Update **confidential** designator here

Red Hat Developer

ML Ops for Java Developers

A Hands-On Guide with Kubeflow and Quarkus

Eder Ignatowicz
Senior Principal Software Engineer

Elder Moraes
Principal Developer Advocate

Red Hat

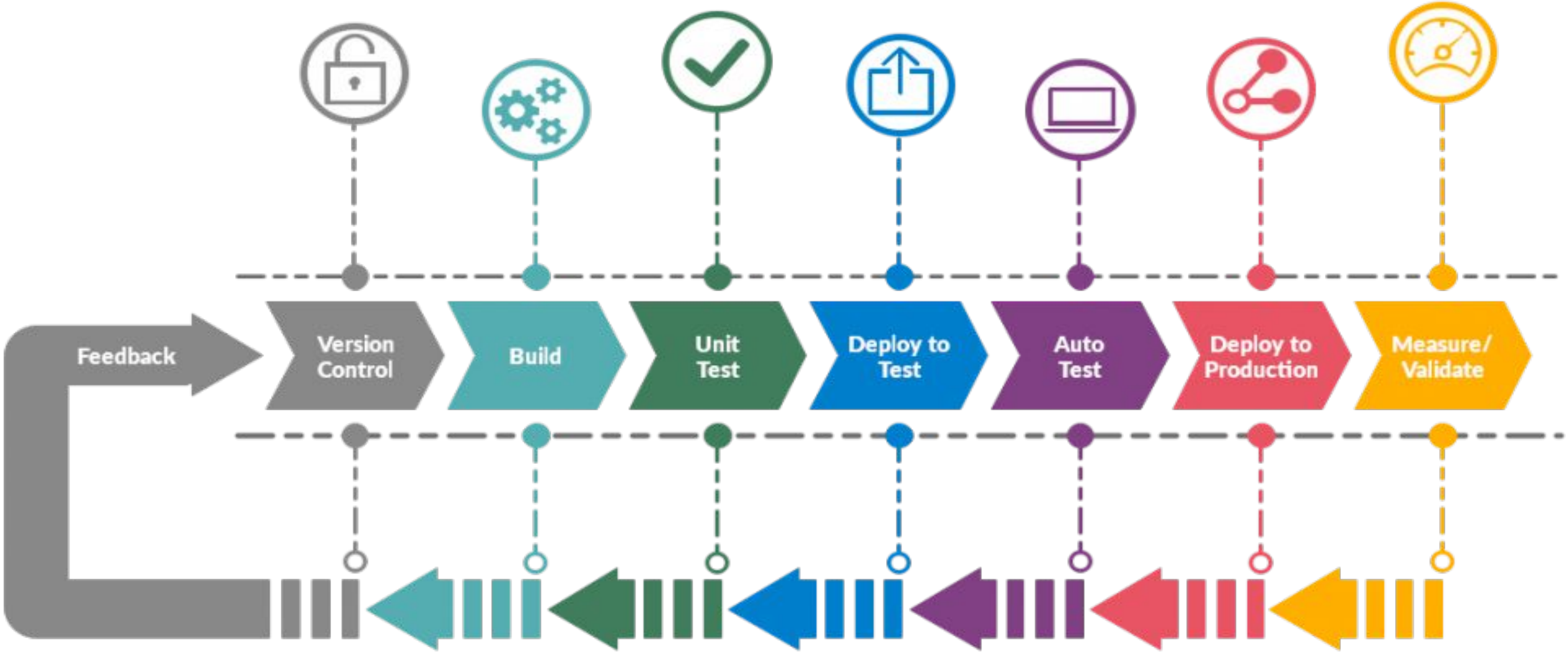

The AI revolution is coming
Red Hat
OpenShift AI
The AI revolution is coming
GENILDO
made with mematic

What are our goals with this talk?

- Bootstrap your understanding of how a ML Ops platform works under the hood;
- Understand how people put ML models in production
- How Kubeflow is designed to simplify ML workflows on Kubernetes.
- Where the Java developer fits into this picture
- How Quarkus, the Kubernetes-native Java framework, is the best way to consume 'Kubernetes-based' Machine Learning Models

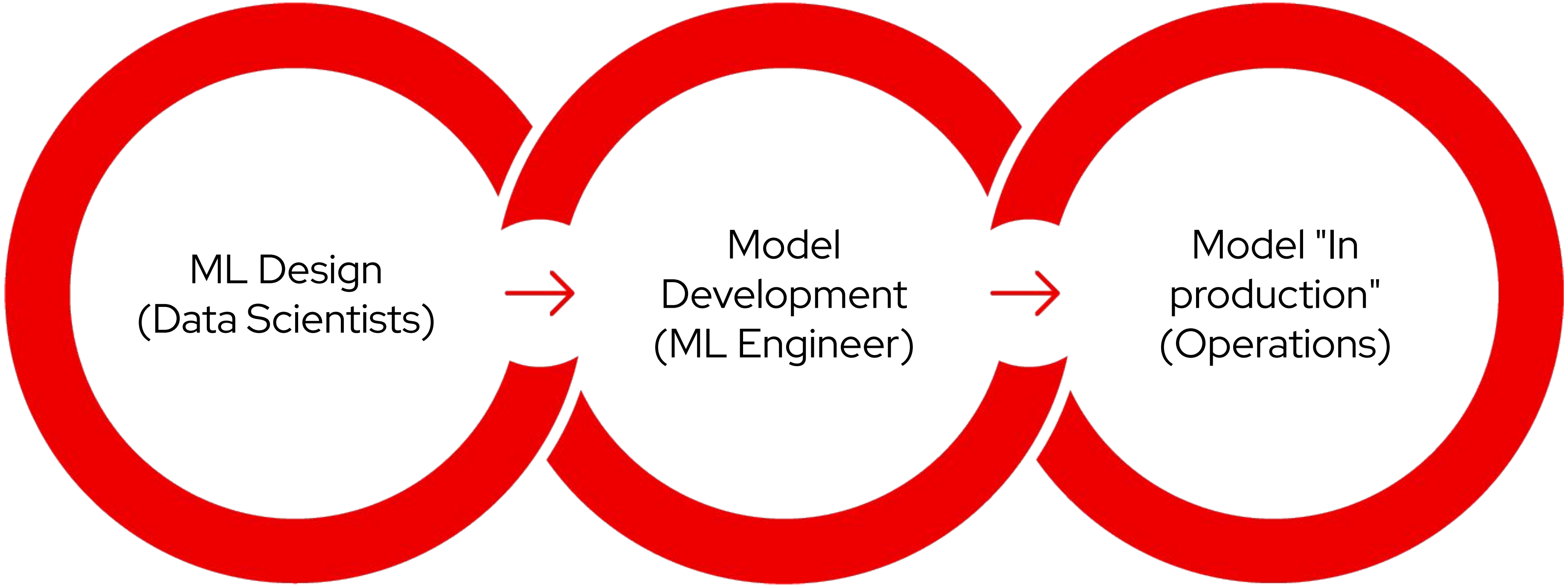
Source:Insert source data here
Insert source dat here

Devops



Feedback
Version Control
Build
Unit Test
Deploy to Test
Auto Test
Deploy to Production
Measure/ Validate

What is Machine Learning Operations (MLOps)?



ML models are more complex than traditional software because:

- Data is constantly changing (drift, bias, new patterns).
- Models are non-deterministic (outputs vary with training).
- Training, tuning, and deployment require heavy compute resources.
- Collaboration is harder (data scientists, engineers, ops teams all involved).

MLOps (Machine Learning Operations) applies DevOps principles to ML, ensuring **scalable, reproducible, and automated ML workflows.**

Some key prerequisites before diving into the ML lifecycle

- What is a Machine Learning **Model**?

A **model** is a program that takes an input (e.g., text, image, or data) and produces an output (e.g., classification, prediction)

In fraud detection, a model predicts whether a transaction is fraudulent or legitimate based on past data

Some key prerequisites before diving into the ML lifecycle

- What is **Inference**?

Inference is the process of using a trained model to make predictions on new data.

In Java terms, it's like calling a pre-trained function to get a result.

Some key prerequisites before diving into the ML lifecycle

- What is a **Feature**?

  A feature is a measurable property used as input for the model.

In fraud detection, common features include:

- Transaction amount (higher amounts might indicate fraud).
- Location (transaction from an unusual country).
- Number of transactions in the last hour (high frequency could be suspicious).

Some key prerequisites before diving into the ML lifecycle

- What are **Parameters** and **Hyperparameters**?

Parameters are internal variables of a model learned during training by the model to make predictions.

Hyperparameters are manually set configurations that affect how the model learns.

Some key prerequisites before diving into the ML lifecycle

- What is **Model Training**?

Training is the process of feeding historical fraud data to the model so it learns patterns.

Uses labeled data:

✅ Legitimate transactions

❌ Fraudulent transactions

The model adjusts parameters to minimize wrong predictions.

Some key prerequisites before diving into the ML lifecycle

- What is **Model Serving**?

Model serving is deploying a trained model to handle live transactions.

The model is exposed as an API or integrated into a real-time system.

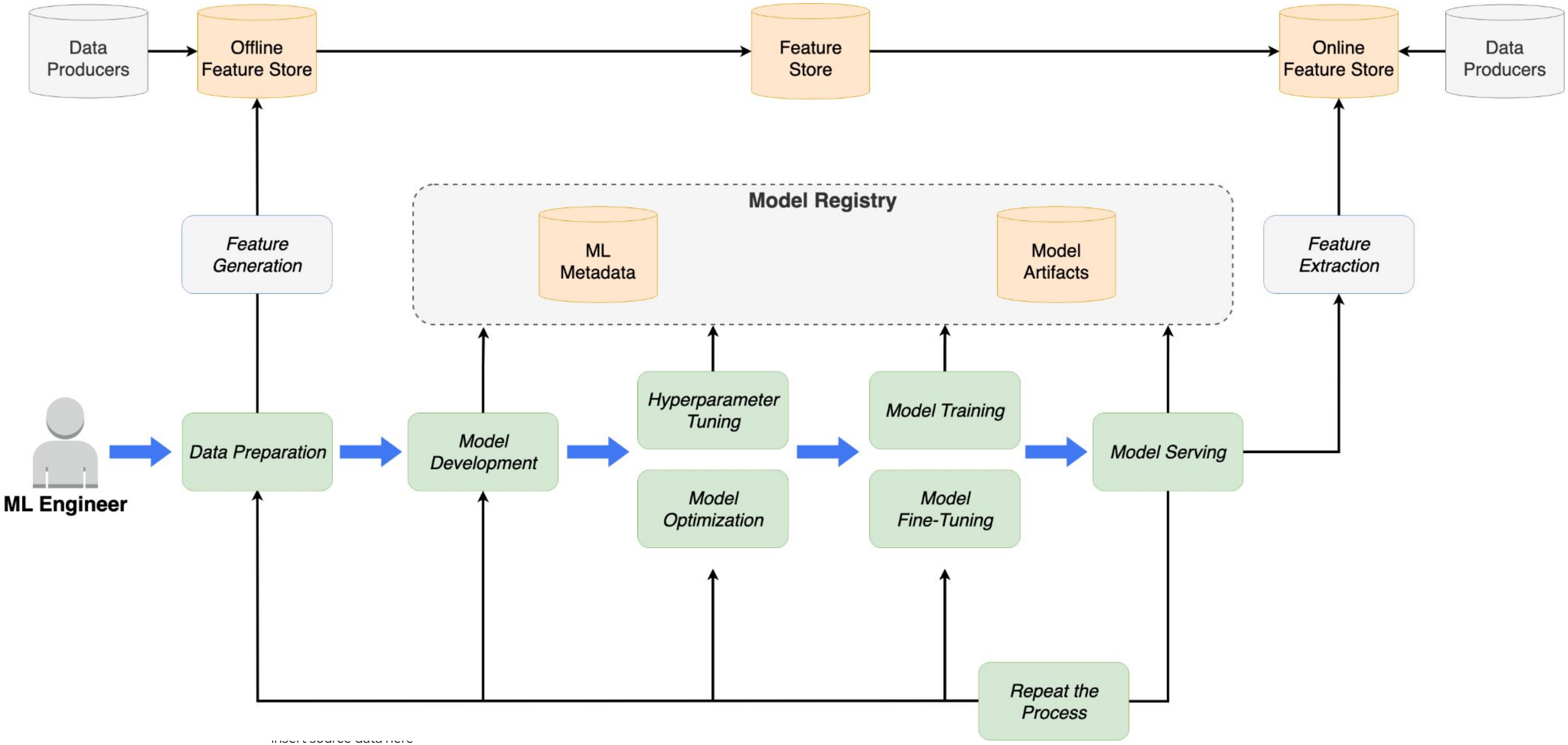
Some key prerequisites before diving into the ML lifecycle

- What is the **Model Registry?**

A Model Registry stores and tracks different versions of trained models.

Helps in versioning, auditing, and rollback of models.

Introducing the ML Lifecycle

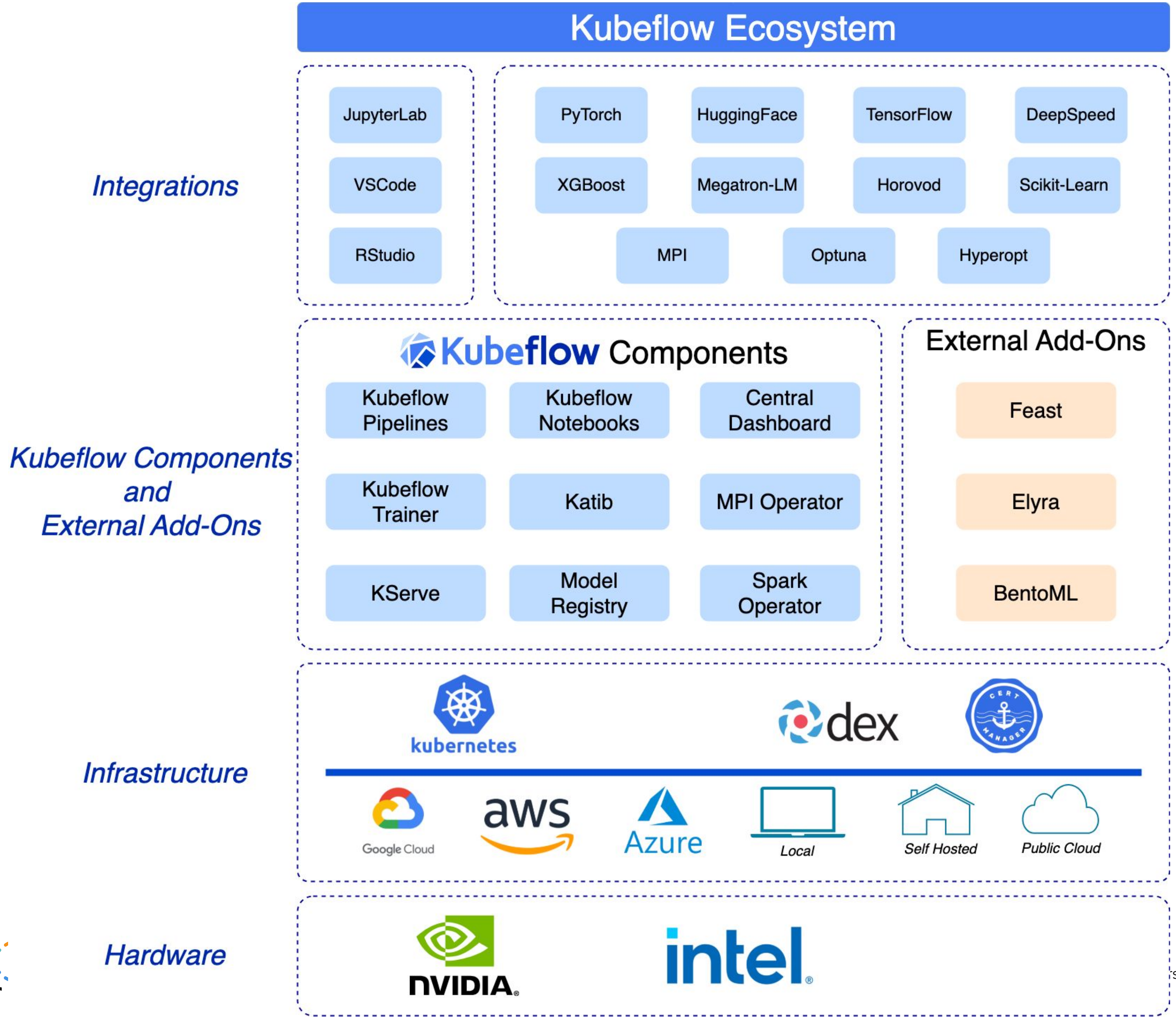


What is Kubeflow?

Kubeflow

- Kubeflow is a community and ecosystem of open-source projects to address each stage in the machine learning (ML) lifecycle with support for best-in-class open source tools and frameworks.
- Kubeflow makes AI/ML on Kubernetes simple, portable, and scalable.

Source:
Insert source data here
Insert source data here


Kubeflow Ecosystem
Integrations
JupyterLab
VSCode
RStudio
PyTorch
HuggingFace
TensorFlow
DeepSpeed
XGBoost
Megatron-LM
Horovod
Scikit-Learn
MPI
Optuna
Hyperopt
Kubeflow Components and External Add-Ons
Kubeflow Components
Kubeflow Pipelines
Kubeflow Notebooks
Central Dashboard
Kubeflow Trainer
Katib
MPI Operator
KServe
Model Registry
Spark Operator
External Add-Ons
Feast
Elyra
BentoML
Infrastructure
kubernetes
dex
CERT MANAGER
Google Cloud
aws
Azure
Local
Self Hosted
Public Cloud
Hardware
NVIDIA
intel

Maintainer Distribution Name	**Kubeflow Version**	**Target Platform**	**Link**
Amazon Web Services	1.7 [release notes]	Amazon Elastic Kubernetes Service (EKS)	Website
Aranui Solutions deployKF	1.8 [version matrix]	Multiple [list]	Website
Canonical Charmed Kubeflow	1.8 [release notes]	Multiple	Website
Google Cloud	1.8 [release notes]	Google Kubernetes Engine (GKE)	Website
IBM Cloud	1.8 [release notes]	IBM Cloud Kubernetes Service (IKS)	Website
Microsoft Azure	1.7 [release notes]	Azure Kubernetes Service (AKS)	Website
Nutanix	1.8	Nutanix Kubernetes Engine	Website
QBO	1.8 [release notes]	QBO Kubernetes Engine (QKE)	Website
Red Hat Open Data Hub	**1.9**	OpenShift	Website
VMware	1.6	VMware vSphere	Website

Red Hat's AI/ML engineering is 100% open source

Upstream projects

jupyter

RAY

PyTorch

Kubeflow

TensorFlow

TrustyAI

KServe

→

Community projects

OPEN DATA HUB

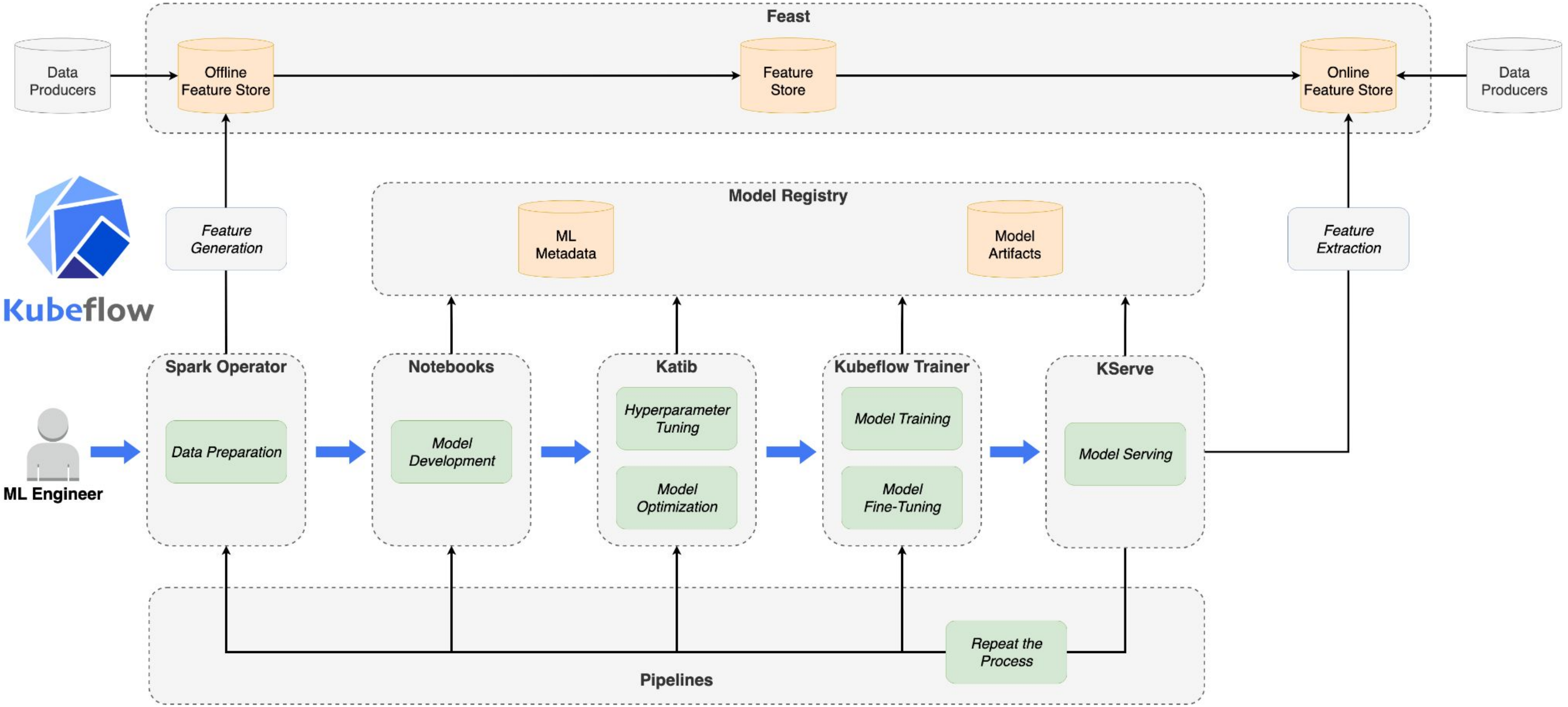
AI Platform powered by Open Source

→

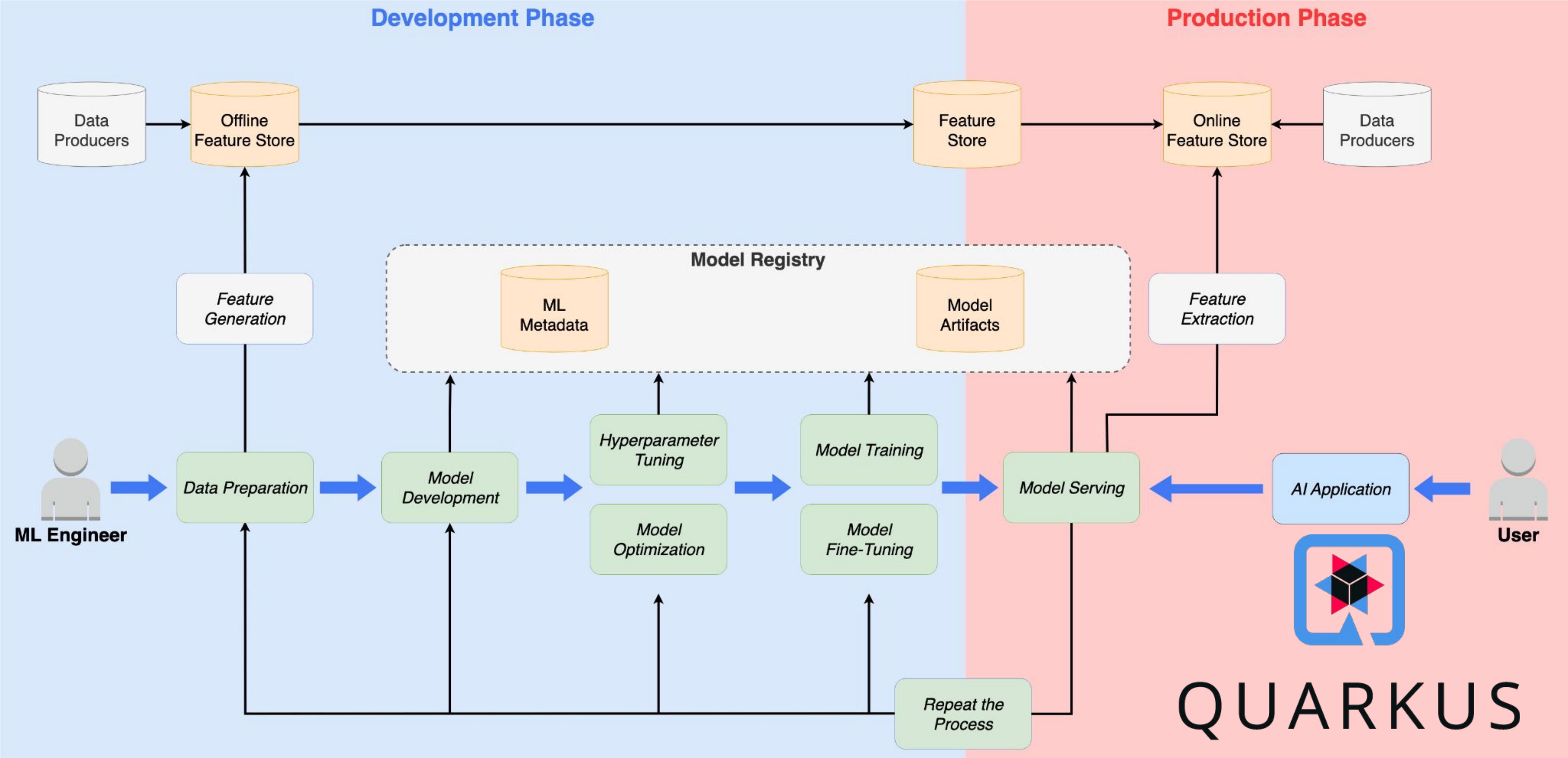
Product

Red Hat OpenShift AI

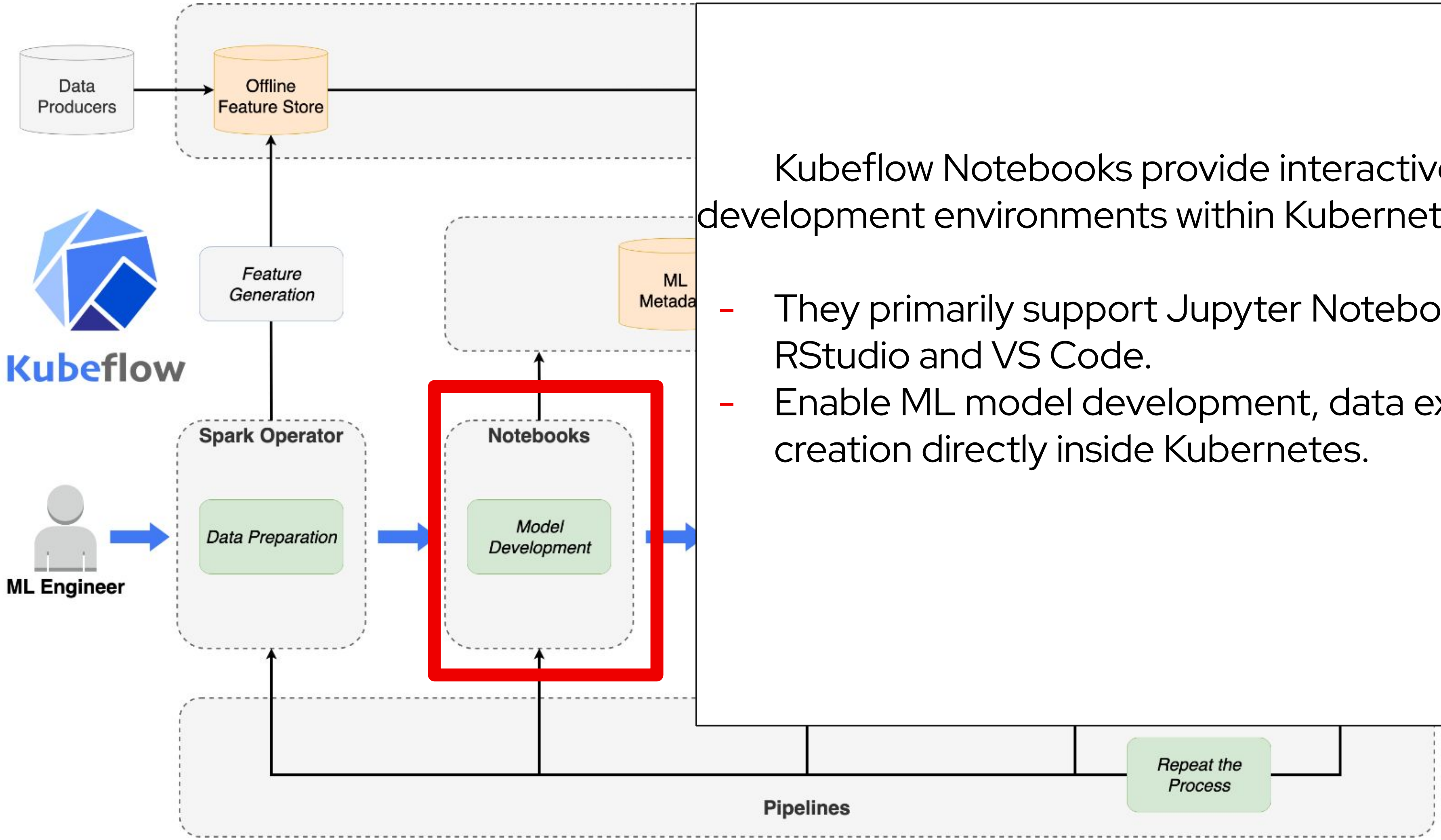
ML Lifecycle for Production and Development Phases



ML Lifecycle for Production and Development Phases

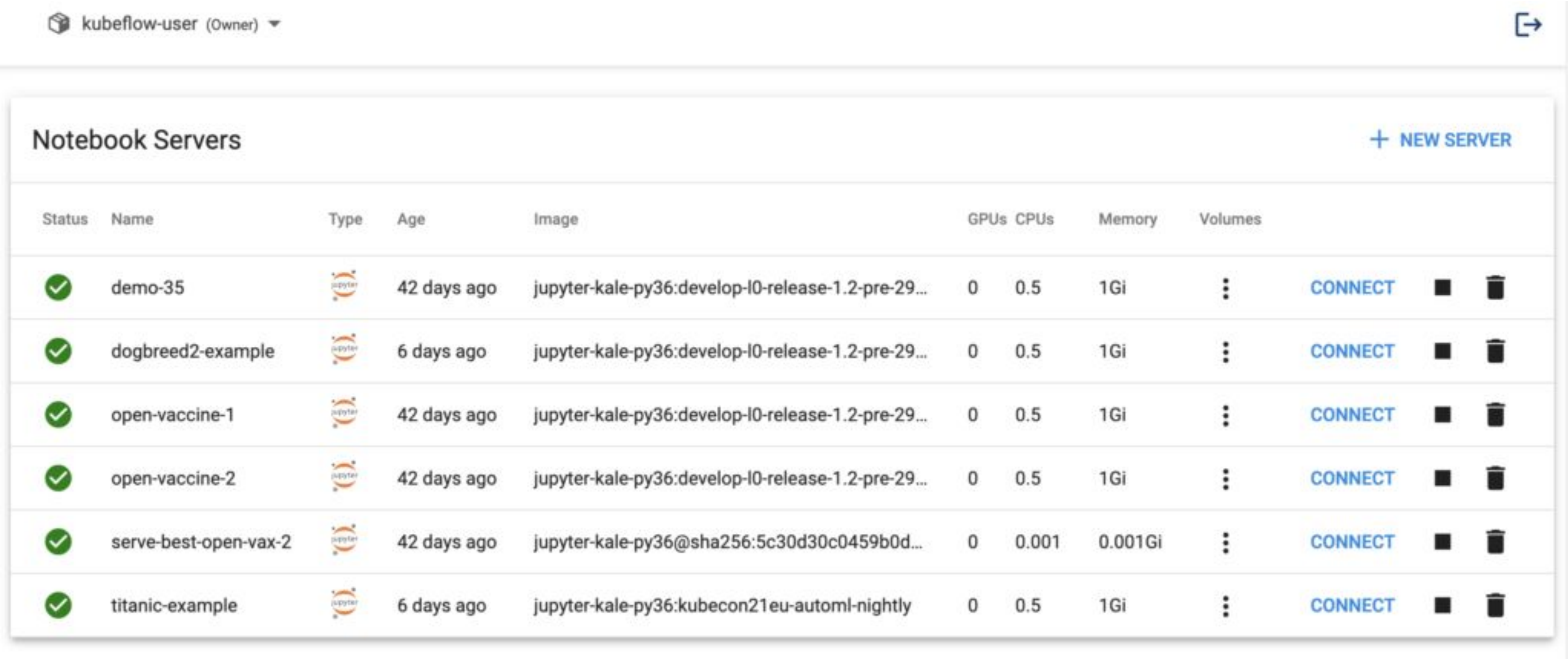


Kubeflow Notebooks



Kubeflow Notebooks provide interactive, containerized development environments within Kubernetes.

- They primarily support Jupyter Notebooks, but also work with RStudio and VS Code.
- Enable ML model development, data exploration, and pipeline creation directly inside Kubernetes.

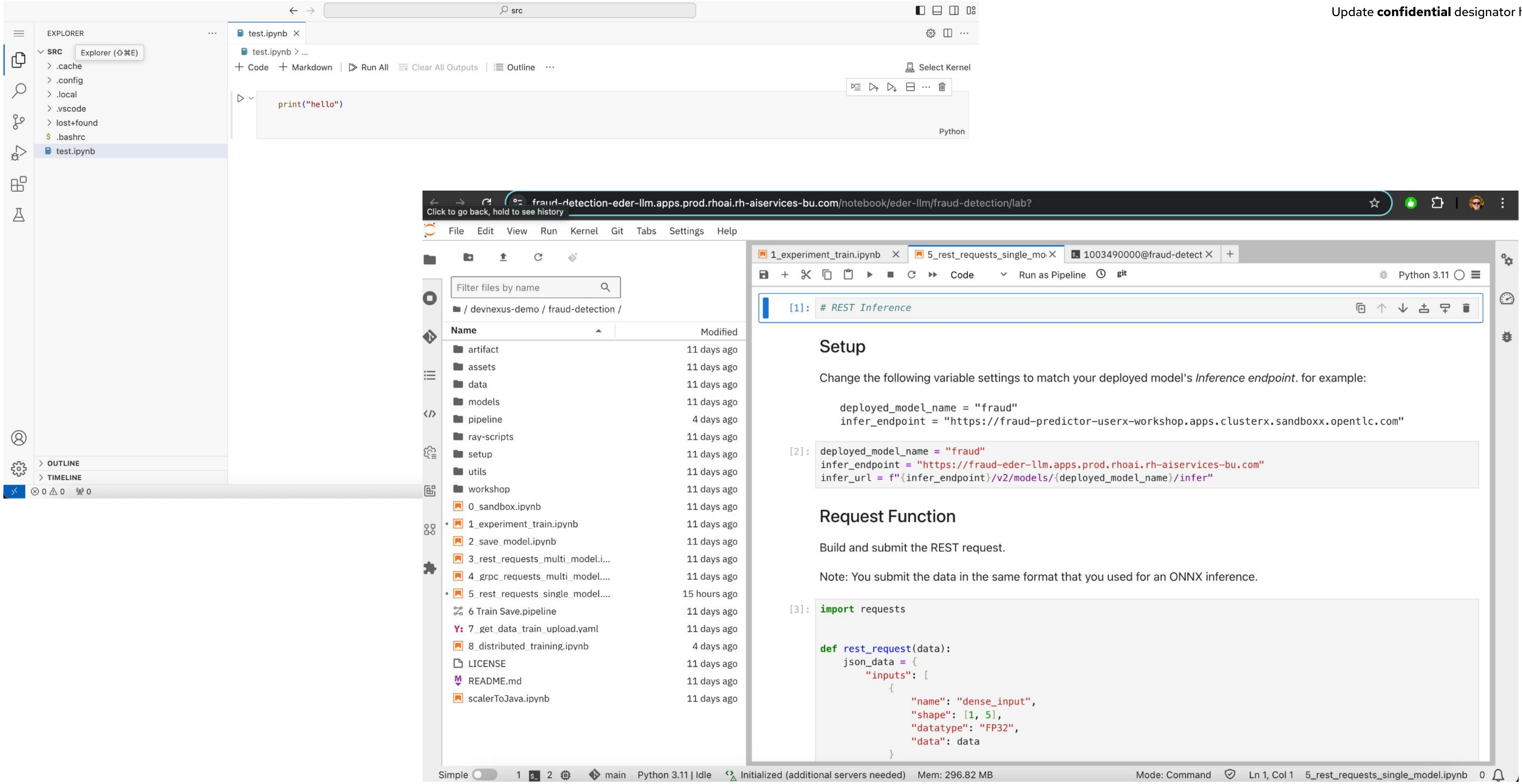


kubeflow-user (Owner)

Notebook Servers

\+ NEW SERVER

Status	Name	Type	Age	Image	GPUs	CPUs	Memory	Volumes			
	demo-35	jupyter	42 days ago	jupyter-kale-py36:develop-l0-release-1.2-pre-29...	0	0.5	1Gi		CONNECT		
	dogbreed2-example	jupyter	6 days ago	jupyter-kale-py36:develop-l0-release-1.2-pre-29...	0	0.5	1Gi		CONNECT		
	open-vaccine-1	jupyter	42 days ago	jupyter-kale-py36:develop-l0-release-1.2-pre-29...	0	0.5	1Gi		CONNECT		
	open-vaccine-2	jupyter	42 days ago	jupyter-kale-py36:develop-l0-release-1.2-pre-29...	0	0.5	1Gi		CONNECT		
	serve-best-open-vax-2	jupyter	42 days ago	jupyter-kale-py36@sha256:5c30d30c0459b0d...	0	0.001	0.001Gi		CONNECT		
	titanic-example	jupyter	6 days ago	jupyter-kale-py36:kubecon21eu-automl-nightly	0	0.5	1Gi		CONNECT		


src
EXPLORER
SRC
Explorer (⇧⌘E)
.cache
.config
.local
.vscode
lost+found
.bashrc
test.ipynb
OUTLINE
TIMELINE
test.ipynb
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
Select Kernel
print("hello")
Python
fraud-detection-eder-llm.apps.prod.rhoai.rh-aiservices-bu.com/notebook/eder-llm/fraud-detection/lab?
Click to go back, hold to see history
File Edit View Run Kernel Git Tabs Settings Help
1_experiment_train.ipynb
5_rest_requests_single_mo
1003490000@fraud-detect
Filter files by name
/ devnexus-demo / fraud-detection /
Name
Modified
artifact 11 days ago
assets 11 days ago
data 11 days ago
models 11 days ago
pipeline 4 days ago
ray-scripts 11 days ago
setup 11 days ago
utils 11 days ago
workshop 11 days ago
0_sandbox.ipynb 11 days ago
1_experiment_train.ipynb 11 days ago
2_save_model.ipynb 11 days ago
3_rest_requests_multi_model.i... 11 days ago
4_grpc_requests_multi_model.... 11 days ago
5_rest_requests_single_model.... 15 hours ago
6 Train Save.pipeline 11 days ago
7_get_data_train_upload.yaml 11 days ago
8_distributed_training.ipynb 4 days ago
LICENSE 11 days ago
README.md 11 days ago
scalerToJava.ipynb 11 days ago
Code
Run as Pipeline
git
Python 3.11
[1]: # REST Inference
Setup
Change the following variable settings to match your deployed model's Inference endpoint. for example:
deployed_model_name = "fraud"
infer_endpoint = "https://fraud-predictor-userx-workshop.apps.clusterx.sandboxx.opentlc.com"
[2]: deployed_model_name = "fraud"
infer_endpoint = "https://fraud-eder-llm.apps.prod.rhoai.rh-aiservices-bu.com"
infer_url = f"{infer_endpoint}/v2/models/{deployed_model_name}/infer"
Request Function
Build and submit the REST request.
Note: You submit the data in the same format that you used for an ONNX inference.
[3]: import requests
def rest_request(data):
json_data = {
"inputs": [
{
"name": "dense_input",
"shape": [1, 5],
"datatype": "FP32",
"data": data
}
Simple
main
Python 3.11 | Idle
Initialized (additional servers needed)
Mem: 296.82 MB
Mode: Command
Ln 1, Col 1
5_rest_requests_single_model.ipynb

Notebook image

Image selection *

TensorFlow

Version selection *

2024.2

CUDA v12.4, Python v3.11, TensorFlow v2.17

Hover over a version to view its included packages.

View package information

Deployment size

Container size

Tiny

Tiny
Limits: 1 CPU, 1GiB Memory Requests: 500m CPU, 1GiB Memory

Small
Limits: 2 CPU, 8GiB Memory Requests: 1 CPU, 8GiB Memory

Medium
Limits: 6 CPU, 24GiB Memory Requests: 3 CPU, 16GiB Memory

Large
Limits: 14 CPU, 56GiB Memory Requests: 7 CPU, 56GiB Memory

X Large
Limits: 30 CPU, 120GiB Memory Requests: 15 CPU, 120GiB Memory

Deployment size

Container size

X Large

Limits: 30 CPU, 120GiB Memory Requests: 15 CPU, 120GiB Memory

Accelerator

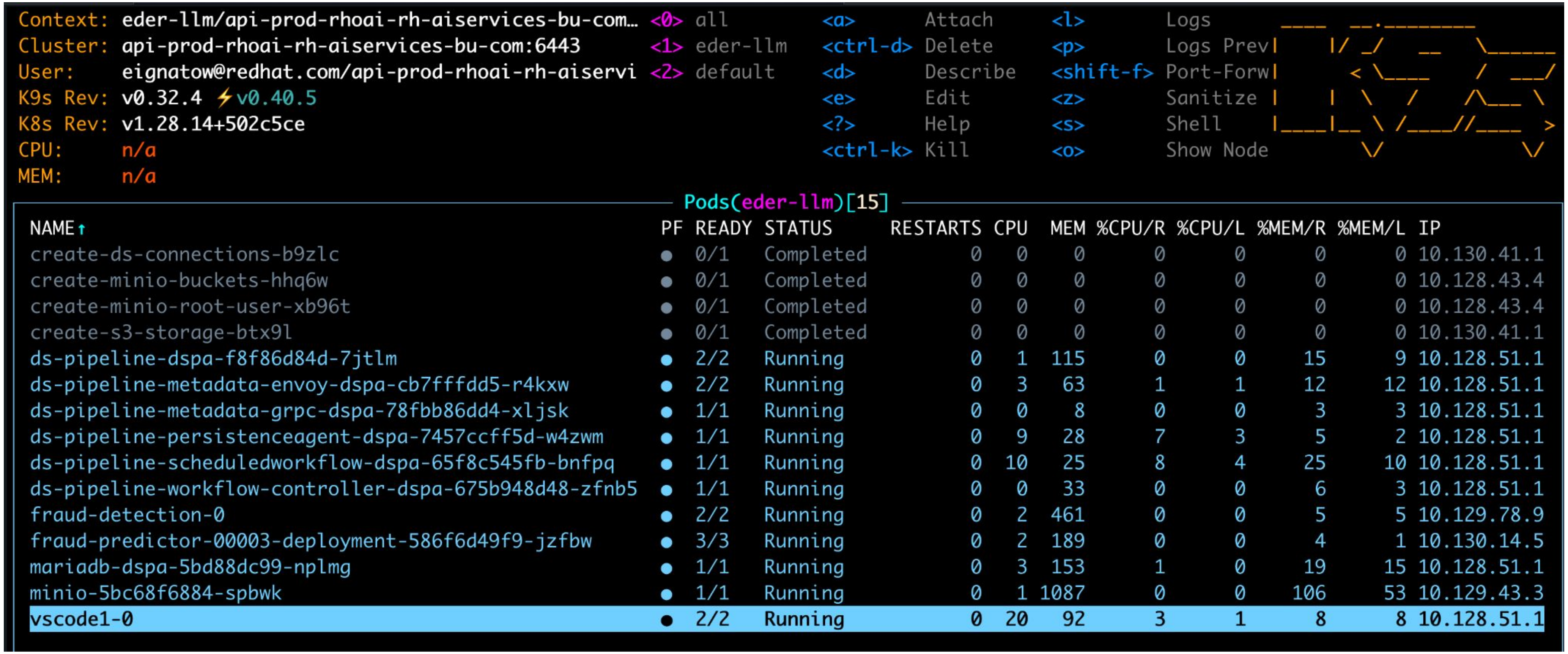
None

Large GPU Card (NVIDIA A10G - 24 GB VRAM)
Restricted use - Do not select without approval

Medium GPU Card (NVIDIA T4 - 16 GB VRAM)
Regular users should select this

None

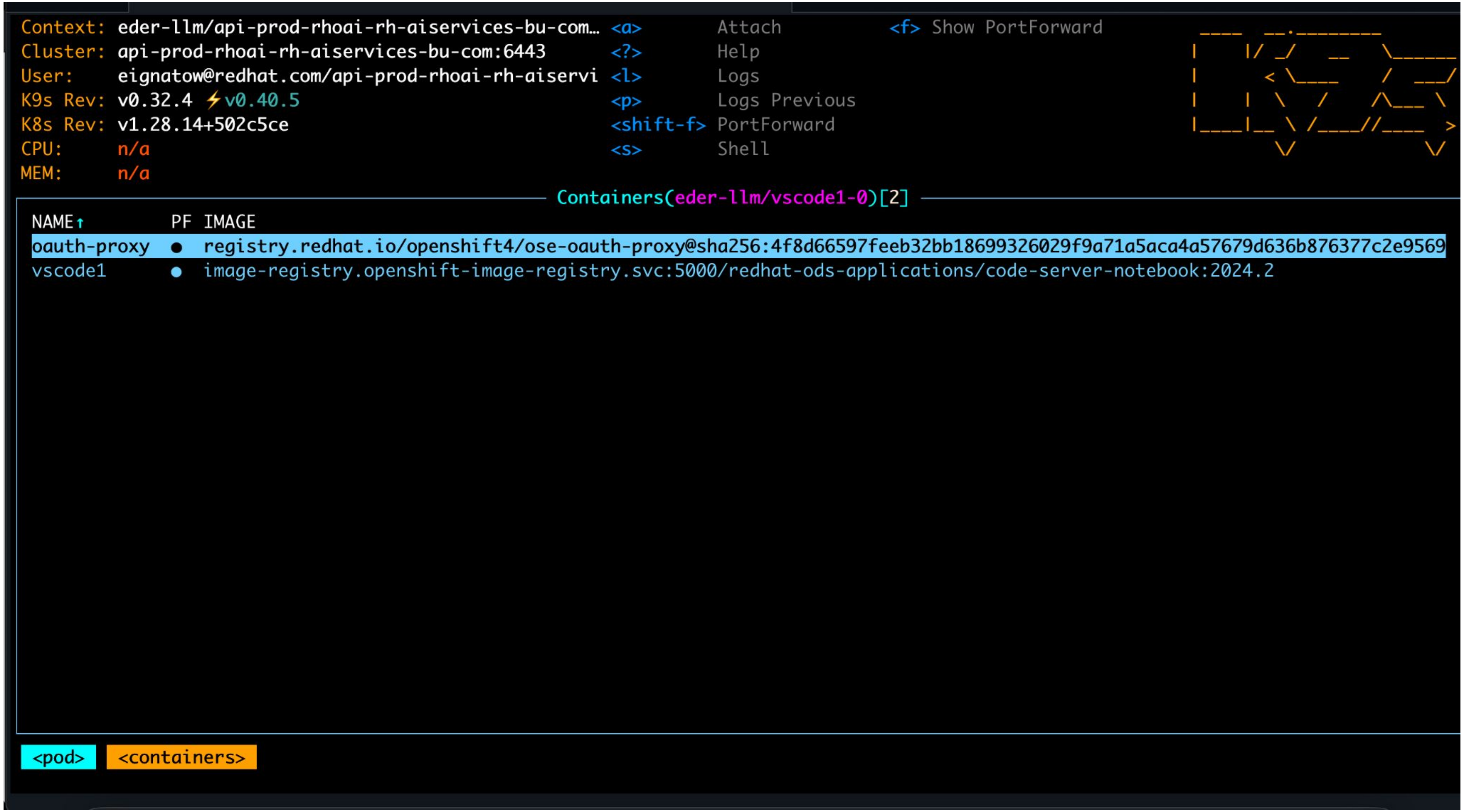
```yaml
apiVersion: kubeflow.org/v1
kind: Notebook
metadata:
  name: my-kubeflow-notebook
  namespace: my-namespace
spec:
  template:
    spec:
      serviceAccountName: kubeflow-notebook
      containers:
        - name: notebook-container
          image: quay.io/jupyter/minimal-notebook
          workingDir: /home/jovyan
          command:
            - "start-notebook.sh"
          resources:
            requests:
              cpu: "2"
              memory: "4Gi"
            limits:
              cpu: "4"
              memory: "8Gi"
          volumeMounts:
            - name: workspace
              mountPath: /home/jovyan
      volumes:
        - name: workspace
          persistentVolumeClaim:
            claimName: my-notebook-pvc
```



Context: eder-llm/api-prod-rhoai-rh-aiservices-bu-com… <0> all <a> Attach <l> Logs
Cluster: api-prod-rhoai-rh-aiservices-bu-com:6443 <1> eder-llm <ctrl-d> Delete <p> Logs Prev
User: eignatow@redhat.com/api-prod-rhoai-rh-aiservi <2> default <d> Describe <shift-f> Port-Forw
K9s Rev: v0.32.4 ⚡v0.40.5 <e> Edit <z> Sanitize
K8s Rev: v1.28.14+502c5ce <?> Help <s> Shell
CPU: n/a <ctrl-k> Kill <o> Show Node
MEM: n/a

Pods(eder-llm)[15]

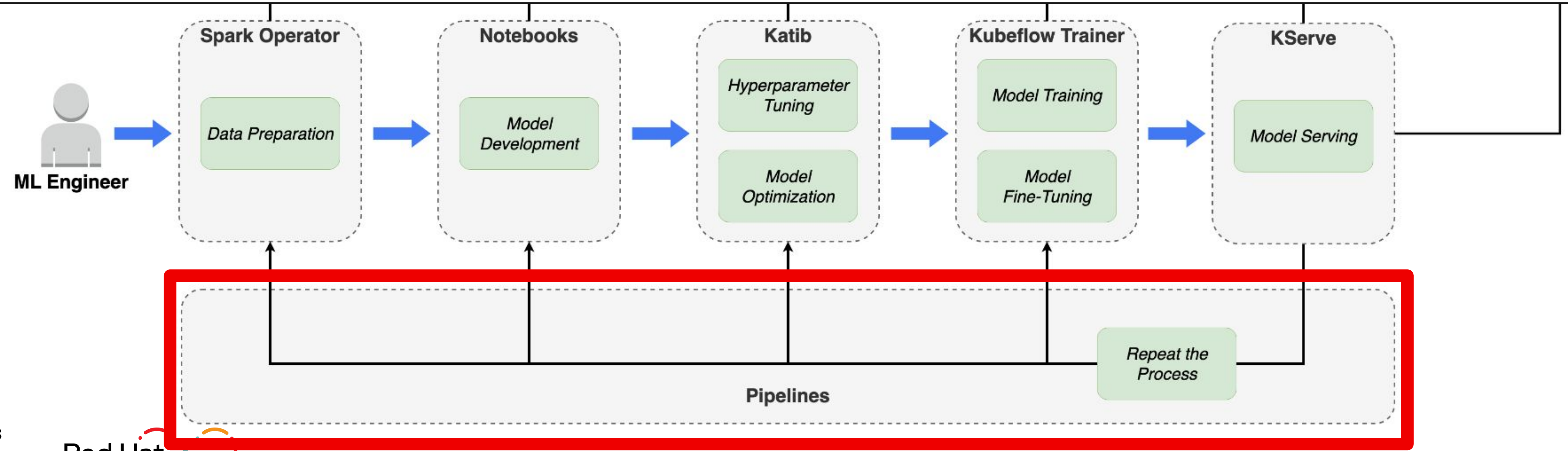
NAME↑	PF	READY	STATUS	RESTARTS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP
create-ds-connections-b9zlc	●	0/1	Completed	0	0	0	0	0	0	0	10.130.41.1
create-minio-buckets-hhq6w	●	0/1	Completed	0	0	0	0	0	0	0	10.128.43.4
create-minio-root-user-xb96t	●	0/1	Completed	0	0	0	0	0	0	0	10.128.43.4
create-s3-storage-btx9l	●	0/1	Completed	0	0	0	0	0	0	0	10.130.41.1
ds-pipeline-dspa-f8f86d84d-7jtlm	●	2/2	Running	0	1	115	0	0	15	9	10.128.51.1
ds-pipeline-metadata-envoy-dspa-cb7fffdd5-r4kxw	●	2/2	Running	0	3	63	1	1	12	12	10.128.51.1
ds-pipeline-metadata-grpc-dspa-78fbb86dd4-xljsk	●	1/1	Running	0	0	8	0	0	3	3	10.128.51.1
ds-pipeline-persistenceagent-dspa-7457ccff5d-w4zwm	●	1/1	Running	0	9	28	7	3	5	2	10.128.51.1
ds-pipeline-scheduledworkflow-dspa-65f8c545fb-bnfpq	●	1/1	Running	0	10	25	8	4	25	10	10.128.51.1
ds-pipeline-workflow-controller-dspa-675b948d48-zfnb5	●	1/1	Running	0	0	33	0	0	6	3	10.128.51.1
fraud-detection-0	●	2/2	Running	0	2	461	0	0	5	5	10.129.78.9
fraud-predictor-00003-deployment-586f6d49f9-jzfbw	●	3/3	Running	0	2	189	0	0	4	1	10.130.14.5
mariadb-dspa-5bd88dc99-nplmg	●	1/1	Running	0	3	153	1	0	19	15	10.128.51.1
minio-5bc68f6884-spbwk	●	1/1	Running	0	1	1087	0	0	106	53	10.129.43.3
vscode1-0	●	2/2	Running	0	20	92	3	1	8	8	10.128.51.1


Context: eder-llm/api-prod-rhoai-rh-aiservices-bu-com…
Cluster: api-prod-rhoai-rh-aiservices-bu-com:6443
User: eignatow@redhat.com/api-prod-rhoai-rh-aiservi
K9s Rev: v0.32.4 ⚡v0.40.5
K8s Rev: v1.28.14+502c5ce
CPU: n/a
MEM: n/a
<a> Attach
<?> Help
<l> Logs
<p> Logs Previous
<shift-f> PortForward
<s> Shell
<f> Show PortForward
Containers(eder-llm/vscode1-0)[2]
NAME↑ PF IMAGE
oauth-proxy ● registry.redhat.io/openshift4/ose-oauth-proxy@sha256:4f8d66597feeb32bb18699326029f9a71a5aca4a57679d636b876377c2e9569
vscode1 ● image-registry.openshift-image-registry.svc:5000/redhat-ods-applications/code-server-notebook:2024.2
<pod>
<containers>

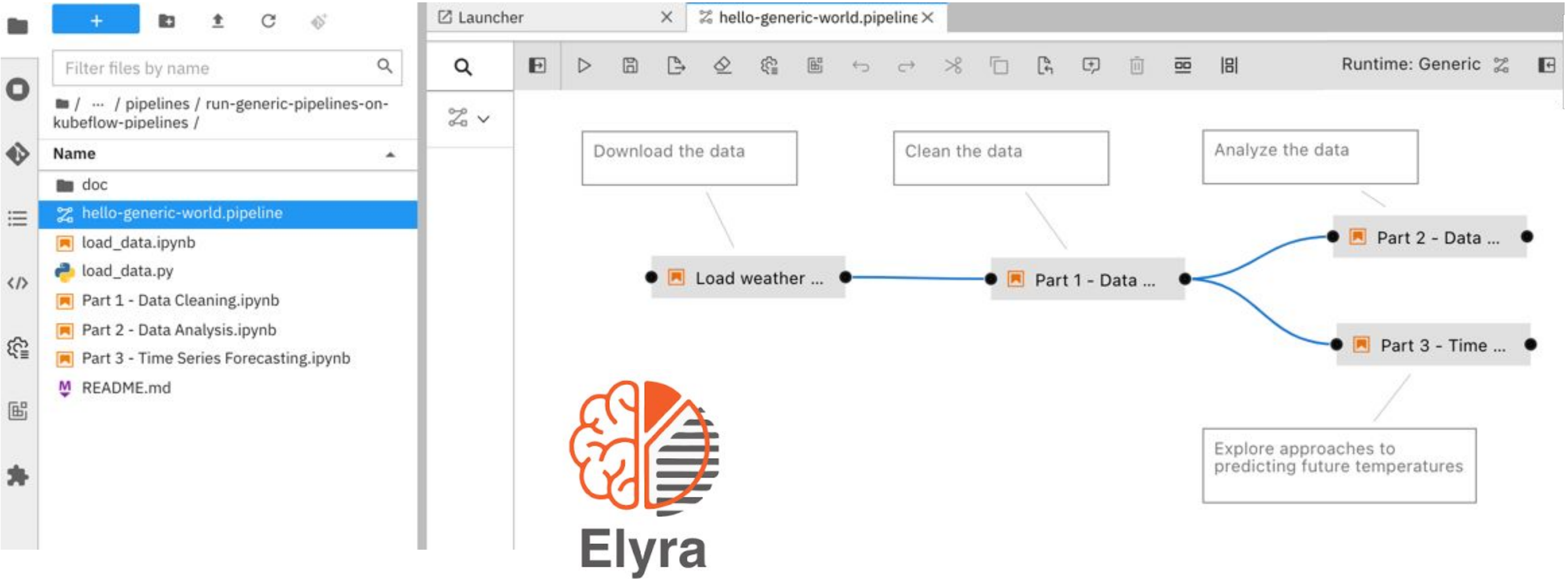
Kubeflow Pipelines

Kubeflow Pipelines (KFP) help orchestrate, automate, and manage ML workflows in Kubernetes.

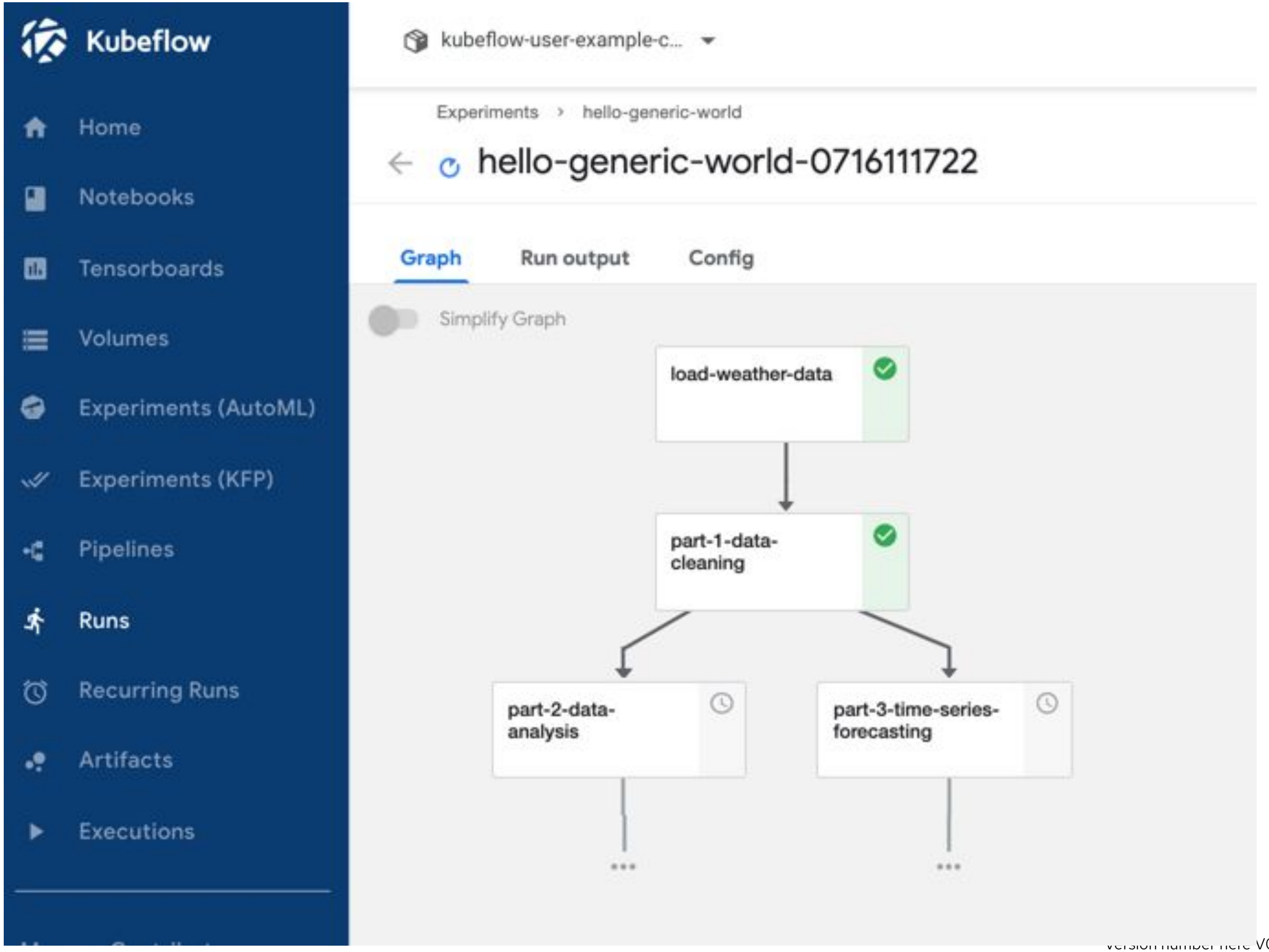
- They define ML tasks as a Directed Acyclic Graph (DAG), ensuring step-by-step reproducible execution.
- Each step (data processing, training, evaluation, deployment) runs in containerized microservices.

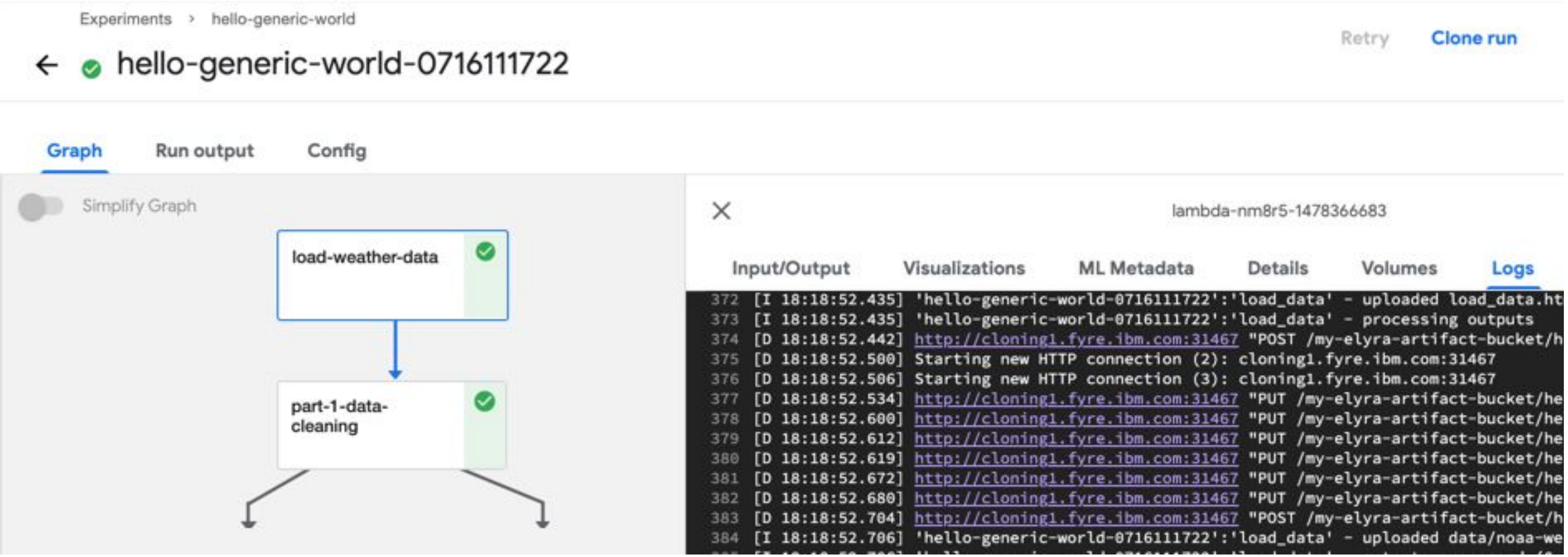


```python
from kfp import dsl

@dsl.component
def say_hello(name: str) -> str:
    hello_text = f'Hello, {name}!'
    print(hello_text)
    return hello_text

@dsl.pipeline
def hello_pipeline(recipient: str) -> str:
    hello_task = say_hello(name=recipient)
    return hello_task.output
```

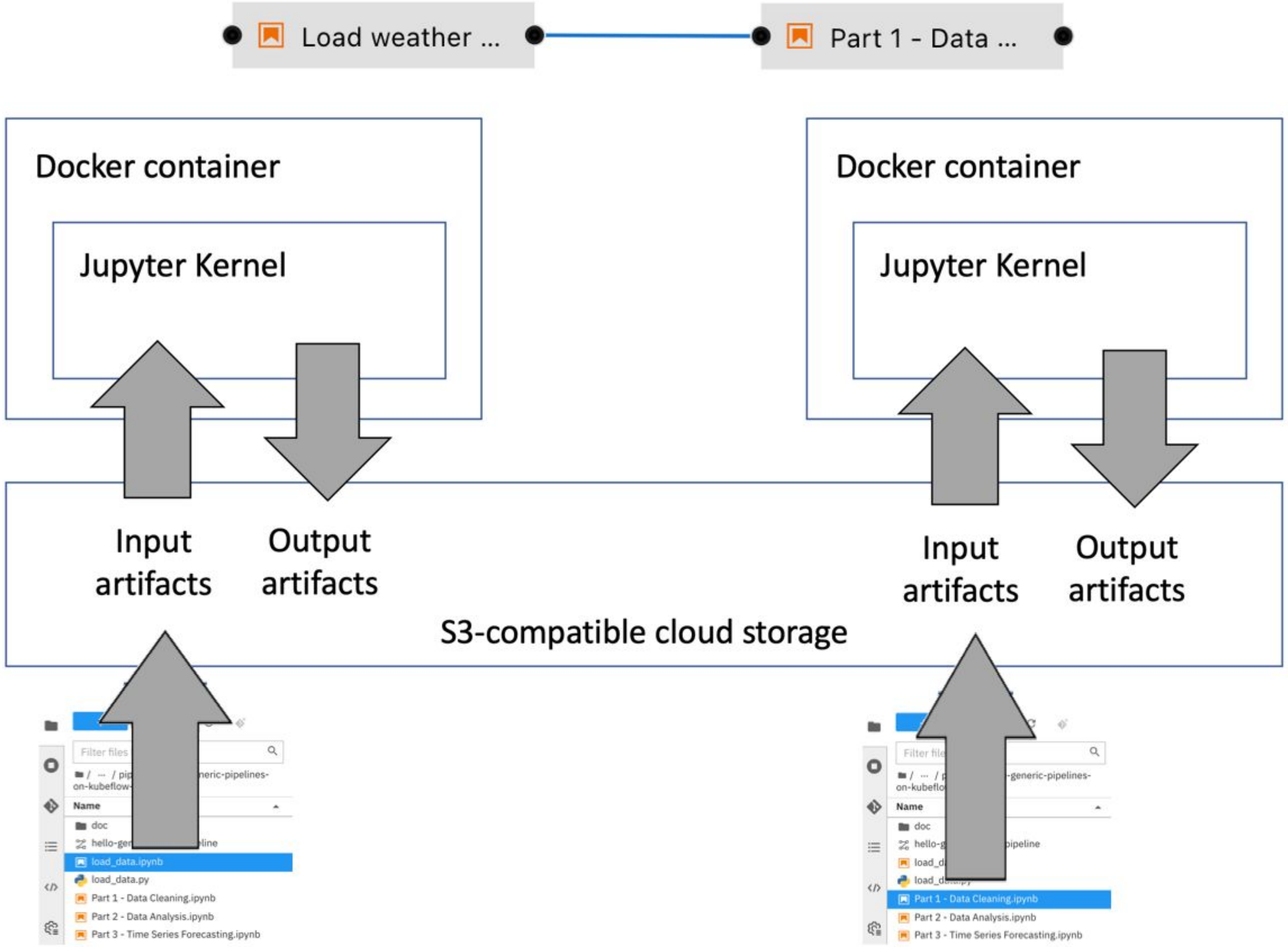


https://github.com/harshad16/data-science-pipeline-example/tree/master/run-pipelines-on-data-science-pipelines


Kubeflow
kubeflow-user-example-c...
Home
Notebooks
Tensorboards
Volumes
Experiments (AutoML)
Experiments (KFP)
Pipelines
Runs
Recurring Runs
Artifacts
Executions
Experiments › hello-generic-world
hello-generic-world-0716111722
Graph
Run output
Config
Simplify Graph
load-weather-data
part-1-data-cleaning
part-2-data-analysis
part-3-time-series-forecasting

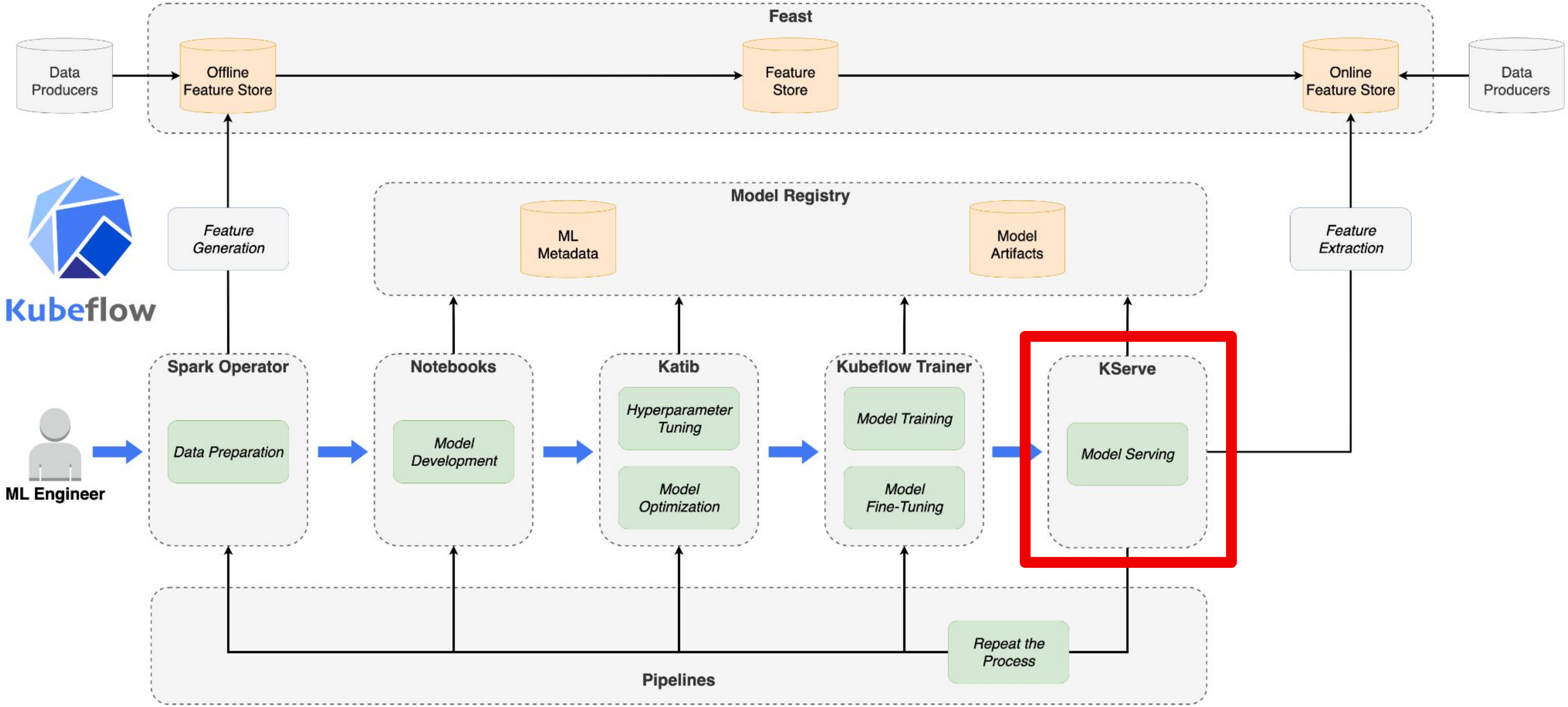


https://github.com/harshad16/data-science-pipeline-example/tree/master/run-pipelines-on-data-science-pipelines

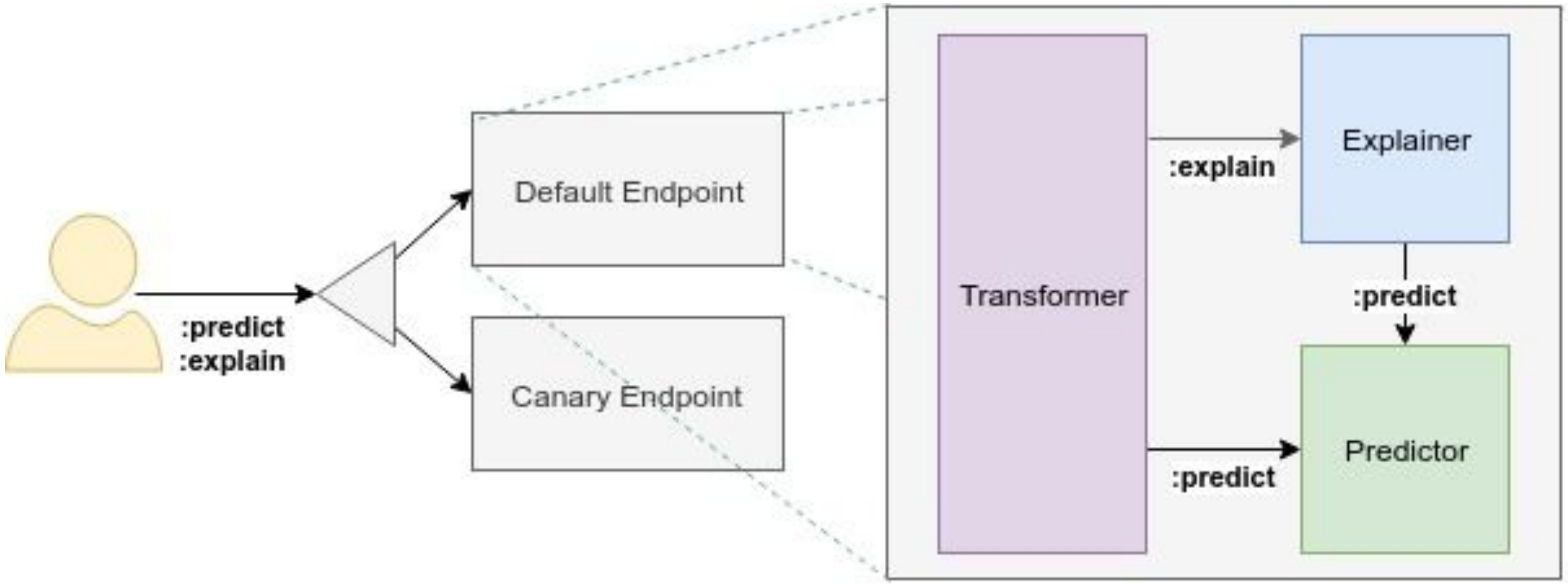


https://github.com/harshad16/data-science-pipeline-example/tree/master/run-pipelines-on-data-science-pipelines

KServe


Feast
Data Producers
Offline Feature Store
Feature Store
Online Feature Store
Data Producers
Model Registry
ML Metadata
Model Artifacts
Kubeflow
Feature Generation
Feature Extraction
Spark Operator
Data Preparation
Notebooks
Model Development
Katib
Hyperparameter Tuning
Model Optimization
Kubeflow Trainer
Model Training
Model Fine-Tuning
KServe
Model Serving
ML Engineer
Repeat the Process
Pipelines

KServe



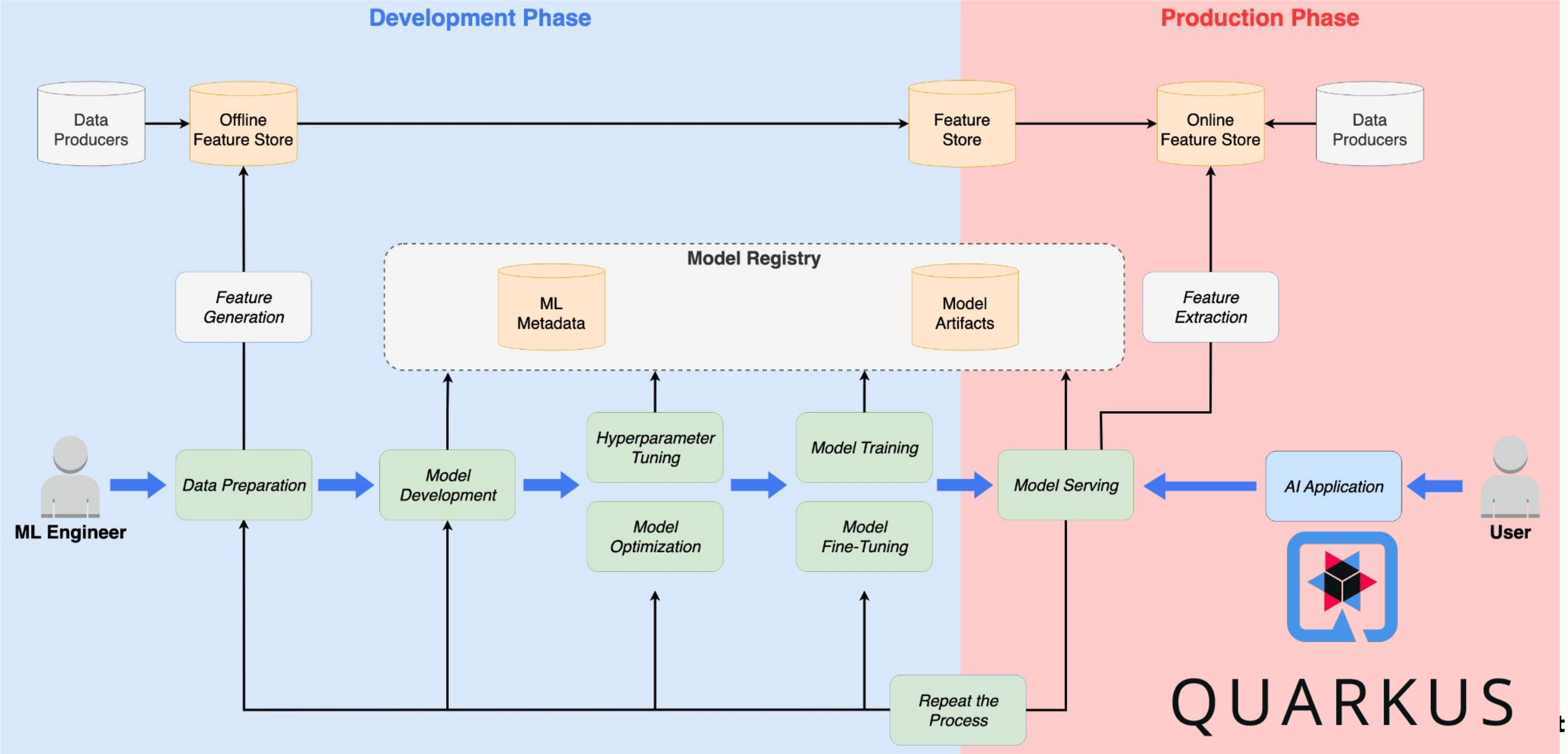

```yaml
apiVersion:
serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: sklearn-iris
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri:
'gs://kfserving-examples/models
/sklearn/1.0/model'
```

Demo Kubeflow Fraud Detection

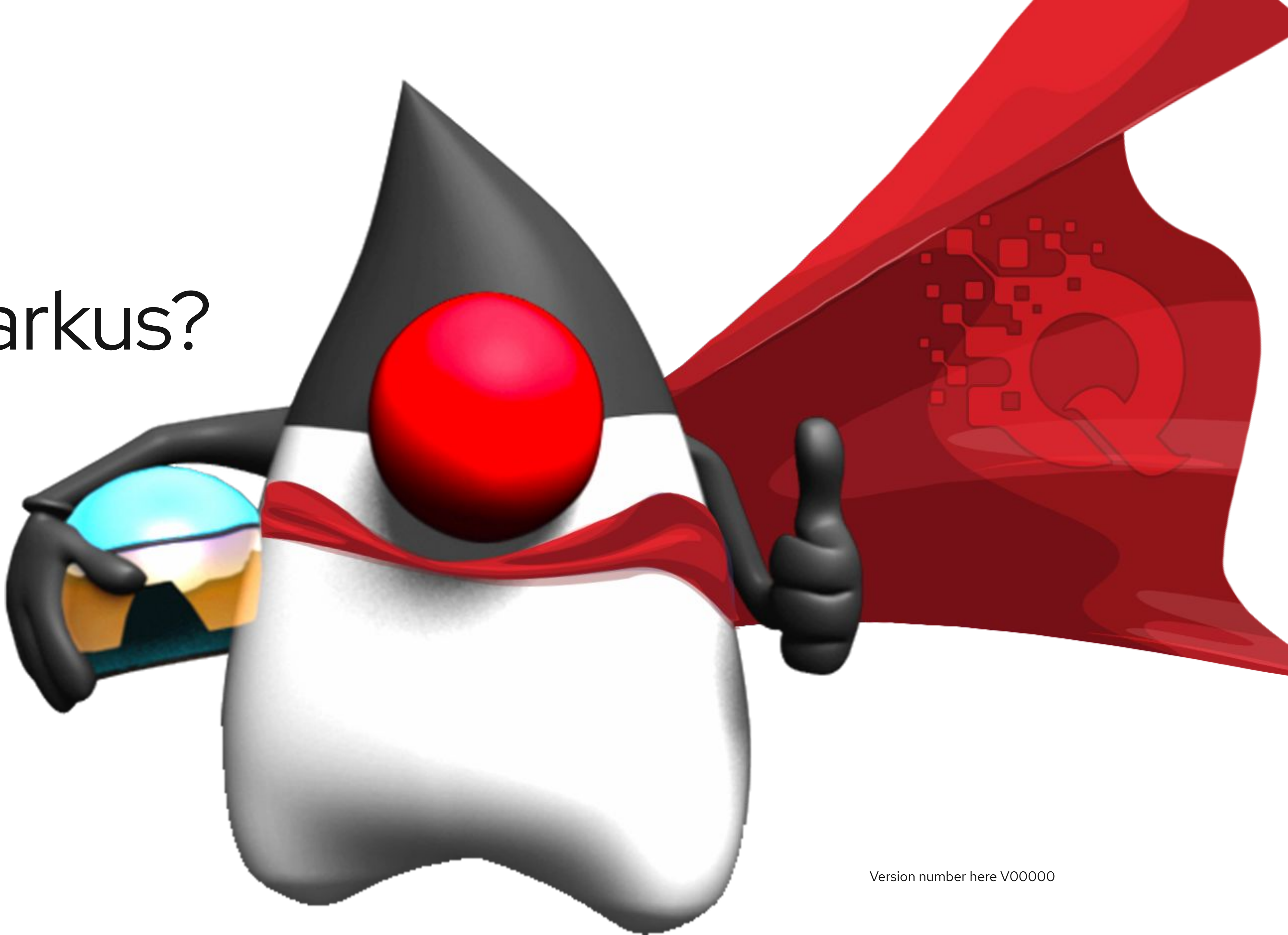
https://ai-on-openshift.io/demos/financial-fraud-detection/financial-fraud-detection/

Update **confidential** designator here

ML Lifecycle for Production and Development Phases



Why Quarkus?



Modern Java Stack

Cloud Native

(Micro)Services

Serverless

Modern Java Stack

Cloud Native

(Micro)Services

Serverless

It’s perfectly fine for Monoliths too :-)

Traditional vs. Quarkus

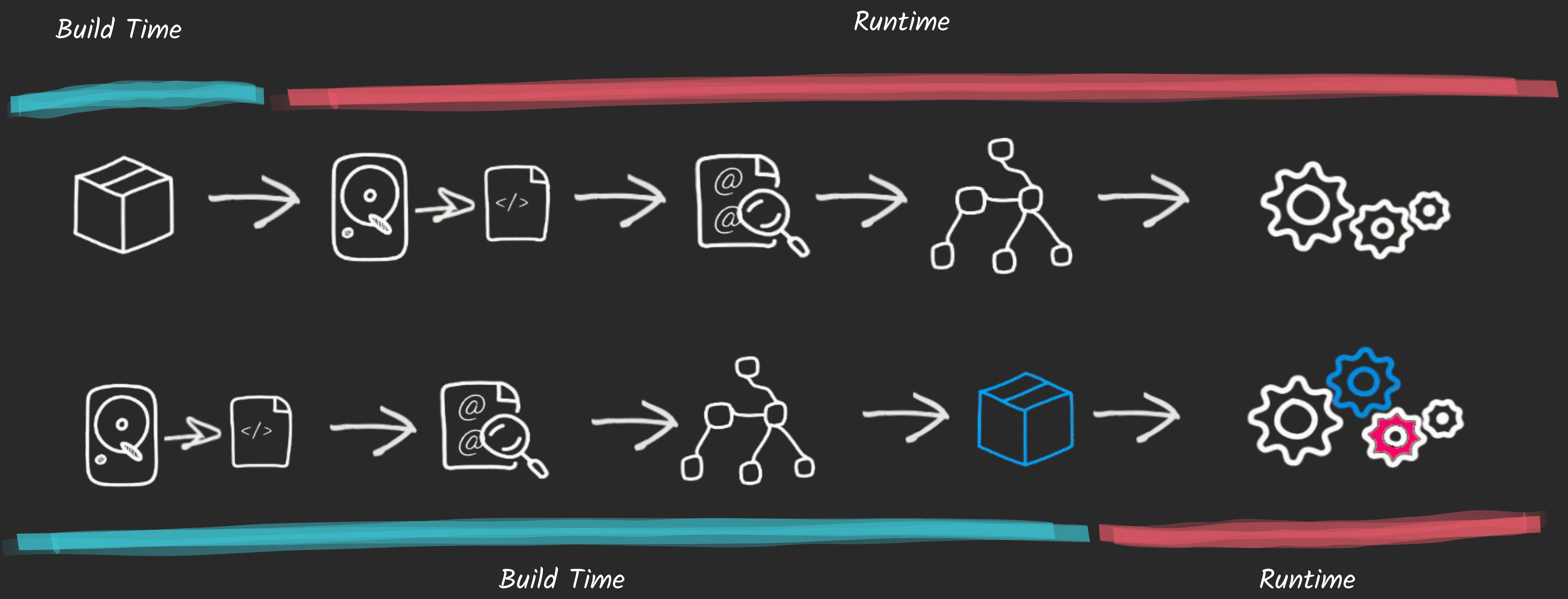
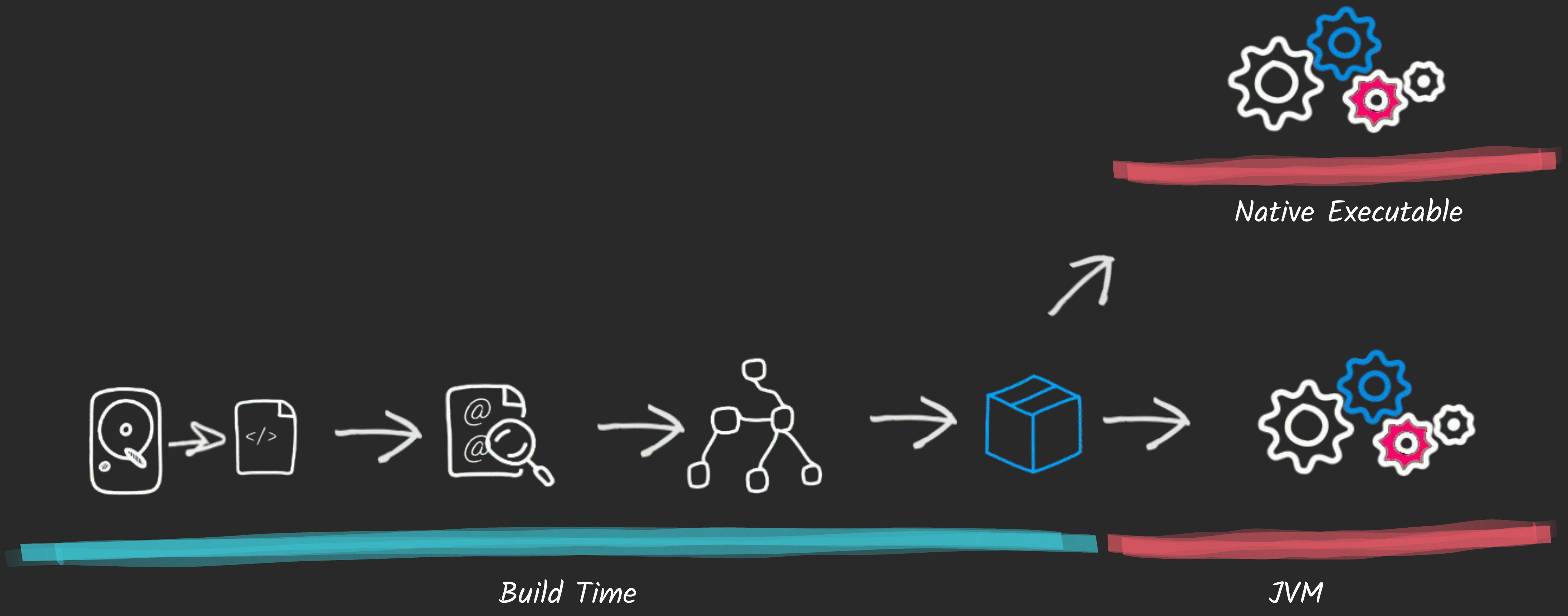



FIGURE OUT EVERYTHING AT RUNTIME
DO MORE STUFF AT BUILD TIME
imgflip.com

Native Compilation



Kubernetes Native

On the shoulders of Giants

et al.

Demo Quarkus
Fraud Detection

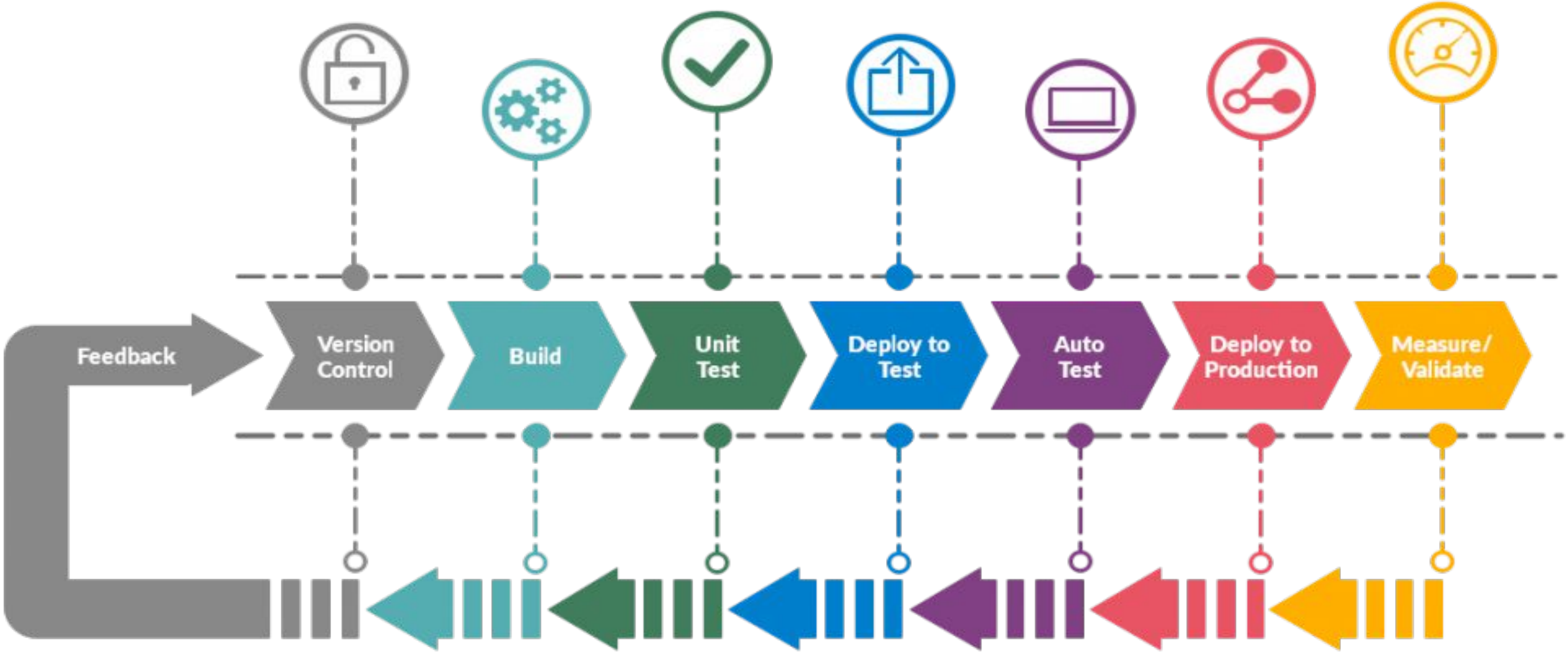
Quarkus & Langchain4j

Langchain4j

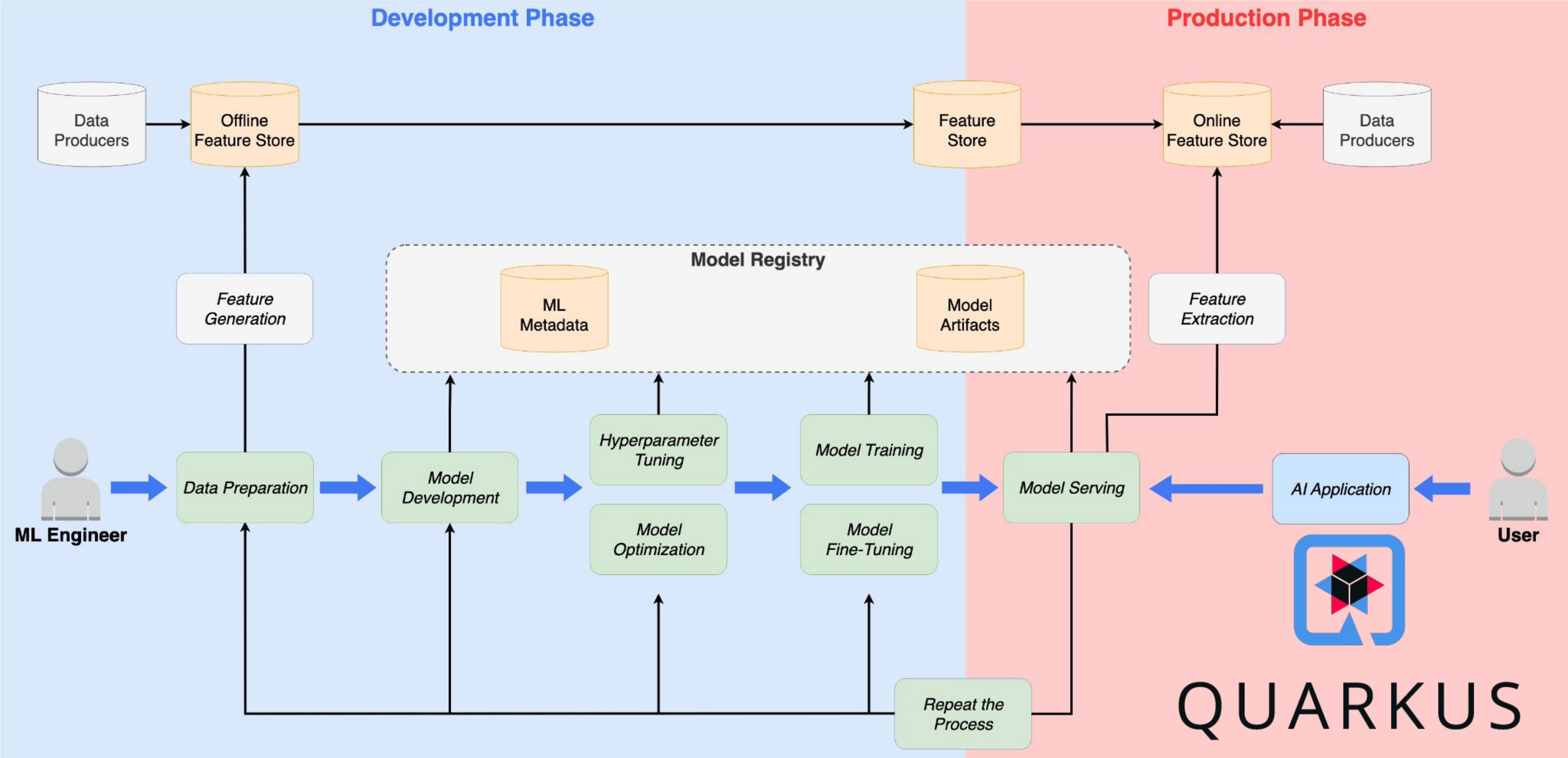
- Unified APIs: LLMs providers and embedding stores use proprietary APIs. LangChain4j abstracts them for you;
- Ready-to-use: prompt templating, memory management, agents, RAGs, etc; you have interfaces and implementations so you can get things done quickly;
- 4j: because Java is fun! :-)

Demo Quarkus & Langchain4j
Models & Multi-models

Devops Pipeline



Kubeflow Pipelines: Reproducible ML Workflow



Red Hat Developer

Thank you

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat