

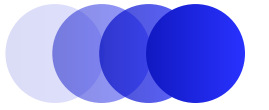
MLops Project:

Orchestrating Continuous NAS with Early Stopping



Prepared by: ELAKIL Hakima

Suggested by: Mr. EL KALLOUBI Fahd



Plan

① General Introduction

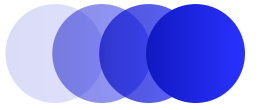
② Kubeflow Fondamntals & Katib

③ Implemented Pipeline

④ Demonstration

⑤ Conclusion





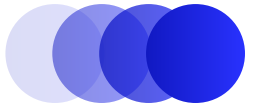
Project Context

Background



- Traditional machine learning workflows are often manual, hard to scale, and difficult to reproduce.
- Hyperparameter tuning requires extensive trial and error, making it a time-consuming and resource-intensive task.
- As a result, many machine learning models never go beyond the experimentation stage and fail to reach production.

Need for an automated, scalable, and production-ready ML workflow



Project Context



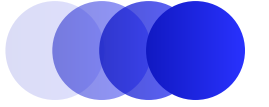
Core Objectives

Research ML



Production ML

- **Automate hyperparameter tuning** to reduce manual effort and experimentation time
- **Enable scalable experimentation** by running multiple trials in parallel
- **Ensure reproducibility** through standardized and controlled ML workflows
- Efficiently utilize resources using Kubernetes-native execution
- Bridge the gap between experimentation and production

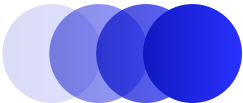


Project Context

MLOps Solution

- MLOps Problems Addressed:
 - Experiment automation
 - Reproducibility
 - Scalability
 - Model deployment
- Our Approach
 - Build a complete end-to-end MLOps pipeline using Kubernetes-based tools



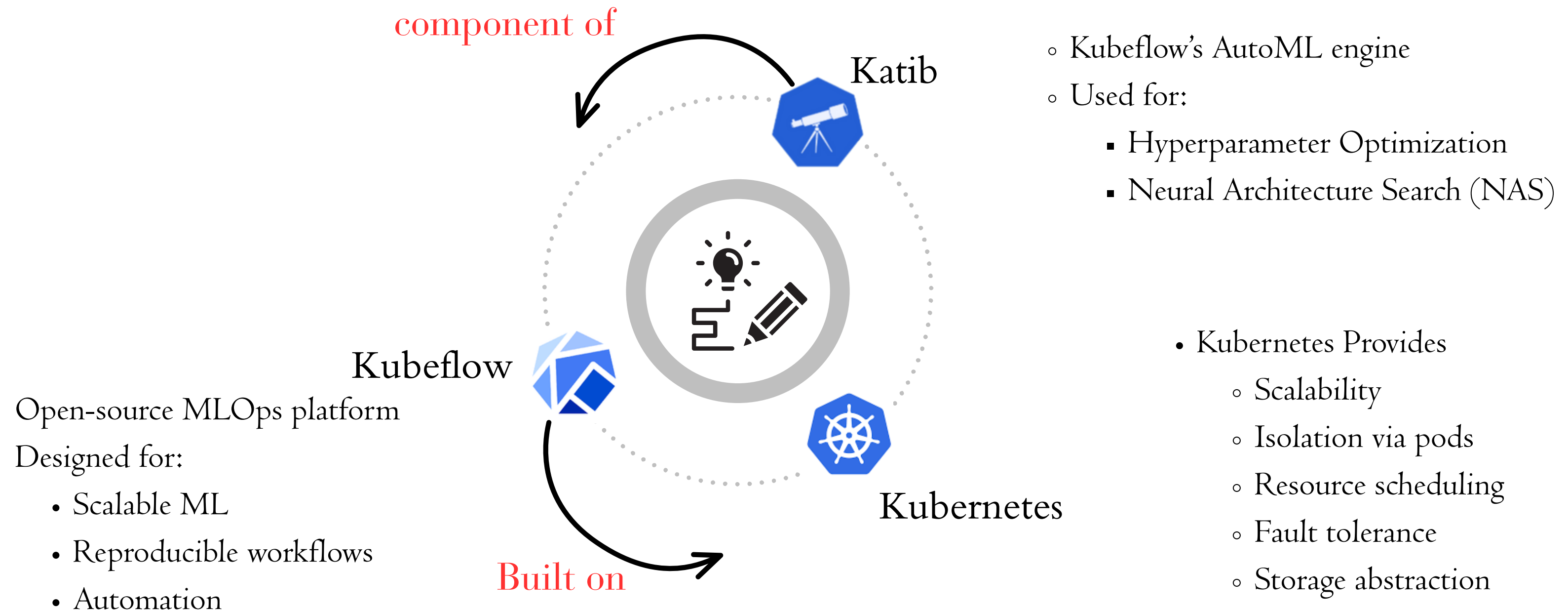


Technologies Used



Technology	Role
MicroK8s	Kubernetes cluster
Docker	Environment reproducibility
Kubeflow Katib	AutoML / NAS
PVC	Model persistence
FastAPI	Model serving

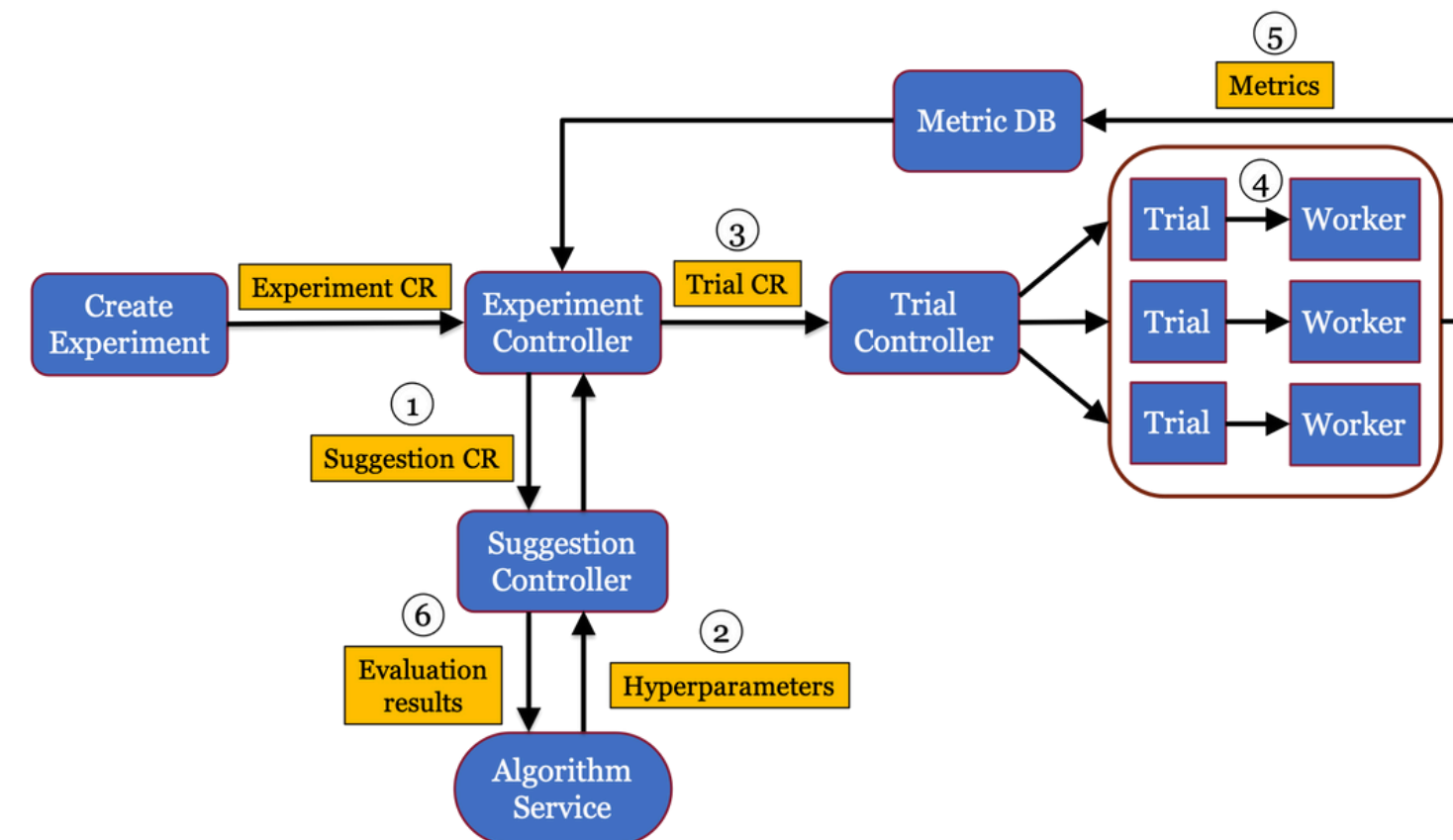
Kubeflow



Katib Experiment

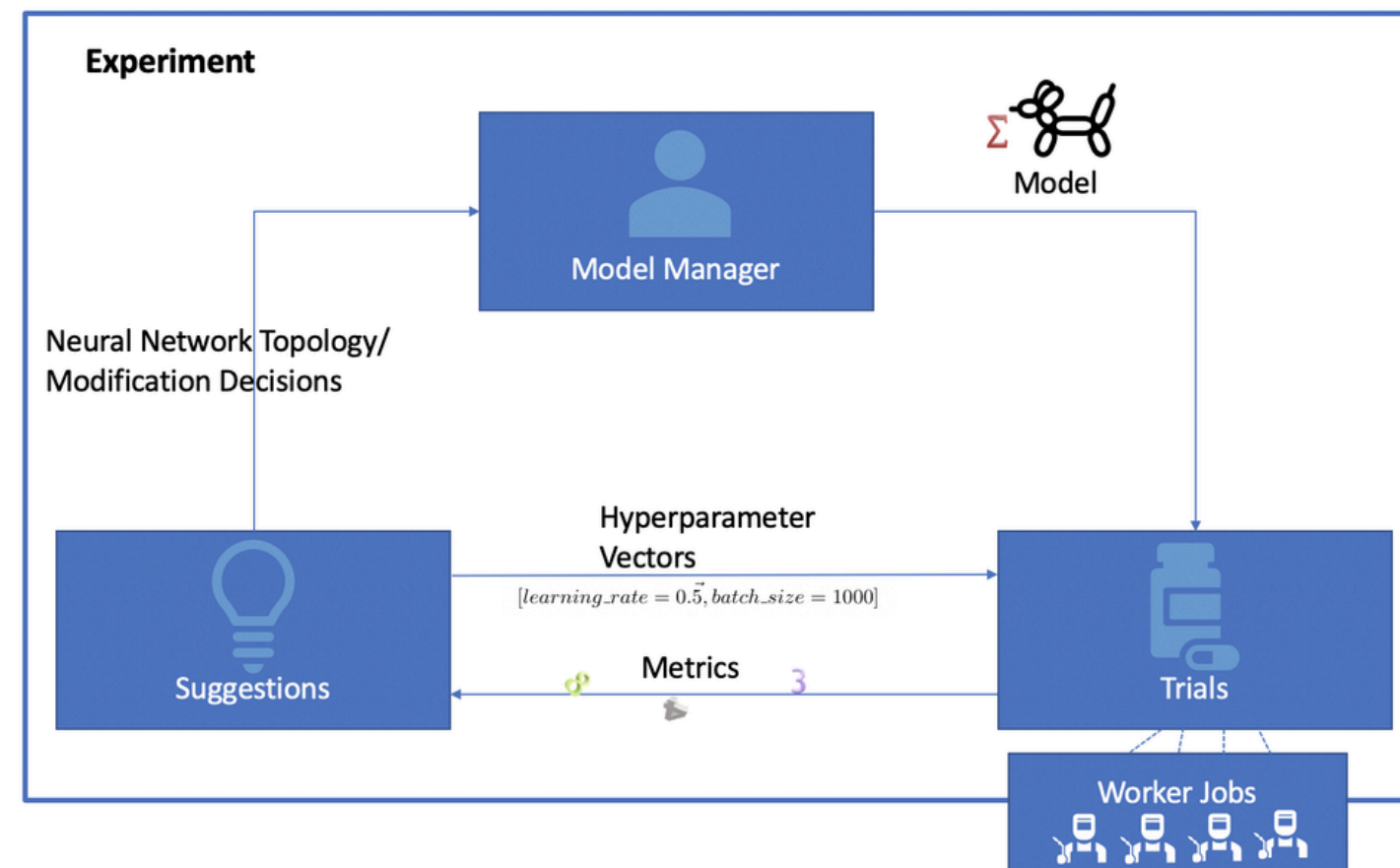
Katib Pipeline:

1. Experiment submitted and a Suggestion is created.
2. Algorithm service generates hyperparameters.
3. Trials are created for each parameter set.
4. Trials run as pods using the trial template.
5. Metrics are collected and stored.
6. Results are fed back to generate new suggestions.



Katib Experiment

Katib Pipeline:



Katib iteratively suggests hyperparameters, runs each trial as a Kubernetes pod, collects metrics, and converges toward the best model.

Project Context

Tools Overview

Pipeline

Workflow

End-to-End Pipeline:

- Containerize training code
- Launch automated experiments
- Optimize hyperparameters
- Save best model
- Deploy model as an API

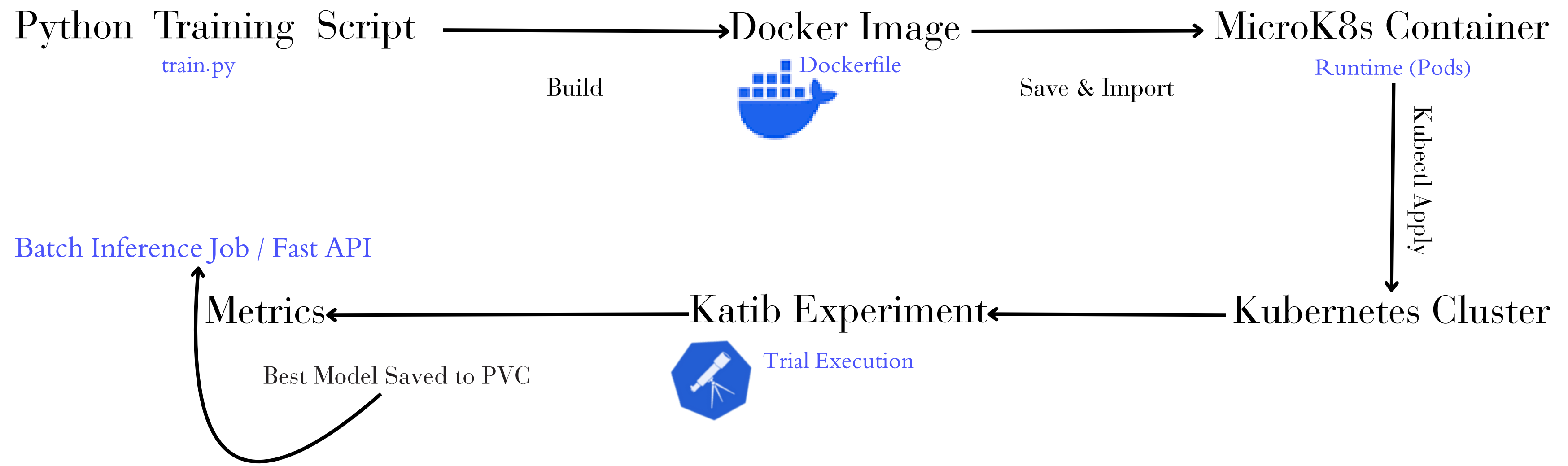


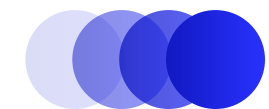
Use Case architecture

katib – nas – atelier /

- | __ train_continuous_nas . py – Core training logic with Early Stopping
- | __ Dockerfile – Environment blueprint for the training pod
- | __ continuous_nas . yaml – Katib Experiment definition (Search Space)
- | __ tmp - pvc - pod . yaml – Helper pod to extract models from PVC
- | __ main . py – FastAPI inference service
- | __ model / – Local mount point for PVC artifacts

Use Case workflow

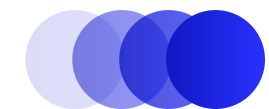




Use Case Steps Definition

Experiment elements Definition

Element	Where it is	Role (what it does)
spec	Experiment level	Global configuration of the Katib experiment
parameters	Experiment spec	Defines what hyperparameters to tune and their
trialTemplate	Experiment spec	Blueprint for how a single trial should run
trialSpec	Inside trialTemplate	Kubernetes Pod spec used for each trial
trialParameters	Inside trialTemplate	Placeholders replaced with actual hyperparameter
primaryContainerName	trialTemplate	Main container where metrics are collected
containers	trialSpec.spec	Defines container image, command, args
command	Container	Program to run (e.g. python train.py)
args	Container	Arguments passed to the program
restartPolicy	trialSpec.spec	Controls what happens when the container exits



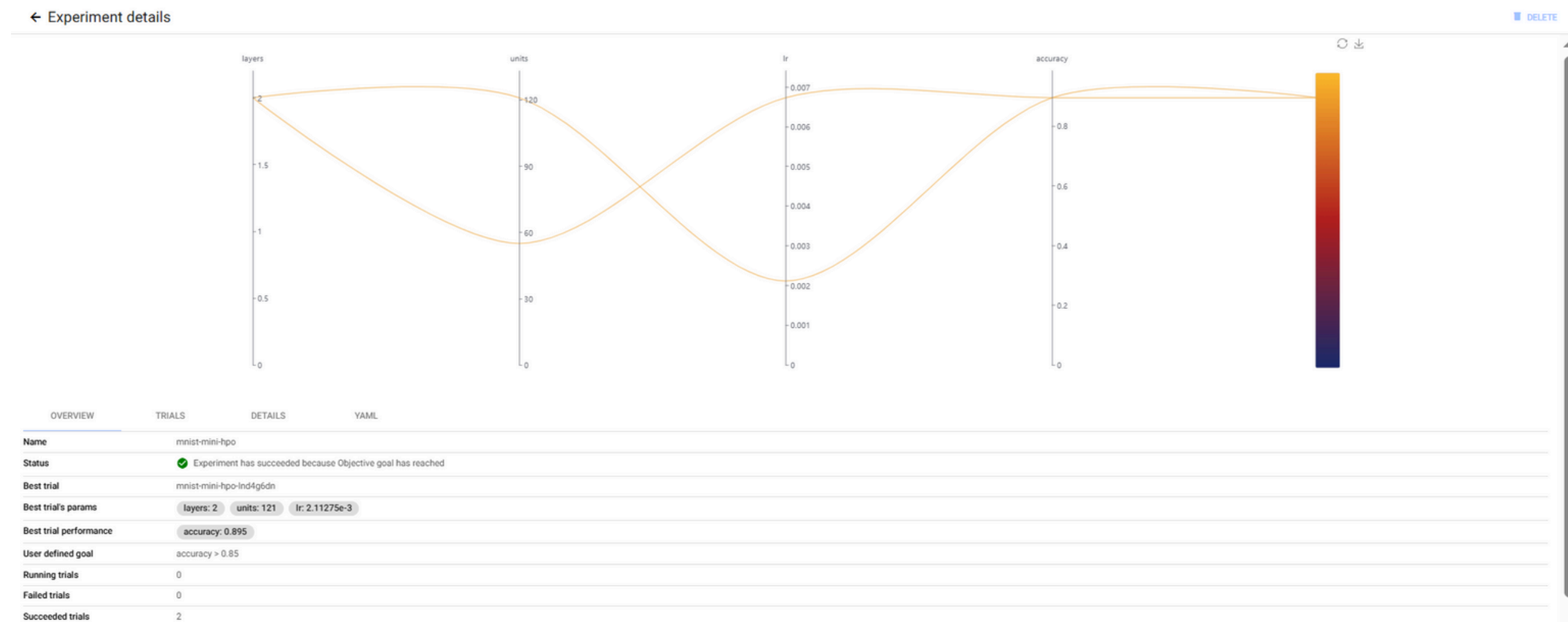
Project Context

Tools Overview

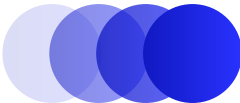
Pipeline

Implementation

Use Case Implementation



Experiment Overview



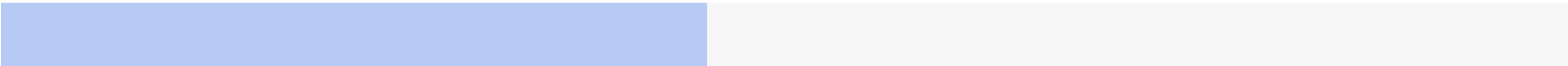
Project Context

Tools Overview

Pipeline

Implementation

Use Case Implementation



OVERVIEW

TRIALS

DETAILS

YAML

Filter

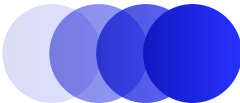
Enter property name or value

Status	Trial name <div></div>	Accuracy	Layers	Units	Lr
<div></div>	mnist-mini-hpo-br8wh5rz	0.895	2	55	6.72864e-3
<div></div>	mnist-mini-hpo-lnd4g6dn	0.895	2	121	2.11275e-3

Items per page: 10

1 - 2 of 2

Trials



Project Context

Tools Overview

Pipeline

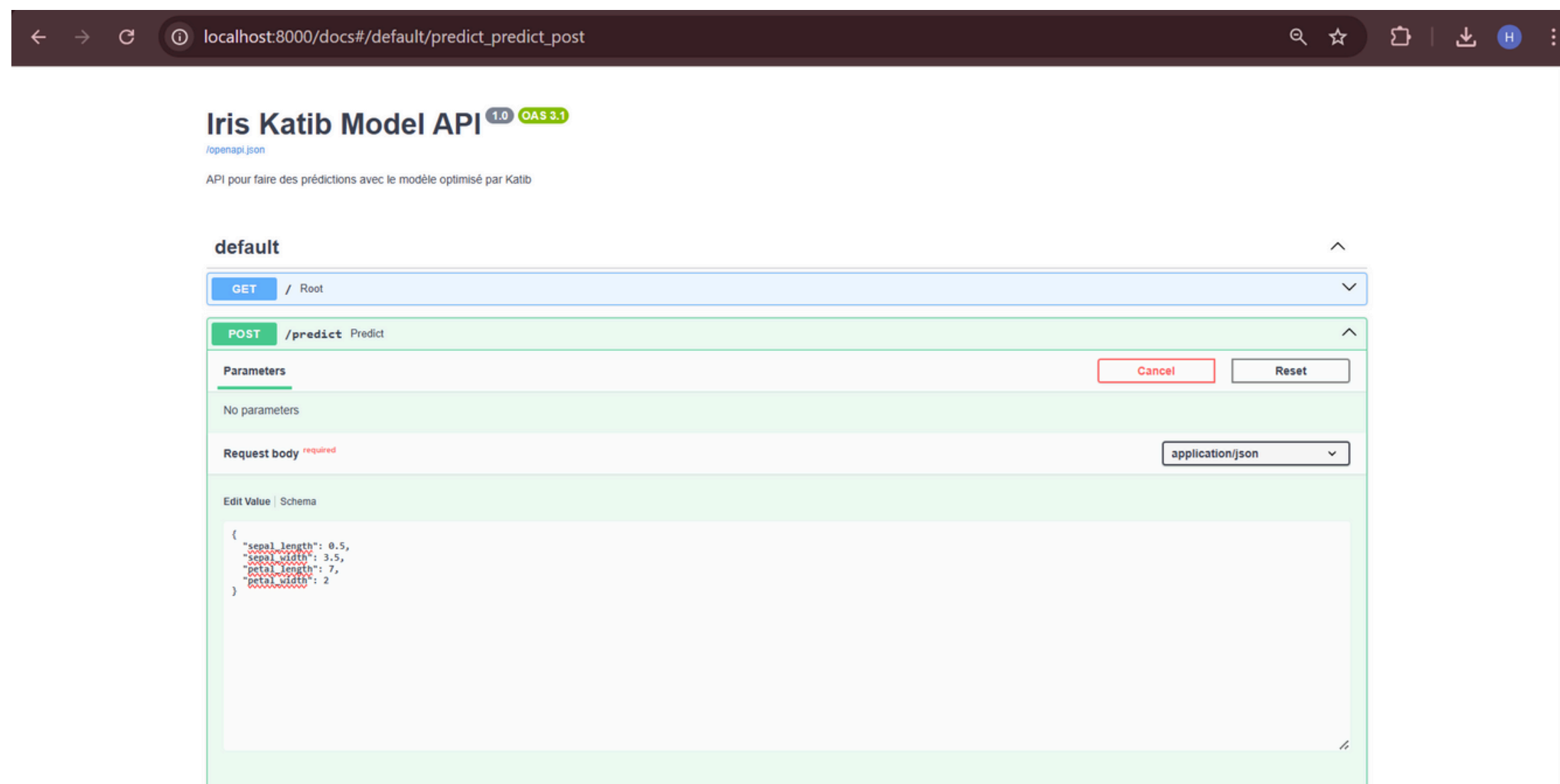
Implementation

Use Case Implementation

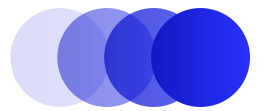
OVERVIEW	TRIALS	DETAILS	YAML
Objective			
Name	accuracy		
Type	maximize		
Goal	0.85		
Additional metrics	No additional metrics		
Trials			
Max failed trials			
Max trials	4		
Parallel trials	2		
Parameters			
layers	<div>Parameter type: int</div> <div>Min: 1</div> <div>Max: 3</div>		
units	<div>Parameter type: int</div> <div>Min: 32</div> <div>Max: 128</div>		
lr	<div>Parameter type: double</div> <div>Min: 0.0005</div> <div>Max: 0.01</div>		
Algorithm			
Name	random		
Metrics collector			
Collector type	StdOut		

Experiment Details

Inference



Fast API service



Project Context

Tools Overview

Pipeline

Implementation

Inference

```
hp@DESKTOP-1FTUR50:~/katib-iris-v3/mnist$ microk8s kubectl get jobs -n kubeflow
NAME                                STATUS    COMPLETIONS   DURATION   AGE
mnist-batch-inference              Running   0/1            24s        24s
hp@DESKTOP-1FTUR50:~/katib-iris-v3/mnist$ microk8s kubectl logs -n kubeflow job/mnist-batch-inference
2025-12-20 19:33:11.323260: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
2025-12-20 19:33:13.061489: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2025-12-20 19:33:20.395646: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:31] Could not find cuda drivers on your machine, GPU will not be used.
Model loaded
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ————— 8s 1us/step
Sample 0: predicted class = 6
Sample 1: predicted class = 5
Sample 2: predicted class = 6
Sample 3: predicted class = 5
Sample 4: predicted class = 7
Sample 5: predicted class = 2
Sample 6: predicted class = 6
Sample 7: predicted class = 4
Sample 8: predicted class = 3
Sample 9: predicted class = 2
```

Batch job Inference Serving

Conclusion

KServe Deployment on MicroK8s — Observations & Constraints

What worked and what didn't:

- All necessary components (Istio, Knative, Ingress, KServe, Storage) deployed
- KServe InferenceService link is displayed
- Pod fails to become Ready

Cause:

Resource constraints on single-node MicroK8s cluster:

- CPU & memory contention between Istio, Knative, and KServe.
- Pods cannot fully initialize → container crash or delay.

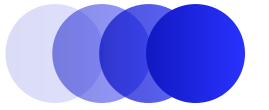


Conclusion



This mini project demonstrates an end-to-end MLOps workflow, from model training to deployment.

- Docker ensures reproducibility
- Kubernetes & Katib automate and scale experiments with tracking
- PVC enables persistent storage for models
- FastAPI exposes the trained model as a production-ready API



Conclusion

Thank you for your attention

