MINDSONAI

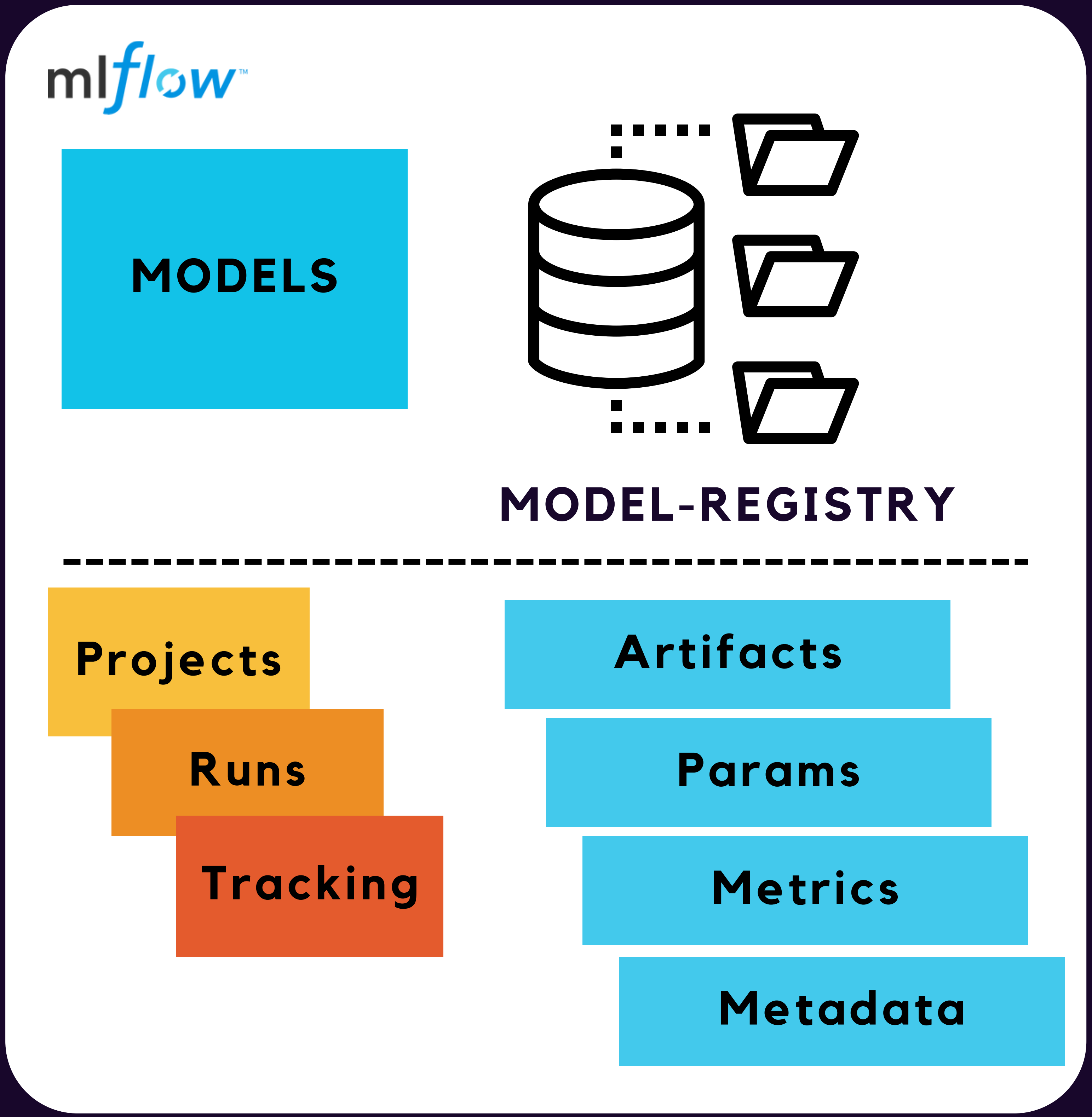
ML OPERATIONS

ML-FLOW

https://linkedin.com/rishabhio

JIO ML OPS 24

Intro to ML-Flow



Intro to ML-Flow

Artifacts — mlflow.log_artifact()

- Model Files
- Plot Files

Params — mlflow.log_param()

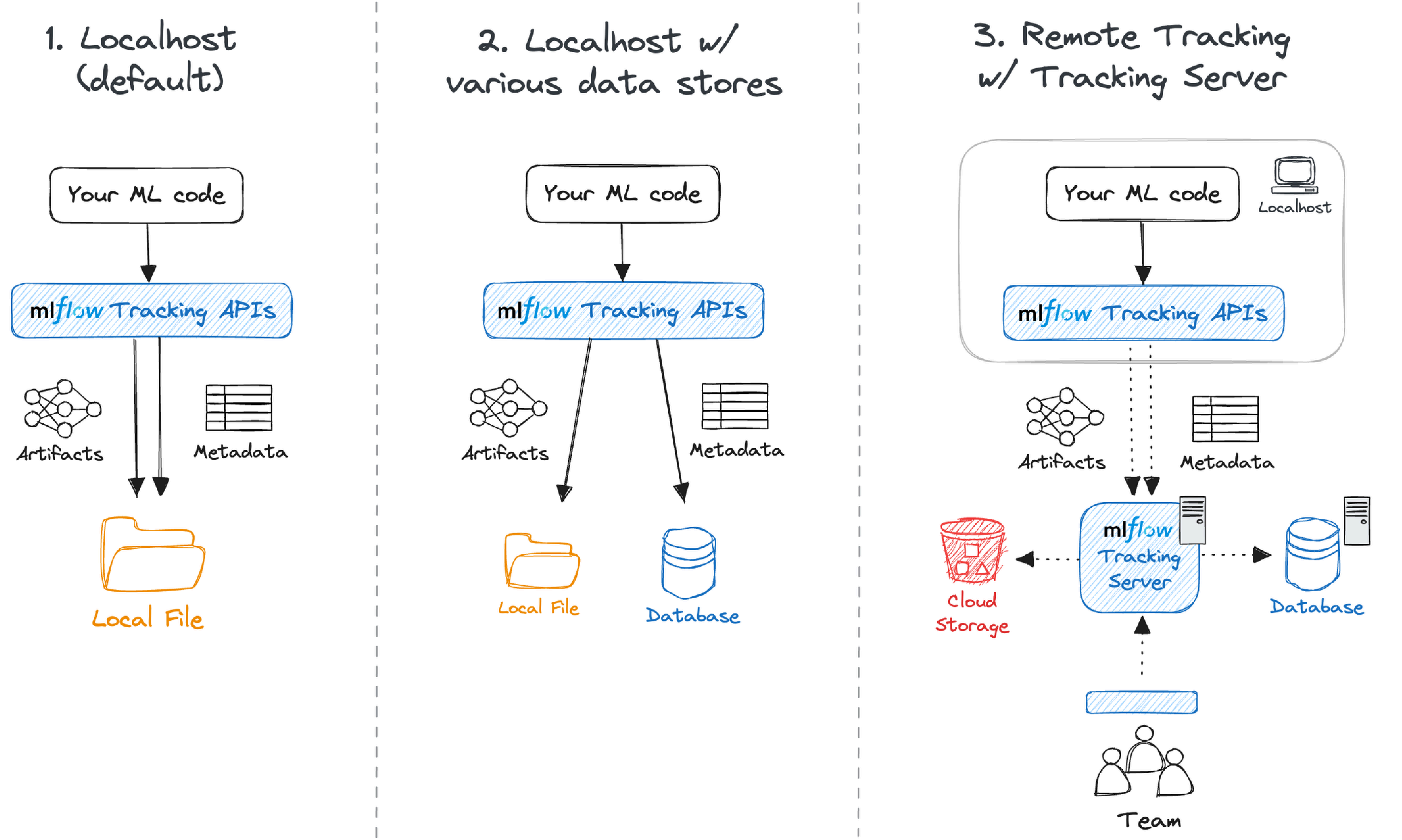
- Learning Rate
- Batch Size

Metadata — mlflow.set_tag()

- Model Version
- Experiment Name

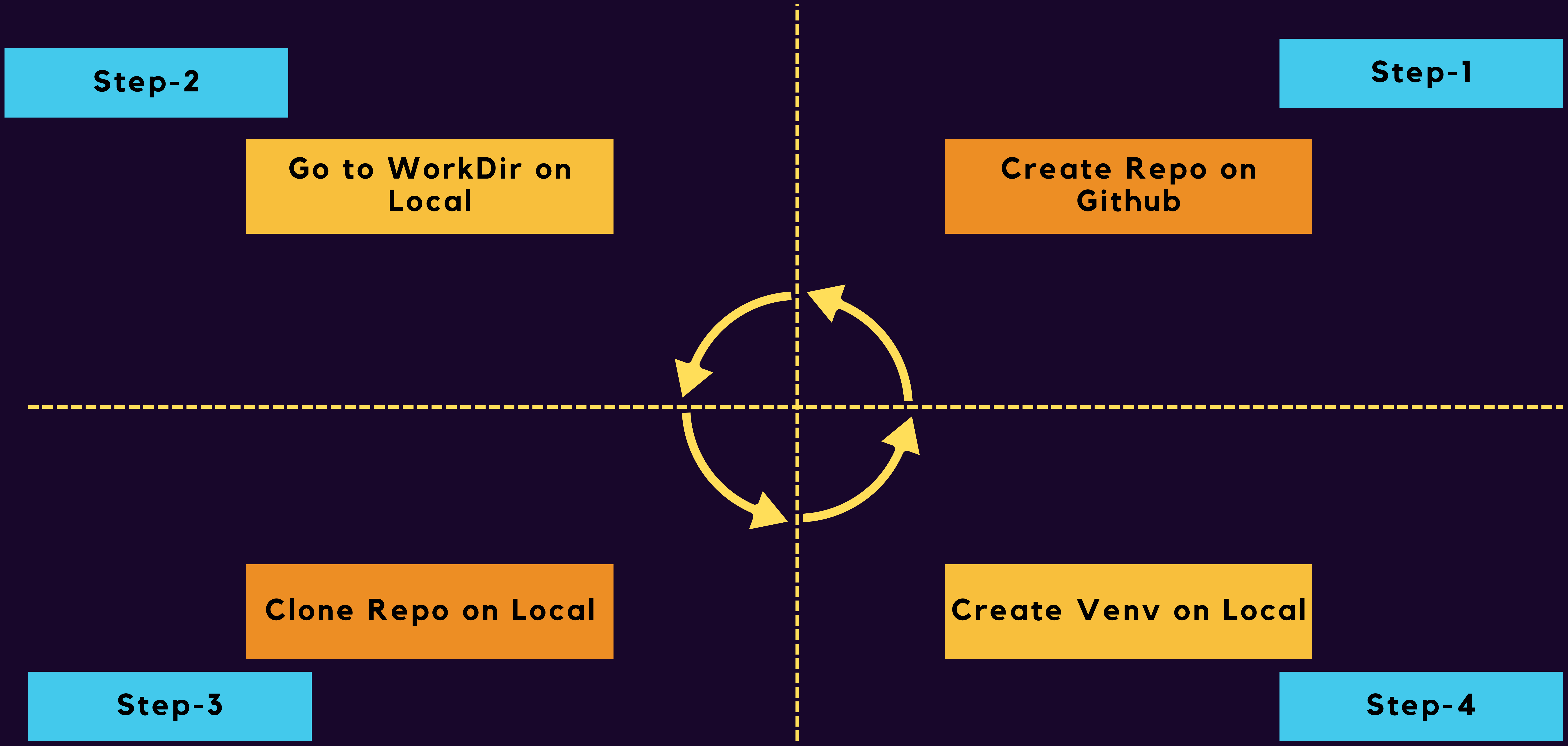
Metrics — mlflow.set_metric()

- Accuracy
- Loss


1. Localhost (default)
Your ML code
mlflow Tracking APIs
Artifacts
Metadata
Local File
2. Localhost w/ various data stores
Your ML code
mlflow Tracking APIs
Artifacts
Metadata
Local File
Database
3. Remote Tracking w/ Tracking Server
Your ML code
Localhost
mlflow Tracking APIs
Artifacts
Metadata
Cloud Storage
mlflow Tracking Server
Database
Team

Step by Step Guide to setting up tracking, logging & model registry with ml-flow

Start Your Project



Step: Env Setup

Inside WorkDir

```
pip install mlflow
pip install jupyterlab
pip install scikit-learn pandas numpy
pip freeze>requirements.txt
```

Step: Use custom Venv in Jupyter

Install Packages

```
pip install ipython
```

```
pip install ipykernel
```

Add Your env to Jupyter

```
ipython kernel install --user --name=myenv
```

```
python -m ipykernel install --user --name=myenv
```

Step: Start ML Flow

Inside WorkDir

```
mlflow ui --<port_number>
```

Inside WorkDir

```
mlflow ui --port 8090
```

Step: ML Flow UI

On your Browser

http://localhost:8090

Explore the UI

We'll get back to this step later

Step: Enable ML Flow

Inside Your Experiment

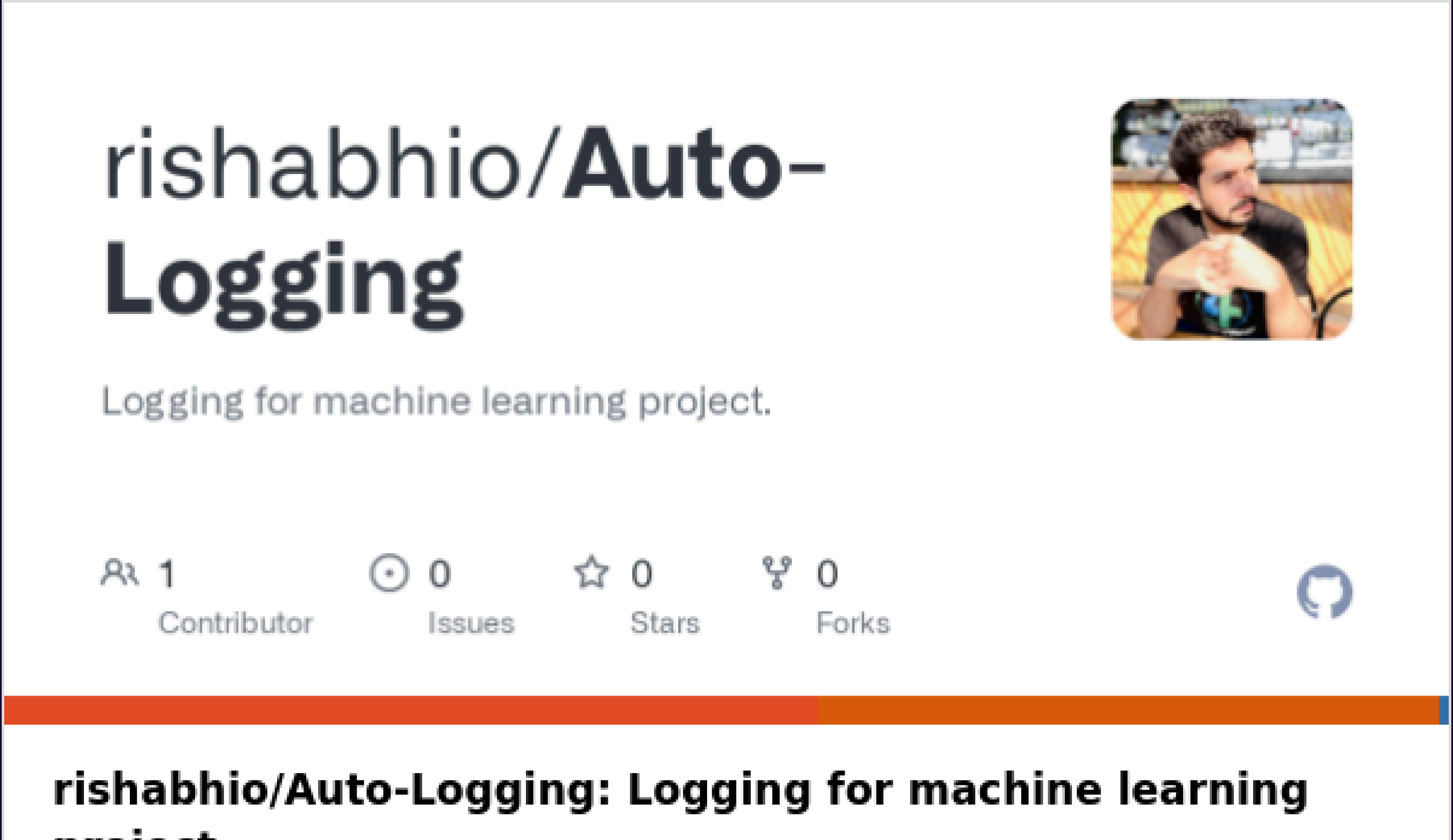
```
import mlflow
```

Inside Your Experiment

```
mlflow.set_tracking_uri("<ml_flow_uri>")
```

http://localhost:8090/

Step: Clone Repo for Reference



Step: Quick & Dirty

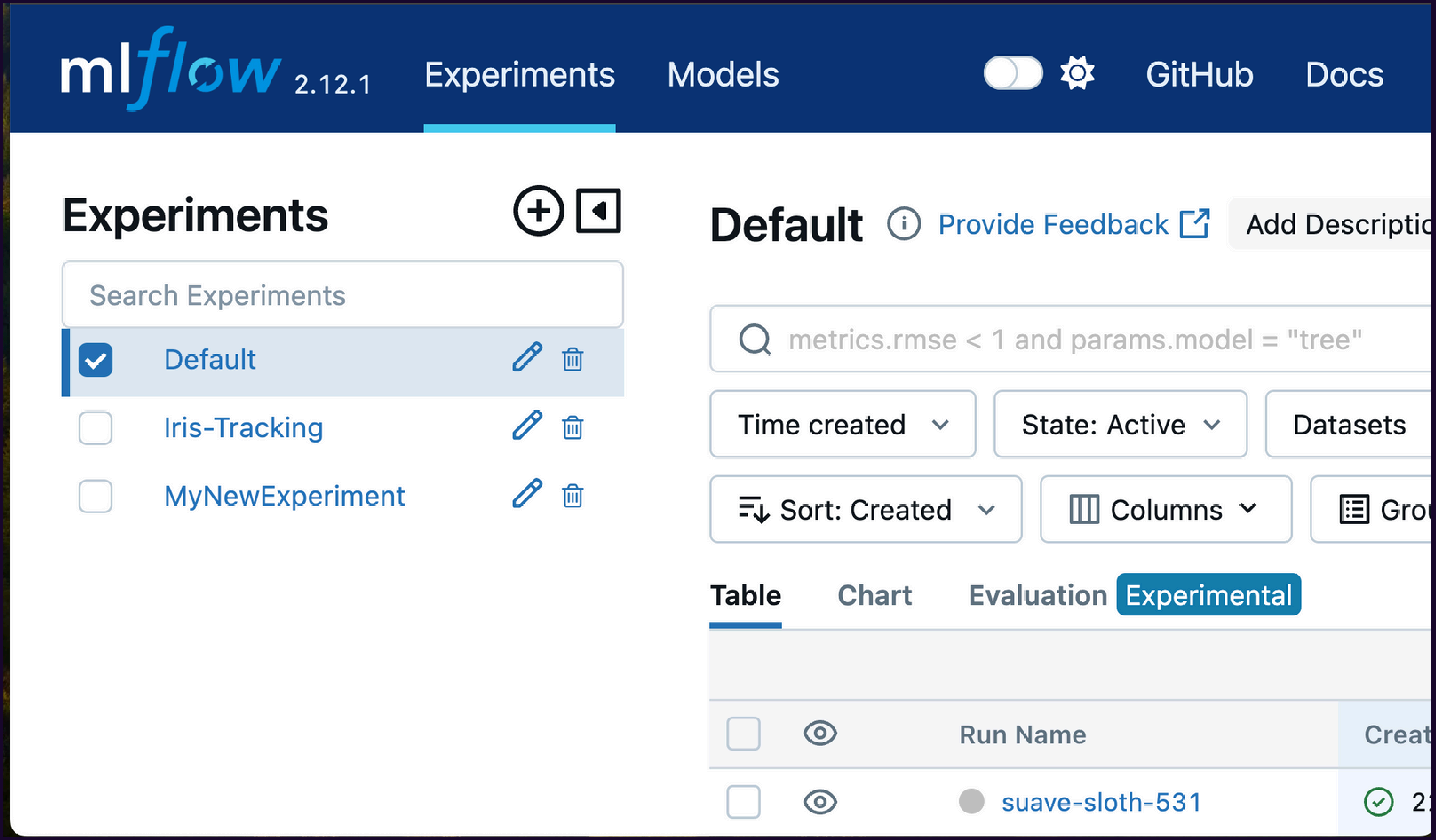
mlflow.autolog()

Most of the common ds/ml packages support the auto-logging feature.

Auto means you don't have to define what to log (some defaults are already set)

Step: Explore ML Flow UI

http://localhost:8090



Step: What all can we Track

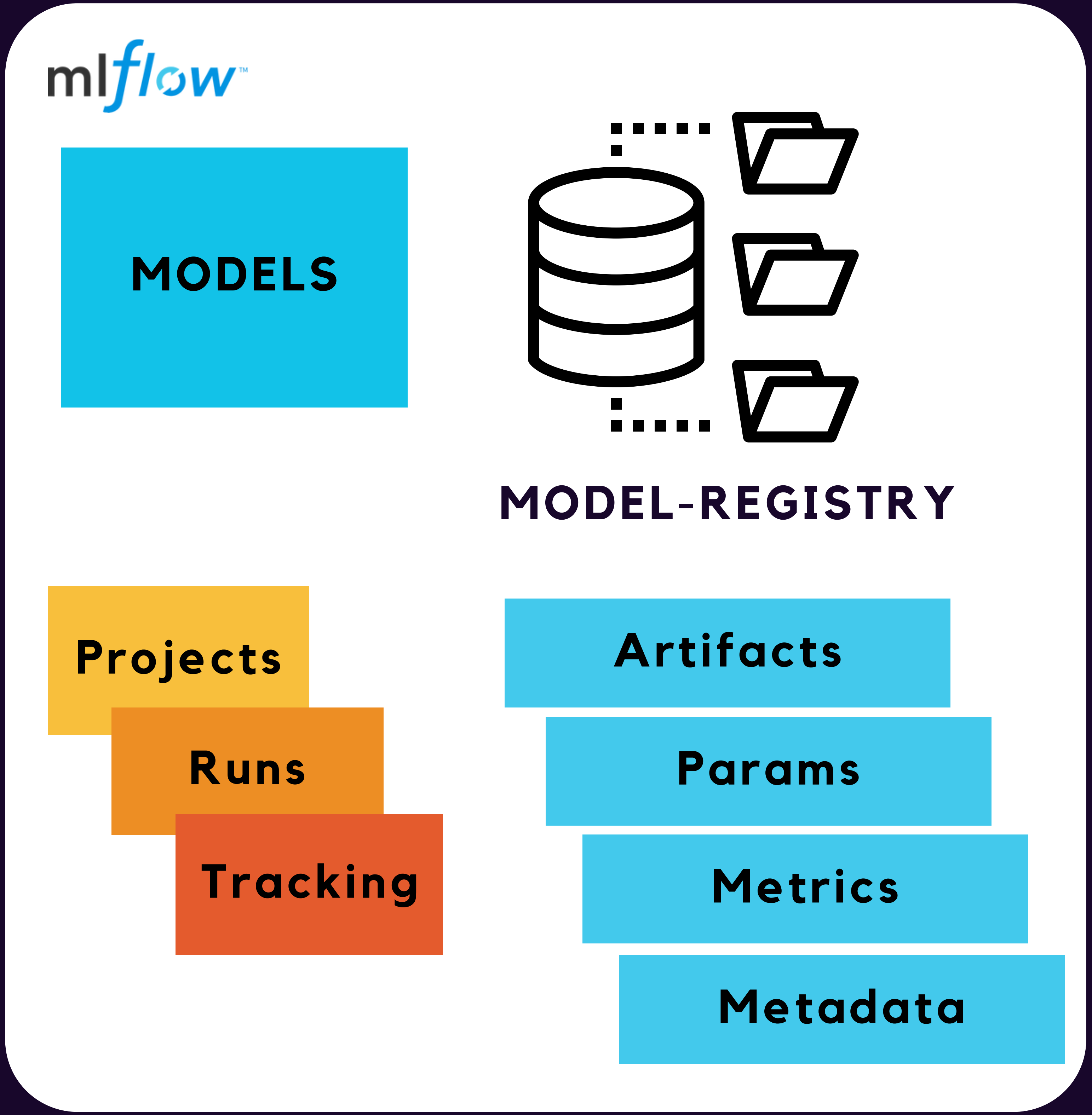
Models can be Tracked

Artifacts (models) can be registered

Params can be tracked by run

Metrics can be tracked

Metadata can be stored



Step: Using the Artifact

```
import mlflow

model = mlflow.pyfunc.load_model("runs:/<run_id>/model")
model.serve(port=5000, enable_mlserver=True)
```

Step: Running Batch Inference

```
import mlflow

model = mlflow.pyfunc.load_model("runs:/<run_id>/model")
predictions = model.predict(pd.read_csv("input.csv"))
predictions.to_csv("output.csv")
```

Step: Support for Bash

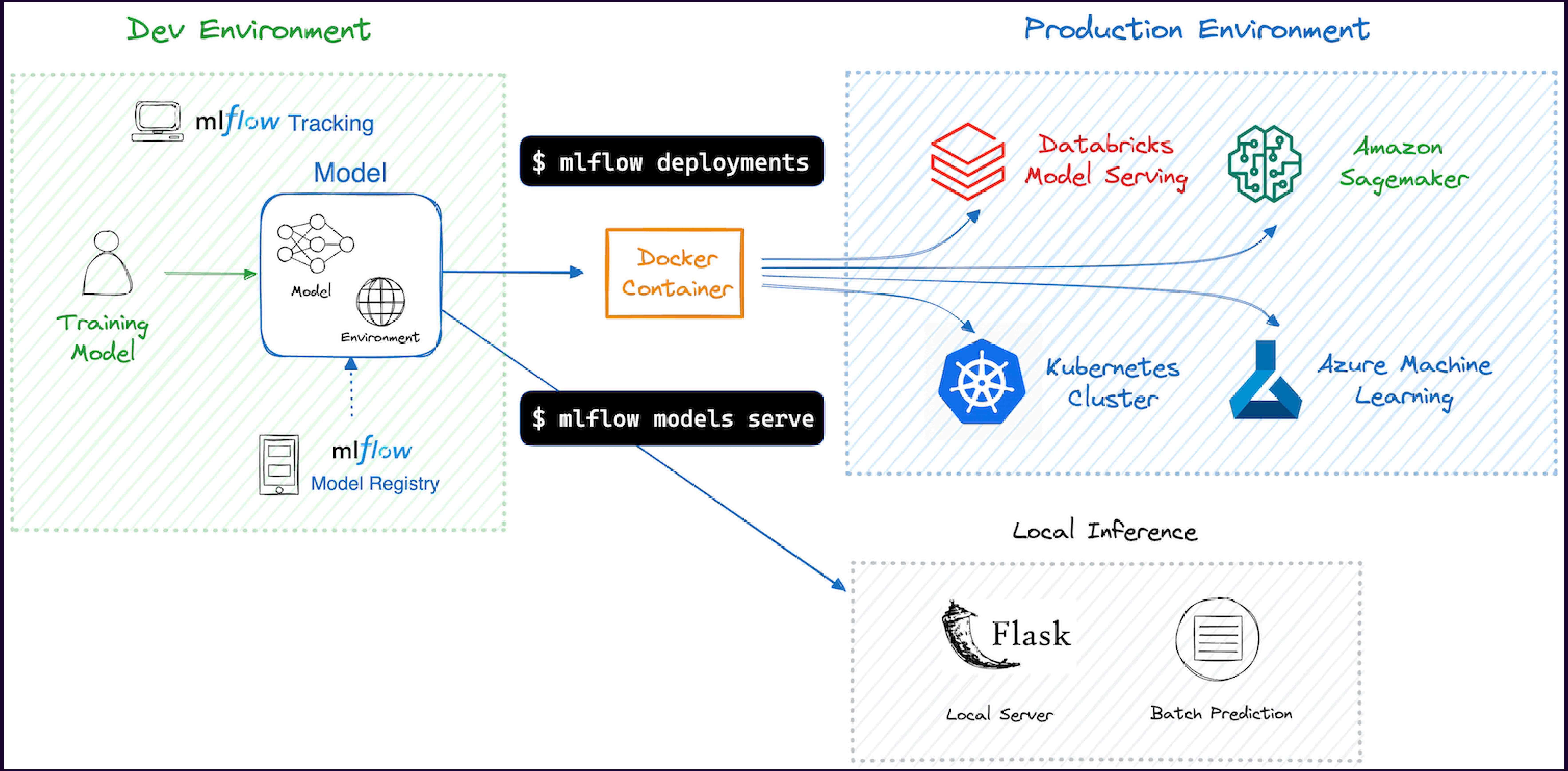
```
mlflow models predict -m runs:/<run_id>/model -i input.csv -o
output.csv
```

Available in the UI

Step: Building a Docker Image for MLflow Model

```
mlflow models build-docker -m runs:/<run_id>/model -n <image_name> --enable-mlserver
```

Available in the UI


Dev Environment
Production Environment
mlflow Tracking
Model
$ mlflow deployments
Databricks Model Serving
Amazon Sagemaker
Model
Environment
Training Model
Docker Container
Kubernetes Cluster
Azure Machine Learning
$ mlflow models serve
mlflow
Model Registry
Local Inference
Flask
Local Server
Batch Prediction

Step: Summary

mlflow™ can be used to:

1. Track Experiments via (metrics, params, meta, artifacts)
2. Go back in time to view logs of any run
3. Register Models (Model Registry)
4. Serve models right from the platform (via cmdline or Python code)
5. Deploy models (build docker containers and deploy them)

Step: Next Steps

mlflow™ is Customizable

1. ML Flow is written in the familiar Python programming language and React based UI
2. ML Flow can be customized
3. Custom frontend application can be written by utilising the Flask based Restful API
4. APIs can be called directly from other tools / python code
5. ML Flow can be integrated into current system (using Python)
6. Most tools compatible with Python can be made to work equally well with ML Flow as well
7. ML Flow is open source, which gives great opportunity to contribute code to the core platform or plugins

Step: Task -> ML Flow Setup

1. Integrate ML Flow in any of your existing project.
2. Install ML Flow locally
3. Setup your jupyter lab to log to ML Flow via adding the uri
4. Track the experiment via logging meta, metrics, params and artifacts
5. Visualize the same on the platform
6. Serve the model (using local deployment option for now)
7. Write about your experience in the form of a medium article (1 article per team)
8. Share the link of the article as a deliverable
9. Feel free to explore the docs to add more functionality then we have covered
10. You get score based on:
    a. Uniqueness, Content Coverage, ScreenShots of Work, Reproducibility of your Code, Link shared for Github Repo, Language of the Article (should be easy to read)

Switching Gears towards final Project.

Step: Note on Final Project

1. Your final project score will be based on the following:
    a. How many ML-OPs practices are you able to showcase via your work: e.g.
        i. Version Control (Data / Code)
        ii. Experiment Tracking (Practice implementation)
        iii. Experiment Reproducibility
        iv. Capturing the relevant metadata, metrics, params, artifacts
        v. Automation in (Data collections, Feature Engineering, Model Training, Building Containers etc)
        vi. CI-CD (for models / application code etc)
        vii. Maintaining models in Model Registry
        viii. Retraining (Auto / Manual ?)
        ix. What new tools you research (not covered in the class) to implement the practices
        x. How you share your models (via app links or APIs ?)
        xi. Do you implement any governance on your data / models / app ? (How well you control your assets)

Step: Observe how others are doing. . .

MLOps at GreenSteam: Shipping Machine Learning [Case Study]

GreenSteam's MLOps journey: shared codebases, reproducibility, model versioning, serving, auditing, and lessons

Step: What / How to Observe ?

1. What practices are being implemented and what tools are in use?
2. Try to assess the level of ML-OPs maturity
3. Try to critique the approaches or even better think of better approaches based on your understanding
4. Find out new keywords (you never heard before)
5. Try to replicate the work (where possible)
6. Share learnings via Medium / Linkedin etc

Step: Observe People in this Space

Just an example, do your research

MLOps guide

I help companies deploy machine learning into production. I write about AI applications, tooling, and best practices.

Chip Huyen

Noah Gift

Machine Learning, Data Science, Cloud & AI Lecturer

Noah Gift / Oct 15, 2017

Step: Observe Tools in this Space

MLOps Tools: Types, Key Features, and Top 6 Solutions

MLOps seeks to automate the entire lifecycle of developing, deploying, and monitoring models by combining machine learning, DevOps, and data engineering.

run.ai

MLOps Landscape in 2024: Top Tools and Platforms

Check the key players in the MLOps and FMOps (or LLMOps) ecosystems of 2024, encompassing both open-source and closed-source tools.

neptune.ai / Jun 27, 2023

databricks

What is MLOps?

MLOps is a core function of Machine Learning engineering, focused on streamlining the process of taking ML models to production, and then maintaining and monitoring them.

Databricks

Step: Follow Relevant Github Repos

visenger/**awesome-mlops**

A curated list of references for MLOps

61 Contributors · 19 Used by · 12k Stars · 2k Forks

visenger/awesome-mlops: A curated list of references for MLOps

A curated list of references for MLOps . Contribute to visenger/awesome-