KubeCon | CloudNativeCon

North America 2019

KubeCon | CloudNativeCon
North America 2019

From Notebook to Kubeflow Pipelines

An End-to-End Data Science Workflow

Michelle Casbon, Google *@texasmichelle*

Stefano Fioravanzo, FBK *@sfioravanzo*

Ilias Katsakioris, Arrikto *@elikatsis*

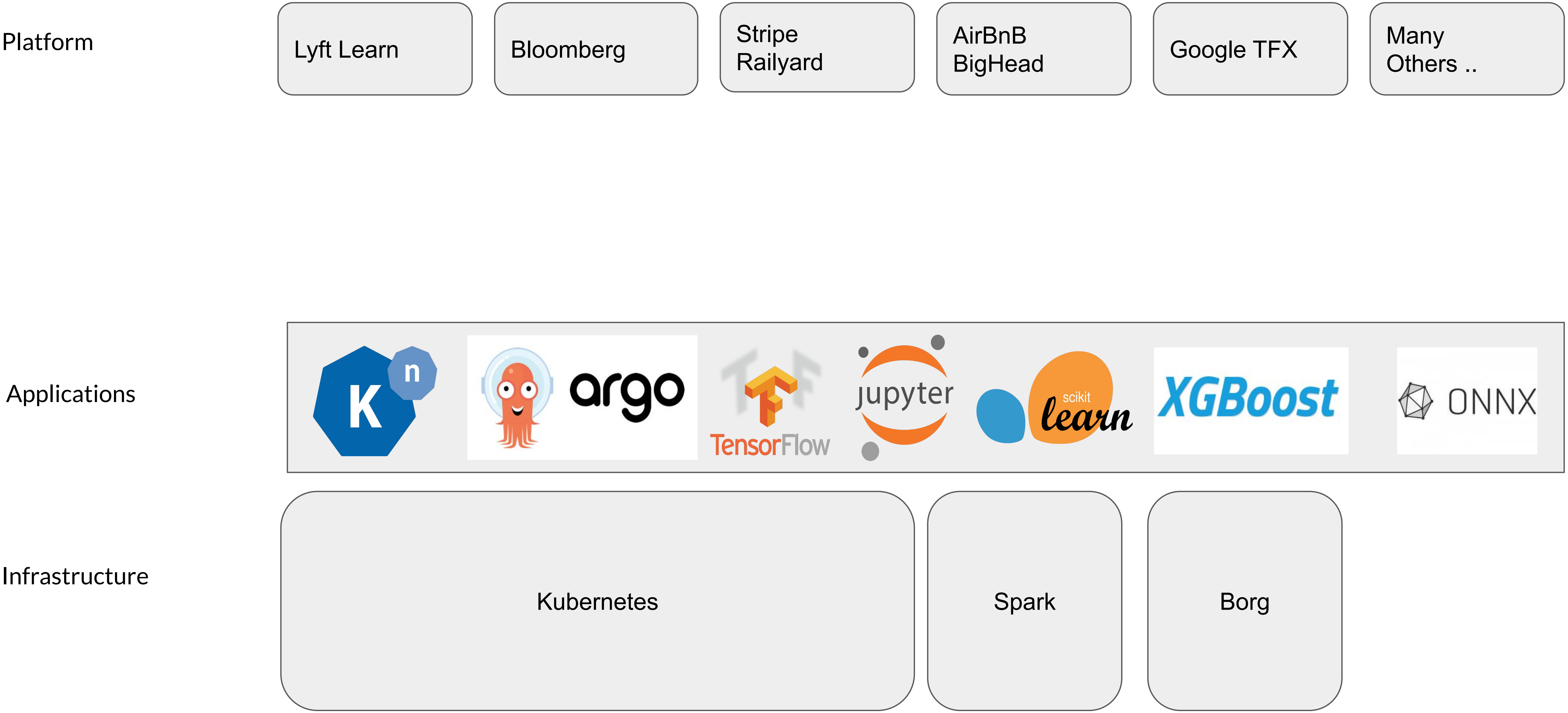
What is Kubeflow

The Kubeflow project is dedicated to making deployments of machine learning (ML) workflows on Kubernetes: simple, portable and scalable.

Why Kubeflow

- End-to-end solution for ML on Kubernetes
- Containerized workload
- Experiment & exploration with state-of-the-art AI technologies
- Easy on-boarding
- Outstanding community and industry support

Platforms Critical to Success With ML



Platform

Lyft Learn

Bloomberg

Stripe Railyard

AirBnB BigHead

Google TFX

Many Others ..

Applications

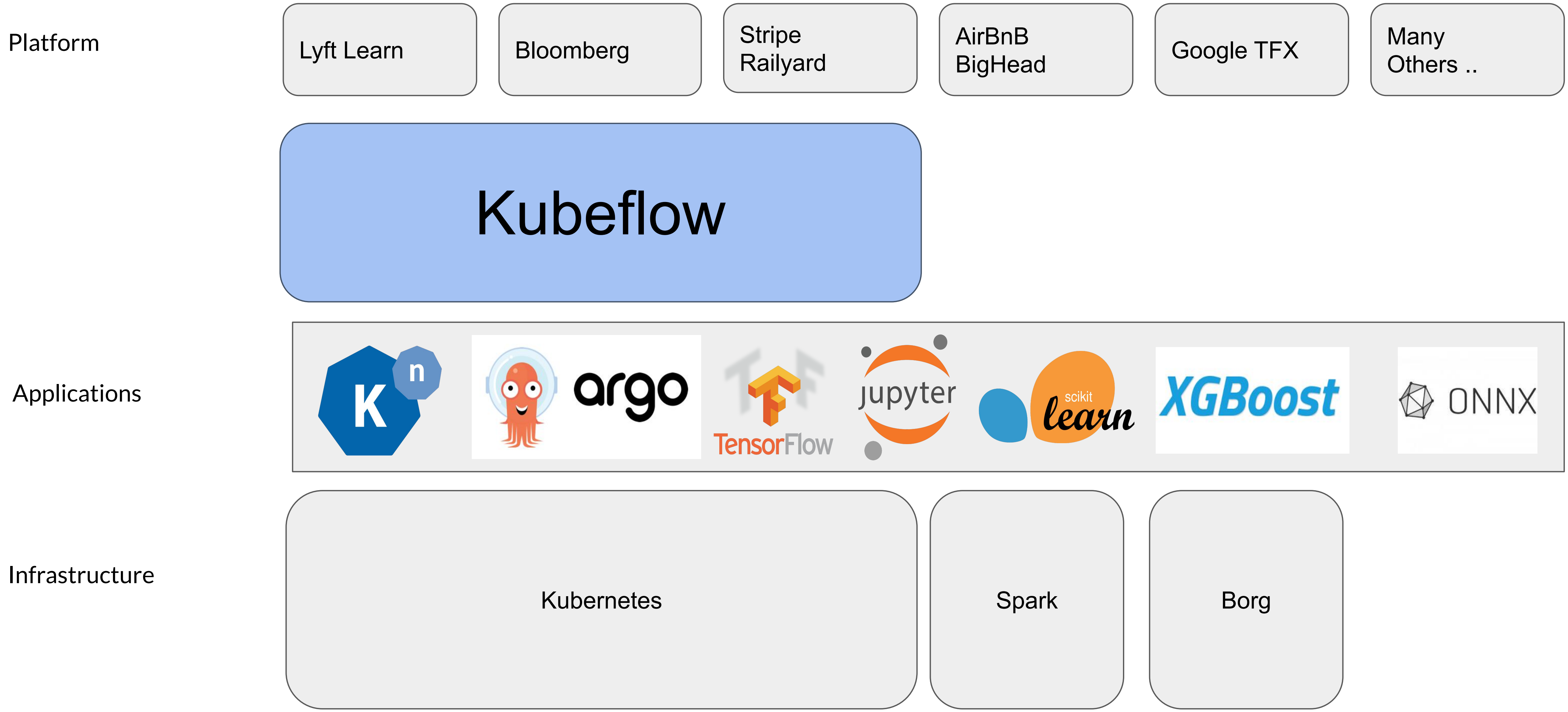
Infrastructure

Kubernetes

Spark

Borg

An Open Platform For Everyone



Platform

Lyft Learn

Bloomberg

Stripe Railyard

AirBnB BigHead

Google TFX

Many Others ..

Kubeflow

Applications

Infrastructure

Kubernetes

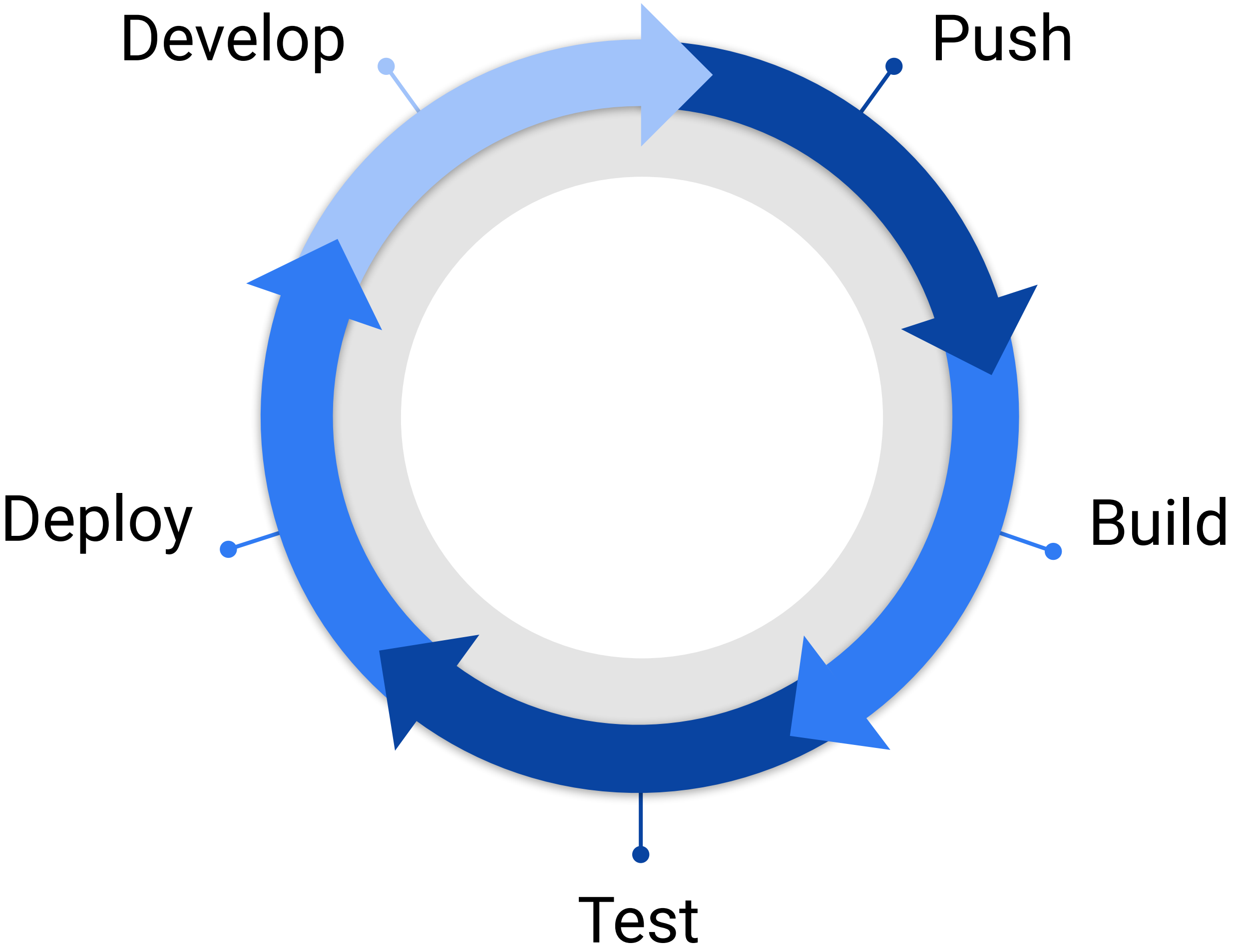
Spark

Borg

ML Applications Are Distributed Systems

"Hidden Technical Debt in Machine Learning Systems"
https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

CI/CD Critical For Managing Complexity

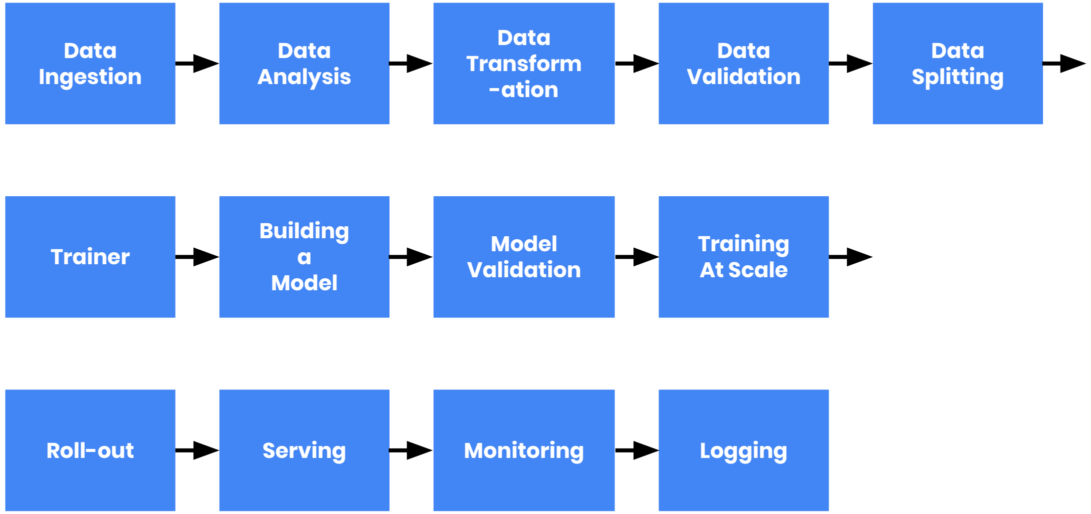


Data Science with Kubeflow

Kubeflow Pipelines exists because Data Science and ML are inherently **pipeline processes**

This workshop will focus on two essential aspects:

- **Low barrier to entry**: deploy a Jupyter Notebook to Kubeflow Pipelines in the Cloud using a fully GUI-based approach
- **Reproducibility**: automatic data versioning to enable reproducibility and better collaboration between data scientists

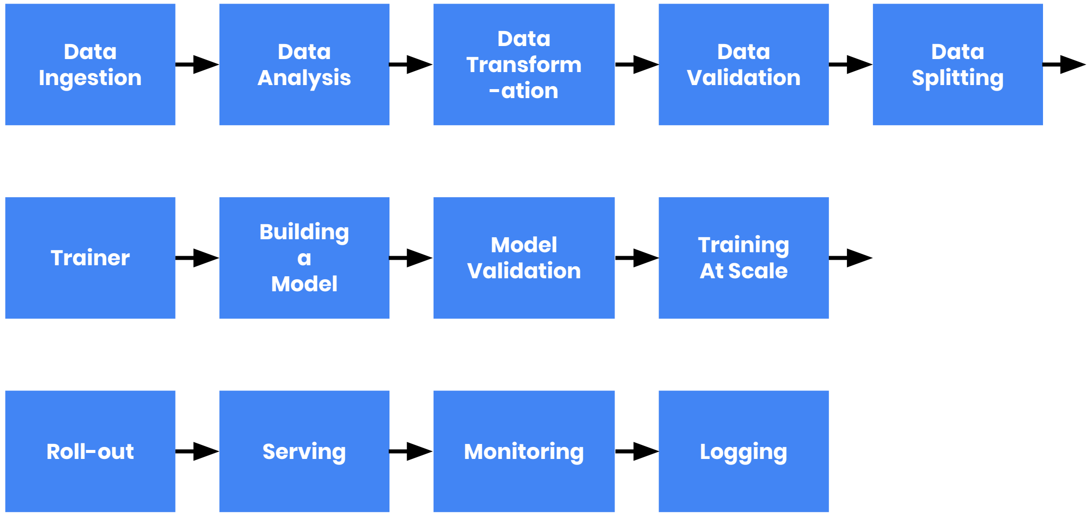


Data Science with Kubeflow

Kubeflow Pipelines exists because Data Science and ML are inherently **pipeline processes**

This workshop will focus o spects:

- **Low barrier to entry**: Kale Notebook to Kubeflow Pipelines on the cloud using a fully GUI-based approach
- **Reproducibility**: Arrikto ersioning to enable reproducibility and better collaboration between data scientists

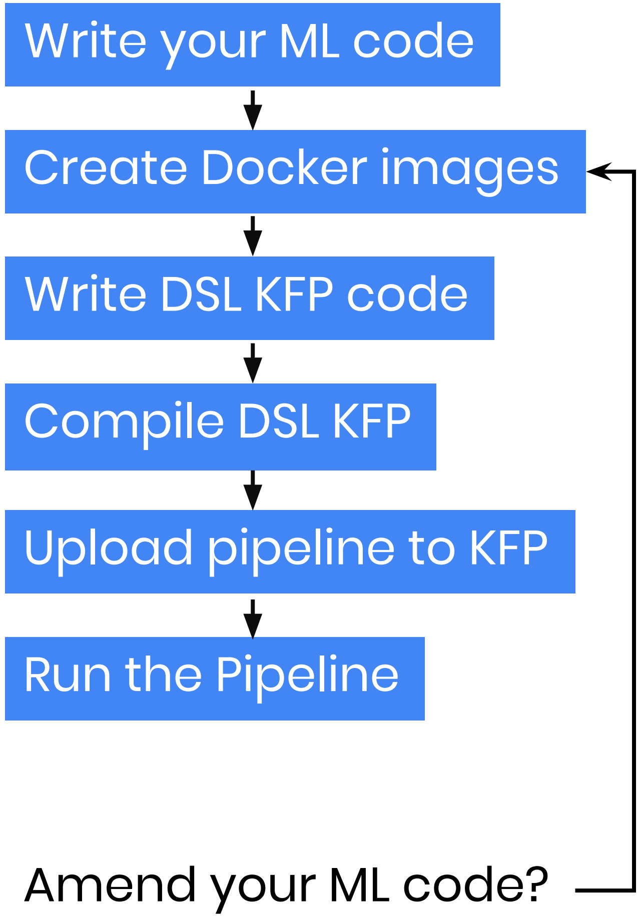


Benefits of running a Notebook as a Pipeline

- The steps of the workflow are clearly defined
- Parallelization & isolation
    - Hyperparameter tuning
- Data versioning
- Different infrastructure requirements
    - Different hardware (GPU/CPU)

Workflow

Before



Write your ML code

Create Docker images

Write DSL KFP code

Compile DSL KFP

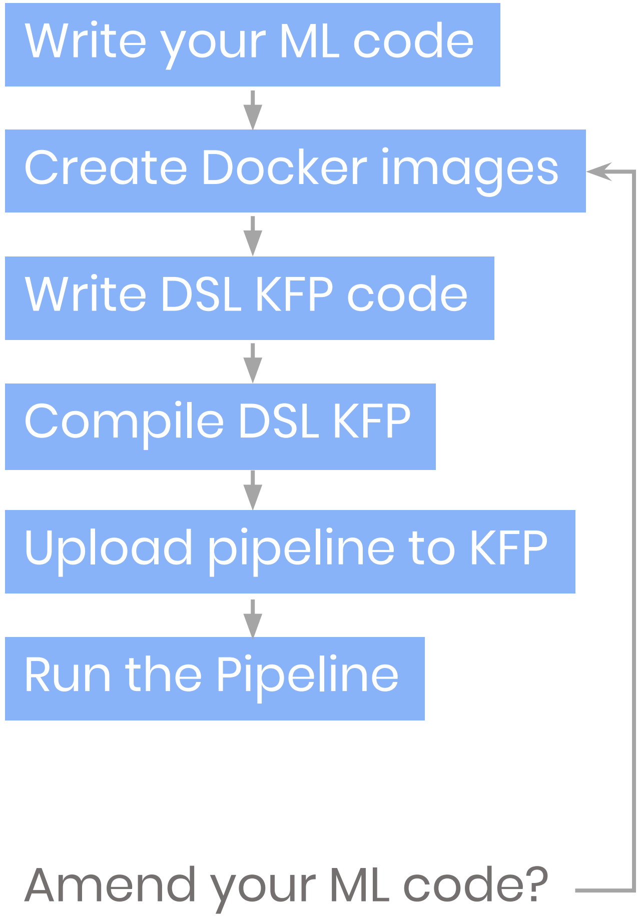
Upload pipeline to KFP

Run the Pipeline

Amend your ML code?

Workflow

Before



Write your ML code

Create Docker images

Write DSL KFP code

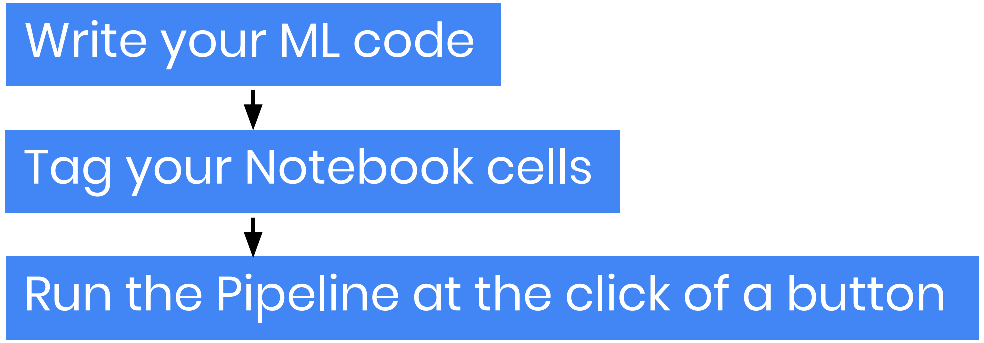
Compile DSL KFP

Upload pipeline to KFP

Run the Pipeline

Amend your ML code?

After



Write your ML code

Tag your Notebook cells

Run the Pipeline at the click of a button

Amend your ML code? ⟶ Just edit your Notebook!

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1. Set up GCP and install MiniKF

2. Explore the ML code of the Titanic challenge

3. Convert notebook to a Kubeflow pipeline

4. Reproducibility with Volume Snapshots

5. Debugging the pipeline

6. Clean up

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1. Set up GCP and install MiniKF
2. Explore the ML code of the Titanic challenge
3. Convert notebook to a Kubeflow pipeline
4. Reproducibility with Volume Snapshots
5. Debugging the pipeline
6. Clean up

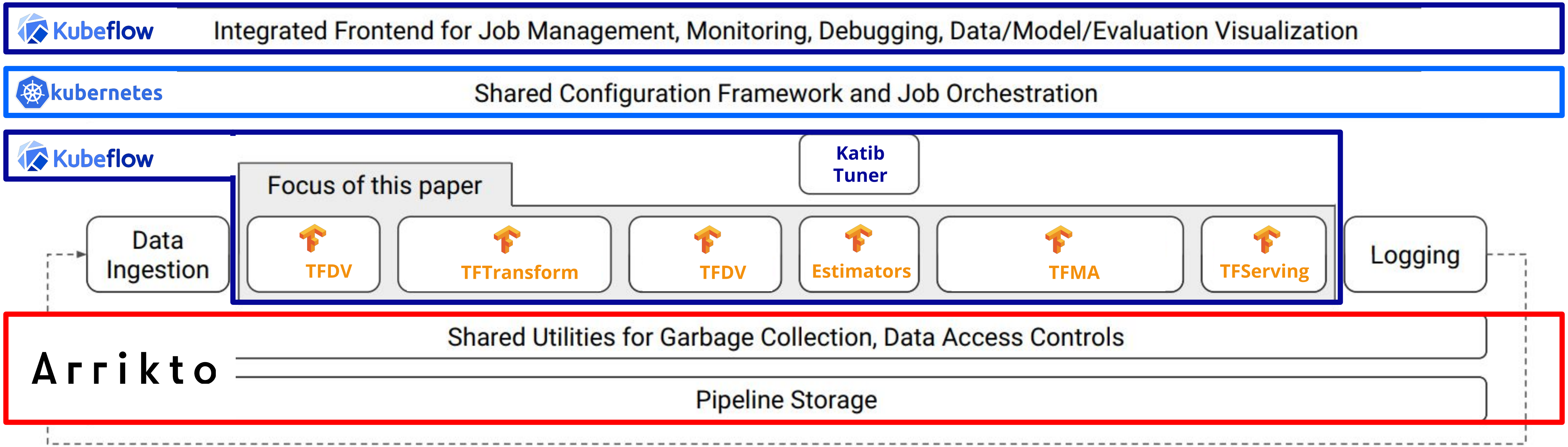
What is MiniKF?

- Kubeflow on GCP, your laptop, or on-prem infrastructure in just a few minutes
- All-in-one, single-node, Kubeflow distribution
- Very easy to spin up on your own environment on-prem or in the cloud
- MiniKF = MiniKube + Kubeflow + Arrikto’s Rok Data Management Platform

Arrikto Rok

KDD 2017 Applied Data Science Paper

KDD’17, August 13–17, 2017, Halifax, NS, Canada



Kubeflow — Integrated Frontend for Job Management, Monitoring, Debugging, Data/Model/Evaluation Visualization

kubernetes — Shared Configuration Framework and Job Orchestration

Kubeflow

Katib Tuner

Focus of this paper

Data Ingestion

TFDV

TFTransform

TFDV

Estimators

TFMA

TFServing

Logging

Arrikto

Shared Utilities for Garbage Collection, Data Access Controls

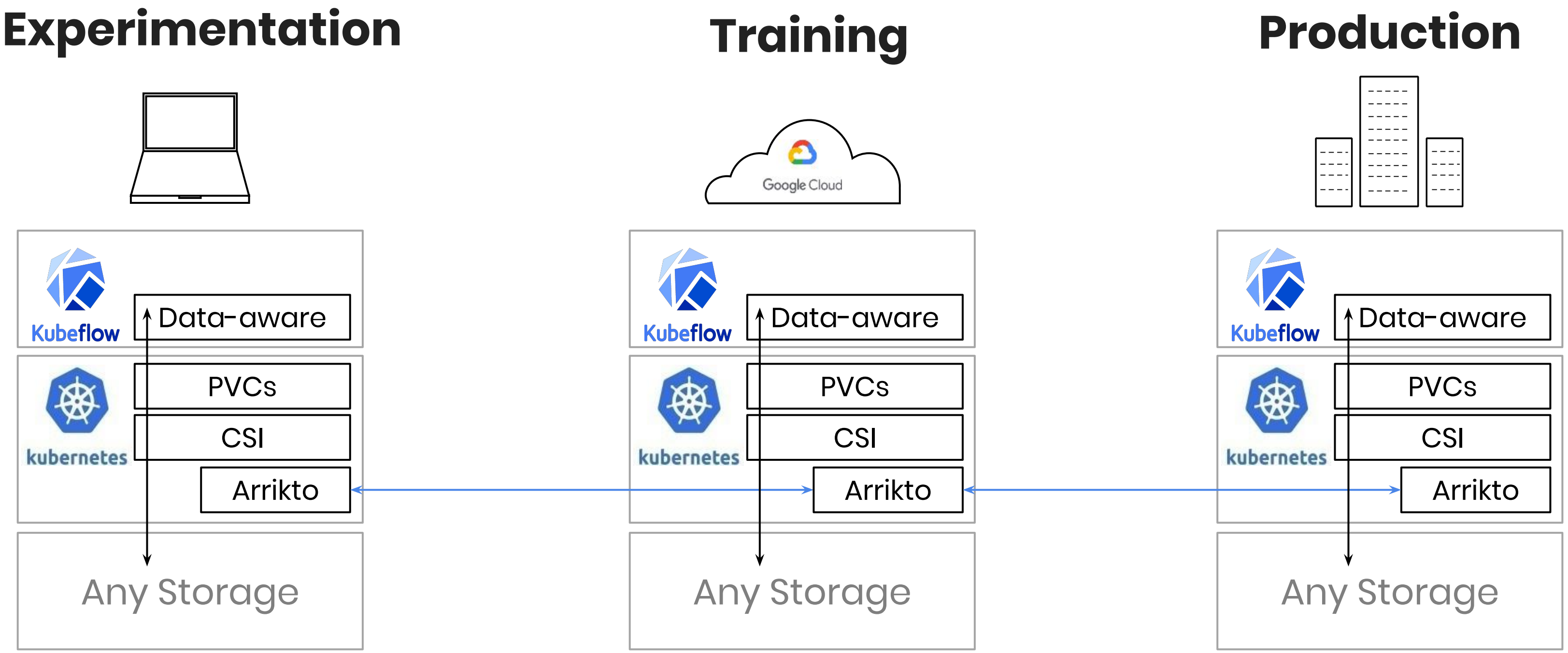
Pipeline Storage

Figure 1: High-level component overview of a machine learning platform.

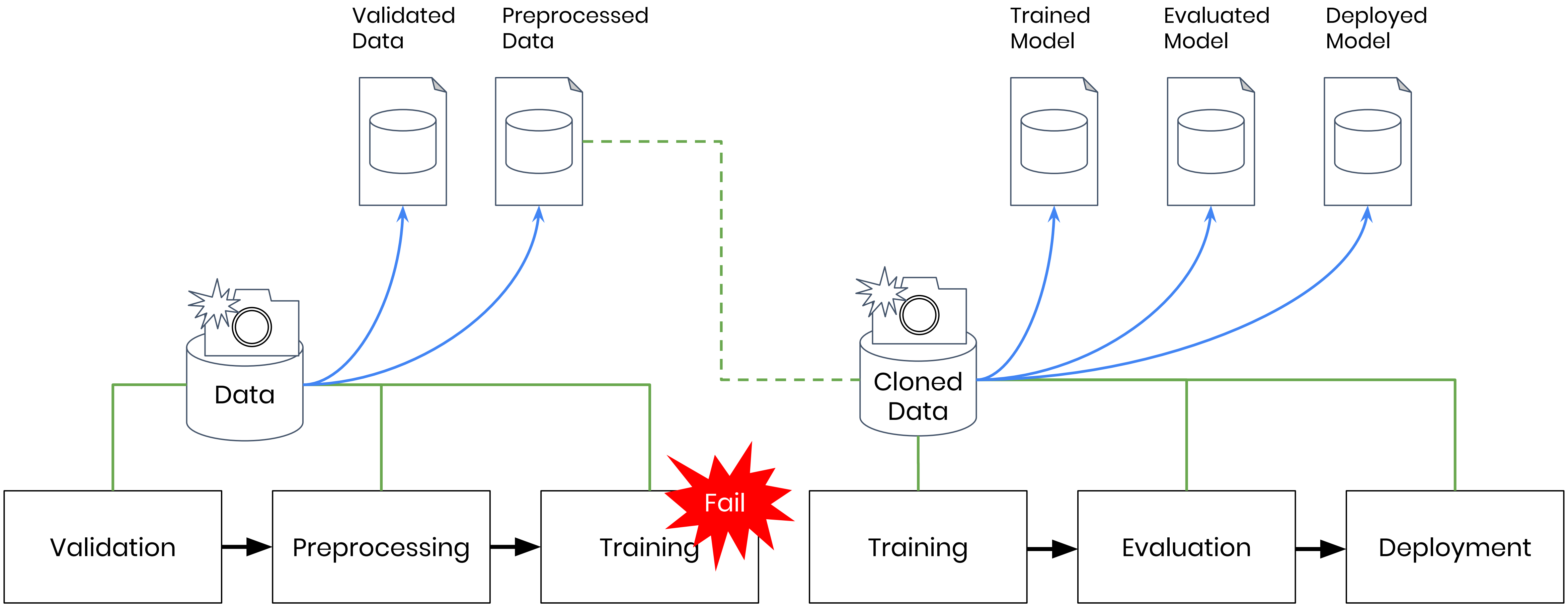
Arrikto Rok

Data Versioning, Packaging, and Sharing

Across teams and cloud boundaries for complete Reproducibility, Provenance, and Portability



Arrikto Rok


Validated
Data
Preprocessed
Data
Trained
Model
Evaluated
Model
Deployed
Model
Data
Cloned
Data
Fail
Validation
Preprocessing
Training
Training
Evaluation
Deployment

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1. Set up GCP and install MiniKF
2. Explore the ML code of the Titanic challenge
3. Convert notebook to a Kubeflow pipeline
4. Reproducibility with Volume Snapshots
5. Debugging the pipeline
6. Clean up

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1. Set up GCP and install MiniKF

2. Explore the ML code of the Titanic challenge

3. Convert notebook to a Kubeflow pipeline

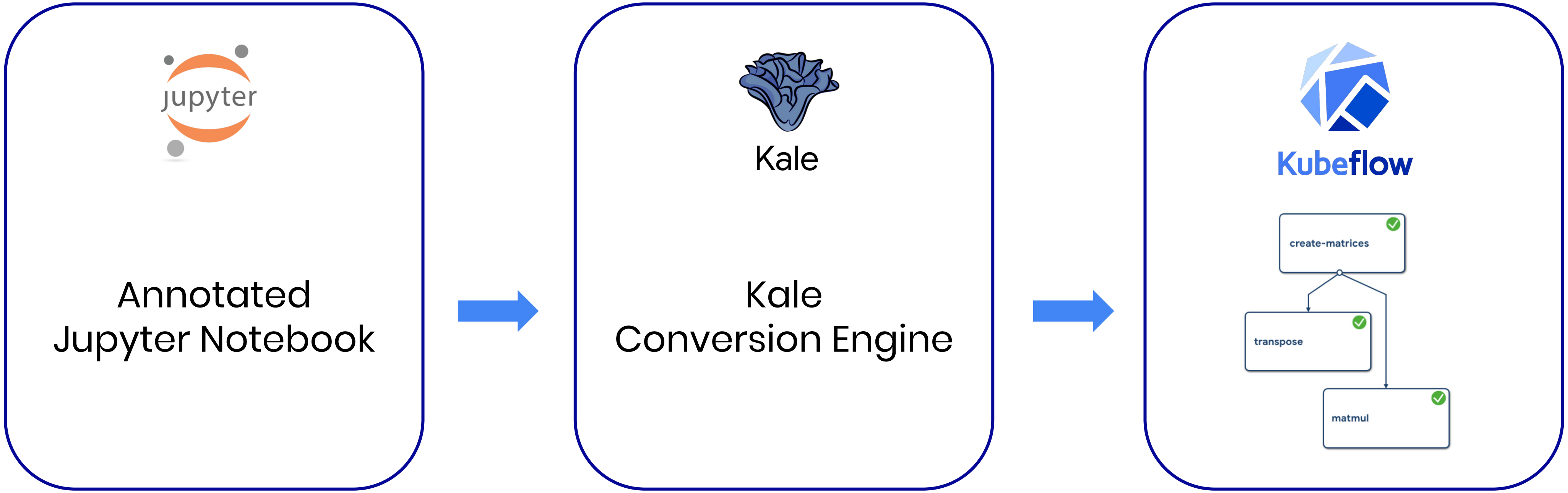
4. Reproducibility with Volume Snapshots

5. Debugging the pipeline

6. Clean up

KALE – **K**ubeflow **A**utomated Pipe**L**ines **E**ngine

- Python package + JupyterLab extension
- Convert a Jupyter Notebook to a KFP workflow
- No need for Kubeflow SDK

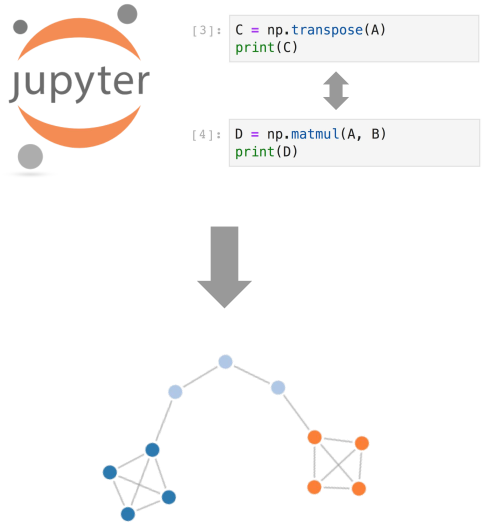


Annotated Jupyter Notebook

Kale Conversion Engine

KALE – Modules

nbparser

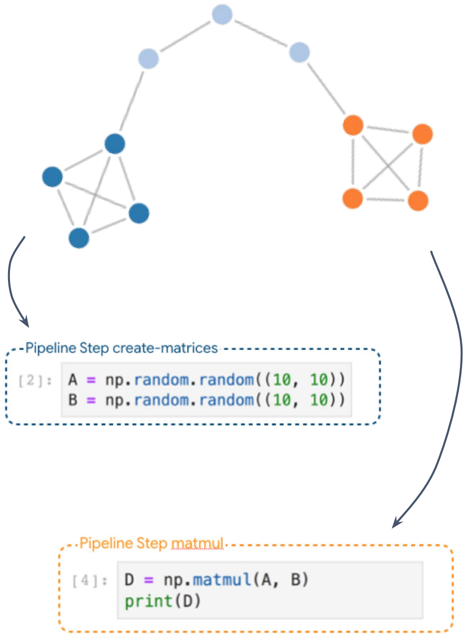


```
[3]: C = np.transpose(A)
     print(C)
```

```
[4]: D = np.matmul(A, B)
     print(D)
```

Derive pipeline structure

static_analyzer



Pipeline Step create-matrices

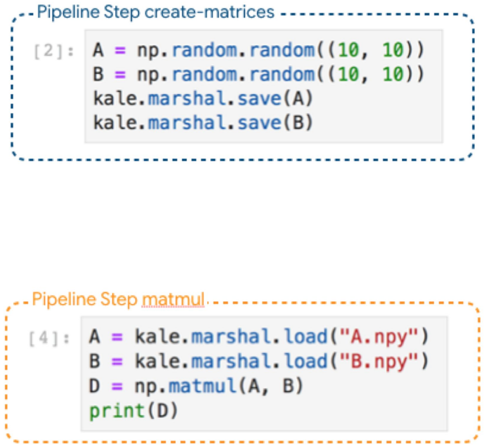
```
[2]: A = np.random.random((10, 10))
     B = np.random.random((10, 10))
```

Pipeline Step matmul

```
[4]: D = np.matmul(A, B)
     print(D)
```

Identify dependencies

marshal



Pipeline Step create-matrices

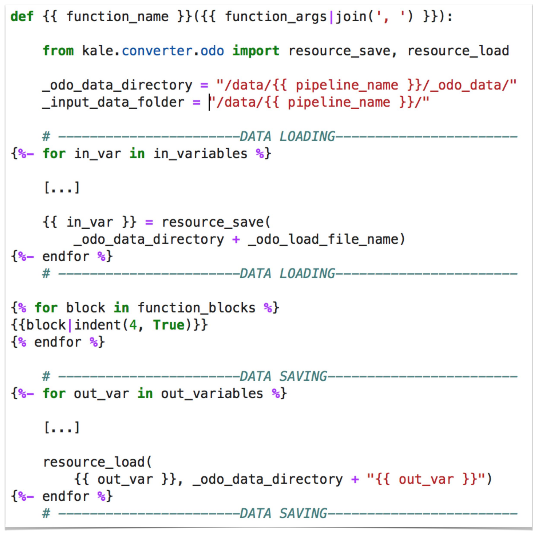
```
[2]: A = np.random.random((10, 10))
     B = np.random.random((10, 10))
     kale.marshal.save(A)
     kale.marshal.save(B)
```

Pipeline Step matmul

```
[4]: A = kale.marshal.load("A.npy")
     B = kale.marshal.load("B.npy")
     D = np.matmul(A, B)
     print(D)
```

Inject data objects

codegen

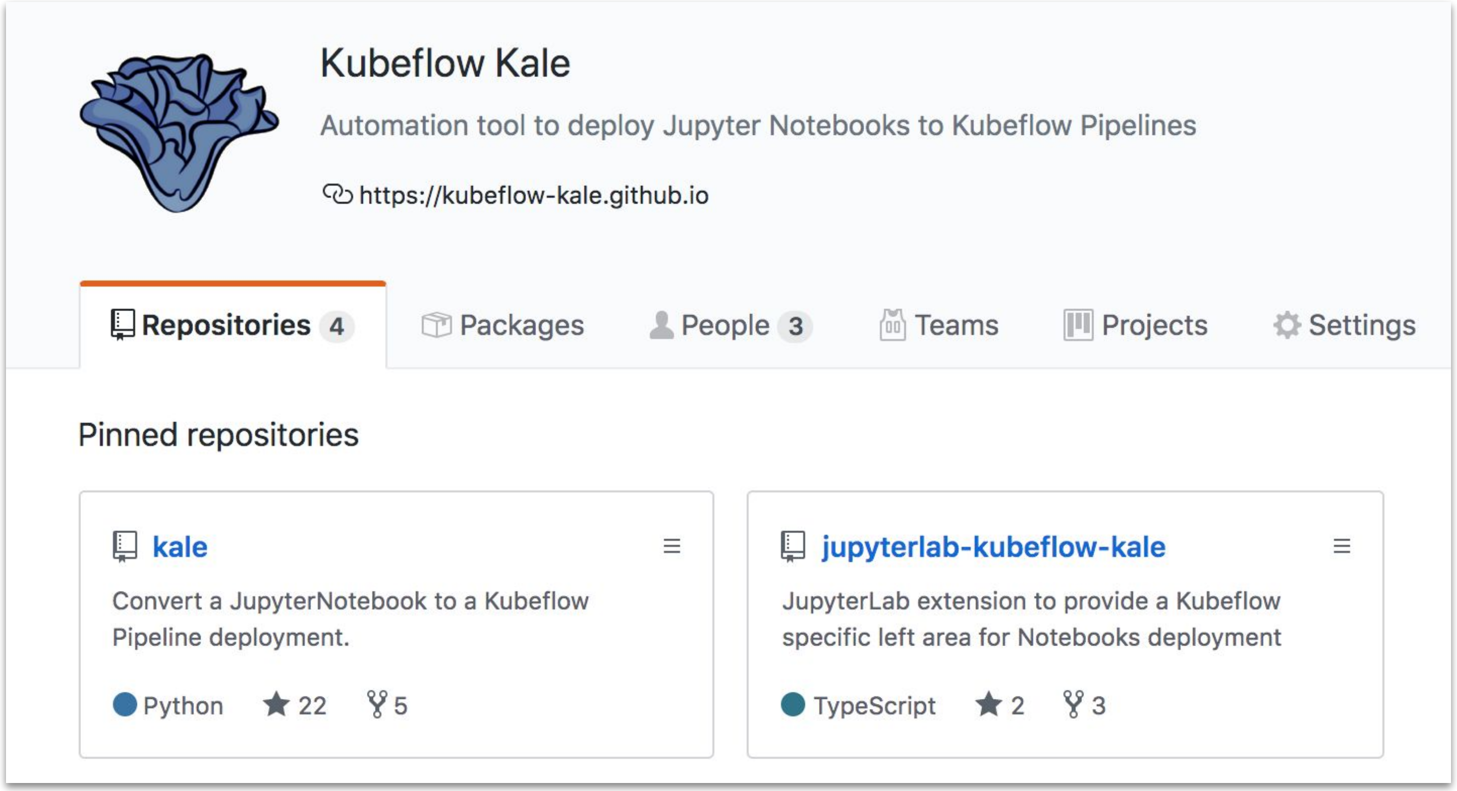


```
def {{ function_name }}({{ function_args|join(', ') }}):

    from kale.converter.odo import resource_save, resource_load

    _odo_data_directory = "/data/{{ pipeline_name }}/_odo_data/"
    _input_data_folder = "/data/{{ pipeline_name }}/"

    # -----------------------DATA LOADING-----------------------
{%- for in_var in in_variables %}

    [...]

    {{ in_var }} = resource_save(
        _odo_data_directory + _odo_load_file_name)
{%- endfor %}
    # -----------------------DATA LOADING-----------------------

{% for block in function_blocks %}
{{block|indent(4, True)}}
{% endfor %}

    # -----------------------DATA SAVING-----------------------
{%- for out_var in out_variables %}

    [...]

    resource_load(
        {{ out_var }}, _odo_data_directory + "{{ out_var }}")
{%- endfor %}
    # -----------------------DATA SAVING-----------------------
```

Generate & deploy pipeline

Contribute!

github.com/kubeflow-kale



Kubeflow Kale

Automation tool to deploy Jupyter Notebooks to Kubeflow Pipelines

https://kubeflow-kale.github.io

Pinned repositories

kale

Convert a JupyterNotebook to a Kubeflow Pipeline deployment.

● Python ★ 22 5

jupyterlab-kubeflow-kale

JupyterLab extension to provide a Kubeflow specific left area for Notebooks deployment

● TypeScript ★ 2 3

Kale Intro on Medium: https://bit.ly/2qjXXhF

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1. Set up GCP and install MiniKF
2. Explore the ML code of the Titanic challenge
3. Convert notebook to a Kubeflow pipeline
4. Reproducibility with Volume Snapshots
5. Debugging the pipeline
6. Clean up

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1
Set up GCP and install MiniKF

2
Explore the ML code of the Titanic challenge

3
Convert notebook to a Kubeflow pipeline

4
Reproducibility with Volume Snapshots

5
Debugging the pipeline

6
Clean up

Agenda

g.co/codelabs/kubeflow-minikf-kale

Zones:
us-central1-*
us-west1-*
us-west2-*

1
Set up GCP and install MiniKF

2
Explore the ML code of the Titanic challenge

3
Convert notebook to a Kubeflow pipeline

4
Reproducibility with Volume Snapshots

5
Debugging the pipeline

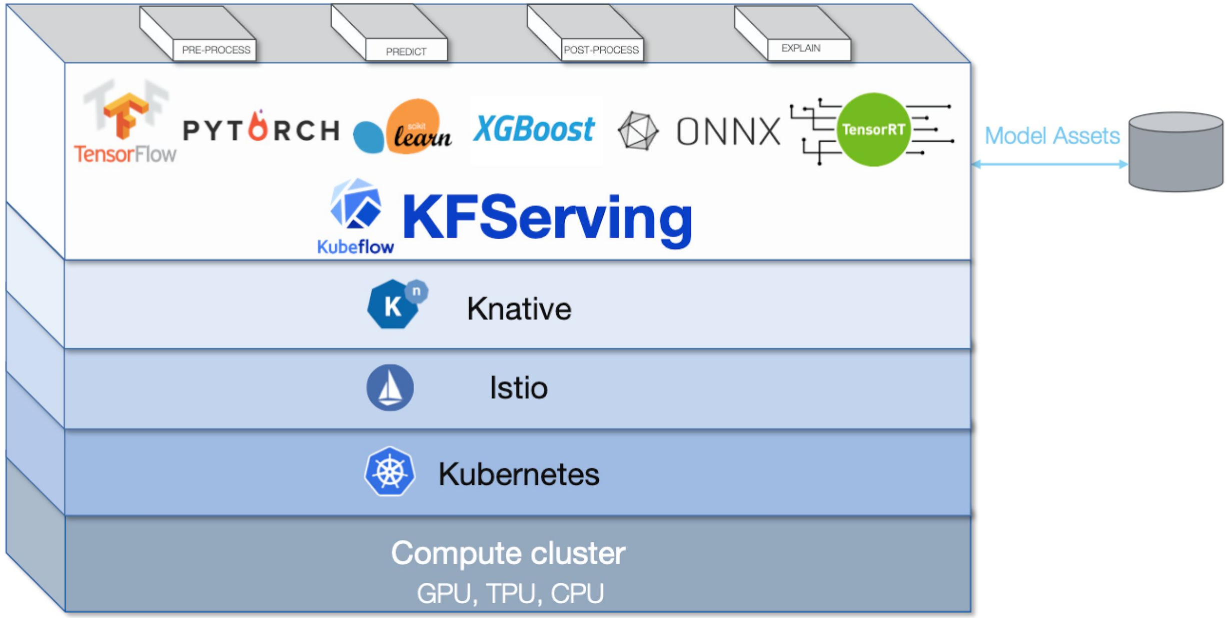
6
Clean up

What’s new in v0.7

- KFServing for model deployment and management
- kfctl simpler syntax - deploy with 1 command

```
kfctl apply  -f kfdef.yaml
```

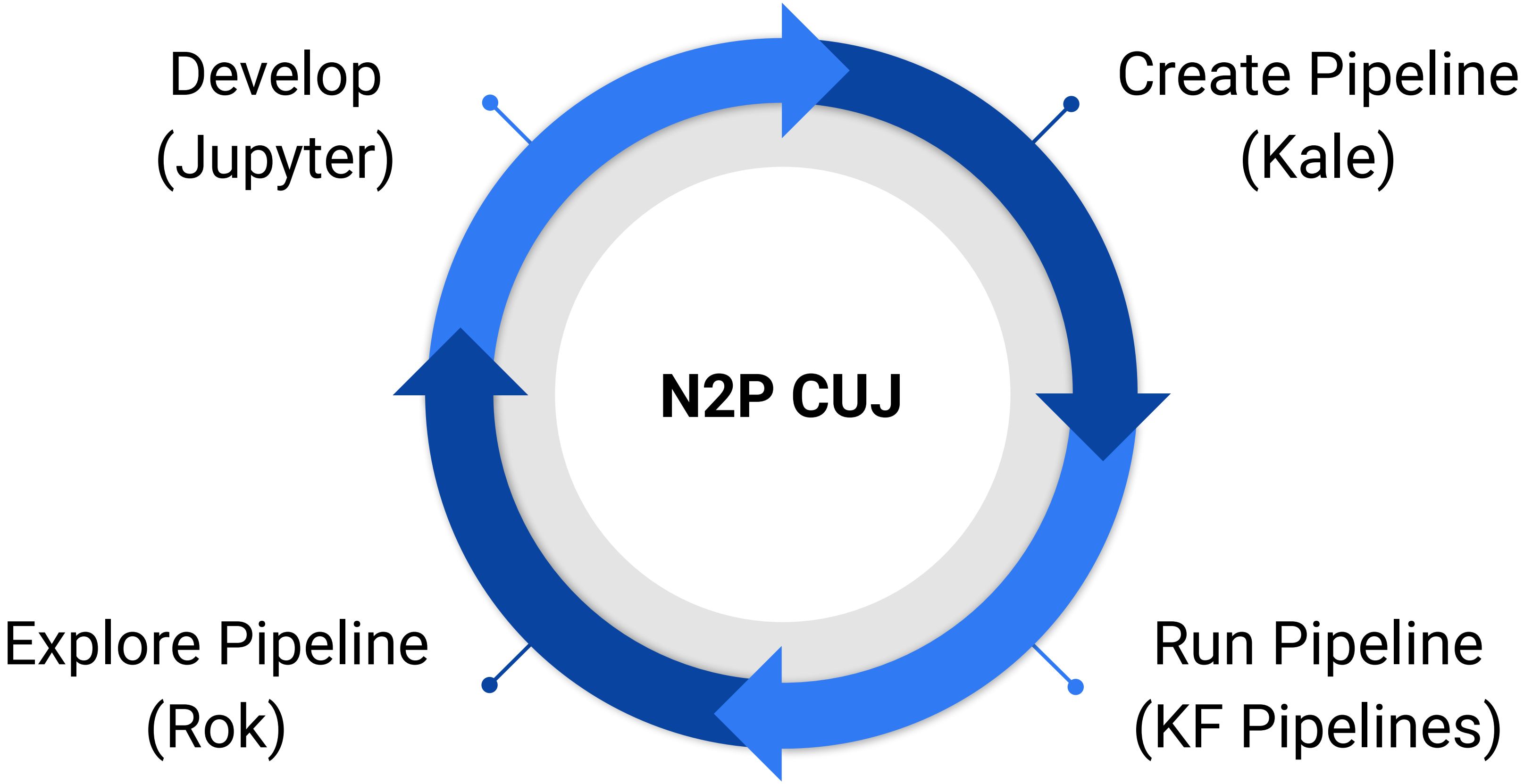
- Improved multi-user support
  - Aggregated roles
- Hyperparameter tuning
  - A “Suggestions CR” that provides suggestions to improve experiments
  - A more robust metric collector and prometheus runtime metrics and counters
  - More back-end database options



What's new in v0.7

- Pipelines
  - Performance improvements
  - Automatic metadata logging for TFX pipelines
  - New looping constructs withItems and withParams

Notebook-to-Pipeline CUJ



N2P CUJ

Develop
(Jupyter)

Create Pipeline
(Kale)

Run Pipeline
(KF Pipelines)

Explore Pipeline
(Rok)

Ecosystem-supported CUJ for Kubeflow 1.0 coming in Jan 2020

Community

Kubeflow is open

- Open community
- Open design
- Open source
- Open to ideas

Get involved

- github.com/kubeflow
- kubeflow.slack.com
- @kubeflow
- kubeflow-discuss@googlegroups.com
- Community call on Tuesdays