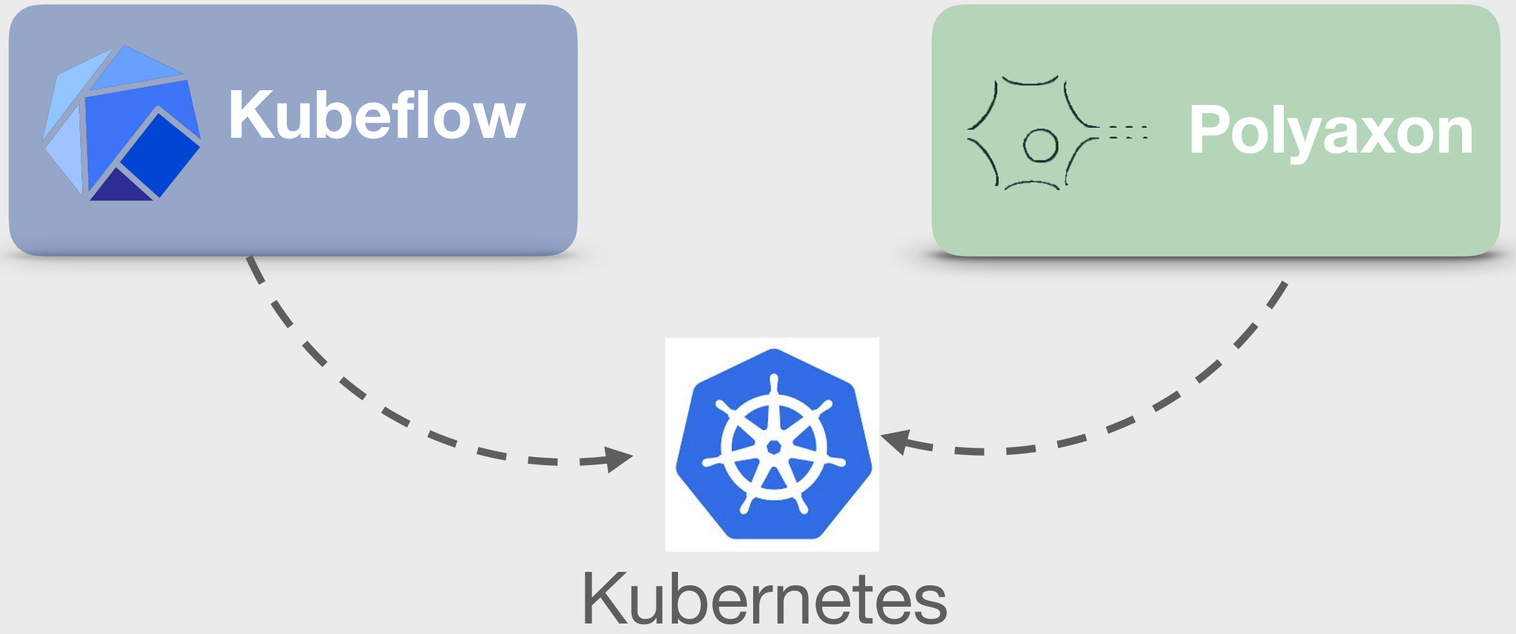
Kubeflow

- ML toolkit for Kubernetes
- Open-source and community-driven
- Support for multiple ML frameworks
- End-to-end workflows which can be shared, scaled and deployed

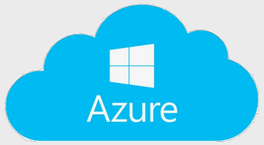
Source:
https://github.com/kubeflow/kubeflow/issues/187

State of the Art Technologies

Support Machine Learning workloads in the Cloud



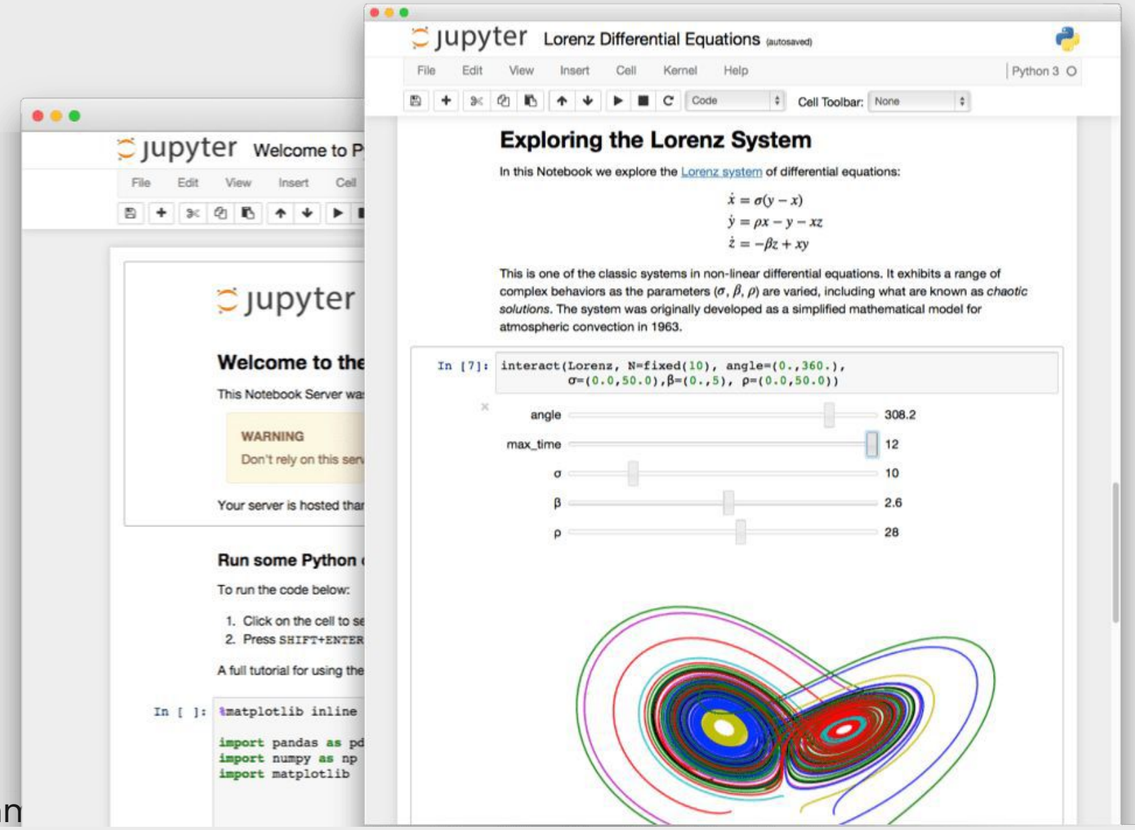
Cloud Agnostic



Components

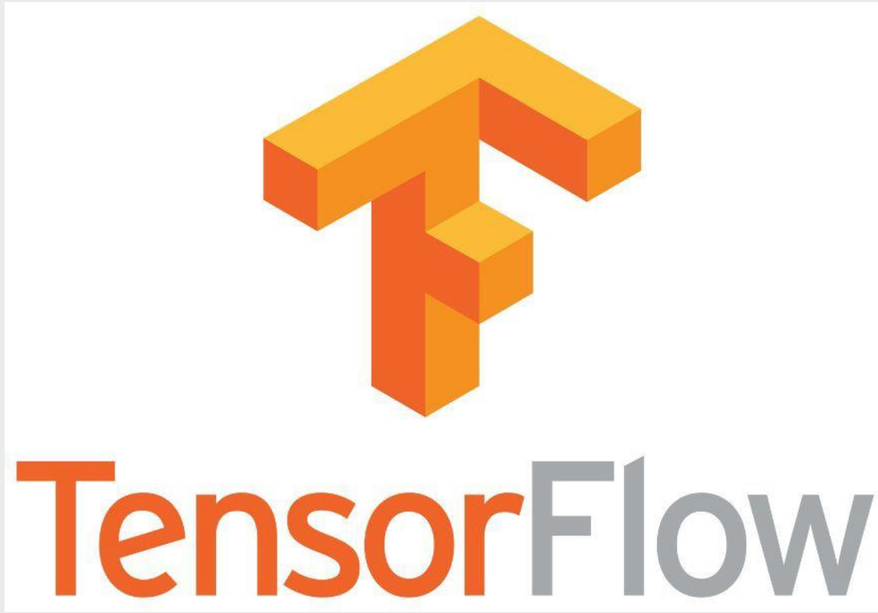
- Jupyterhub (collaboration and interactivity)
- K8s- native tensorflow controller (model building)
- K8s- native tensorflow serving deployment (model deployment)
- Ambassador (reverse proxy)
- Current and upcoming components for model tuning, model building and much more...
- Out-of-the-box setup for putting all of this together!

Jupyterhub


jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None
Exploring the Lorenz System
In this Notebook we explore the Lorenz system of differential equations:
$\dot{x} = \sigma(y - x)$
$\dot{y} = \rho x - y - xz$
$\dot{z} = -\beta z + xy$
This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as chaotic solutions. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
angle 308.2
max_time 12
σ 10
β 2.6
ρ 28
jupyter Welcome to P
File Edit View Insert Cell
jupyter
Welcome to the
This Notebook Server wa
WARNING
Don't rely on this ser
Your server is hosted tha
Run some Python
To run the code below:
1. Click on the cell to se
2. Press SHIFT+ENTER
A full tutorial for using the
In [ ]: %matplotlib inline
import pandas as pd
import numpy as np
import matplotlib

Tensorflow

- Open source numerical computing and ML
- Developed by Google, open-sourced in 2015
- Huge community and ecosystem
- Support for multiple ML models
- Tf-serving (model deployment), tensorboard (training visualization), etc.
- **Supports distributed training and deployment of models**



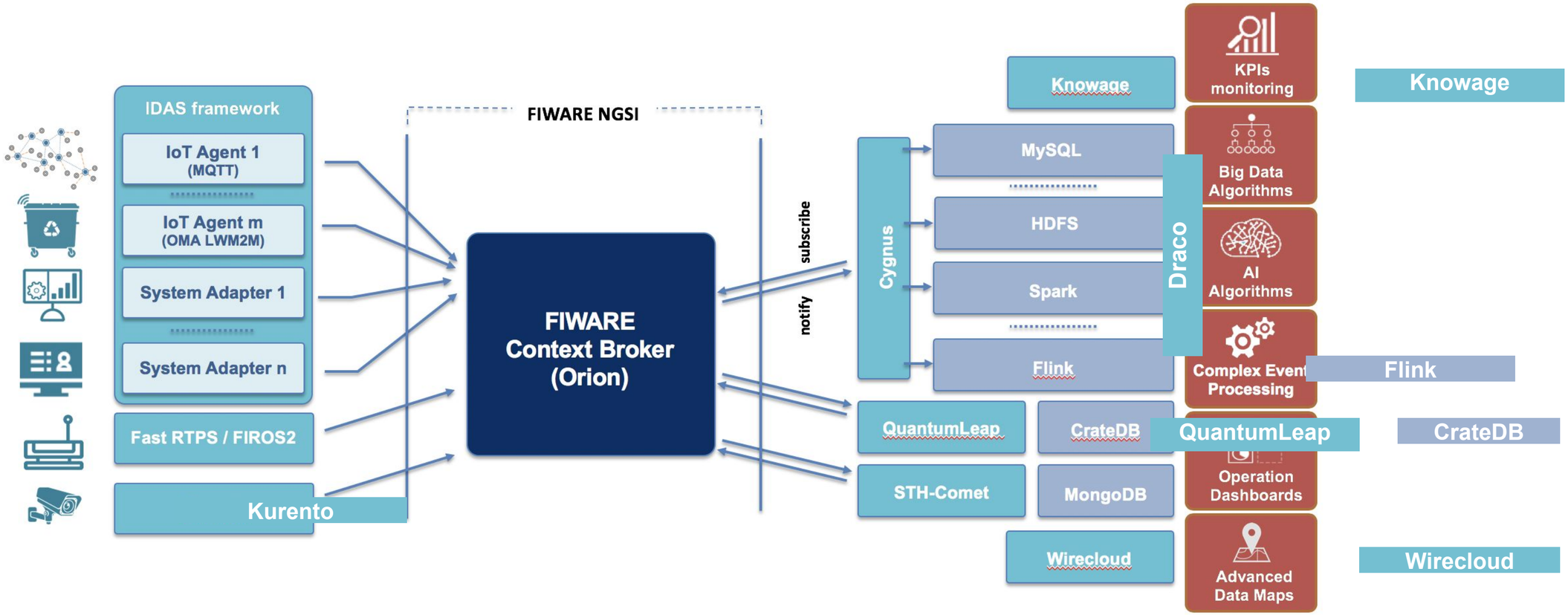
Why Kubeflow?

Based on current functionality you should consider using Kubeflow if:

- You want to train/serve TensorFlow models in different environments (e.g. local, on prem, and cloud)
- You want to use Jupyter notebooks to manage TensorFlow training jobs
- You want to launch training jobs that use resources – such as additional CPUs or GPUs – that aren't available on your personal computer
- You want to combine TensorFlow with other processes
  - For example, you may want to use tensorflow/agents to run simulations to generate data for training reinforcement learning models.

Refer https://www.kubeflow.org/docs/started/getting-started/ for more info.

Simple Smart solutions: Reference Architecture



FIWARE
Global
Summit

From Data
to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Draco: Persisting Context Data to MongoDB

José Andrés Muñoz Arcentales, Javier Conde Díaz, Joaquín Salvachúa

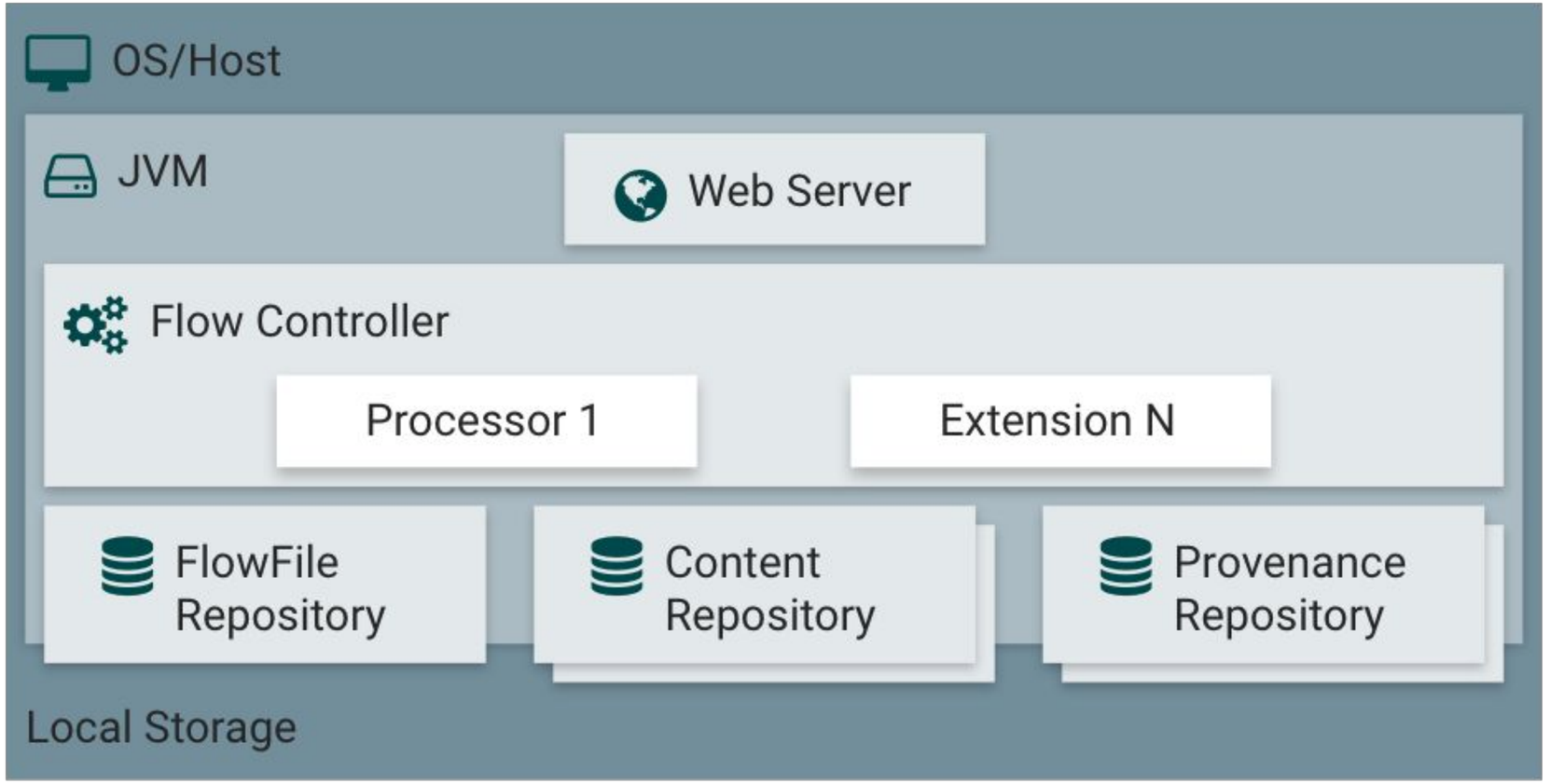
Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

The Draco GE

- The Draco Generic Enabler takes care of the data ingestion and persistence. Is a is an easy to use, powerful, and reliable system for processing and distributing data. Internally, Draco is based on Apache NiFi.
- NiFi is a dataflow system based on the concepts of flow-based programming. It supports powerful and scalable directed graphs of data routing, transformation, and system mediation logic. It was built to automate the flow of data between systems.

Apache Nifi Architecture



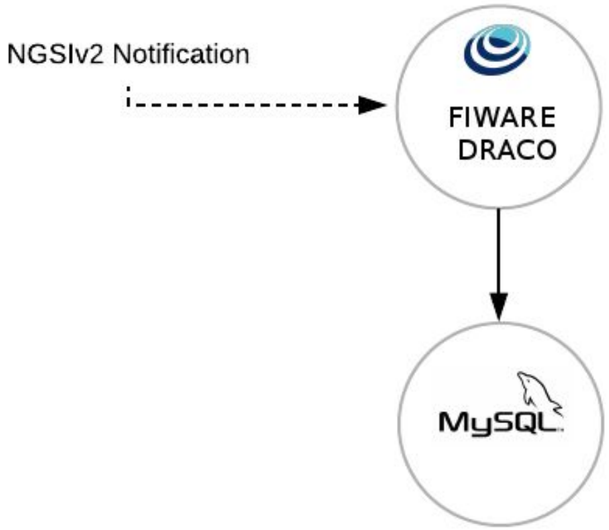
Draco integration in the FIWARE ecosystem

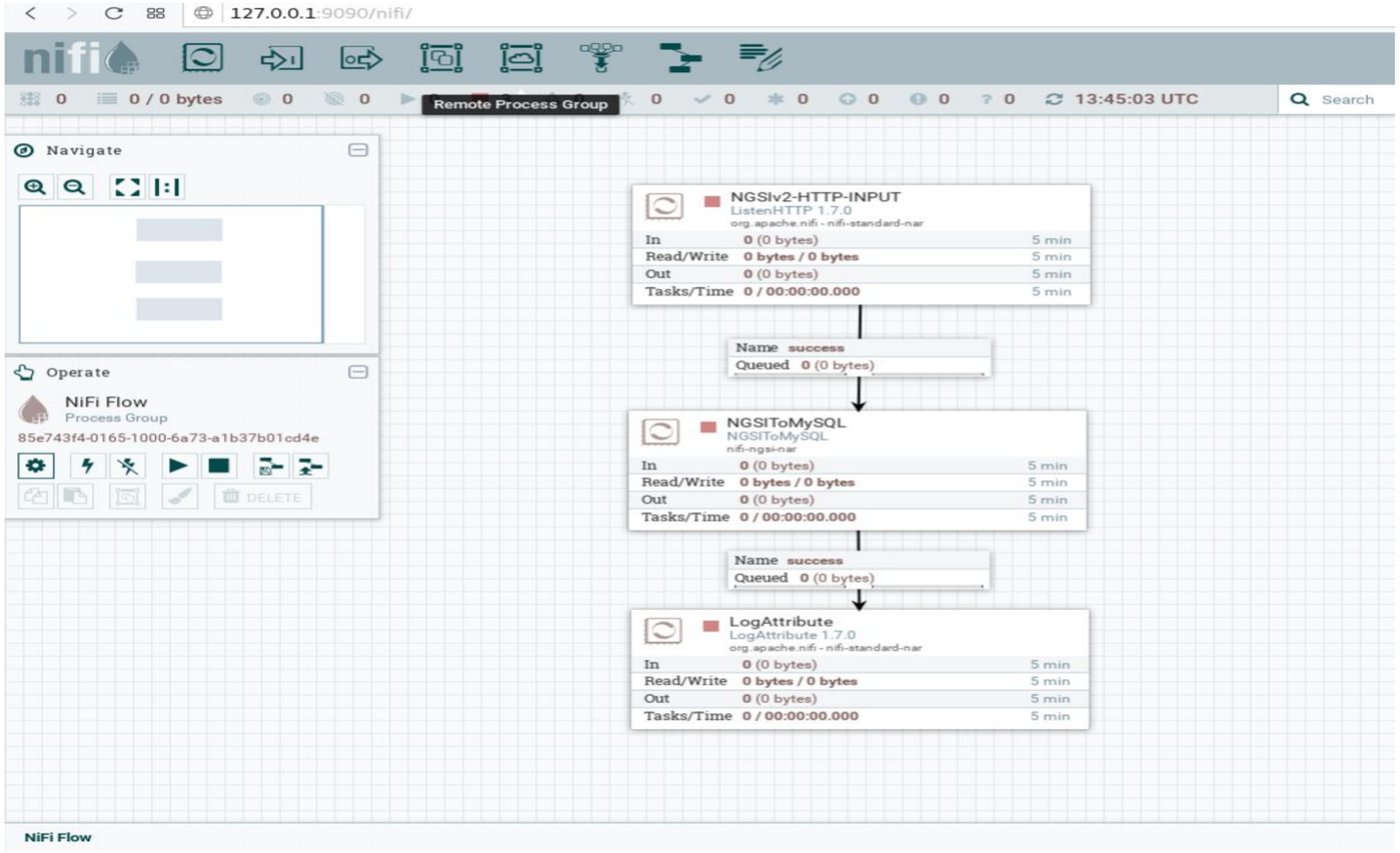
MiniNiFI (low profile version)

Features

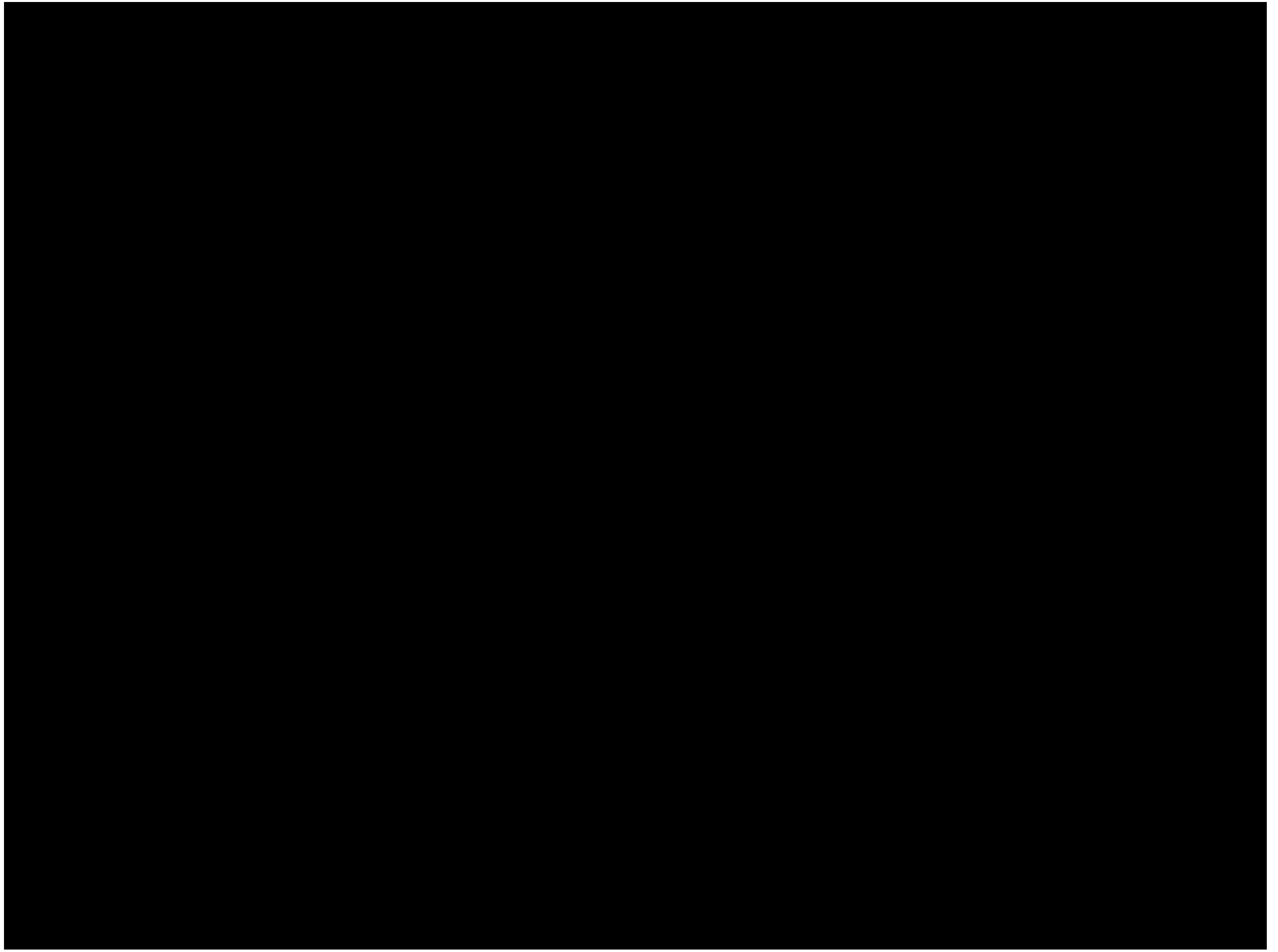
- **Based on Apache NiFi.**
- **NGSI 2 Support both for ingestion and serialization to have full integration with the Orion Context Broker.**
- **Several persistent backends :**
  - **MySQL**, the well-know relational database manager.
  - **MongoDB**, the NoSQL document-oriented database.
  - **PostgreSQL**, the well-know relational database manager.
  - **HDFS**, Hadoop distributed file system.
  - **Cassandra**, Distributed database.
  - **CartoDB**, for geospatial Data
- **Templates for some common scenarios**
- **Rest API**

Basic Example





Demo



Get the code!

https://github.com/ging/fiware-draco-data-ingestion-supermaket

FIWARE Global Summit

From Data to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Cosmos: Loading Streaming Data using Flink and Spark

José Andrés Muñoz Arcentales, Javier Conde Díaz, Joaquín Salvachúa

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

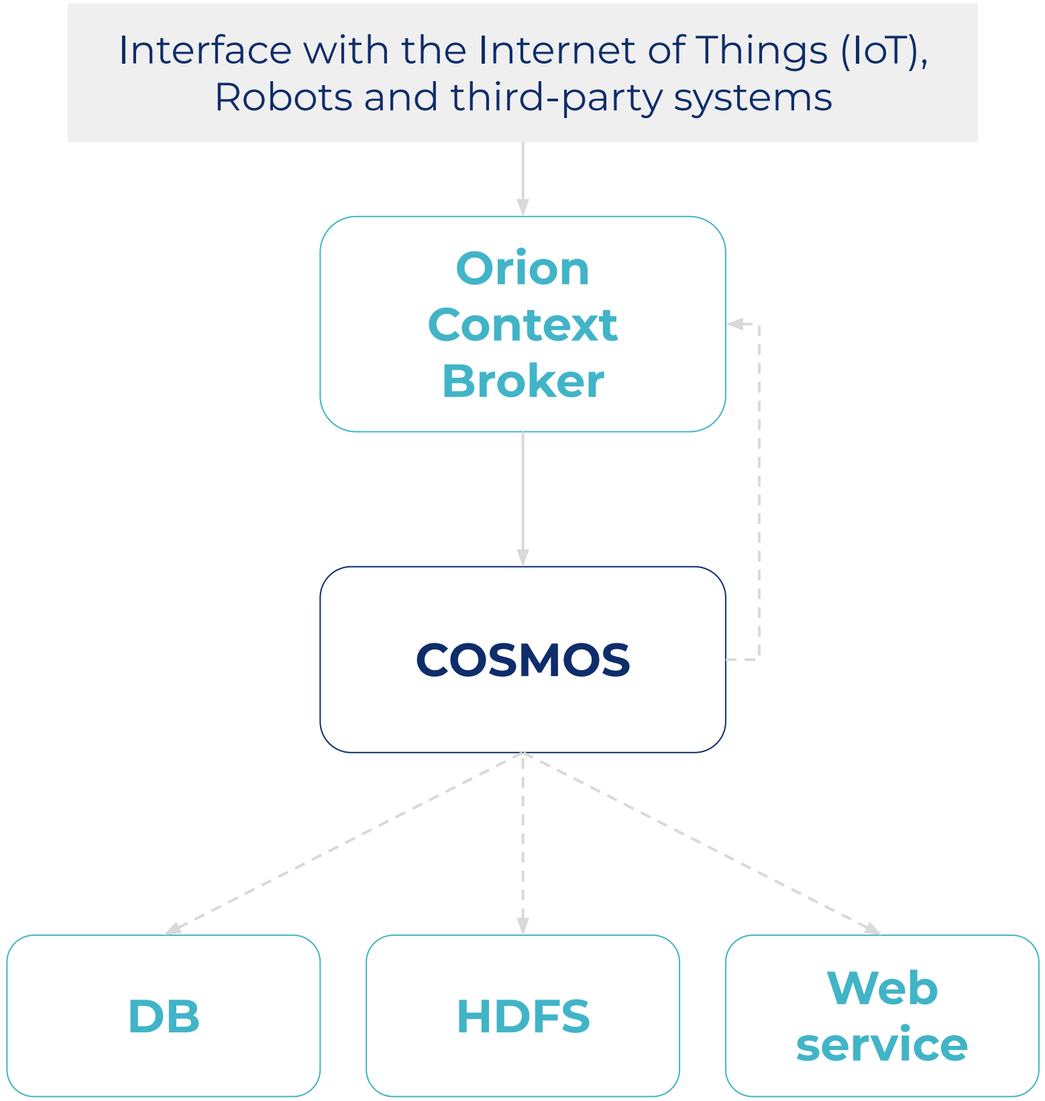
The Cosmos GE

The Cosmos Generic Enabler enables an easier BigData analysis over context integrated with some of the most popular BigData platforms.

Features

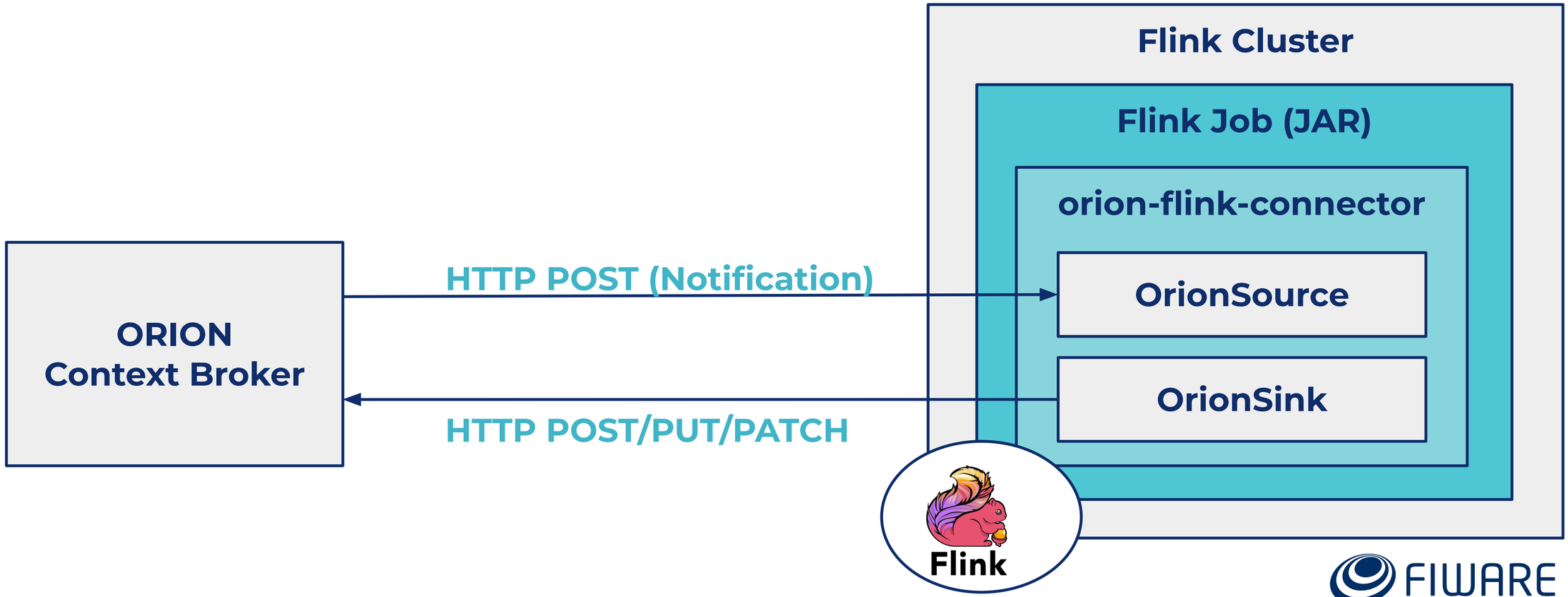
- ✔ Batch Processing
- ✔ Stream Processing (Real-time)
- ✔ Direct data ingestion
- ✔ Direct connection with Orion
- ✔ Multiple Sinks

https://github.com/ging/fiware-cosmos



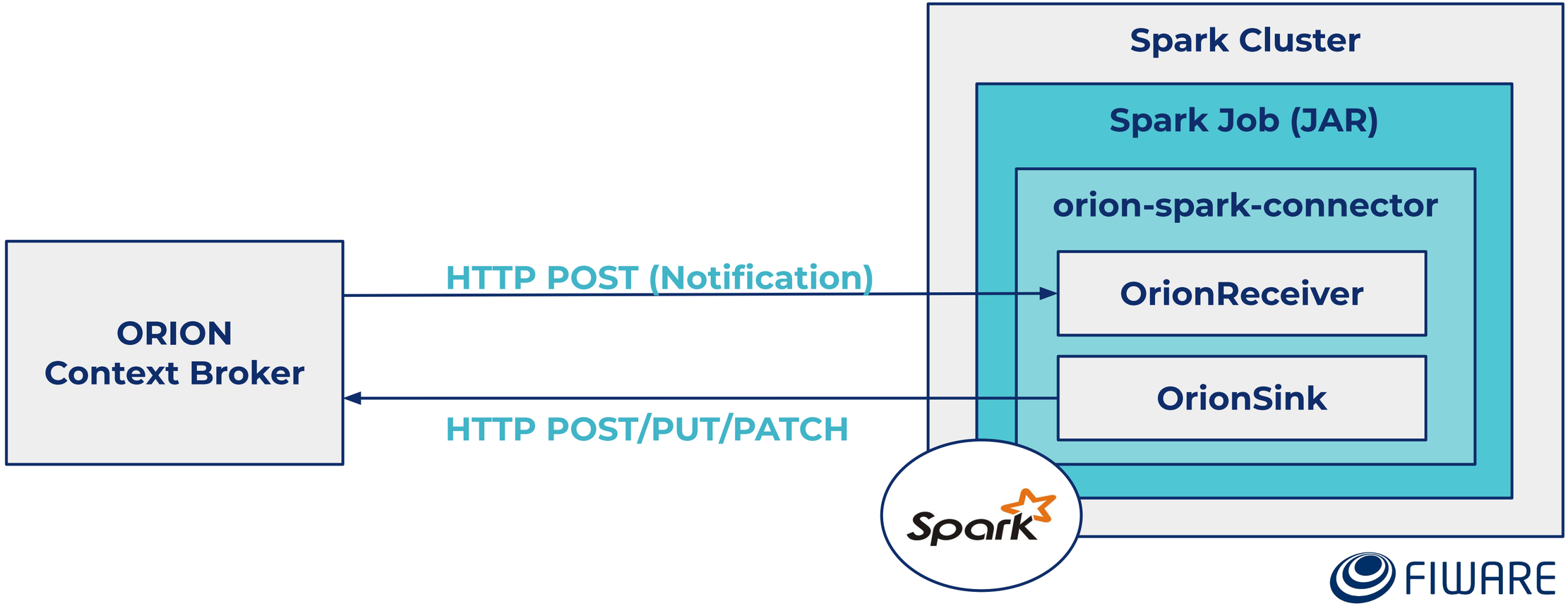
fiware-cosmos-orion-flink-connector

- https://github.com/ging/fiware-cosmos-orion-flink-connector
- https://github.com/ging/fiware-cosmos-orion-flink-connector-examples



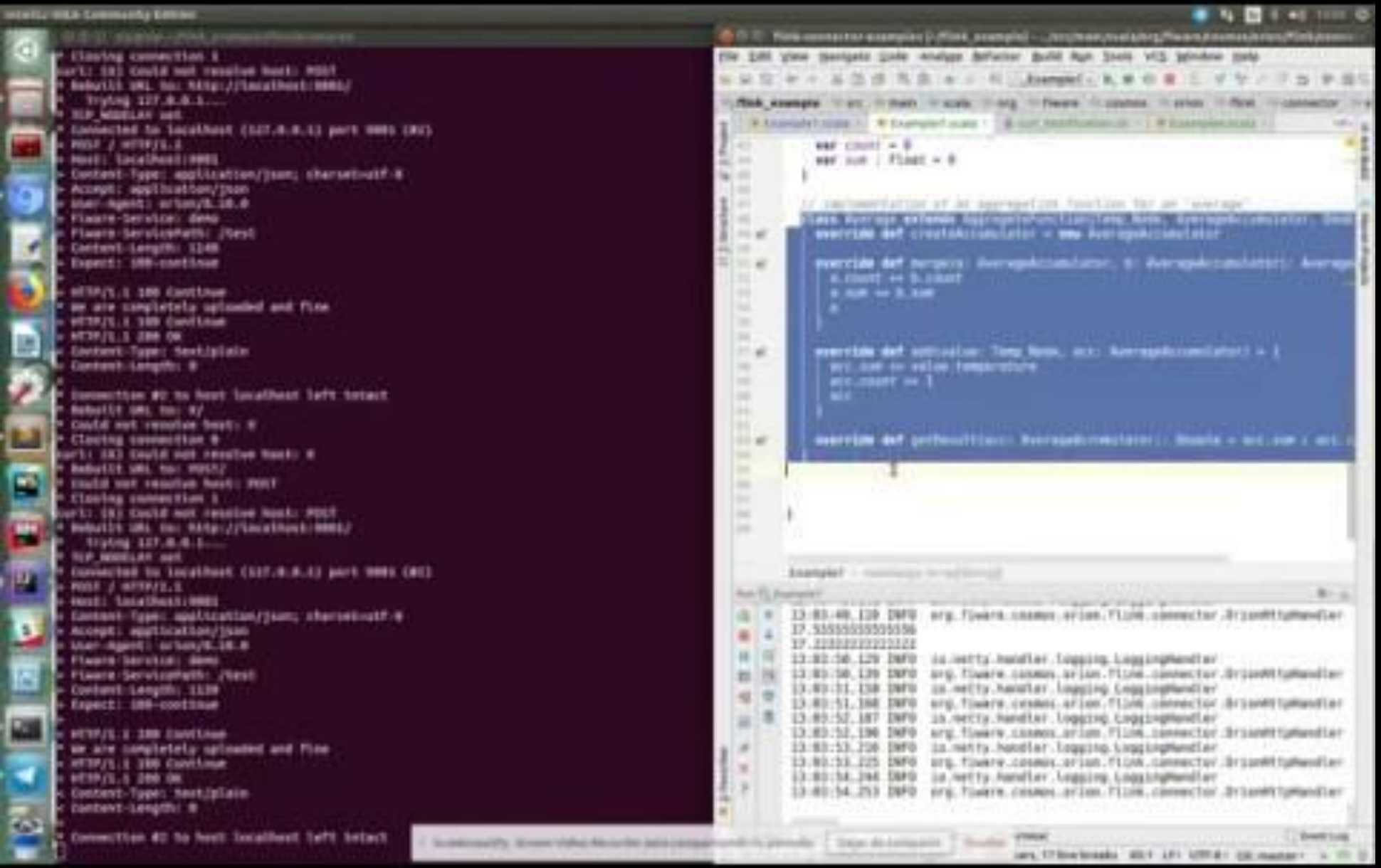
fiware-cosmos-orion-spark-connector

- https://github.com/ging/fiware-cosmos-orion-spark-connector
- https://github.com/ging/fiware-cosmos-orion-spark-connector-examples



Demo: Average temperature for each entity

```
def main(args: Array[String]): Unit = {
 val env = StreamExecutionEnvironment.getExecutionEnvironment

// Create Orion Source. Receive notifications on port 9001
 val eventStream = env.addSource(new OrionSource(9001))

 // Process event stream
 val processedDataStream = eventStream
   .flatMap(event => event.entities)
   .map(entity => {
     val temp = entity.attrs("temperature").value.asInstanceOf[Number].floatValue()
    (entity.id, temp)
   })
   .keyBy(0)
   .timeWindow(Time.seconds(10))
   .aggregate(new Average)

 // print the results with a single thread, rather than in parallel
 processedDataStream.print().setParallelism(1)
```


* Closing connection 1
curl: (6) Could not resolve host: POST
* Rebuilt URL to: http://localhost:9001/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 9001 (#1)
> POST / HTTP/1.1
> Host: localhost:9001
> Content-Type: application/json; charset=utf-8
> Accept: application/json
> User-Agent: orion/0.10.0
> Fiware-Service: demo
> Fiware-ServicePath: /test
> Content-Length: 1140
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
* We are completely uploaded and fine
< HTTP/1.1 200 Continue
< HTTP/1.1 200 OK
< Content-Type: text/plain
< Content-Length: 0
<
* Connection #0 to host localhost left intact
* Rebuilt URL to: 0/
* Could not resolve host: 0
* Closing connection 0
curl: (6) Could not resolve host: 0
* Rebuilt URL to: POST/
* Could not resolve host: POST
* Closing connection 1
curl: (6) Could not resolve host: POST
* Rebuilt URL to: http://localhost:9001/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 9001 (#1)
> POST / HTTP/1.1
> Host: localhost:9001
> Content-Type: application/json; charset=utf-8
> Accept: application/json
> User-Agent: orion/0.10.0
> Fiware-Service: demo
> Fiware-ServicePath: /test
> Content-Length: 1139
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
* We are completely uploaded and fine
< HTTP/1.1 100 Continue
< HTTP/1.1 200 OK
< Content-Type: text/plain
< Content-Length: 0
<
* Connection #0 to host localhost left intact
var count = 0
var sum : Float = 0
override def createAccumulator = new AverageAccumulator
override def merge(a: AverageAccumulator, b: AverageAccumulator): Average
override def add(value: Temp Name, acc: AverageAccumulator) = {
override def getResult(acc: AverageAccumulator): Double = acc.sum / acc.
13:03:49.119 INFO org.fiware.cosmos.orion.flink.connector.OrionHttpHandler
17.555555555555556
17.222222222222222
13:03:50.129 INFO io.netty.handler.logging.LoggingHandler
13:03:50.139 INFO org.fiware.cosmos.orion.flink.connector.OrionHttpHandler
13:03:51.150 INFO io.netty.handler.logging.LoggingHandler
13:03:51.160 INFO org.fiware.cosmos.orion.flink.connector.OrionHttpHandler
13:03:52.187 INFO io.netty.handler.logging.LoggingHandler
13:03:52.196 INFO org.fiware.cosmos.orion.flink.connector.OrionHttpHandler
13:03:53.216 INFO io.netty.handler.logging.LoggingHandler
13:03:53.225 INFO org.fiware.cosmos.orion.flink.connector.OrionHttpHandler
13:03:54.244 INFO io.netty.handler.logging.LoggingHandler
13:03:54.253 INFO org.fiware.cosmos.orion.flink.connector.OrionHttpHandler

Current status

	Flink	Spark
Orion Connector Orion Source/Receiver + Orion Sink	✔	✔
RTD Documentation	✔	✔
Unit Tests	✔	✔
Examples	✔	✔
Step-by-step tutorial	✔	
Support NGSI LD		

FIWARE
Global
Summit

From Data
to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Inference system with ML

José Andrés Muñoz Arcentales, Javier Conde Díaz, Joaquín Salvachúa

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

Predicting supermarket purchases

An use case

Purchase data

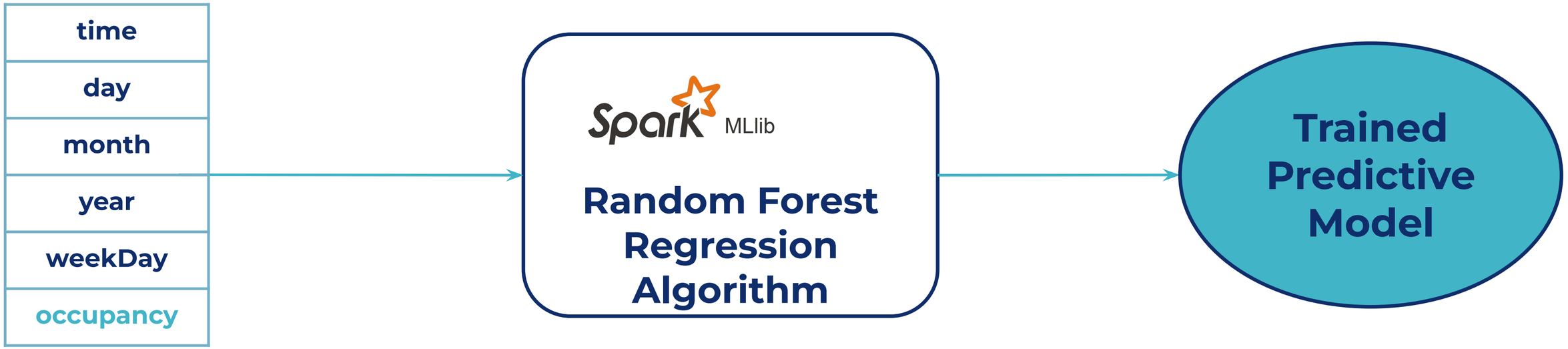
How we model each purchase:

- **date**
- **client_id**
- **supermarket_id**
- **product_list**
  - description
  - n_items
  - price

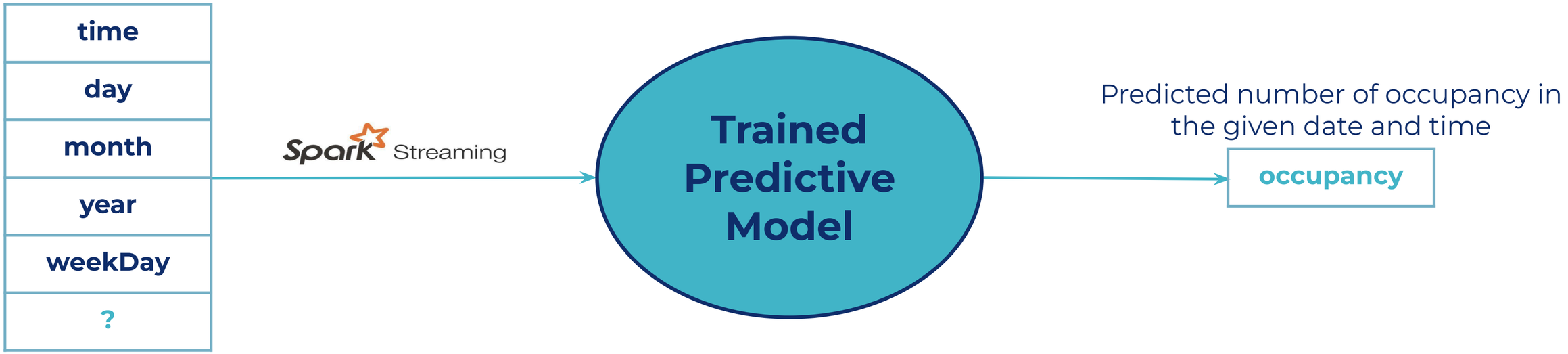
Data aggregation (for each store)

time	day	month	year	weekDay	ocupaccy
0	14	1	2016	3	**5**
1	14	1	2016	3	**3**
2	14	1	2016	3	**4**
3	14	1	2016	3	**3**
4	14	1	2016	3	**2**
5	14	1	2016	3	**8**
6	14	1	2016	3	**12**
7	14	1	2016	3	**12**
8	14	1	2016	3	**23**
9	14	1	2016	3	**45**
10	14	1	2016	3	**55**
11	14	1	2016	3	**37**
12	14	1	2016	3	**42**
13	14	1	2016	3	**41**
14	14	1	2016	3	**38**
15	14	1	2016	3	**29**
16	14	1	2016	3	**33**

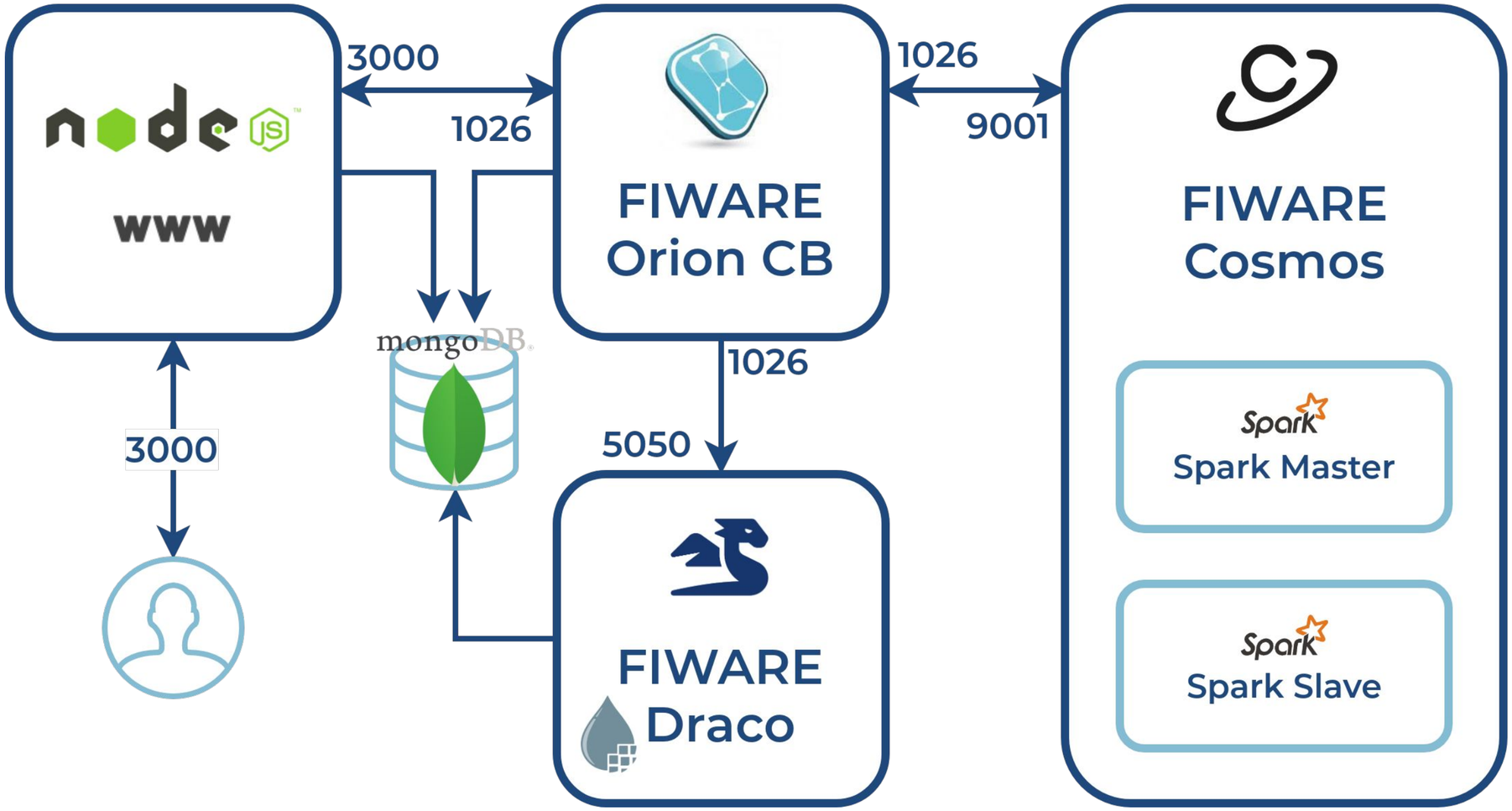
Training our model



Using our model to predict purchases



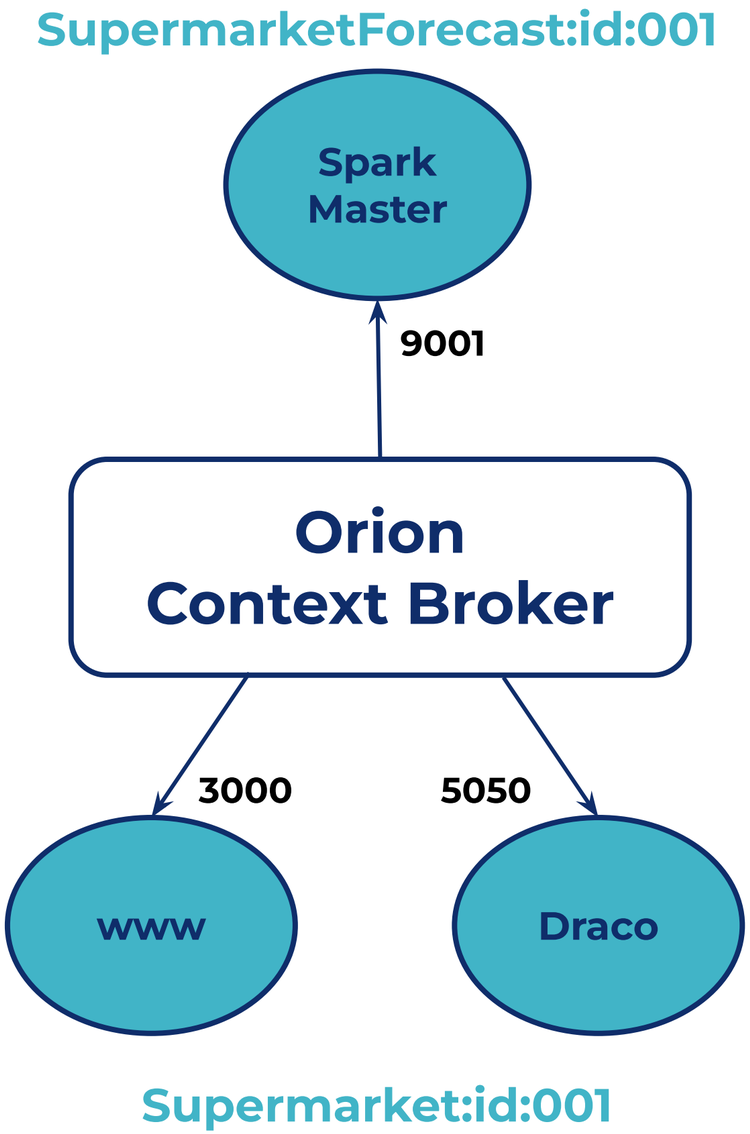
Architecture



Orion entities and subscriptions

```
{
    "id": "urn:ngsi-ld:Supermarket:id:001",
    "type": "Supermarket",
    "name": "F Summit Supermarket",
    "capacity": 200,
    "occupancy": 50,
    "dateObserved": "2023-06-11T21:09:57Z",
    "@context": [
        "https://smartdatamodels.org/context.jsonld"
    ]
}
```

```
{
    "id": "urn:ngsi-ld:SupermarketForecast:id:001",
    "type": "Supermarket",
    "name": "F Summit Supermarket Forecast",
    "capacity": 200,
    "occupancy": 50,
    "dateObserved": "2023-06-11T23:09:57Z",
    "@context": [
        "https://smartdatamodels.org/context.jsonld"
    ]
}
```



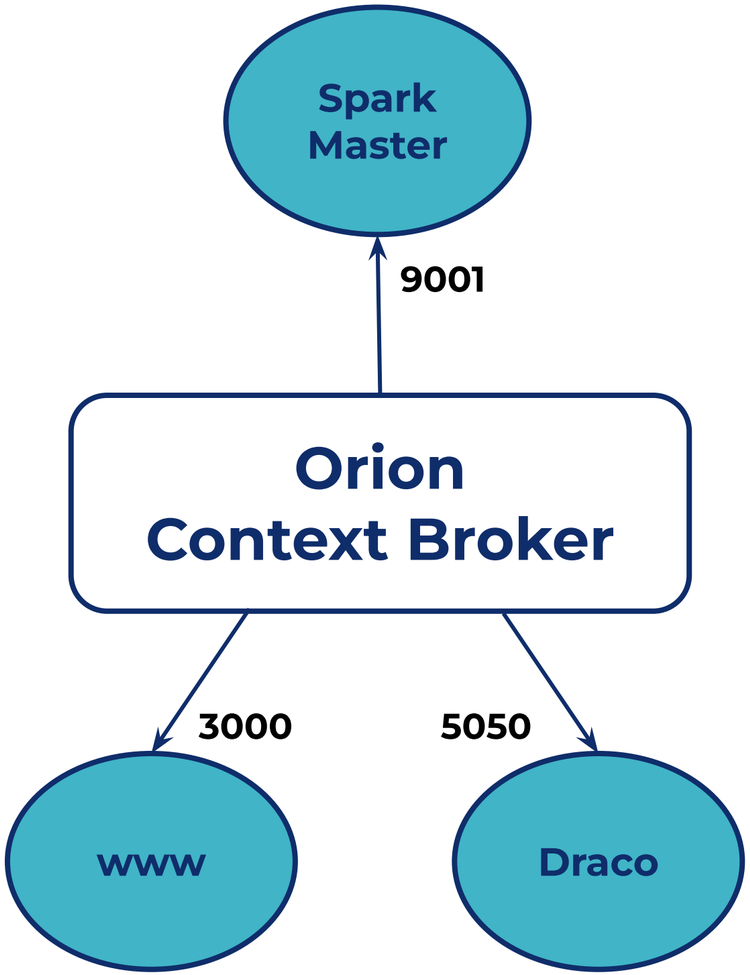
Orion entities and subscriptions


```
{
    "id": "urn:ngsi-ld:Supermarket:001",
    "type": "Supermarket",
    "name": {
      "value": "F. Summit Supermarket"
      "type": "Property"
    },
    "capacity": {
        "value": 50,
        "type": "Property"
      },
      "occupancy": {
        "value": 20,
        "type": "Property"
      },
      "year":{
        "value": 2023,
        "type": "Property"
      },
      "month":{
        "value": 11,
        "type": "Property"
      },
      "day":{
        "value": 22,
        "type": "Property"
      },
      "time": {
        "value": 22,
        "type": "Property"
      },
      "weekDay": {
        "value": 5,
        "type": "Property"
      },
      "dateObserved": {
        "value": "2023-11-22T22:01:12Z",
        "type": "Property"
      }
    "@context": [

"https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ]
}
```



```
{
    "id": "urn:ngsi-ld:SupermarketForecast:001",
    "type": "Supermarket",
    "name": {
      "value": "F. Summit Supermarket Forecast"
      "type": "Property"
    },
    "capacity": {
        "value": 70,
        "type": "Property"
      },
      "occupancy": {
        "value": 50,
        "type": "Property"
      },
      "year":{
        "value": 2023,
        "type": "Property"
      },
      "month":{
        "value": 11,
        "type": "Property"
      },
      "day":{
        "value": 22,
        "type": "Property"
      },
      "time": {
        "value": 23,
        "type": "Property"
      },
      "weekDay": {
        "value": 5,
        "type": "Property"
      },
      "dateObserved": {
        "value": "2023-11-22T23:01:12Z",
        "type": "Property"
      }
    "@context": [

"https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ]
}
```


SupermarketForecast:id:001



Supermarket:id:001

Get the code!

https://github.com/ging/fiware-mlops-supermarket

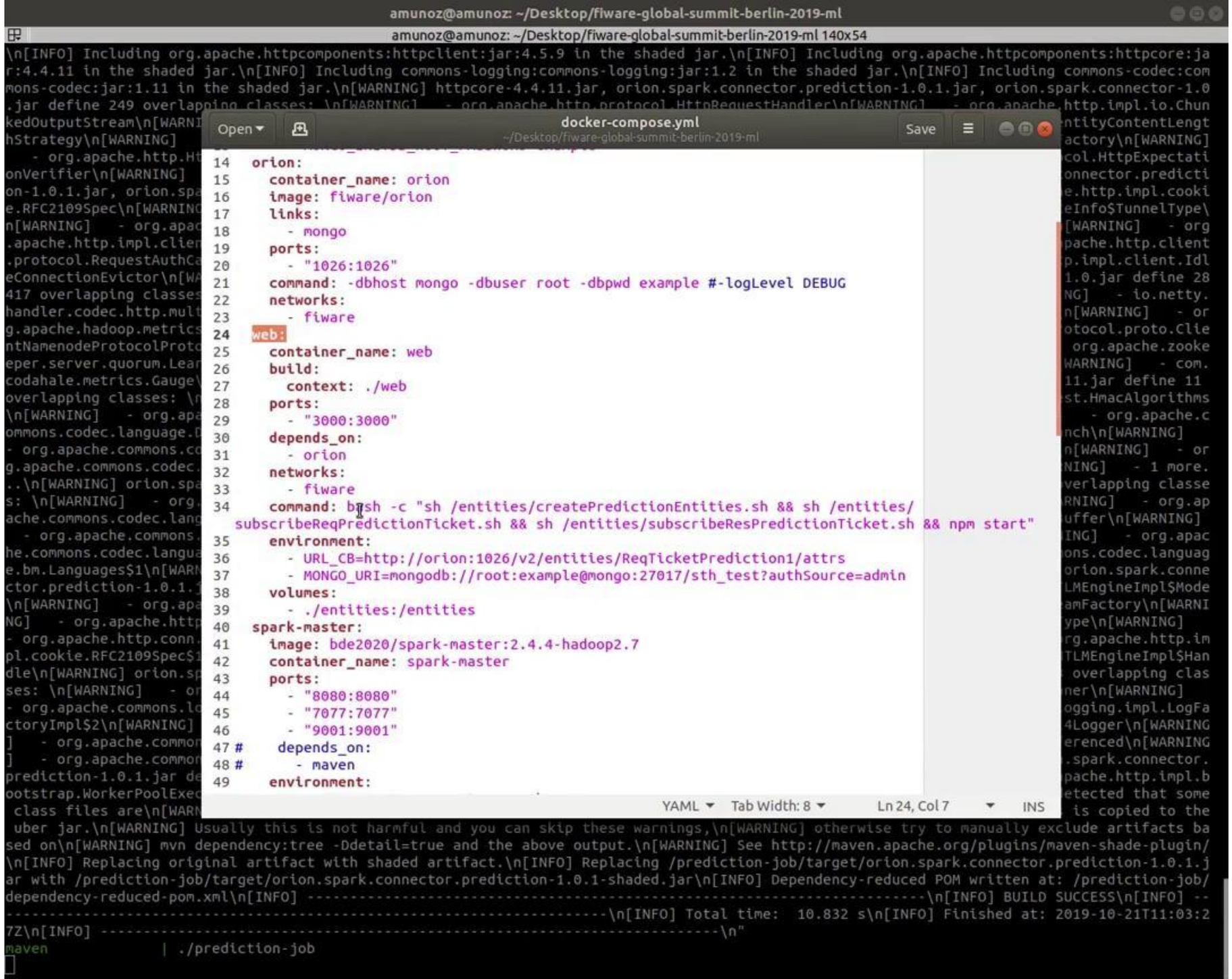
Open your laptop

```
git clone https://github.com/ging/fiware-mlops-supermarket

docker-compose up
```

Open your browser: http://localhost:3000

IT WORKS !!!!

Demo


amunoz@amunoz: ~/Desktop/fiware-global-summit-berlin-2019-ml
amunoz@amunoz: ~/Desktop/fiware-global-summit-berlin-2019-ml 140x54
Open
docker-compose.yml
~/Desktop/fiware-global-summit-berlin-2019-ml
Save
14 orion:
15 container_name: orion
16 image: fiware/orion
17 links:
18 - mongo
19 ports:
20 - "1026:1026"
21 command: -dbhost mongo -dbuser root -dbpwd example #-logLevel DEBUG
22 networks:
23 - fiware
24 web:
25 container_name: web
26 build:
27 context: ./web
28 ports:
29 - "3000:3000"
30 depends_on:
31 - orion
32 networks:
33 - fiware
34 command: bash -c "sh /entities/createPredictionEntities.sh && sh /entities/subscribeReqPredictionTicket.sh && sh /entities/subscribeResPredictionTicket.sh && npm start"
35 environment:
36 - URL_CB=http://orion:1026/v2/entities/ReqTicketPrediction1/attrs
37 - MONGO_URI=mongodb://root:example@mongo:27017/sth_test?authSource=admin
38 volumes:
39 - ./entities:/entities
40 spark-master:
41 image: bde2020/spark-master:2.4.4-hadoop2.7
42 container_name: spark-master
43 ports:
44 - "8080:8080"
45 - "7077:7077"
46 - "9001:9001"
47 # depends_on:
48 # - maven
49 environment:
YAML
Tab Width: 8
Ln 24, Col 7
INS
maven | ./prediction-job

FIWARE
Global
Summit

From Data
to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

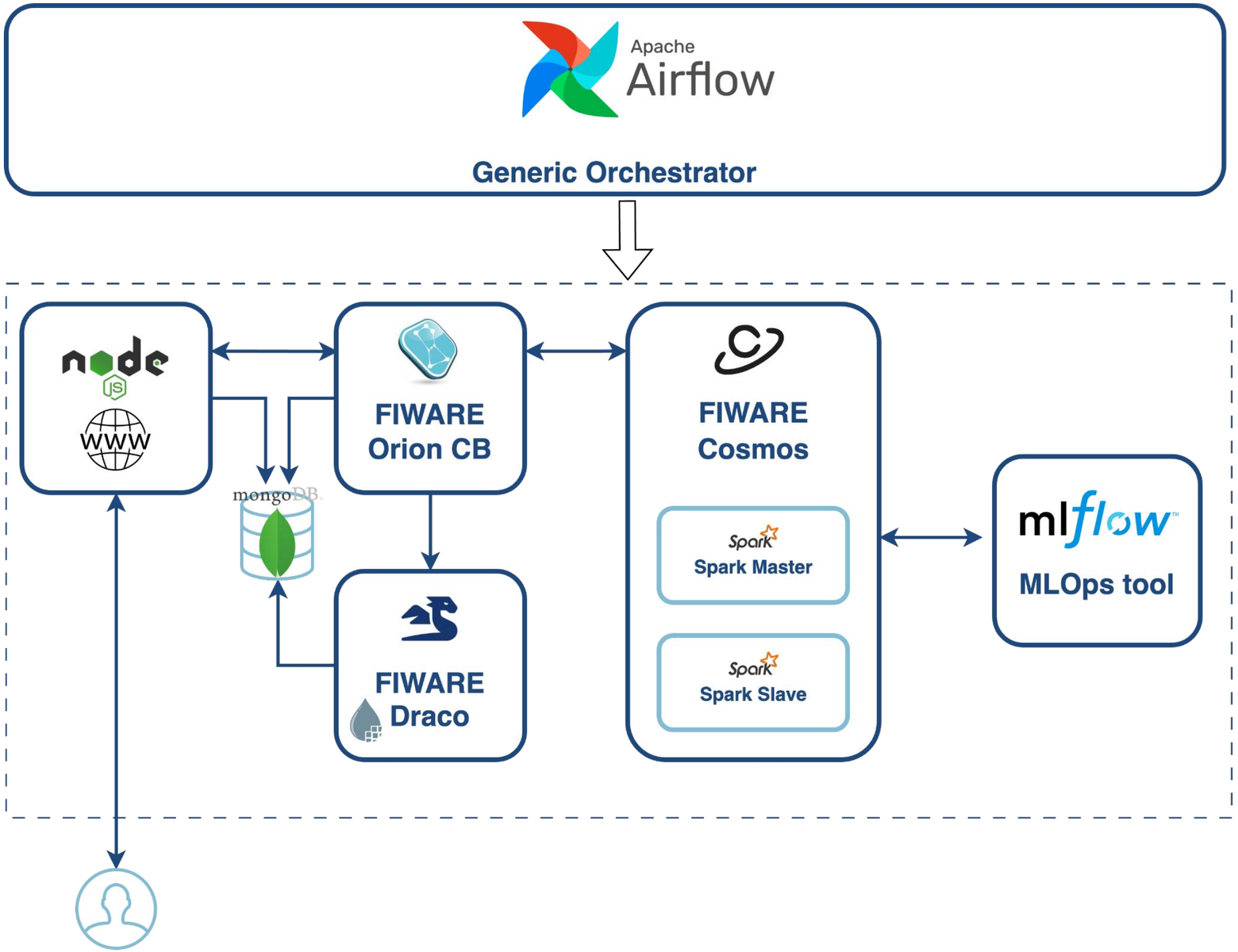
MLOps Life cycle real-time application

José Andrés Muñoz Arcentales, Javier Conde Díaz, Joaquín Salvachúa

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds

Architecture



FIWARE Global Summit

Find Us On

Stay up to date

JOIN OUR NEWSLETTER

Be certified and featured

FIWAREMarketplace

Hosting Partner

City of Vienna

vienna business agency

Keystone Sponsors

aws

ms.gis

رسيل Raseel

Red Hat

Media Partners

bee smart city

CITIES FORUM

CitiesToday

Our Future Water

RM RENEWABLE MATTER

Smart Cities World

the smartcity journal

FIWARE
Global
Summit

Thanks!

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds