Supercharging the Machine Learning Lifecycle

Anniina Sallinen
Data Engineer @ Oura
She / her

CS major in University of Helsinki 2011-2018

- Algorithms and machine learning (+ software systems + distributed systems)

Worked in IT since 2014

- data domain since 2018
- Oura since 2021

Web development:

- Java

Data related development:

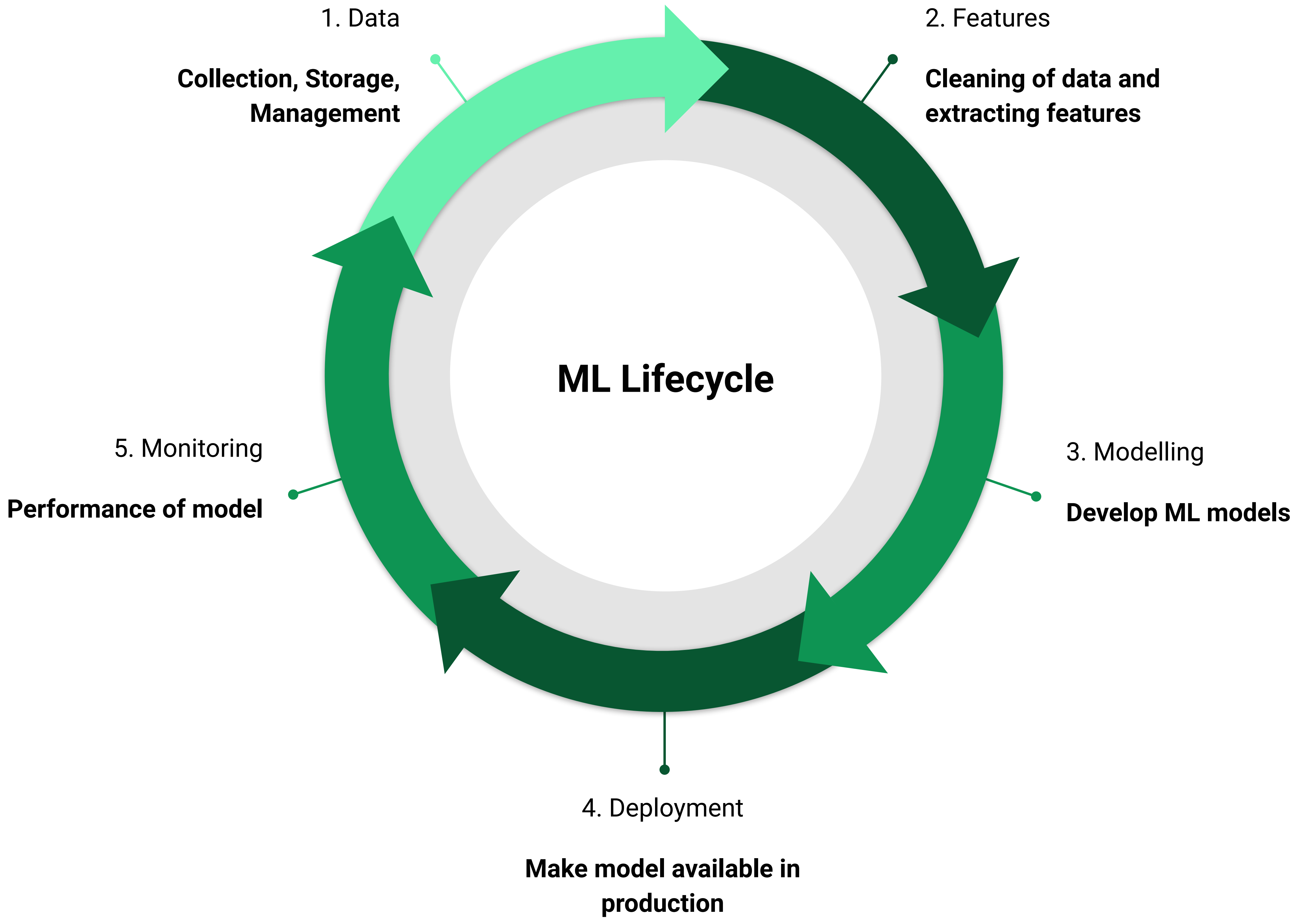
- Clojure
- Python
- Typescript
- AWS
- SQL
- IaC

Anniina Sallinen

Agenda

- Introduction to machine learning life cycle
- Introduction to MLOps
- MLOps at Oura
- QA

Machine learning lifecycle


1. Data
Collection, Storage, Management
2. Features
Cleaning of data and extracting features
ML Lifecycle
3. Modelling
Develop ML models
4. Deployment
Make model available in production
5. Monitoring
Performance of model

Data and features

Querying data from data lake / data warehouse

Sources of data for example: user data from software, internal systems (CRM), external systems (externals APIs)

Data scientists usually need to join the data, calculate features based on existing data

Real world data is messy, so it often needs cleaning

Exploring and visualizing data to understand it

Garbage in,
garbage out

Modelling

Selecting suitable machine learning algorithm

Hyperparameter tuning

Usually repeated multiple times with different algorithms and hyperparameters, possibly different data

Slow and compute heavy part

Needs careful data selection and handling

Essential that training and validation of the model is done with different sets of data

Deployment

CI/CD pipelines to deploy models and ML systems to production

Continuous and automated re-training vs manual re-training and slower updates

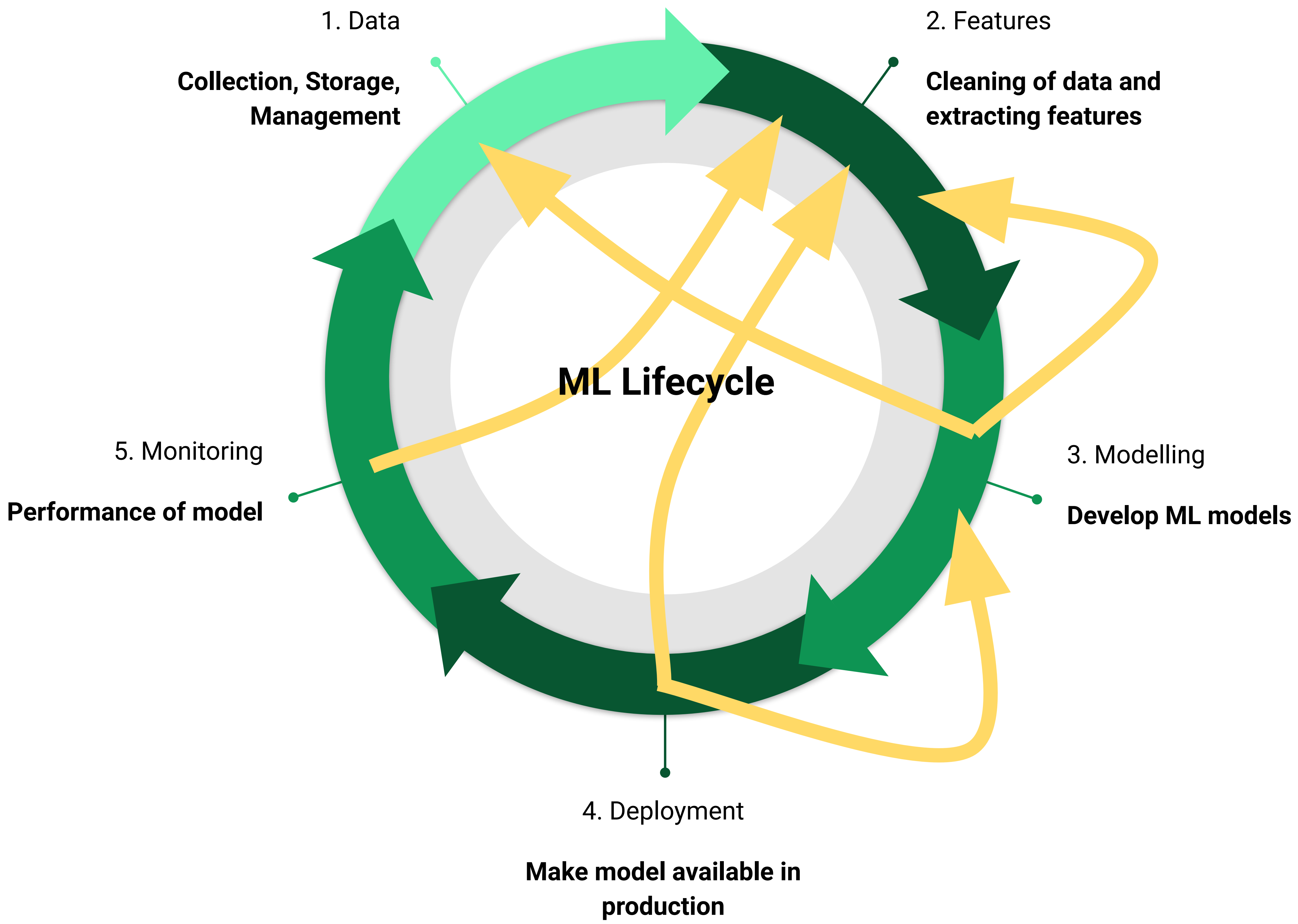
Testing that the model performs better than the previous one: A/B testing, shadow models, canary deployments

Monitoring

Monitoring system metrics such as latency, CPU load, memory usage

Monitoring of the model and data: performance of the model, data drift

Model monitoring often not automated, based on analysis performed by a data scientist


1. Data
Collection, Storage, Management
2. Features
Cleaning of data and extracting features
ML Lifecycle
3. Modelling
Develop ML models
4. Deployment
Make model available in production
5. Monitoring
Performance of model

Common challenges

Lack of
- Reproducibility
- Visibility
- Reliability
- Collaboration and sharing

“MLOps (machine learning operations) is a practice that aims to make developing and maintaining production machine learning seamless and efficient.”

-Valohai

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

-Google

MLOps concepts

Feature store

Model training metadata store

Model registry

Feature store

Storage for data that is used in training machine learning models and for inference.

Features can be calculated ahead of time, before training or making predictions

Standardizes data and calculations. Metadata is also stored in the feature store

Can be used for monitoring data to detect changes in it over time

Model training metadata store

Storage for metadata about training machine learning models

In the metadata store we can store:

- Algorithm and hyperparameters used
- Compute environment
- Test / validation results
- Other metrics about training (execution time etc)

Can be combined with information about data used in training and model artifact to debug and selection of a model

Model registry

Registry for models

Storage for current and old versions of models

Can also contain information about when the model was trained and references to metadata about training

MLOps at Oura

Oura

Founded in 2013 in Oulu

Smart ring to improve well-being of individuals by providing insights of sleep, readiness and activity

Ring generations launched:

- Gen1 (2015)
- Gen 2 (2017)
- Gen 3 (2021)

Almost 500 employees globally, offices in Helsinki, Oulu and Tampere in Finland

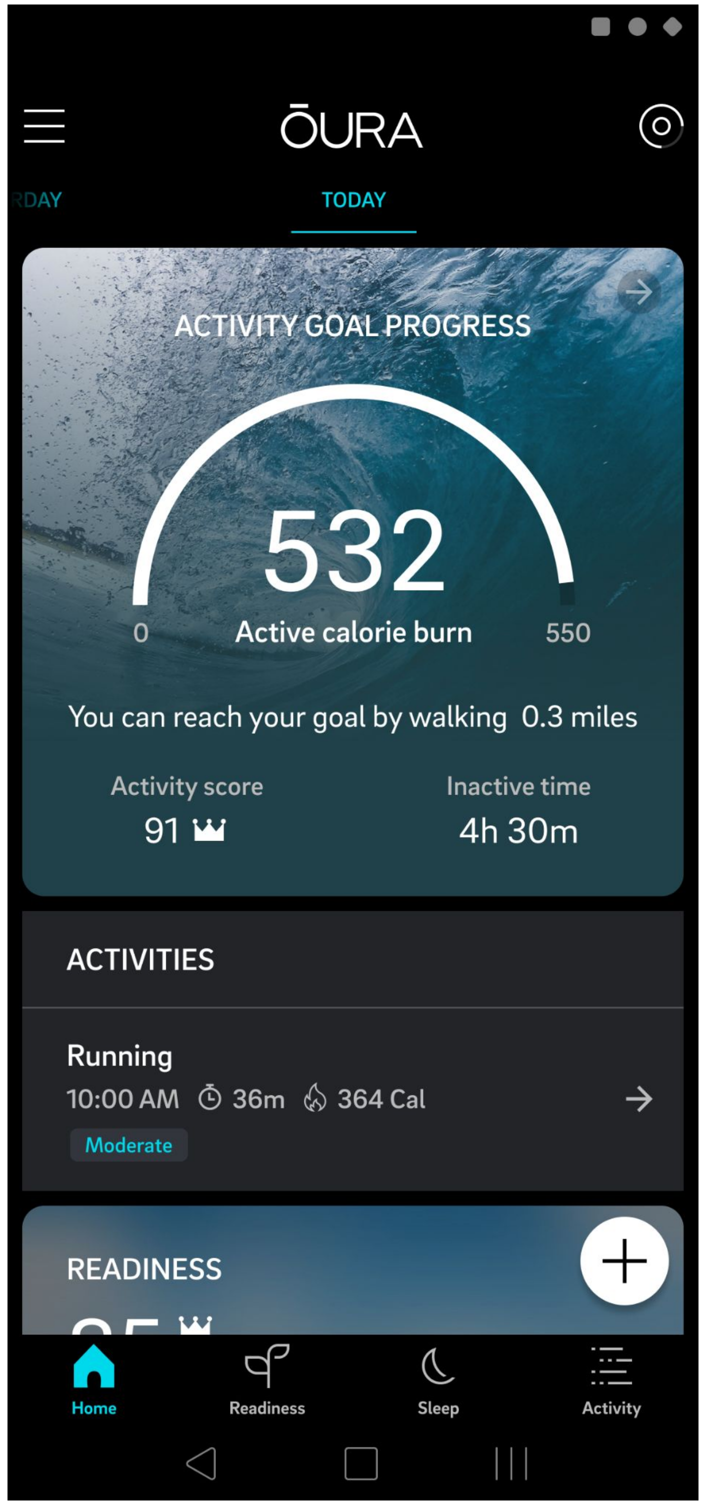
Engineers working on hardware, mobile applications and cloud

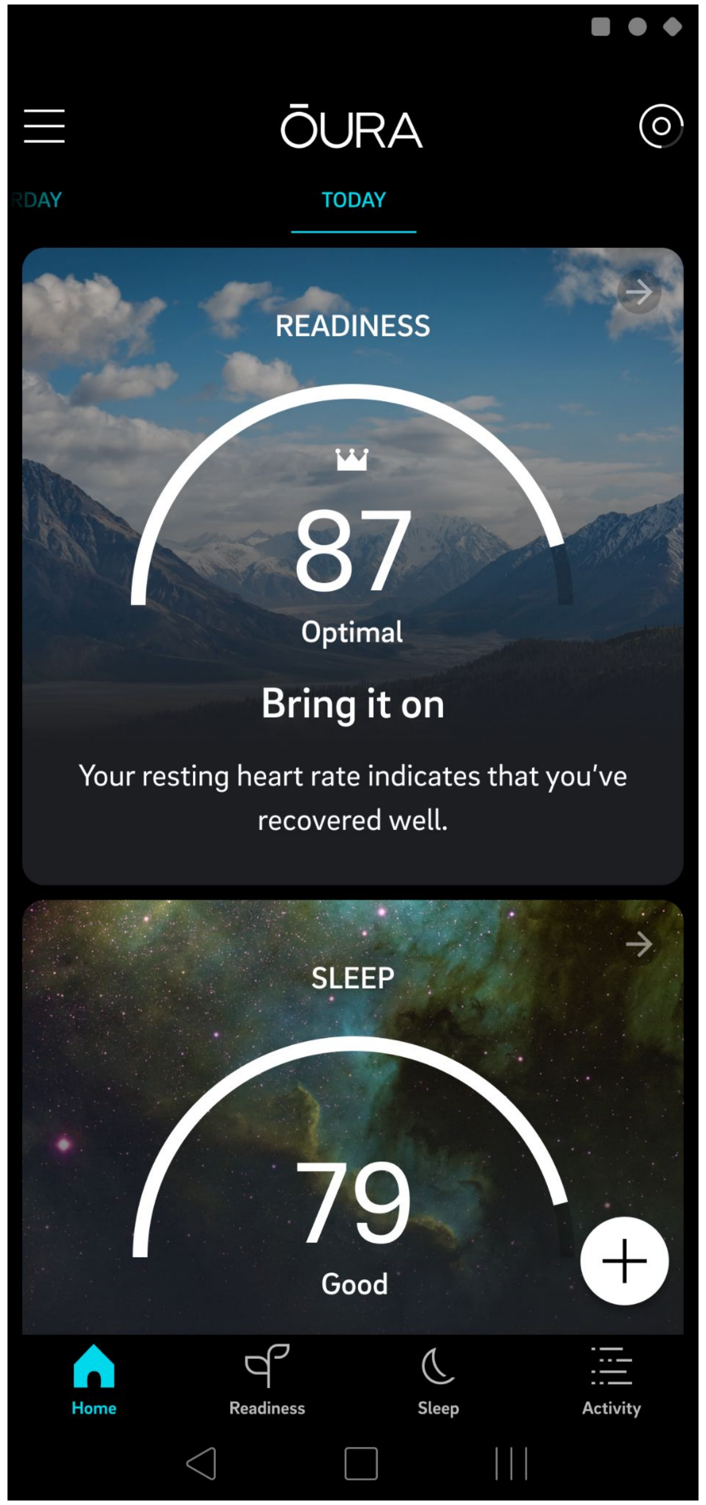
Oura’s data

Oura’s data is mostly IoT data from Oura ring, in total multiple TB and increasing quickly

When the data arrives to the cloud, it is already aggregated

The data arriving cloud might be already several days old, because ring and the mobile app can be used without internet connection





Machine learning models at Oura

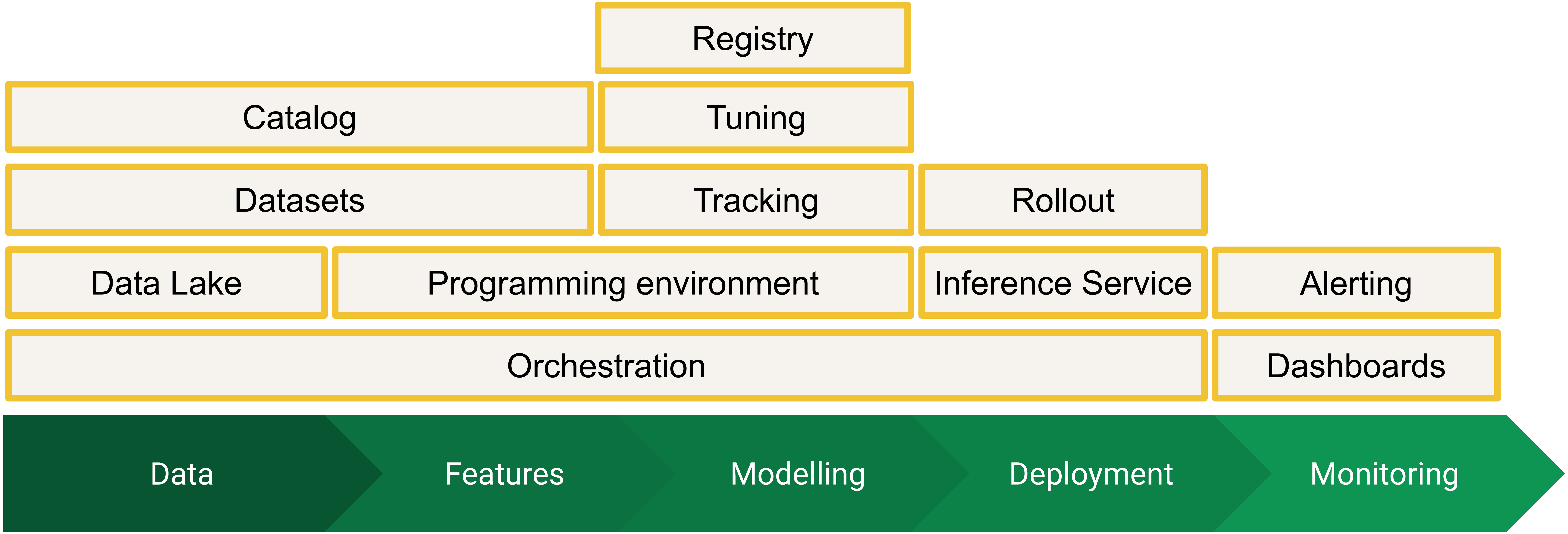
Due to the nature of our data, machine learning models are quite stable per ring release

Models running in ring, on mobile app and on cloud

When deciding whether to run the models on a device or in the cloud, we need to consider pros and cons of both approaches:

- Online vs offline usage
- Latency requirements
- Cost
- Compute power / hardware requirements

ML Stack


Registry
Catalog
Tuning
Datasets
Tracking
Rollout
Data Lake
Programming environment
Inference Service
Alerting
Orchestration
Dashboards
Data
Features
Modelling
Deployment
Monitoring

Data: DBT

Data from multiple sources ingested to data lake

To prepare data sets, we use dbt transformation tooling

DBT together with Elementary enables us to monitor our data and get alerts automatically when something is up with the data

DBT docs contains documentation for the data, including data schemas and dependencies between data

Data Lake

Alerting

Orchestration

Datasets

Tracking: MLFlow

Open source platform for tracking machine learning training runs / experiments

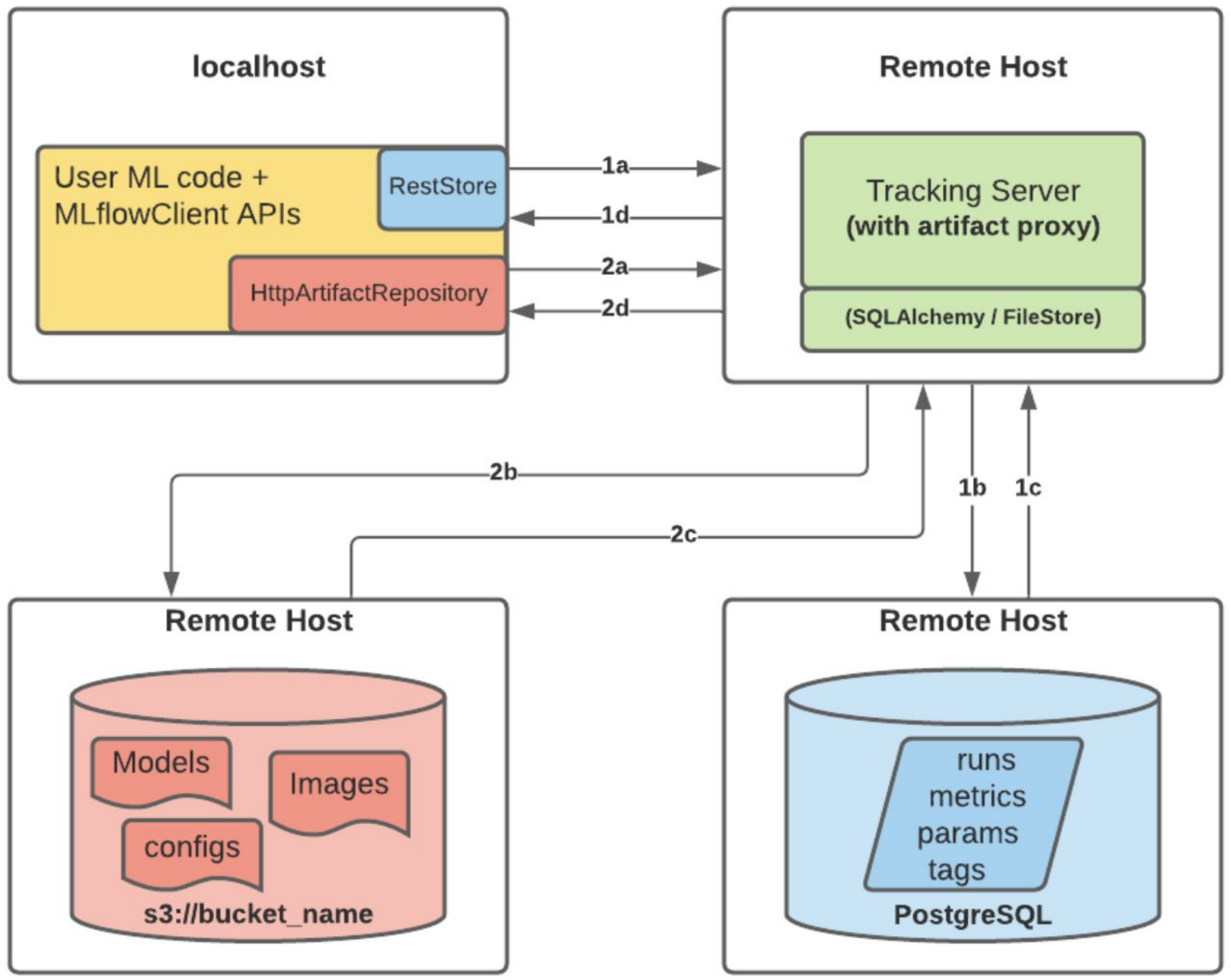
Stores metadata about the runs, metrics, artifacts

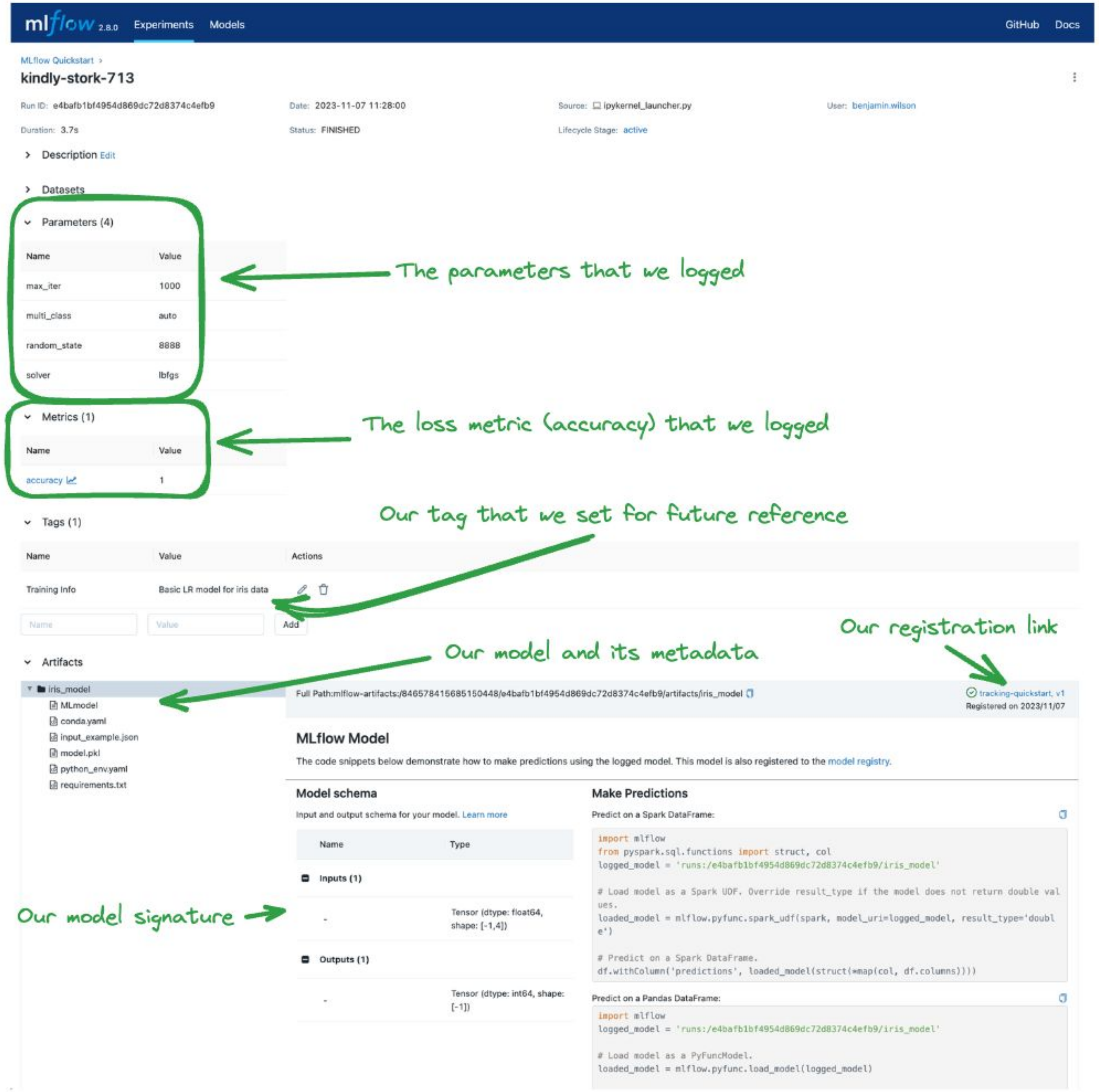
Can be also used as model registry

API + UI

Tracking

Registry





Screenshot source: https://mlflow.org/docs/latest/getting-started/intro-quickstart/index.html

› Datasets

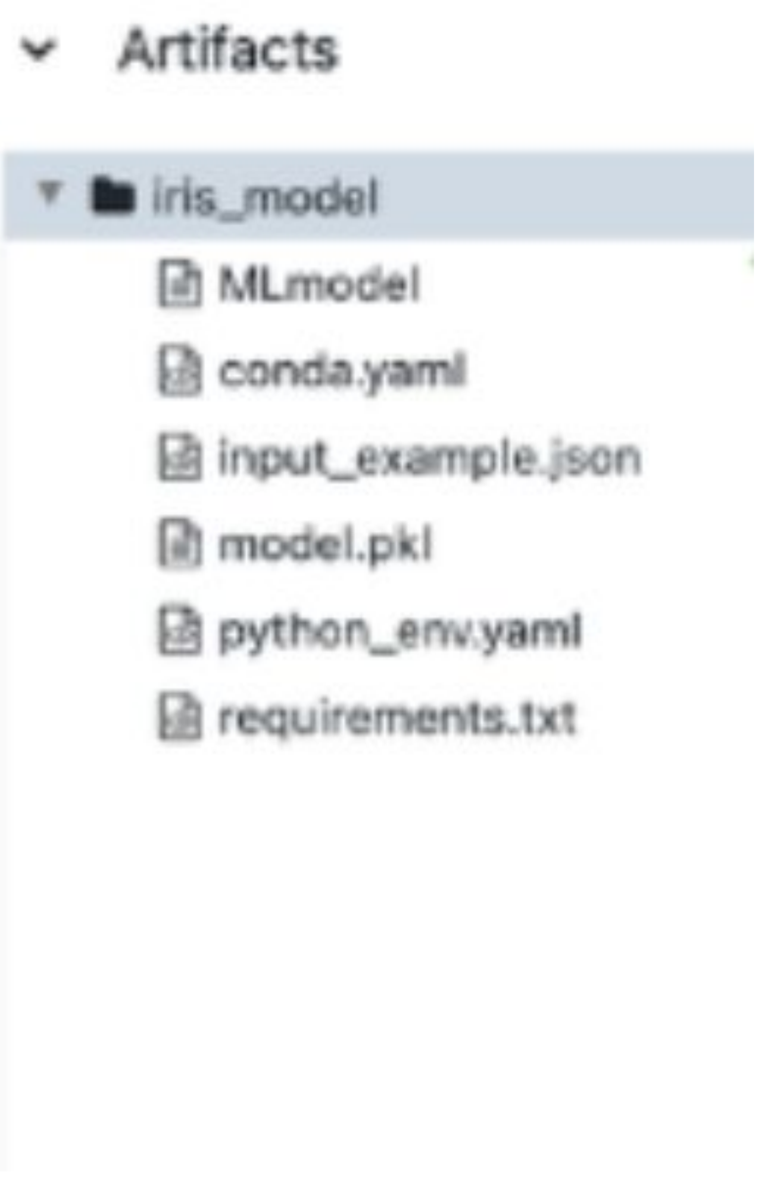
⌄ Parameters (4)

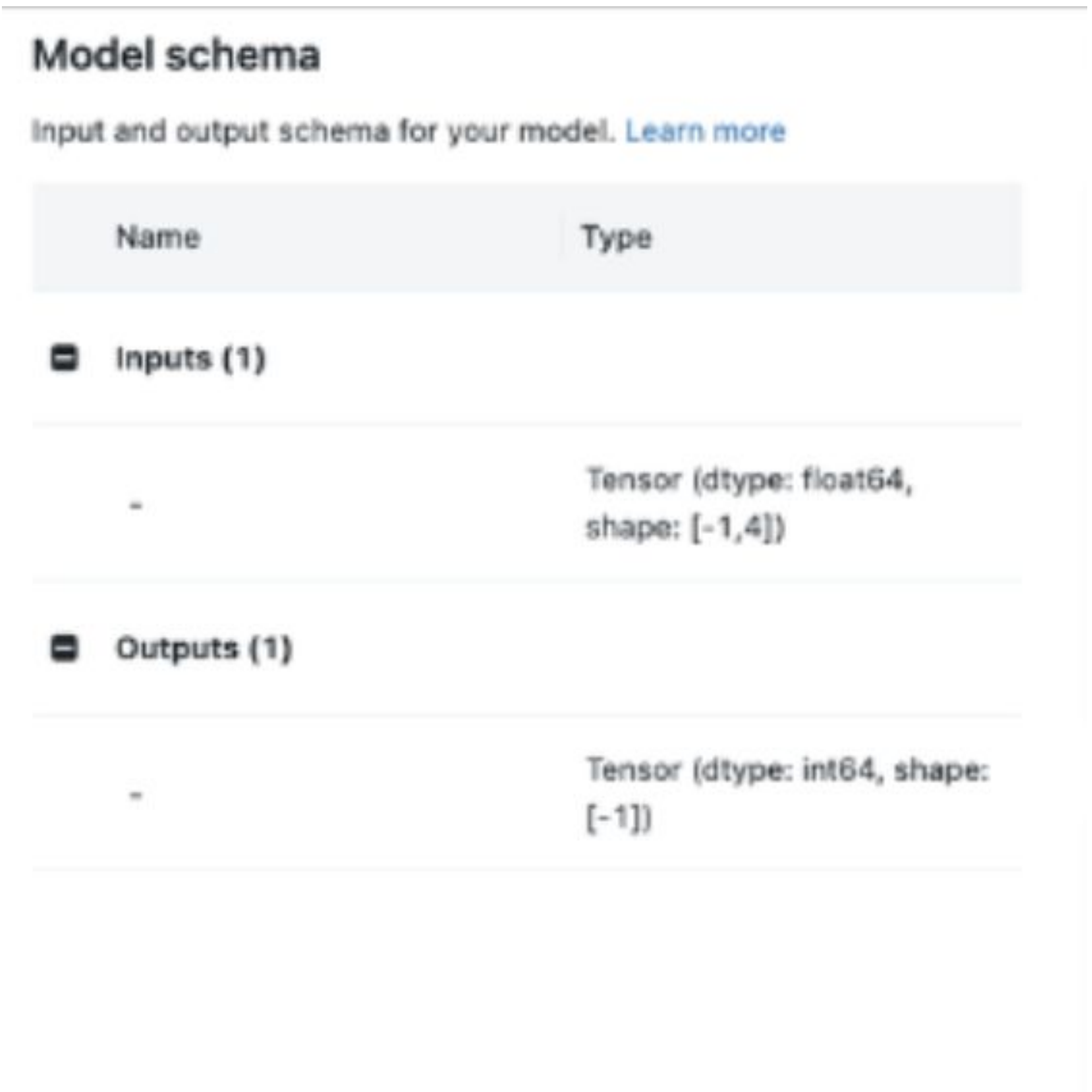
Name	Value
max_iter	1000
multi_class	auto
random_state	8888
solver	lbfgs

⌄ Metrics (1)

Name	Value
accuracy	1

Screenshot source:
https://mlflow.org/docs/latest/getting-started/intro-quickstart/index.html





Screenshot source:
https://mlflow.org/docs/latest/getting-started/intro-quickstart/index.html

Orchestration: Flyte

Orchestration

Registry

Tuning

Workflow tooling that enables easy remote execution for code

Distributed computation: speed up, scale easily

Storing artifacts

Works well with MLFlow

Plain python, organise to workflows and tasks

```python
@task
def train_iris_model(iris_dataset: pd.DataFrame) -> dict:
    y = iris_dataset[target_col]
    X = iris_dataset[features]
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.5, random_state=42
    )
    log_reg = LogisticRegression(max_iter=1000)
    log_reg.fit(X_train, y_train)

    test_prediction = log_reg.predict(X_test)
    return metrics.classification_report(
        y_train, test_prediction, digits=3, output_dict=True
    )

@workflow
def test_workflow() -> dict:
    df = load_iris_dataset()
    visualize_iris_data(iris_data=df)
    df_chunks = create_chunks(iris_dataset=df, chunk_amount=3)
    processed_dfs = map_task(process_iris_data)(iris_data=df_chunks)
    full_df = combine_data(chunks=processed_dfs)
    return train_iris_model(iris_dataset=full_df)
```

SUCCEEDED flyte-test/development/iris_example.test_workflow/**fdb023ebfabc344be87e**

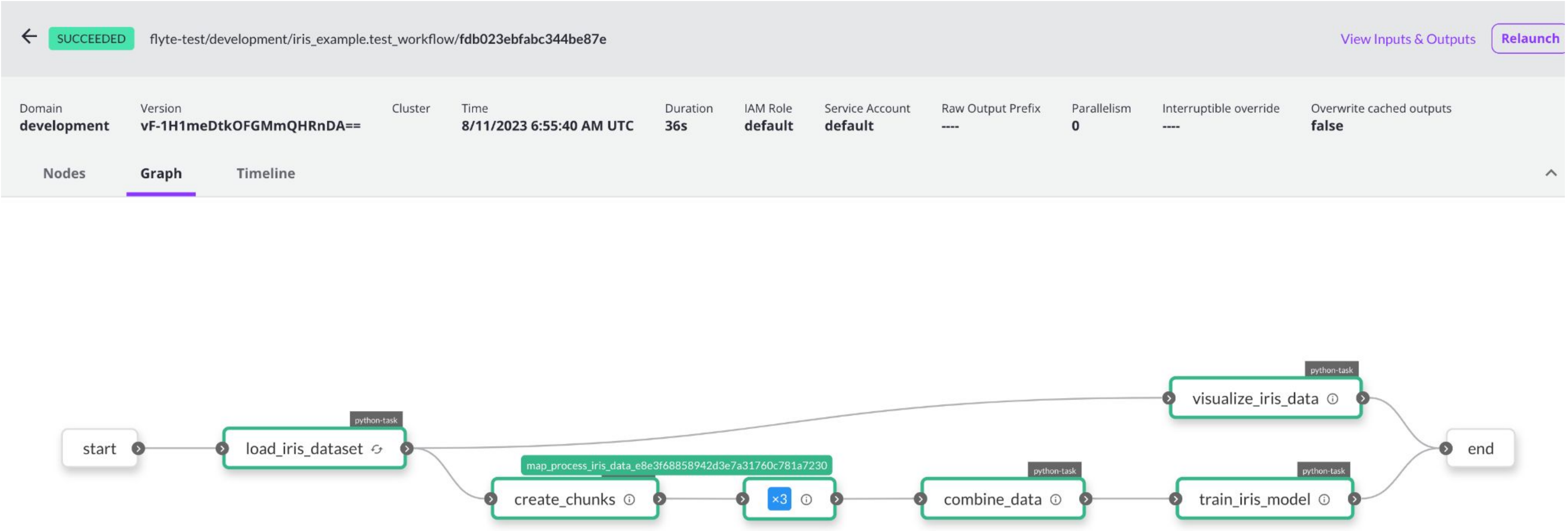
View Inputs & Outputs | **Relaunch**

Domain	Version	Cluster	Time	Duration	IAM Role	Service Account	Raw Output Prefix	Parallelism	Interruptible override	Overwrite cached outputs
development	**vF-1H1meDtkOFGMmQHRnDA==**		**8/11/2023 6:55:40 AM UTC**	**36s**	**default**	**default**	----	**0**	----	**false**

Nodes | Graph | Timeline

Status | Start Time | Duration

load_iris_dataset iris_example.load_iris_dataset	n0	Python Task	SUCCEEDED		
visualize_iris_data iris_example.visualize_iris_data	n1	Python Task	SUCCEEDED	8/11/2023 6:55:41 AM UTC 8/11/2023 9:55:41 AM EEST	8s
create_chunks iris_example.create_chunks	n2	Python Task	SUCCEEDED	8/11/2023 6:55:41 AM UTC 8/11/2023 9:55:41 AM EEST	8s
map_process_iris_data_e8e3f68858942d3e7a31760c781a7230 iris_example.map_process_iris_data_e8e3f68858942d3e7a31760c781a7230	n3	Map Task	SUCCEEDED	8/11/2023 6:55:51 AM UTC 8/11/2023 9:55:51 AM EEST	9s
combine_data iris_example.combine_data	n4	Python Task	SUCCEEDED	8/11/2023 6:56:01 AM UTC 8/11/2023 9:56:01 AM EEST	7s
train_iris_model iris_example.train_iris_model	n5	Python Task	SUCCEEDED	8/11/2023 6:56:09 AM UTC 8/11/2023 9:56:09 AM EEST	7s


SUCCEEDED
flyte-test/development/iris_example.test_workflow/fdb023ebfabc344be87e
View Inputs & Outputs
Relaunch
Domain
development
Version
vF-1H1meDtkOFGMmQHRnDA==
Cluster
Time
8/11/2023 6:55:40 AM UTC
Duration
36s
IAM Role
default
Service Account
default
Raw Output Prefix

Parallelism
0
Interruptible override

Overwrite cached outputs
false
Nodes
Graph
Timeline
start
python-task
load_iris_dataset
map_process_iris_data_e8e3f68858942d3e7a31760c781a7230
create_chunks
×3
python-task
visualize_iris_data
python-task
combine_data
python-task
train_iris_model
end

Challenges at Oura

Running models in ring, mobile, cloud

All of them have unique challenges:
- Updating models in ring / mobile
- Models that work both on iOS and android
- Selected technologies need to be relatively easy for the data scientists to learn
- Monitoring for models in mobile or ring
- Supporting multiple versions in cloud
- Privacy and security

Dependencies between models

Dependencies between predictions and user metrics

End to end testing with ring, mobile and cloud

Debugging of the models

Q&A