PyCon Greece
29-30 August 2025
Technopolis City of Athens

# Hot-Swapping MLflow models

A pythonic approach on AWS lambda

## Stavros Niafas

ML Engineer | MLflow Ambassador

https://github.com/sniafas/PyconGR

# Table of contents

PyCon Greece 29-30 August 2025

## 01 Keeping models fresh w/out downtime

# Keeping model fresh w/out downtime

## Updating Models in production 🔄

- Model Drift

- Changes in data / Missing data trends

- Updating a model can cause downtime

## Dynamic Updates 🎯

- Zero Downtime: Important for high availability services

- Speeding up the ML lifecycle

- Avoids the need for a full re-deployment of application.

## Hot-Swapping 🔥

- Event driven architecture

- Replace the current model with the updated, while the service is still running

- Pythonic way to achieve that

## 02 ML Inference w/ Serverless

# ML Inference w/ Serverless

## Serverless over simplicity 🙌

– Confusing term, servers are there, you just not manage them.

- Zero infra management, self-managed provisioning & auto-scaling

- Ideal for simple inference environments

- Straightforward to implement. Just like a regular Python script

## Reduced time-to-market 📉

- No overhead for API specifics

- Seamless integration, easy to consume

- Unlimited space for automations and parametrization

- Faster experimentation: Commit, Rebuild, Push

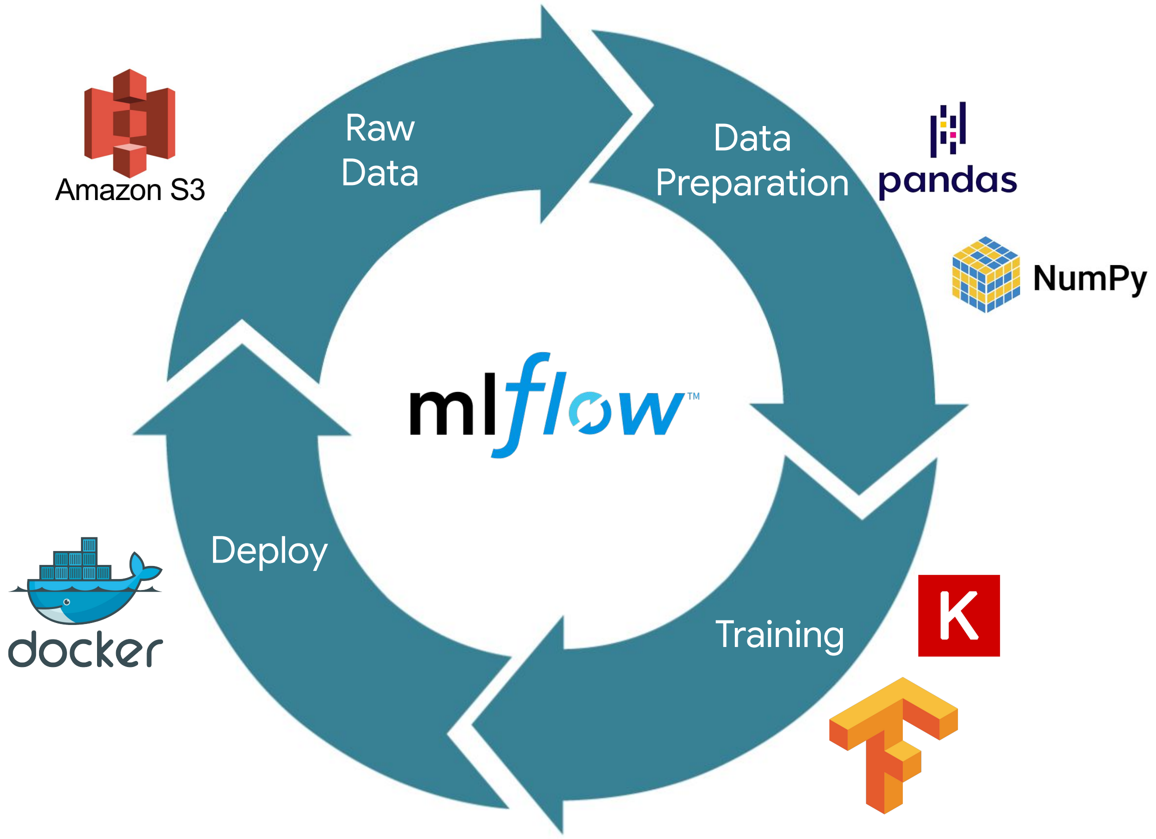
## What to look at ⚠️

- Docker packaging & dependencies

- Cold starts 🥶

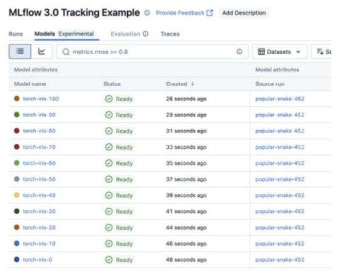
## 03 ML Lifecycle & MLflow

# ML Lifecycle



# MLflow: Managing the ML lifecycle

1. Implement ML lifecycle



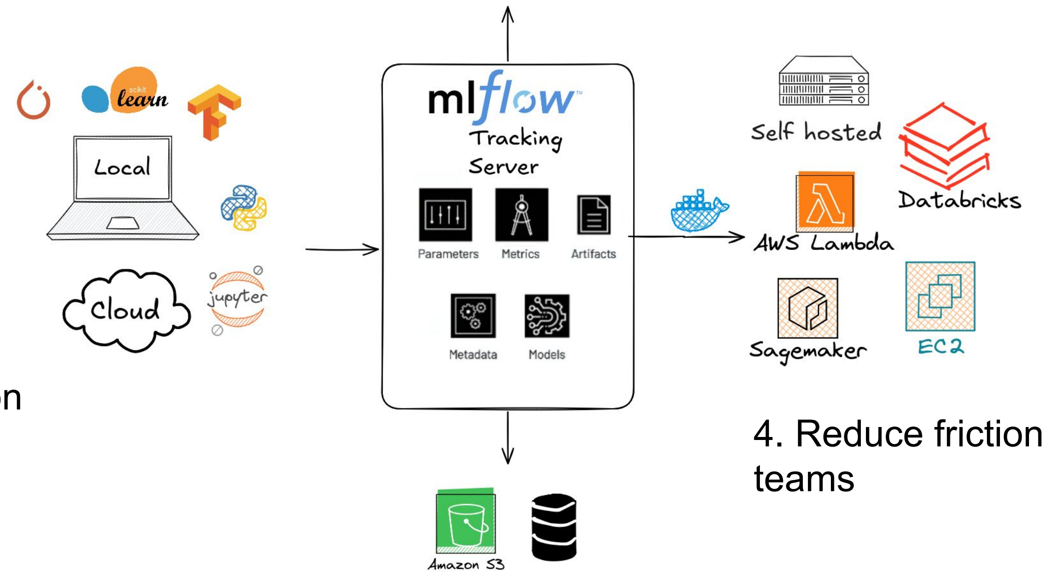
MLflow 3.0 Tracking Example  Provide Feedback  Add Description

Runs  Models Experimental  Evaluation  Traces

metrics.rmse >= 0.8  Datasets

| Model attributes | | | Model attributes |
|---|---|---|---|
| Model name | Status | Created ↓ | Source run |
| torch-iris-100 | Ready | 26 seconds ago | popular-snake-452 |
| torch-iris-90 | Ready | 29 seconds ago | popular-snake-452 |
| torch-iris-80 | Ready | 31 seconds ago | popular-snake-452 |
| torch-iris-70 | Ready | 33 seconds ago | popular-snake-452 |
| torch-iris-60 | Ready | 35 seconds ago | popular-snake-452 |
| torch-iris-50 | Ready | 37 seconds ago | popular-snake-452 |
| torch-iris-40 | Ready | 39 seconds ago | popular-snake-452 |
| torch-iris-30 | Ready | 41 seconds ago | popular-snake-452 |
| torch-iris-20 | Ready | 44 seconds ago | popular-snake-452 |
| torch-iris-10 | Ready | 46 seconds ago | popular-snake-452 |
| torch-iris-0 | Ready | 48 seconds ago | popular-snake-452 |

3. Governance to track experiments and models efficiently



2. Fluent API in Python

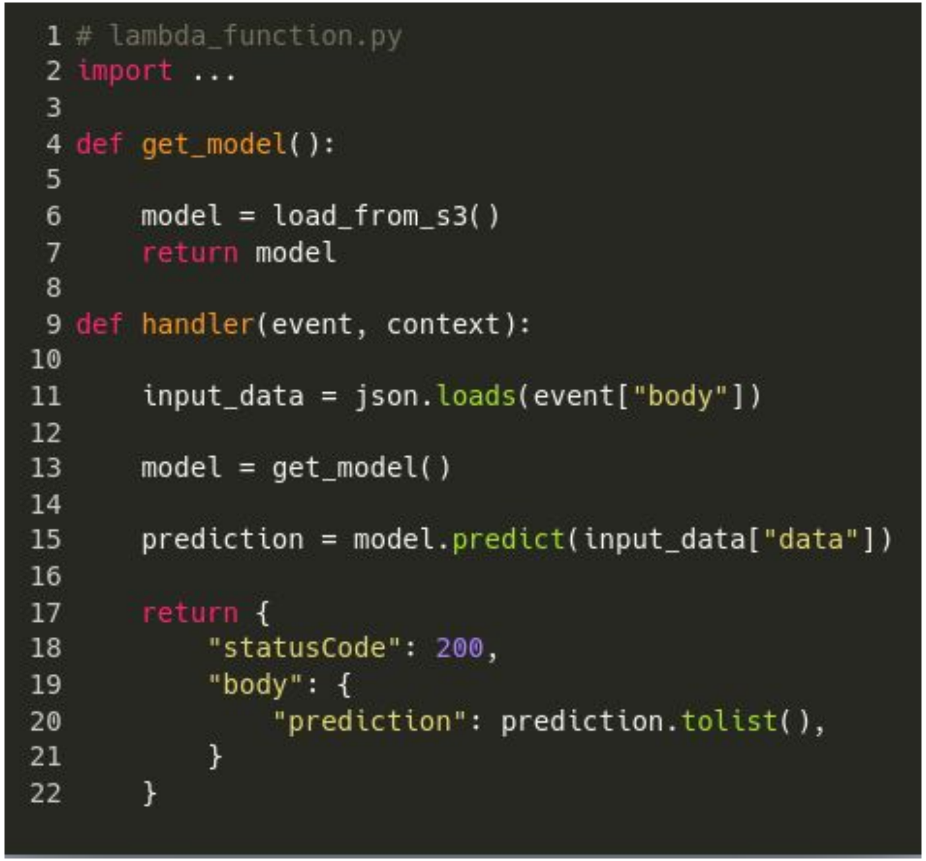
4. Reduce friction between the teams

## 04 AWS λ

# AWS λ

Cold start initialization

Invocation



```python
# lambda_function.py
import ...

def get_model():

    model = load_from_s3()
    return model

def handler(event, context):

    input_data = json.loads(event["body"])

    model = get_model()

    prediction = model.predict(input_data["data"])

    return {
        "statusCode": 200,
        "body": {
            "prediction": prediction.tolist(),
        }
    }
```

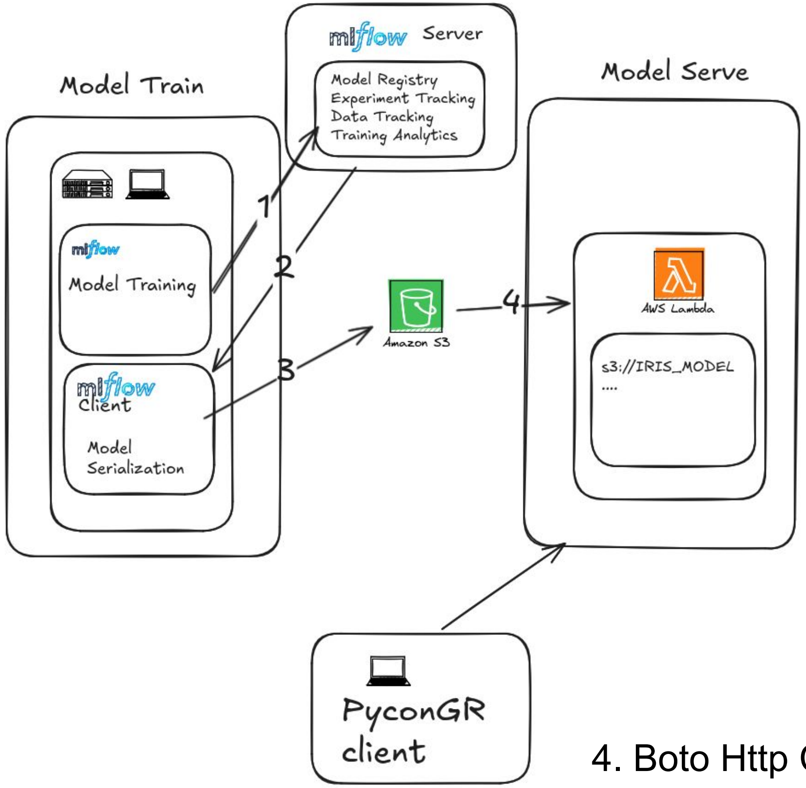
Execution Environment

## 05 Hot-Swapping!

# Hot-Swapping

1. Model Traning (MLflow)

2. Model Serialization (Boto)



3. Model Server (PyLambda + Boto)

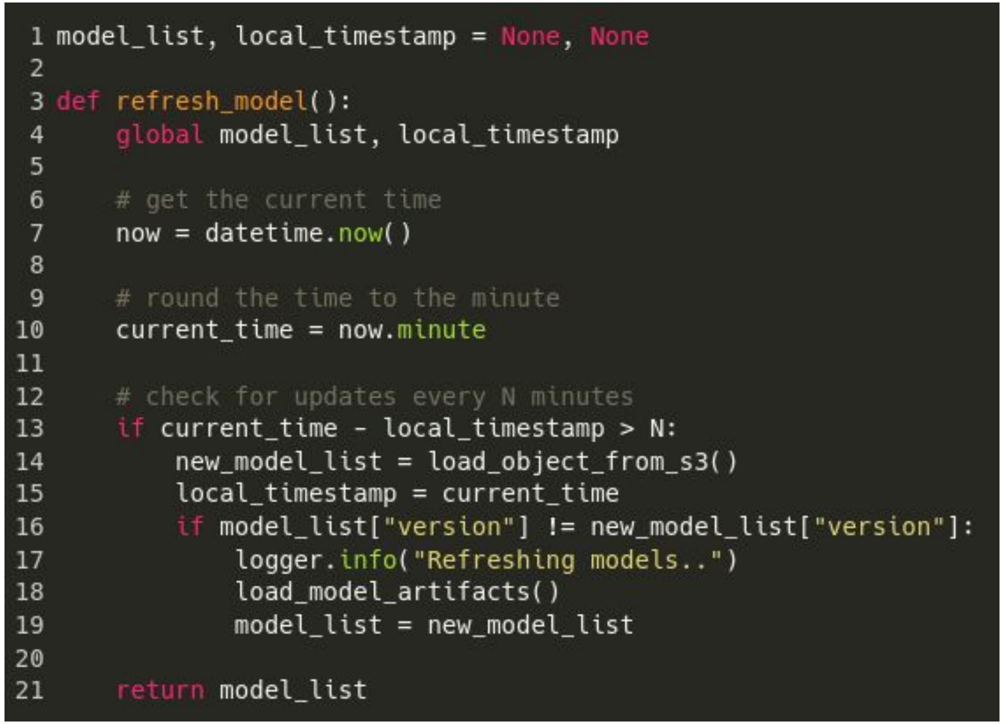
4. Boto Http Client

# Hot-Swapping

# What's behind zero downtime

Globals

Time check



```python
model_list, local_timestamp = None, None

def refresh_model():
    global model_list, local_timestamp

    # get the current time
    now = datetime.now()

    # round the time to the minute
    current_time = now.minute

    # check for updates every N minutes
    if current_time - local_timestamp > N:
        new_model_list = load_object_from_s3()
        local_timestamp = current_time
        if model_list["version"] != new_model_list["version"]:
            logger.info("Refreshing models..")
            load_model_artifacts()
            model_list = new_model_list

    return model_list
```

## 05 Live Demo 🤞 🚀

# Thank you