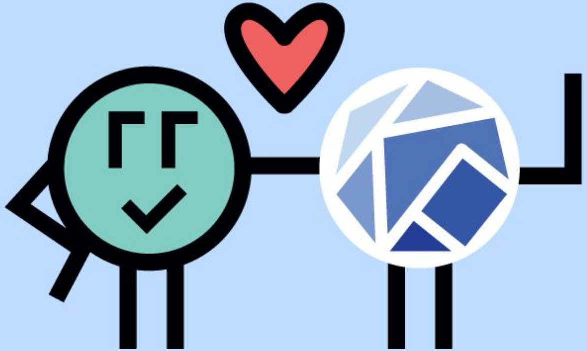
Arrikto

Scalable ML Workflows with Advanced Data Management on Kubeflow



Vangelis Koukis, Founder & CTO, Arrikto

The Problem

- Setting up an ML stack/pipeline is incredibly hard
- Setting up a production ML stack/pipeline is even harder
- Setting up an ML stack/pipeline that works across the 81% of enterprises that use multi-cloud* environments is EVEN HARDER

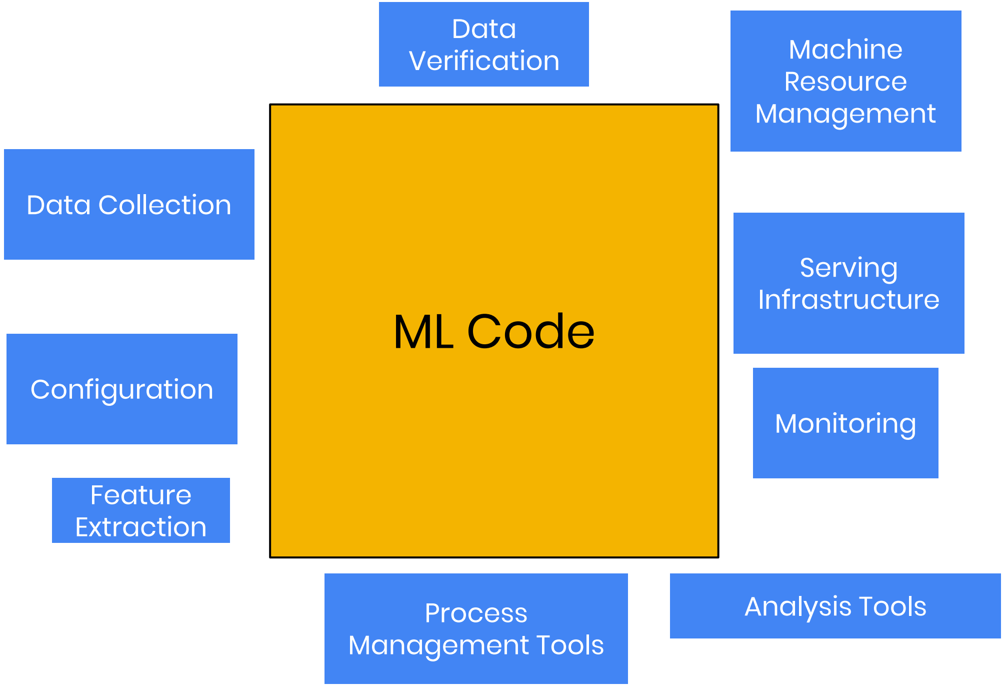
* Note: For the purposes of this presentation, "**local**" is a specific type of "**multi-cloud**"

Source: "Building an ML stack with Kubeflow" by Abhishek Gupta, Google AI Huddle - Bay Area

Arrikto

Kubeflow

Perception: ML Products are mostly about ML

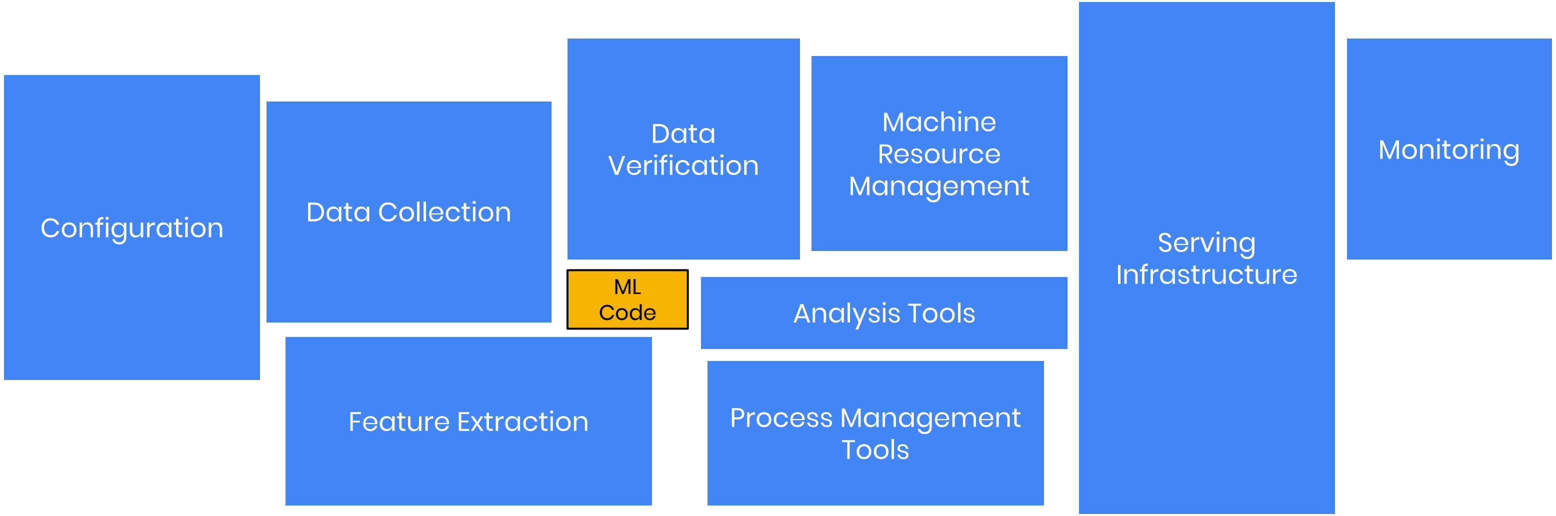


Credit: Hidden Technical Debt of Machine Learning Systems, D. Sculley, et al.

Arrikto

Kubeflow

Reality: ML Requires DevOps; lots of it



Configuration

Data Collection

Data Verification

Machine Resource Management

Serving Infrastructure

Monitoring

ML Code

Analysis Tools

Feature Extraction

Process Management Tools

Credit: Hidden Technical Debt of Machine Learning Systems, D. Sculley, et al.

Why Kubeflow

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

What is MiniKF?

- Kubeflow on your laptop or on-prem infrastructure in just a few minutes
- All-in-one, single-node, Kubeflow distribution
- Featuring the latest Kubeflow version, 0.6.x
- Very easy to spin up on your own local environment
- MiniKF = MiniKube + Kubeflow + Arrikto's Rok Data Management Platform

Arrikto

How to install MiniKF

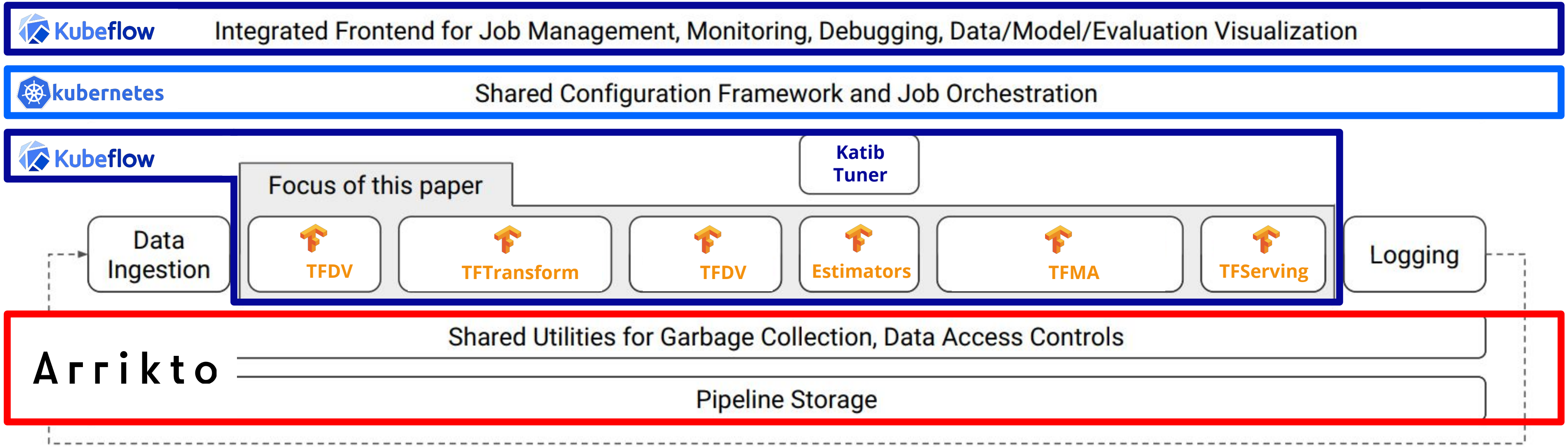
- Watch the webinar recording
- Watch the installation video
- Read the docs
- TL;DR
  - vagrant init arrikto/minikf
  - vagrant up

Arrikto

Live demo of MiniKF installation

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

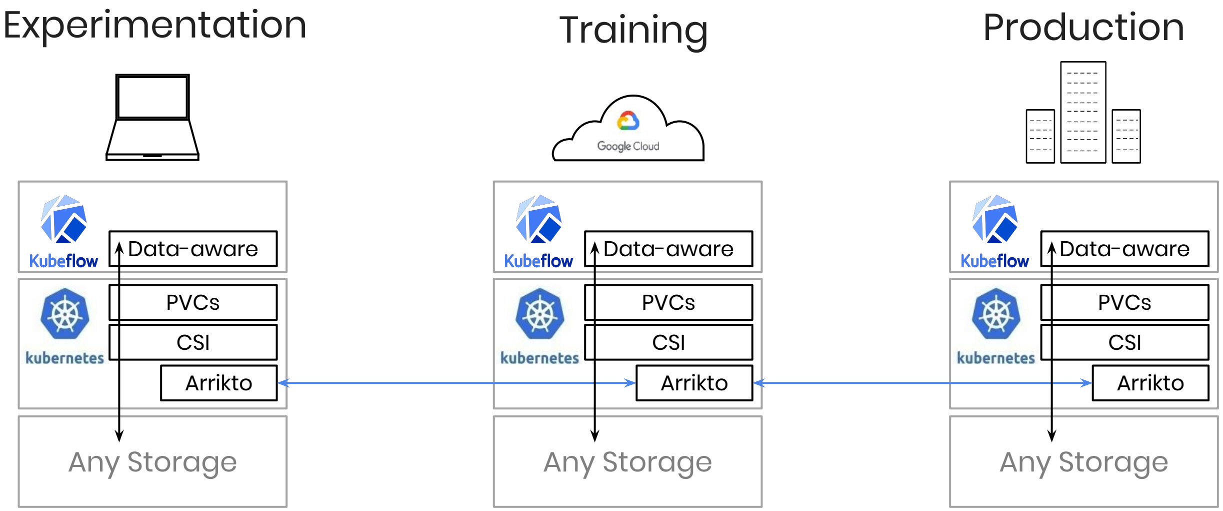
Shared Utilities for Garbage Collection, Data Access Controls

Pipeline Storage

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

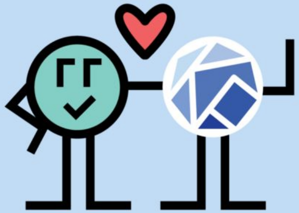
Data Versioning, Packaging, and Sharing

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

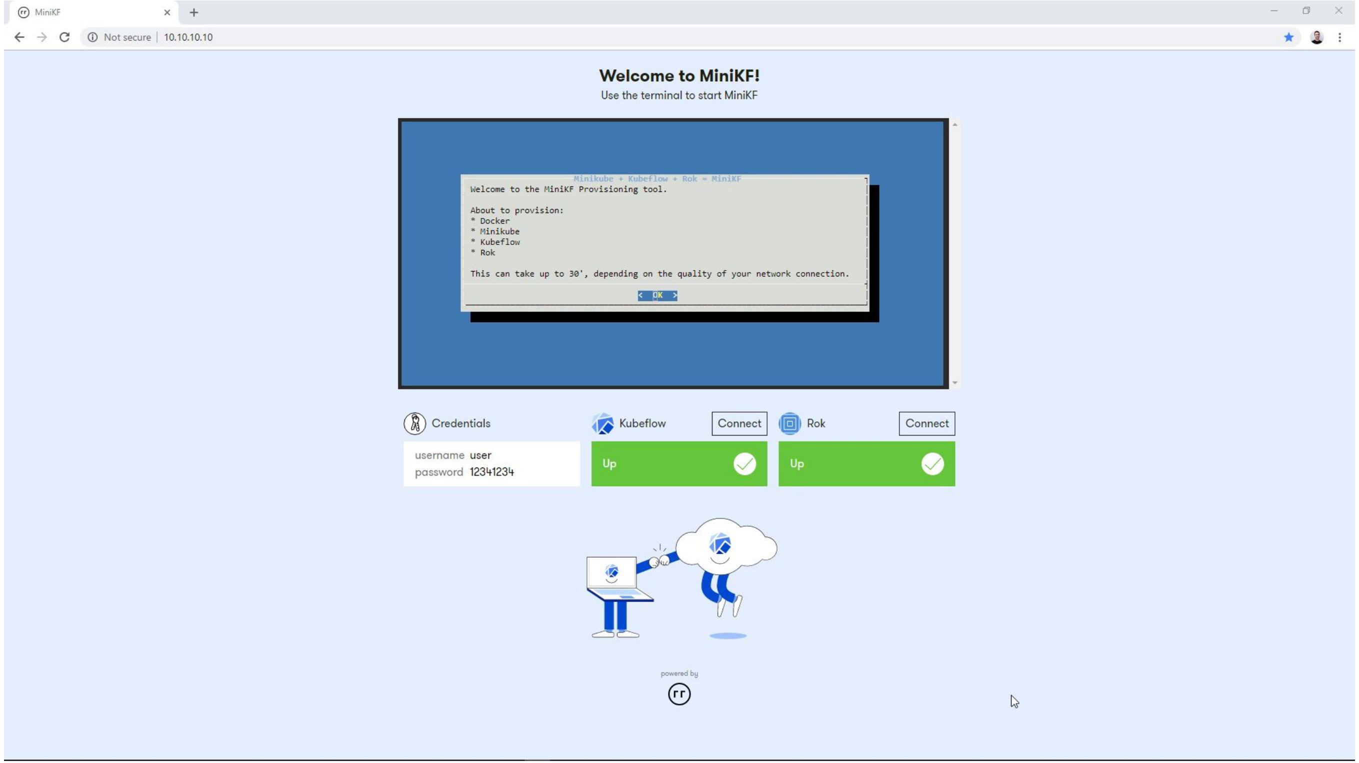


What's new in the latest MiniKF?

- Kubeflow v0.6.2
- Kubeflow authentication with Istio and Dex
- Authorization for Notebooks
- Faster, near-instantaneous snapshot restore with Rok
- Significantly improve time for snapshotting Notebooks (using Arrikto's Rok)
- Ability to snapshot every step of a pipeline (using Arrikto's Rok)



MiniKF landing page



Welcome to MiniKF!

Use the terminal to start MiniKF

```
Minikube + Kubeflow + Rok = MiniKF
Welcome to the MiniKF Provisioning tool.

About to provision:
* Docker
* Minikube
* Kubeflow
* Rok

This can take up to 30', depending on the quality of your network connection.

<  OK  >
```

Credentials

username user
password 12341234

Kubeflow Connect

Up ✓

Rok Connect

Up ✓

Arrikto

Why we started MiniKF

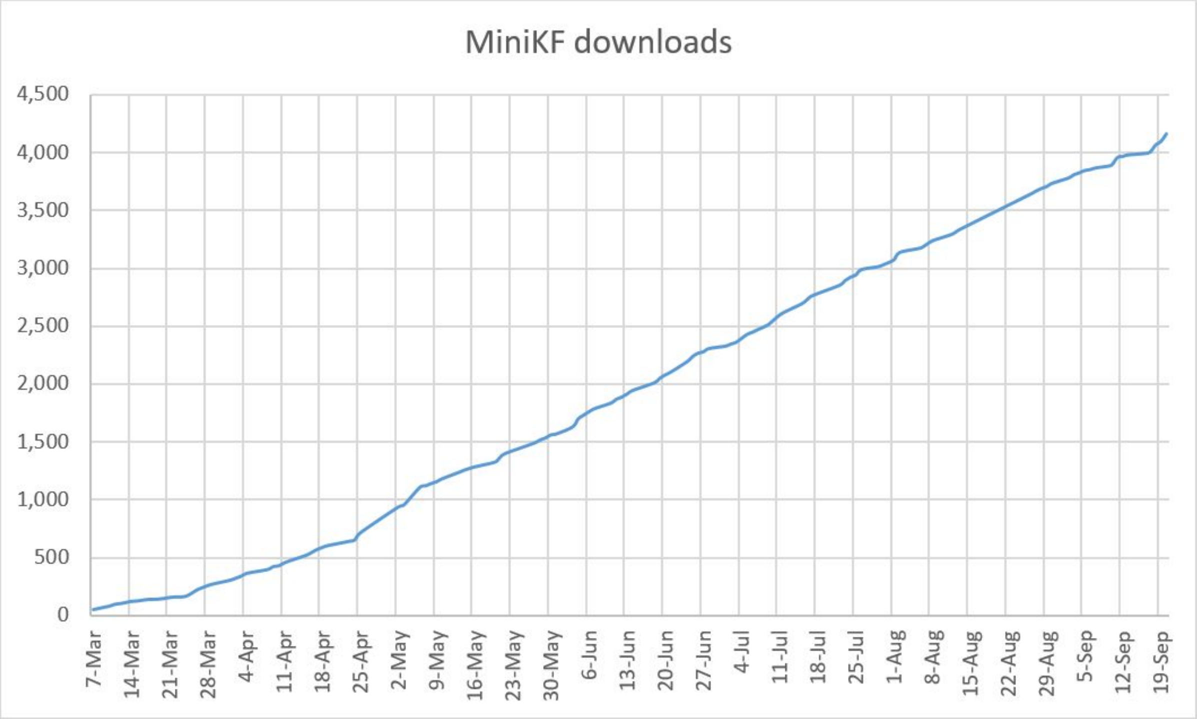
- Exploration and experimentation starts on the data scientist’s laptop
- No easy way to deploy Kubeflow on-prem
- Make get started with Kubeflow dead simple
    - Help democratize access to ML
- Same foundation/APIs everywhere,
    - users can move to a Kubeflow cloud deployment with one click, without having to rewrite anything

Local Kubeflow: Unified UX

- **Exactly** the same environment, on-prem, or on the cloud
- A single, unified User Experience
- Same Kubernetes APIs
- Same Kubeflow components
  - Notebooks
  - Pipelines
  - Fairing

Arrikto

MiniKF adoption



TL;DR of MiniKF installation

```
$ vagrant init arrikto/minikf
$ vagrant up
```

System requirements

- 12GB RAM
- 2 CPUs
- 50GB disk space

Operating systems

MiniKF runs on all major operating systems:

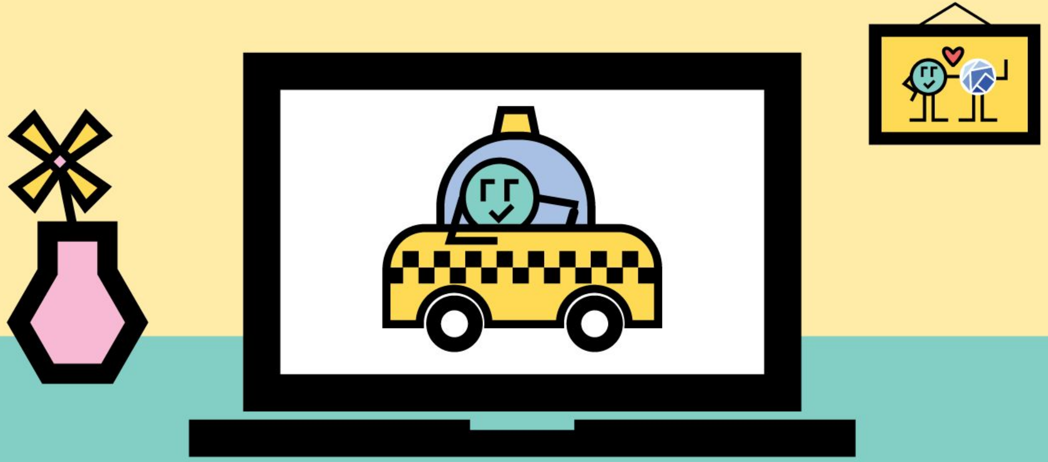
- Linux
- macOS
- Windows

Prerequisites

- Vagrant
- VirtualBox

Arrikto

Live demo: Chicago Taxi on-prem with MiniKF



Arrikto

What is the Chicago Taxi example?

- **Original dataset:**

  More than 100M trips, released by the City of Chicago

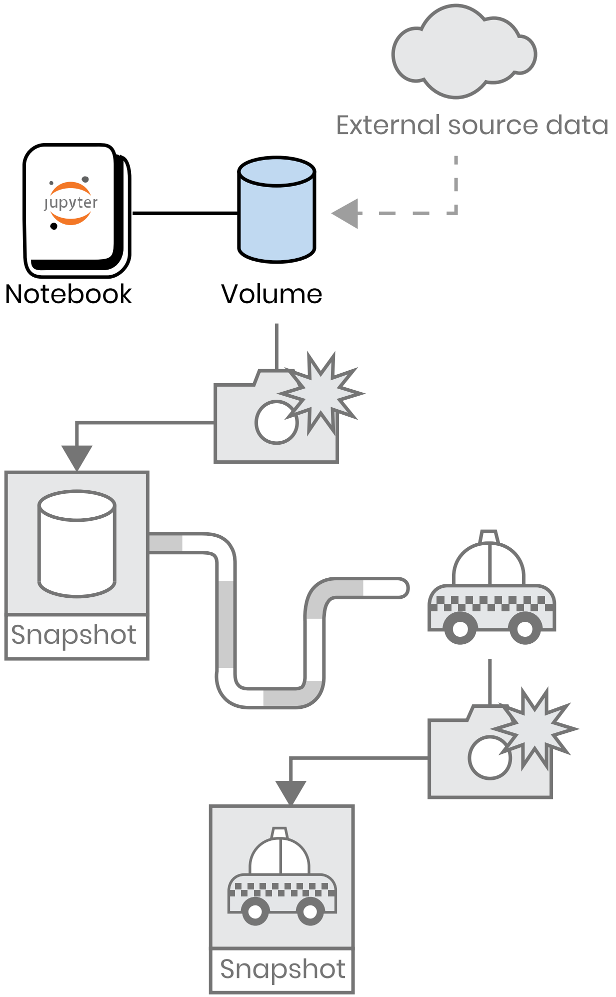
  https://digital.cityofchicago.org/index.php/chicago-taxi-data-released/
- Example fields are: `fare`, `trip_start_month`, `trip_start_hour`, `trip_start_day`, `pickup_latitude`, `pickup_longitude`, `dropoff_latitude`, `dropoff_longitude`, `trip_miles`, `payment_type`, `tips`.

- **End result:**

  A **classifier** that predicts if a trip will result in a tip greater than 20% of the fare

Demo overview

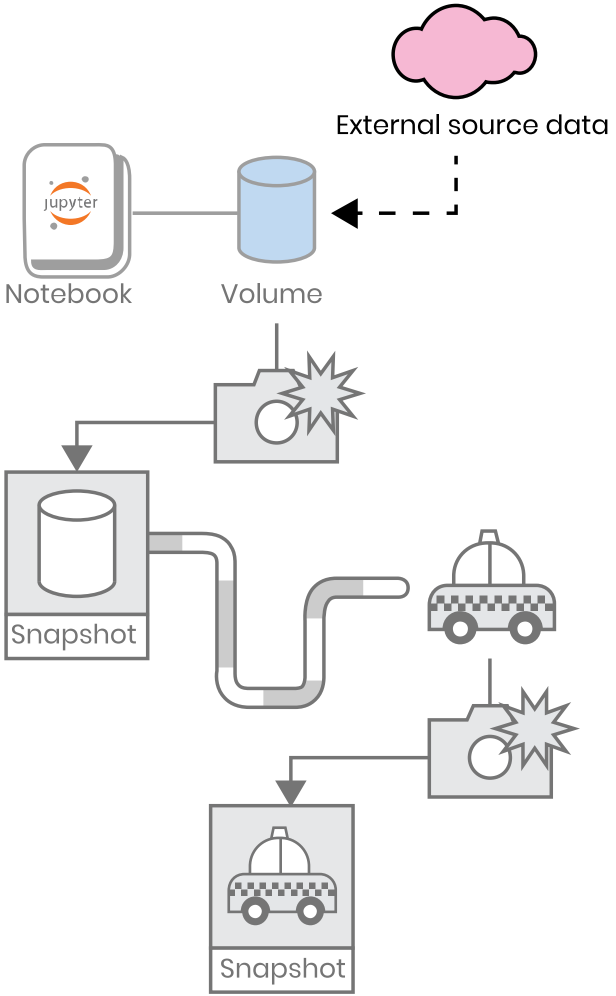
1. Create Notebook, add data volume



Arrikto

Demo overview

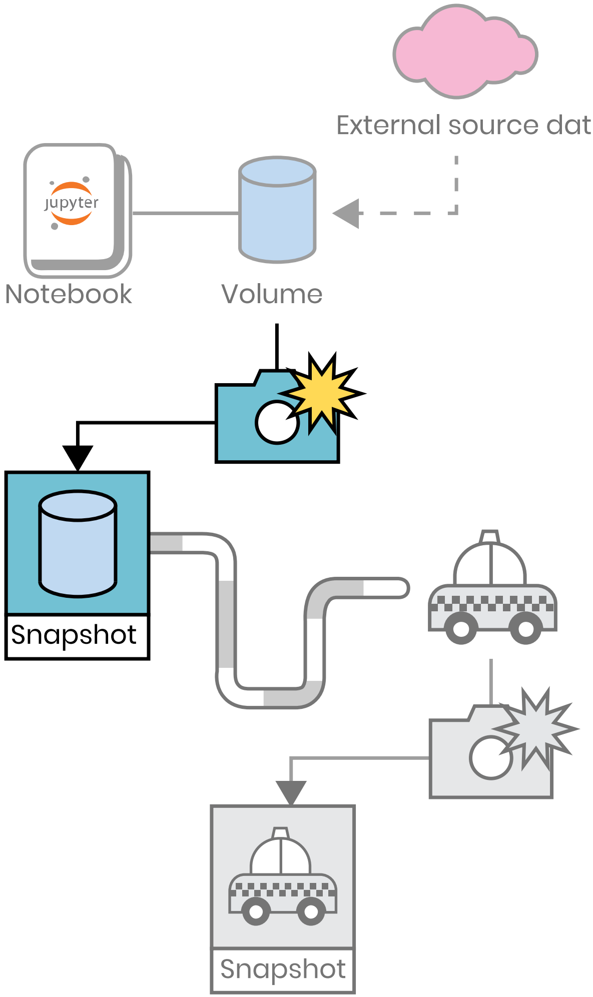
1. Create Notebook, add data volume
2. Ingest data in volume, compile the Taxi Cab Pipeline



Arrikto

Demo overview

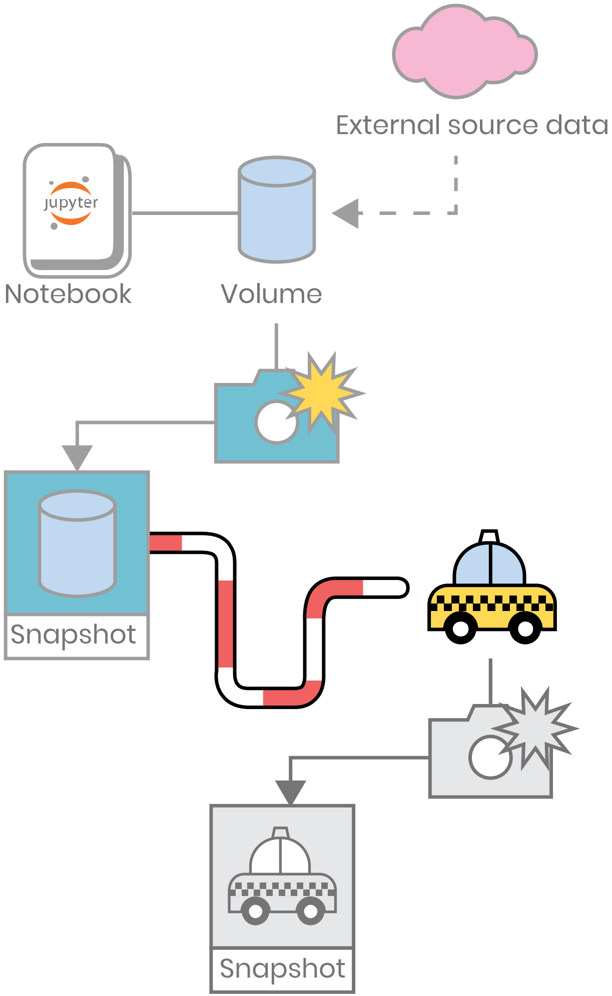
1. Create Notebook, add data volume
2. Ingest data in volume, compile the Taxi Cab Pipeline
3. Take a snapshot of your data using Arrikto Rok



Arrikto

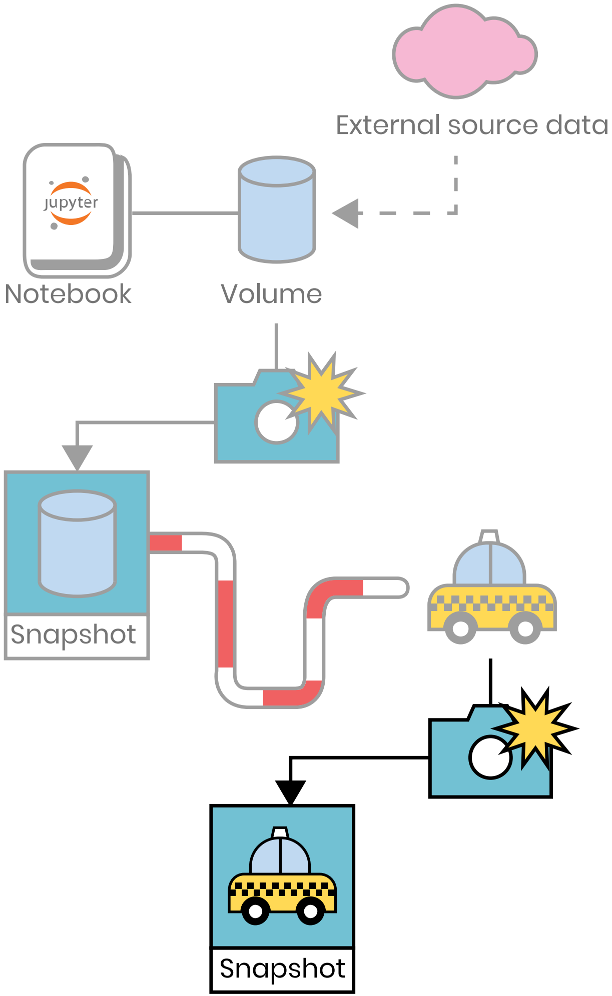
Demo overview

1. Create Notebook, add data volume
2. Ingest data in volume, compile the Taxi Cab Pipeline
3. Take a snapshot of your data using Arrikto Rok
4. Create a new Kubeflow Pipeline and seed it with the Rok snapshot



Demo overview

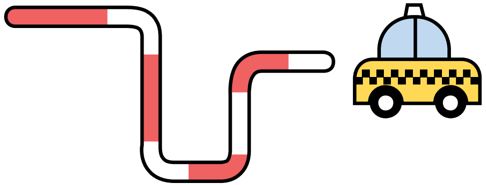
1. Create Notebook, add data volume
2. Ingest data in volume, compile the Taxi Cab Pipeline
3. Take a snapshot of your data using Arrikto Rok
4. Create a new Kubeflow Pipeline and seed it with the Rok snapshot
5. Snapshot the PVC after the pipeline run using Arrikto Rok



Step 1: Data validation

This step uses the TensorFlow Data Validation (TFDV) library to:

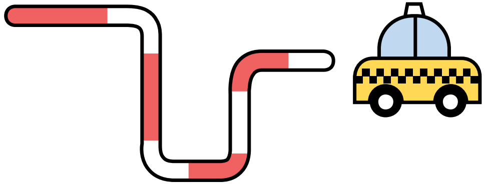
- Validate the training data
- Generate the dataset's schema for use by next steps
- Validate the evaluation data against the schema
- Identify anomalies between the training and evaluation data



Step 2: Data preprocessing

This step uses the TensorFlow Transform (TFT) library to:

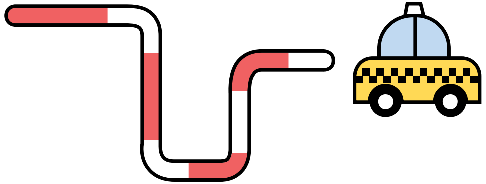
- Preprocess the training dataset, applying transformations to it
- Preprocess the evaluation dataset, applying transformations to it
- Produce a Transform TensorFlow Graph



Step 3: Model training

This step uses the TensorFlow Estimators to:

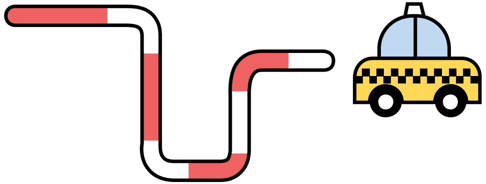
- Train the model using the processed datasets, producing a `SavedModel` for inference
- Produce a TFMA-specific evaluation Graph for deeper analysis with TFMA
- Produce evaluation events for Tensorboard



Step 4: Model analysis

This step uses the TensorFlow TFMA library to:

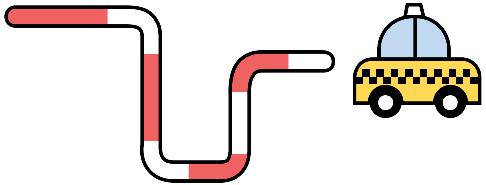
- Evaluate the trained model, using the evaluation dataset



Step 5: Prediction

This step aims to:

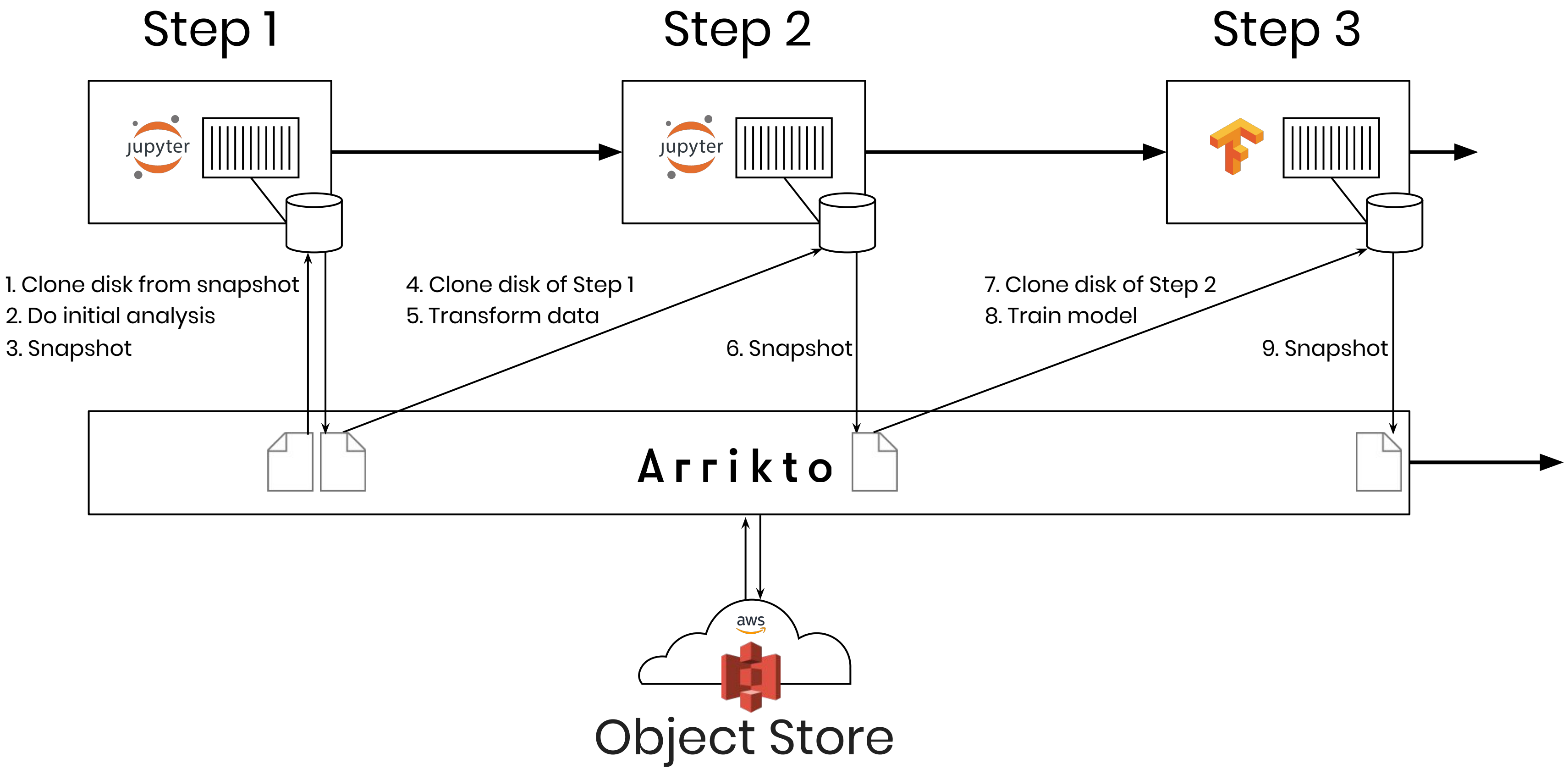
- Make predictions by against the evaluation dataset
- Generate results as CSV file(s), which add the prediction column in the evaluation dataset, to be used by next steps to generate a ROC and a Confusion Matrix.



Data Management in Kubeflow

- Extend Kubeflow to use Persistent Volumes in a vendor-agnostic way

- Arrikto contributions
  - JupyterHub-based Spawner with support for Persistent Volumes (in 0.4)
  - K8s-native Jupyter Notebook Manager with support for Persistent Volumes (in 0.5)
  - Extensions to the Kubeflow Pipelines DSL for Persistent Volumes and Volume Snapshots (in 0.5)



Step 1

Step 2

Step 3

1. Clone disk from snapshot
2. Do initial analysis
3. Snapshot

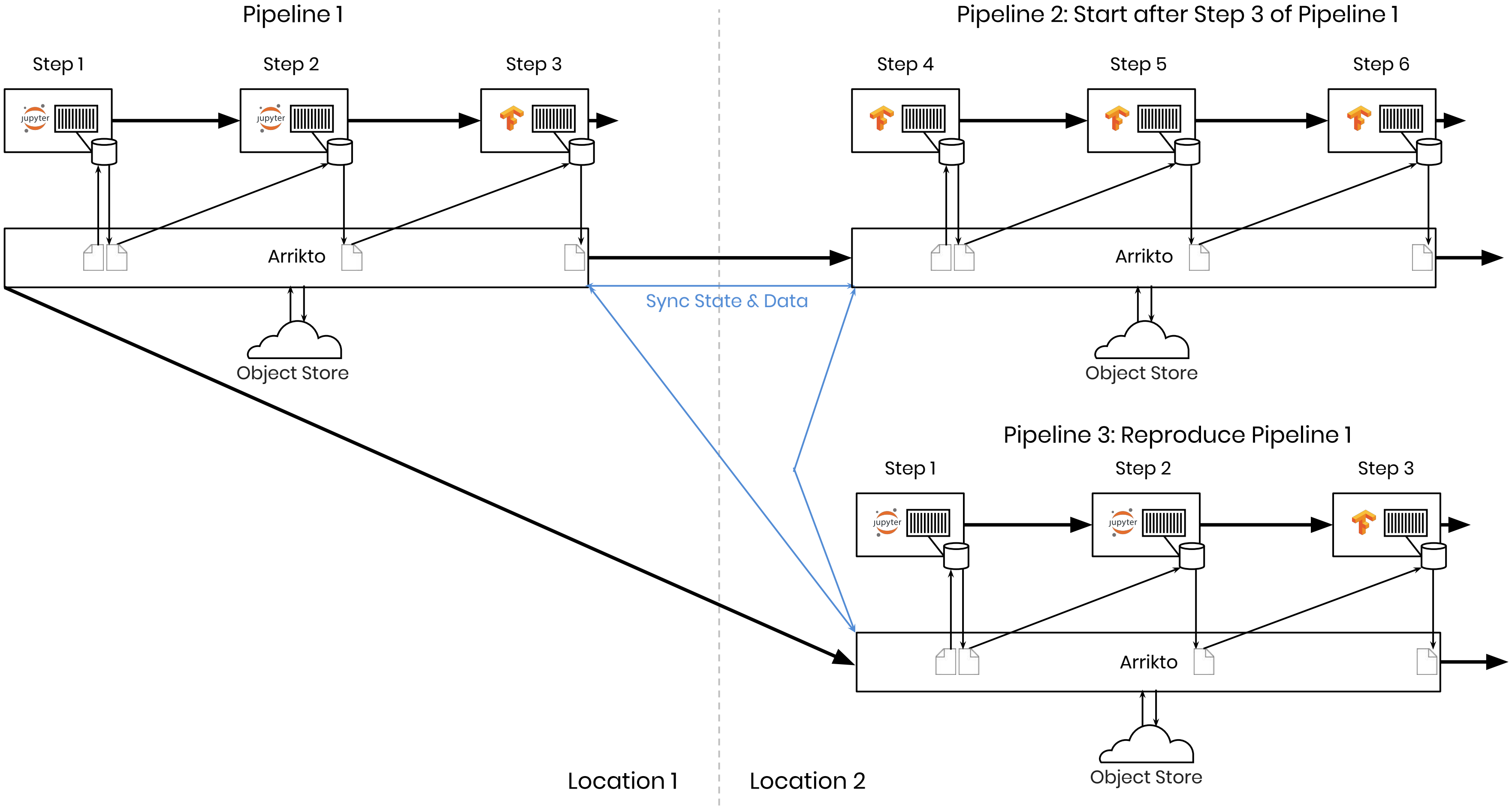
4. Clone disk of Step 1
5. Transform data

6. Snapshot

7. Clone disk of Step 2
8. Train model

9. Snapshot

Object Store


Pipeline 1
Step 1
Step 2
Step 3
jupyter
jupyter
Arrikto
Object Store
Pipeline 2: Start after Step 3 of Pipeline 1
Step 4
Step 5
Step 6
Arrikto
Object Store
Sync State & Data
Pipeline 3: Reproduce Pipeline 1
Step 1
Step 2
Step 3
jupyter
jupyter
Arrikto
Object Store
Location 1
Location 2

Arrikto

Running KFP on-prem before MiniKF

One should have strong Kubernetes knowledge to be able to deal with some steps:

- Understand K8s and be familiar with kubectl
- Understand and compose YAML files
- Manually create PVCs via Kubernetes
- Mount a PVC to a container to fill it up with initial data

Demo Step	Kubeflow System
Create Notebook Server	Notebook Manager UI
Create a Persistent Data Volume	Notebook Manager UI
Ingest Code	JupyterLab, Terminal
Ingest Data	JupyterLab, Terminal
Compile Pipeline	JupyterLab, Terminal
Snapshot JupyterLab Environment	Rok
Download Pipeline	JupyterLab
Upload Pipeline	Pipelines UI, Pipeline
Seed the Data into the Pipeline	Pipelines UI, Pipeline
Run Pipeline	Pipelines UI, Experiment, Run
View Metrics	Pipelines UI, Experiments
View Graph	Pipelines UI, Experiments

Ability to create Persistent Volume from GUI

Ability to transfer snapshot from notebook to pipeline

Demo Step	Kubeflow pre-0.4 System
Create Notebook Server	Notebook Manager UI
Create a Persistent Data Volume	**Local Terminal with Editor** • Compose YAML file to create an empty PVC (empty_pvc.yaml) • kubectl - submit empty_pvc.yaml • K8s creates PVC • kubectl - describe PVC, is it up and have the correct name • Create 2nd YAML file to create a new pod (busybox) and attach a new PVC to the pod. Make sure that the pod doesn't exit, so you can get a shell and work with it • kubectl - submit the 2nd YAML • kubectl - connect to the container and get a shell (kubectl --ti exec -- /bin/sh)
Ingest Data	**Local Terminal with Editor**
Clean up Container	**Local Terminal with Editor** • Remove the busybox container and keep the PVC to give it to KFP (this is mutable) • If you want to snapshot/clone that PVC to be reproducible you need to create new YAML files and know the specifics of your underlying storage
Ingest Code	JupyterLab, Terminal
Compile Pipeline	JupyterLab, Terminal
Snapshot the PVC	**Local Terminal with Editor** • Compose YAML file to snapshot the PVC (snap_pvc.yaml) • kubectl - submit snap_pvc.yaml • K8s creates snapshot
Clone the snapshot	**Local Terminal with Editor** • Compose YAML file to clone the snapshotted PVC (clone_pvc.yaml) • kubectl - submit clone_pvc.yaml • K8s creates new PVC • kubectl - describe to get its name
Download Pipeline	JupyterLab
Upload Pipeline	Pipelines UI, Pipeline
Seed the Data into the Pipeline	Pipelines UI, Pipeline
Run Pipeline	Pipelines UI, Experiments, Run
View Metrics	Pipelines UI, Experiments
View Graph	Pipelines UI, Experiments

50% REDUCTION

Demo Step	Kubeflow System
Create Notebook Server	Notebook Manager UI
Create a Persistent Data Volume	Notebook Manager UI
Ingest Code	JupyterLab, Terminal
Ingest Data	JupyterLab, Terminal
Compile Pipeline	JupyterLab, Terminal
Snapshot JupyterLab Environment	Rok
Download Pipeline	JupyterLab
Upload Pipeline	Pipelines UI, Pipeline
Seed the Data into the Pipeline	Pipelines UI, Pipeline
Run Pipeline	Pipelines UI, Experiment, Run
View Metrics	Pipelines UI, Experiments
View Graph	Pipelines UI, Experiments

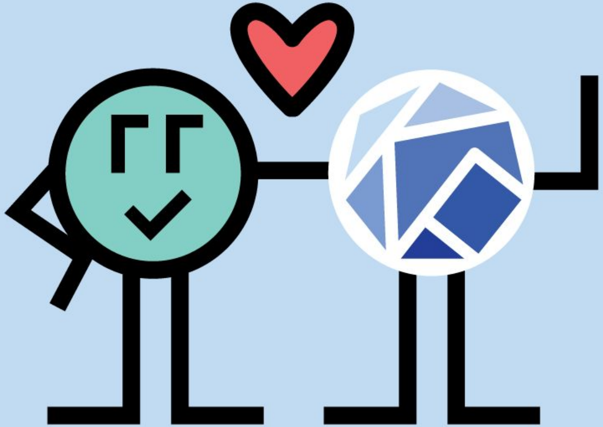
Arrikto

Running KFP on-prem with MiniKF and Rok

Data scientists are more self-sufficient:

- Less interaction with K8s and YAML
- Faster data movement from Notebooks to Pipelines
- Easier mounting of PVCs & seeding with data
- Simplified end-2-end pipeline execution & reproducibility
- Per-step snapshots for notebook-based exploration / iteration / troubleshooting

Running KFP on-prem with MiniKF and Rok



Future improvements

- GPU support
- Support for multi- and hybrid-cloud Kubeflow Pipelines
  - Experiment locally, train and deploy on different clouds
- MiniKF with Kubeflow v0.7 / v1.0 BETA (to be released mid-November)
- Volume Manager UI to browse the files of a Volume
- Request new features
  - #minikf on the Kubeflow Slack

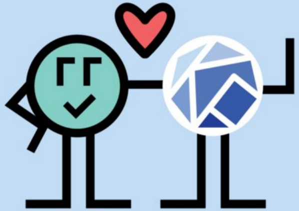
Try it out!

- Installation Instructions:
  - http://www.arrikto.com/minikf
  - https://www.kubeflow.org/docs/started/getting-started-minikf/
- End-to-end ML Pipeline Tutorial
  - Read the blog post
  - View the video
- We need your feedback
  - #minikf on the Kubeflow Slack

Arrikto

Thanks!

www.arrikto.com/minikf



Vangelis Koukis, Founder & CTO, Arrikto
vkoukis@arrikto.com | @vkoukis