

From Notebook to Kubeflow Pipelines with MiniKF & Kale



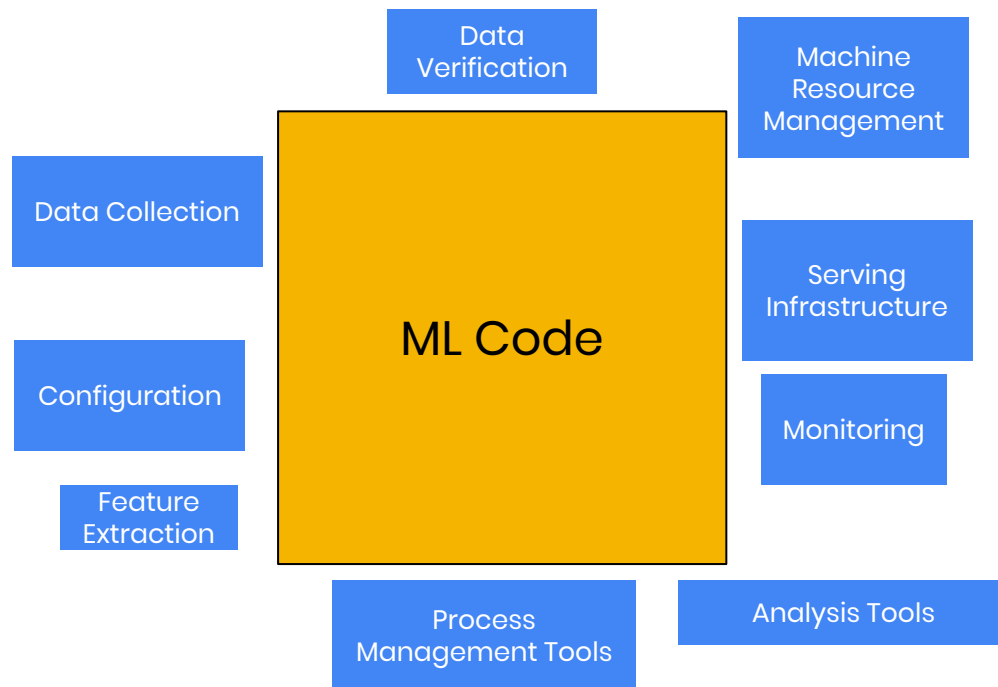
Stefano Fioravanzo, Software Engineer, Arrikto
Vangelis Koukis, Founder & CTO, Arrikto

The Problem

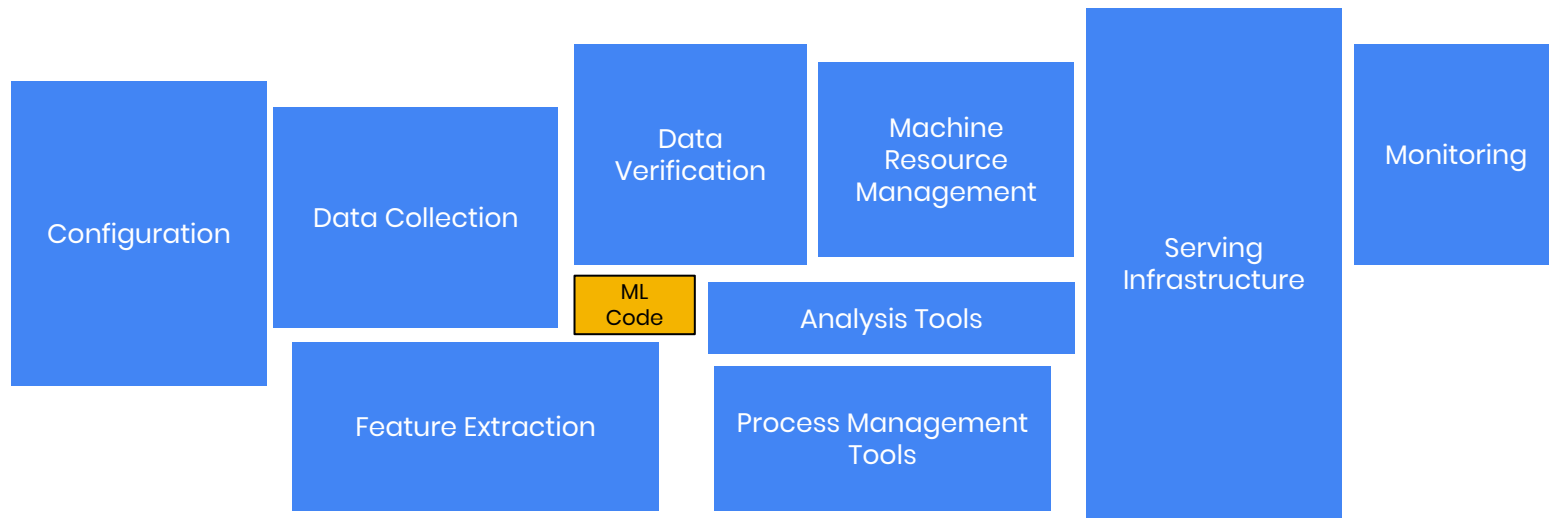
- Setting up an ML stack/pipeline is incredibly hard
- Setting up a production ML stack/pipeline is even harder
- Setting up an ML stack/pipeline that works across the 81% of enterprises that use multi-cloud* environments is EVEN HARDER

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

Perception: ML Products are mostly about ML



Reality: ML Requires DevOps; lots of it



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

Arrikto

Community!

Arrikto



Bloomberg



CANONICAL



GitHub

GOJEK



heptio



One Convergence



PRIMER



SELDON

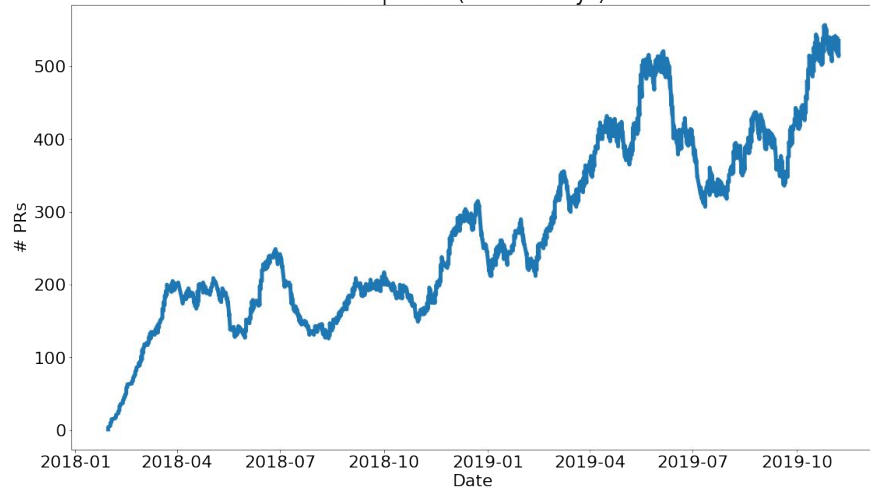


tu simple

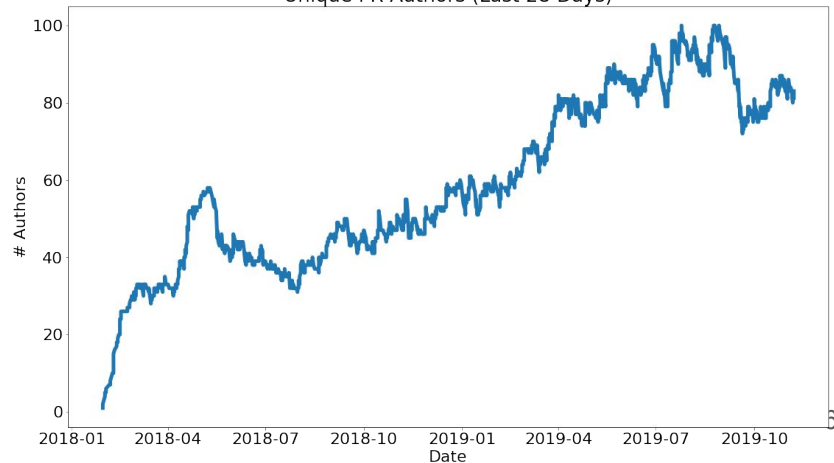


weaveworks

Unique PRs (Last 28 Days)



Unique PR Authors (Last 28 Days)



Just a SMALL sample of community contributions

Arrikto

- Jupyter manager UI
- Pipelines volume support
- MiniKF
- Auth with Istio + Dex
- On-premise installation

Bloomberg

- KFServing

Cisco

- Auth with Istio + Dex
- Katib
- KubeBench
- PyTorch
- On-premise installation

GoJEK

- Feast feature store

IBM

- Pipeline components for spark, ffdl
- Katib
- KFServing
- Faring
- Kubeflow SDK (TFJob, PyTorchJob, KFServing)
- Manifest

Intel

- kfctl (CLI & library) & kustomize
- OpenVino

Intuit

- Argo

RedHat + NVIDIA

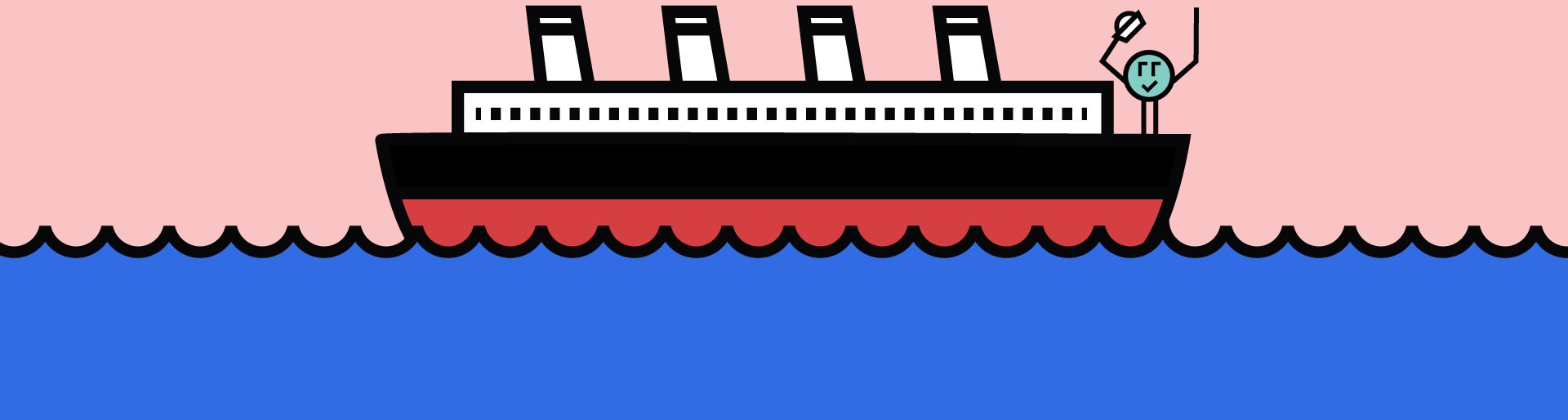
- TensorRT for notebooks

Seldon

- Seldon core

Arrikto

Live demo: Titanic example on MiniKF

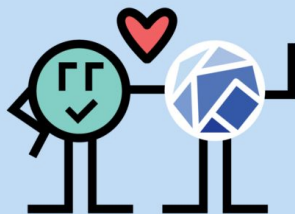


What is MiniKF?

- 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

What's new in the latest MiniKF?

- Kubeflow 0.7.1
 - Stay tuned for Kubeflow 1.0
- Support for GPUs
- 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)



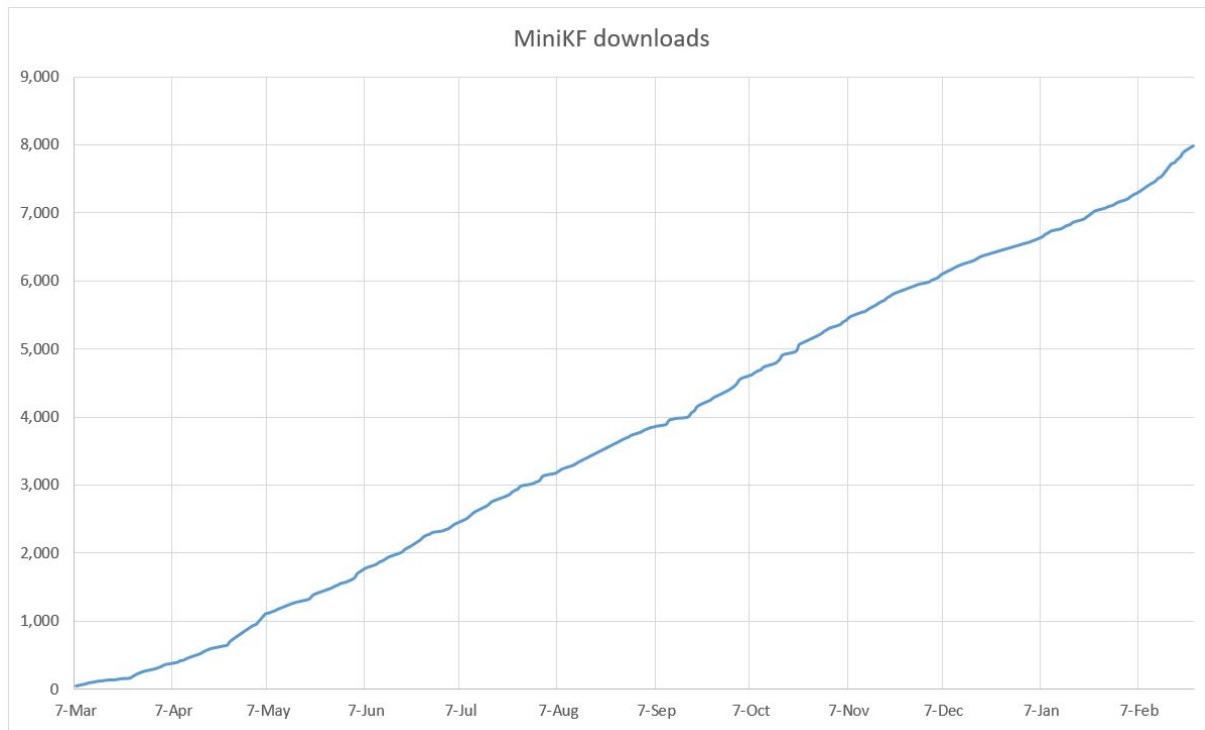
Why we started MiniKF

- 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
 - Katib
 - Kale

MiniKF on laptop adoption

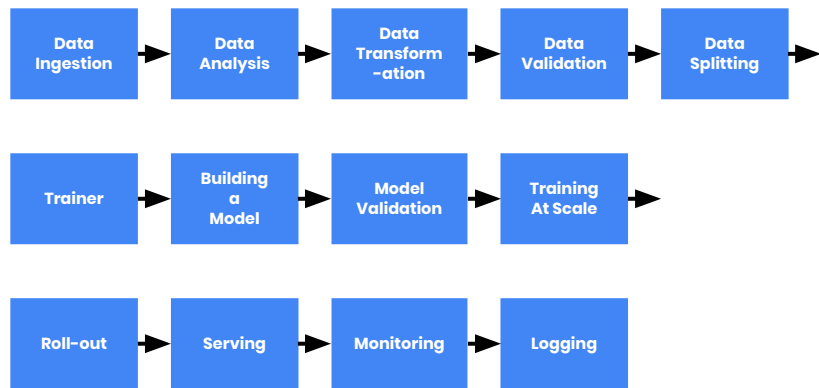


Data Science with Kubeflow

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

This webinar 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

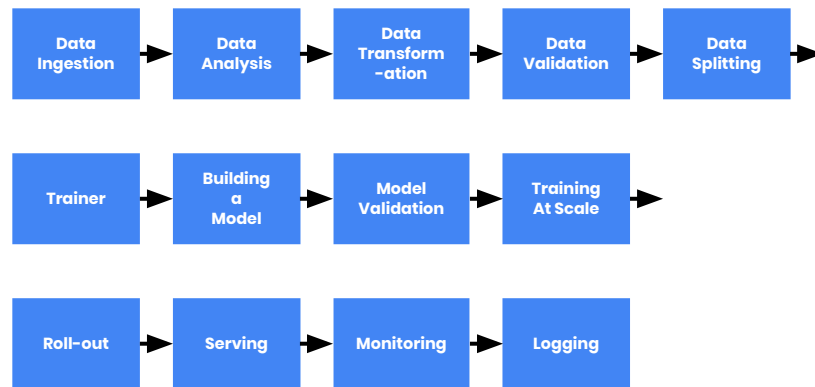


Data Science with Kubeflow

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

This webinar will focus on two essential aspects:

- **Low barrier to entry:**  **Kale** Notebook to Kubeflow Cloud using a fully GUI-based approach
- **Reproducibility:**  **Rok** versioning to enable reproducibility and 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

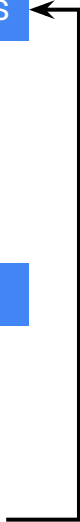


Upload pipeline to KFP



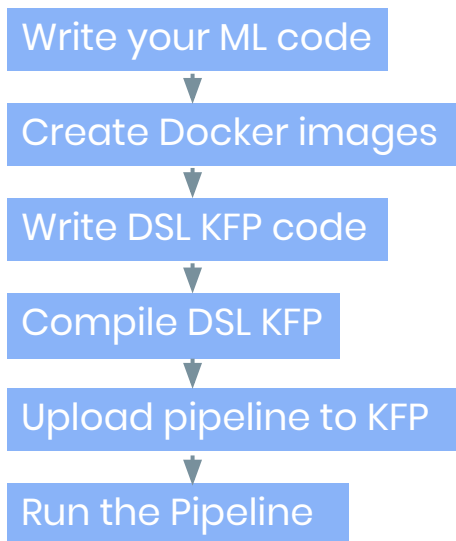
Run the Pipeline

Amend your ML code?



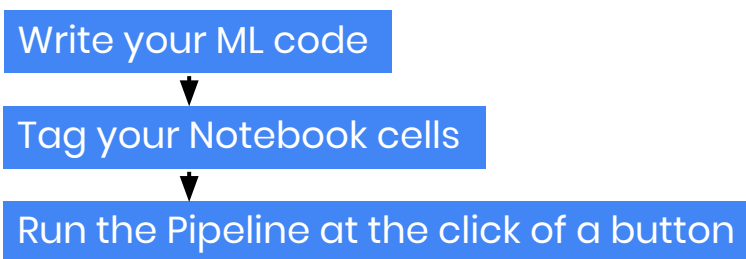
Workflow

Before



Amend your ML code?

After

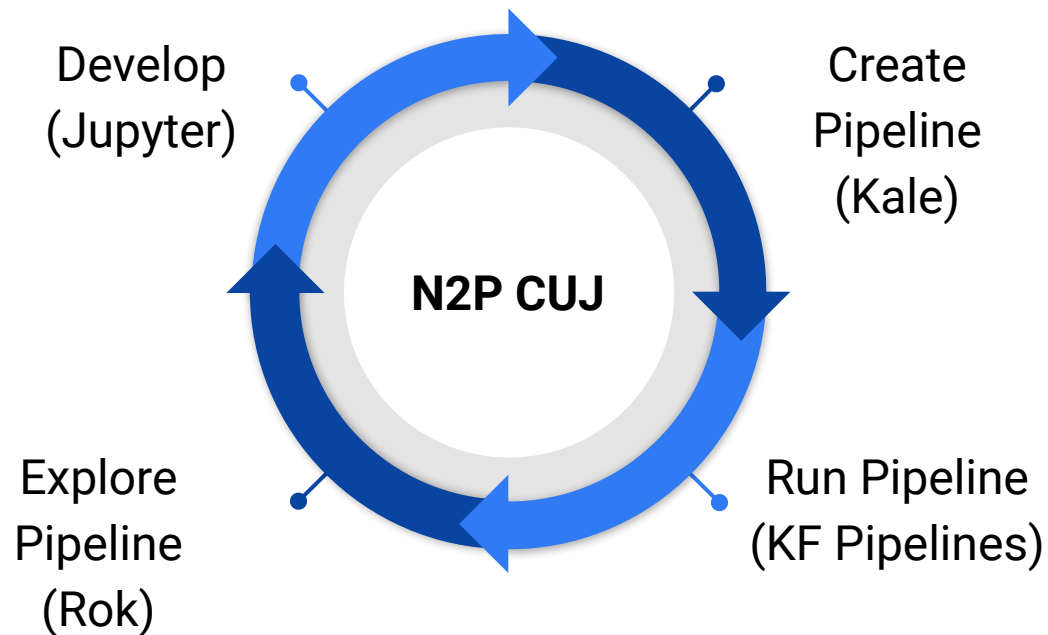


Amend your ML code? → Just edit your Notebook!

CI/CD for ML

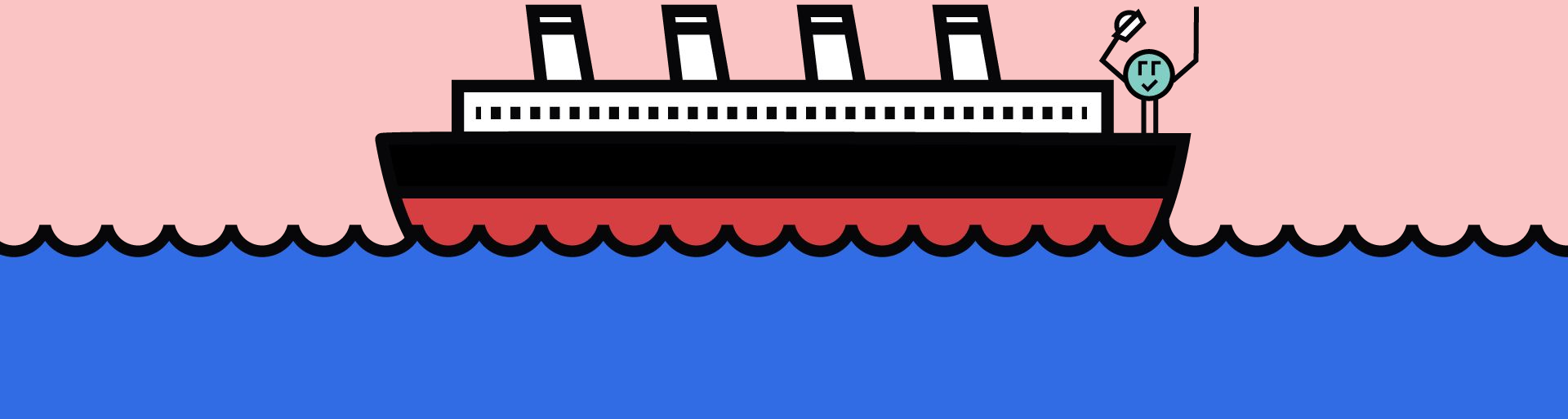
How can data scientists continually improve and validate models?

- Develop models and pipelines in Jupyter
- Convert notebook to pipeline using Kale
- Run pipeline using Kubeflow Pipelines
- Explore and debug pipeline using Rok



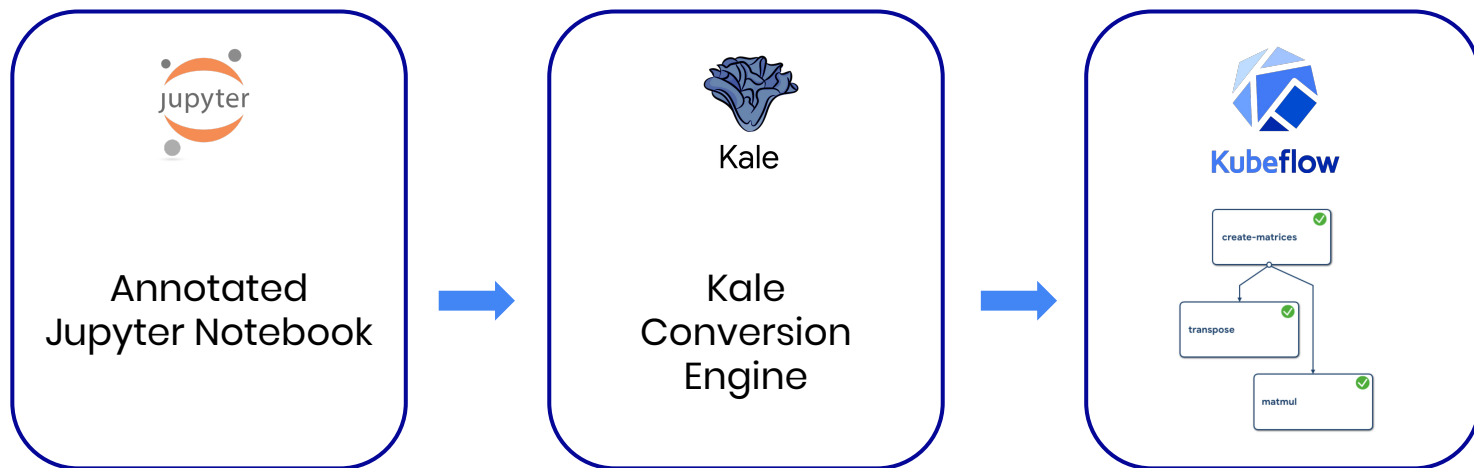
Arrikto

Live demo: Titanic example on MiniKF



KALE – Kubeflow Automated PipeLines Engine

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



Demo steps

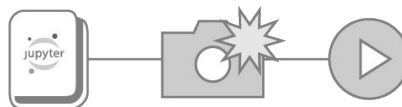
1. **Start Notebook, install new libraries on the fly**



2. Enable Kale and tag your Notebook cells



3. Snapshot your Notebook using Rok and run the pipeline

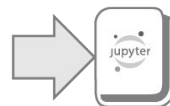


4. Reproduce prior state and debug your ML code



Demo steps

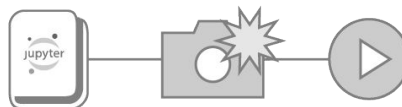
1. Start Notebook, install new libraries on the fly



2. **Enable Kale and tag your Notebook cells**



3. Snapshot your Notebook using Rok and run the pipeline

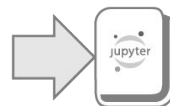


4. Reproduce prior state and debug your ML code



Demo steps

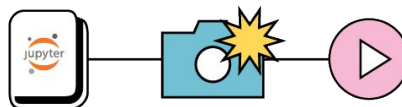
1. Install any missing libraries on the fly



2. Enable Kale and tag your Notebook cells



3. **Snapshot your Notebook using Rok and run the pipeline**




4. Reproduce prior state and debug your ML code



KALE - Modules


nbparser



```

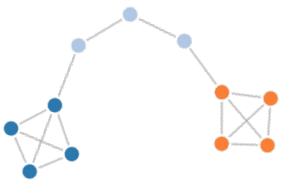
[3]: C = np.transpose(A)
      print(C)
      ↕
[4]: D = np.matmul(A, B)
      print(D)
    
```

↓



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

Data Management in Kubeflow

- Extend Kubeflow to use Persistent Volumes in a vendor-agnostic way
- Arrikto major 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)
 - Authentication and authorization using Istio and Dex (in 0.6)
 - K8s-native Volumes Manager with support for creating new PVCs and viewing their data (coming in 1.0)

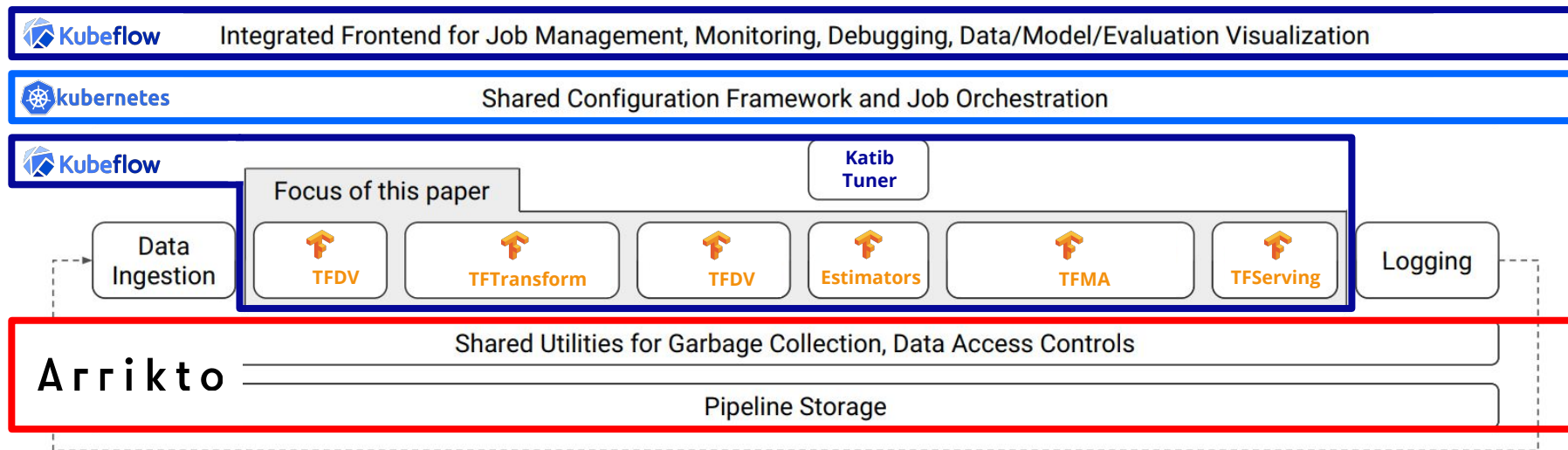
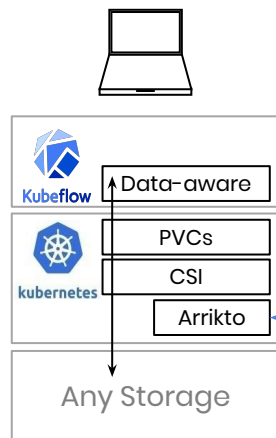


Figure 1: High-level component overview of a machine learning platform.

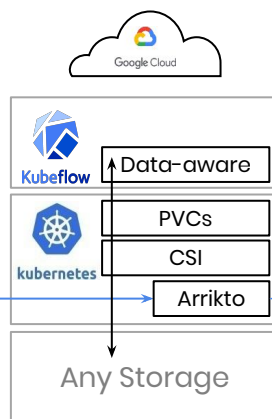
Data Versioning, Packaging, and Sharing

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

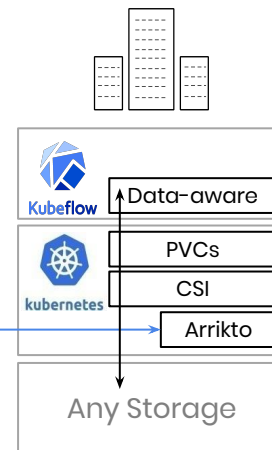
Experimentation



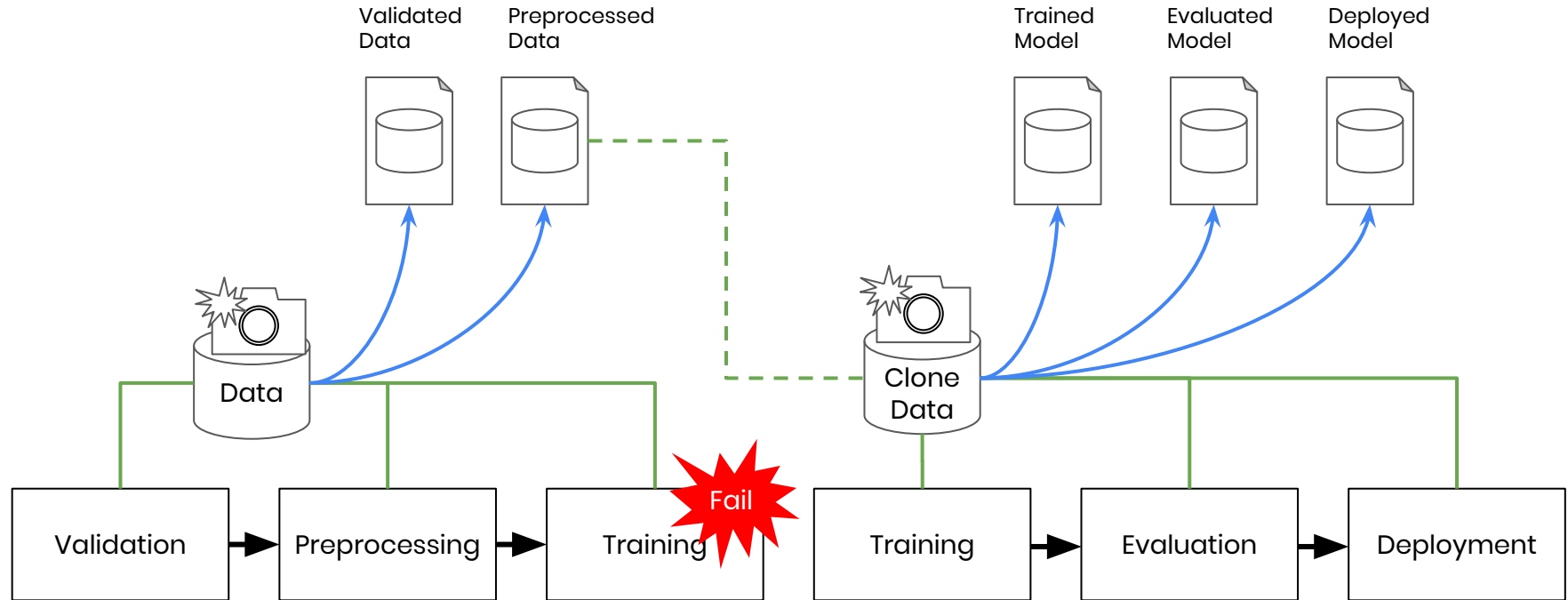
Training



Production

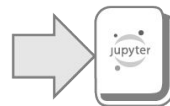


Arrikto Rok



Demo steps

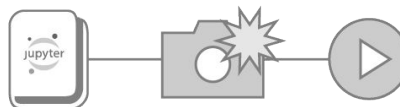
1. Install any missing libraries on the fly



2. Enable Kale and tag your Notebook cells



3. Snapshot your Notebook using Rok and run the pipeline



4. **Reproduce prior state and debug your ML code**



Running KFP: **Without** Kale and Rok

You would need strong Kubernetes knowledge to

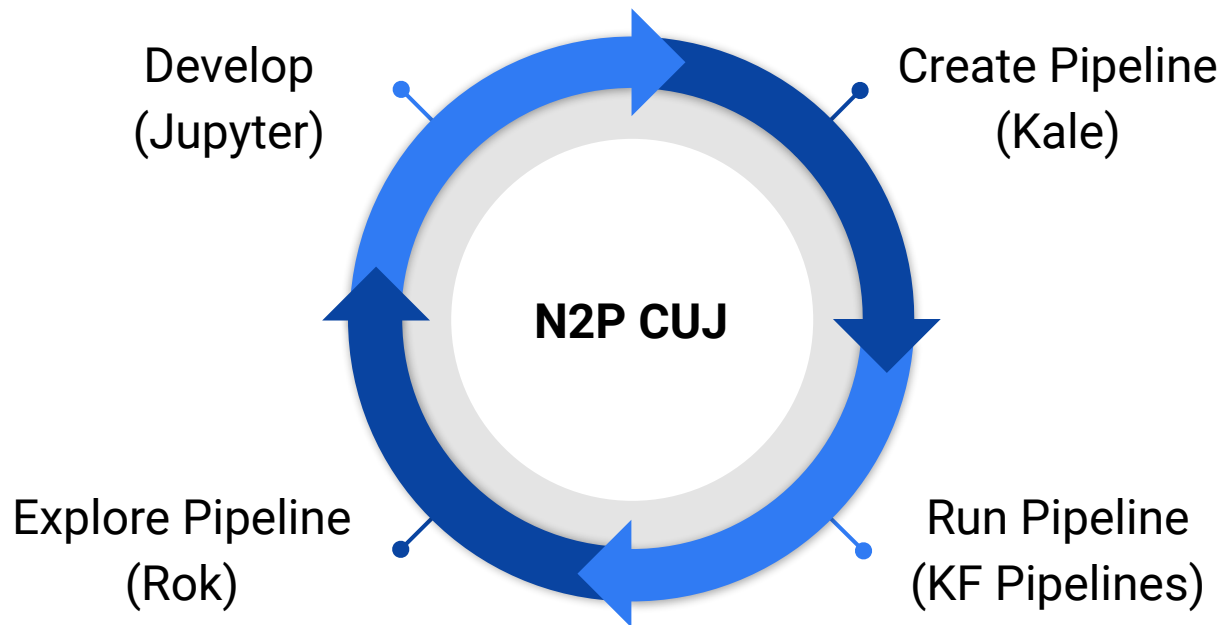
- Understand K8s and be familiar with kubectl
- Understand and compose YAML files
- Manually create PVCs via Kubernetes
- Mount a PVC to a container to fill it up with initial data
- Start a pipeline manually

Running KFP: **With** Kale and Rok

Data scientists are more self-sufficient:

- No interaction at all with K8s and YAML
- Fast data movement from Notebooks to Pipelines
- Start a pipeline with the click of a button
- Seamless mounting of PVCs & seeding with data
- Simplified end-2-end pipeline execution & reproducibility
- Per-step snapshots for notebook-based exploration / iteration / troubleshooting

Notebook-to-Pipeline CUJ

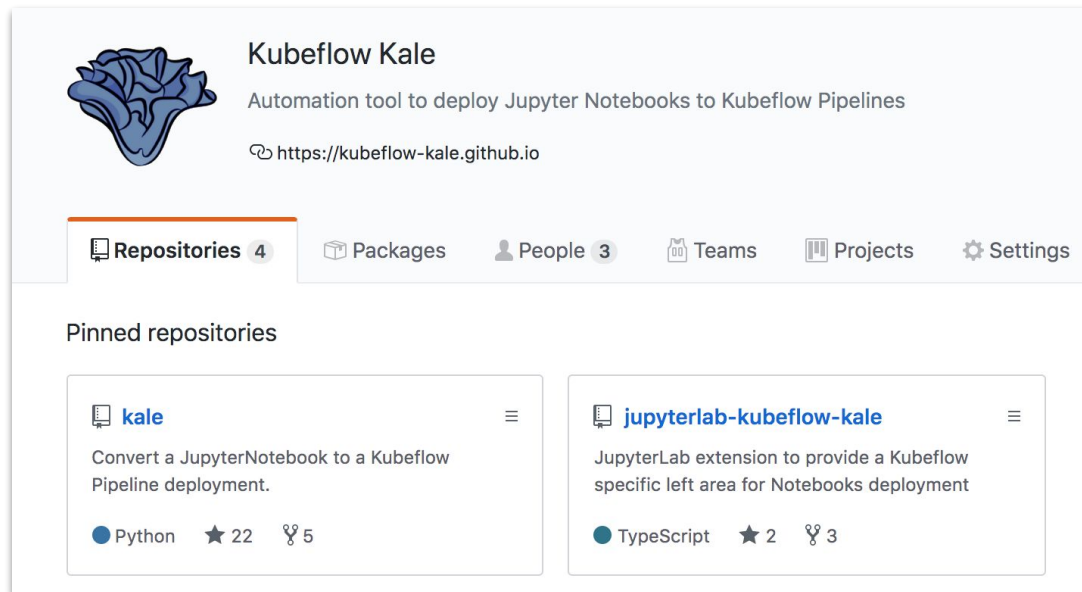


Future improvements

- Support for multi- and hybrid-cloud Kubeflow Pipelines
 - Experiment locally, train and deploy on different clouds
- Hyperparameter Tuning with Kale and Katib
 - 1000s of automated pipeline runs! Caching!
- Data and metadata tracking with Rok and MLMD
 - Explore run history and lineage of artifacts
- MiniKF with Kubeflow 1.0
 - Manage and browse Volumes with a new Volumes Manager UI

Contribute!

github.com/kubeflow-kale



The screenshot shows the GitHub profile for 'Kubeflow Kale'. At the top left is a blue brain icon. The profile name is 'Kubeflow Kale' with the description 'Automation tool to deploy Jupyter Notebooks to Kubeflow Pipelines' and a link to 'https://kubeflow-kale.github.io'. Below this is a navigation bar with 'Repositories 4' (highlighted), 'Packages', 'People 3', 'Teams', 'Projects', and 'Settings'. The 'Pinned repositories' section contains two items: 'kale' (Python, 22 stars, 5 forks) and 'jupyterlab-kubeflow-kale' (TypeScript, 2 stars, 3 forks).

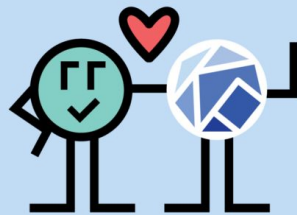
Read more about Kale [on Medium](#)

Try it out!

- Installation Instructions:
 - <http://www.arrikto.com/minikf>
 - <https://www.kubeflow.org/docs/started/getting-started-minikf/>
 - <https://www.kubeflow.org/docs/started/workstation/minikf-gcp/>
- Notebook-to-Pipeline Tutorial
 - Follow the [CodeLab](#)
 - View the [video](#)
- We need your feedback!
 - #minikf on the [Kubeflow Slack](#)

Thanks!

www.arrikto.com/minikf



Stefano Fioravanzo, Software Engineer, Arrikto
stefano@arrikto.com | @sfioravanzo

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