Experiment Tracking

MLOps Zoomcamp

Important concepts

- ML experiment: the process of building an ML model
- Experiment run: each trial in an ML experiment
- Run artifact: any file that is associated with an ML run
- Experiment metadata: information about an ML experiment like the source code used, the name of the user, etc.

What's experiment tracking?

Experiment tracking is the process of keeping track of all the **relevant information** from an **ML experiment**, which includes:

- Source code
- Environment
- Data
- Model
- Hyperparameters
- Metrics
- …

Why is experiment tracking so important?

In general, because of these 3 main reasons:

- Reproducibility
- Organization
- Optimization

Tracking experiments in spreadsheets

Why is not enough?

- Error prone
- No standard format
- Visibility & Collaboration

	Accuracy	TP	FP	FN	TN	TOTAL	FP (%)	FN (%)
Baseline new phash	0.906	8213	721	446	2983	12363	5.83192%	3.6%
Baseline new phash	0.909	8210	693	482	3504	12889	5.37668%	3.7%
a)	0.908	8216	699	443	3005	12363	5.65235%	3.6%
b)	0.905	8137	702	438	2729	12006	5.84708%	3.6%
c)	0.908	8217	694	444	3032	12387	5.60130%	3.6%
d)	0.906	8147	706	431	2756	12040	5.86240%	3.6%
e)	0.906	8201	705	464	3030	12400	5.68548%	3.7%
f)	0.905	8135	700	446	2774	12055	5.80672%	3.7%
g)	0.906	8207	705	461	3036	12407	5.67825%	3.7%
h)	0.905	8126	691	460	2794	12071	5.72308%	3.8%
i)	0.906	8211	702	460	3042	12415	5.65445%	3.7%
j)	0.905	8134	694	456	2796	12080	5.74503%	3.8%
m)	0.908	8226	697	431	2934	12288	5.67220%	3.5%
n)	0.907	8145	686	422	2634	11887	5.77101%	3.6%
k)	0.908	8216	703	443	3072	12434	5.65653%	3.6%
l)	0.907	8142	693	437	2843	12115	5.71881%	3.6%
o)	0.907	6857	569	298	1550	9273	6.13070%	3.2%
p)	0.904	7966	693	411	2397	11467	6.04488%	3.6%
m) from MS	0.909	8223	689	434	2941	12287	5.60620%	3.5%
n) from MS	0.907	8155	687	413	2633	11887	5.78082%	3.5%
q)	0.909	7994	650	422	2682	11748	5.53428%	3.6%
r)	0.911	7445	583	339	1969	10336	5.63887%	3.3%
s)	0.909	6921	527	286	1236	8970	5.87886%	3.2%
r_new)	0.907	8194	707	498	3490	12889	5.48530%	3.9%

MLflow

Definition: *“An open source platform for the machine learning lifecycle”* *

In practice, it’s just a Python package that can be installed with pip, and it contains four main modules:

- Tracking
- Models
- Model Registry
- Projects

* Official MLflow documentation: https://mlflow.org/

Tracking experiments with MLflow

The MLflow Tracking module allows you to organize your experiments into runs, and to keep track of:

- Parameters
- Metrics
- Metadata
- Artifacts
- Models

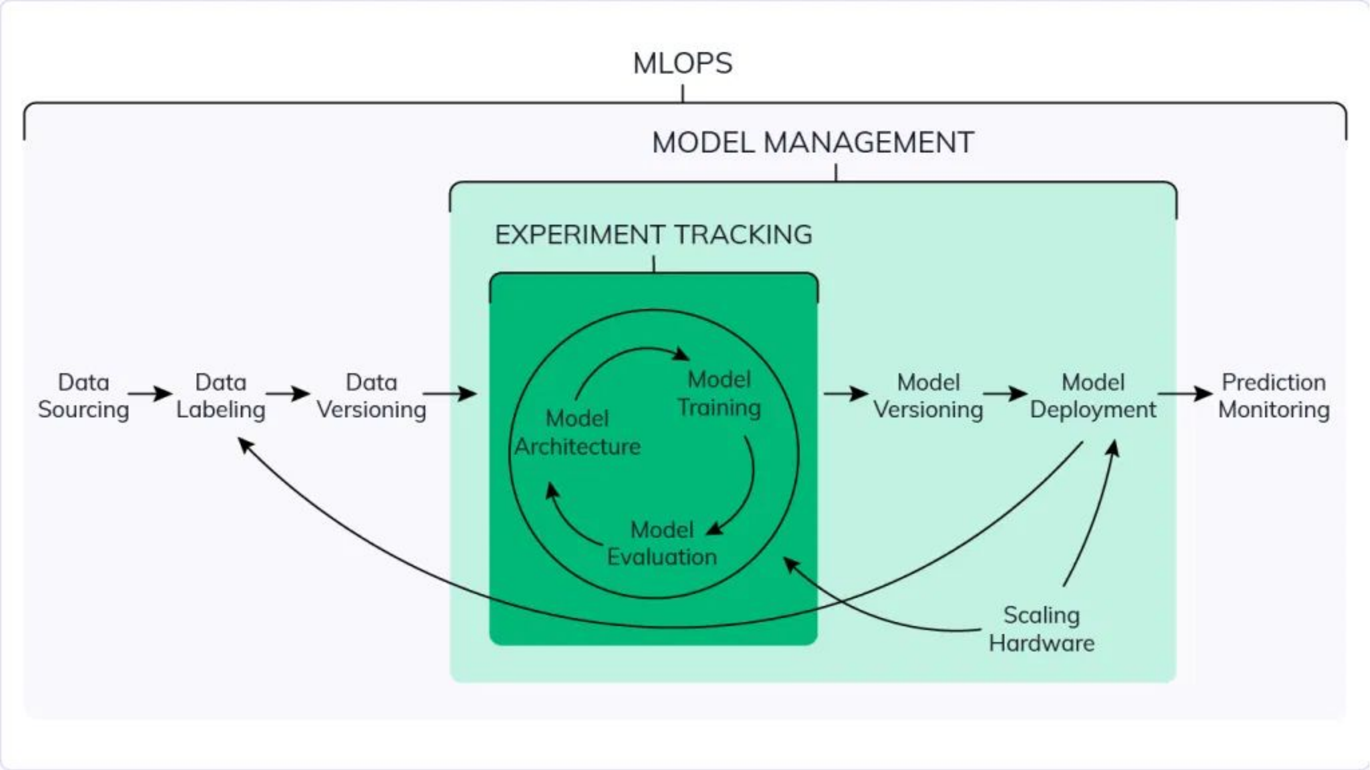
Along with this information, MLflow automatically logs extra information about the run:

- Source code
- Version of the code (git commit)
- Start and end time
- Author

mlflow demo

Model Management

Machine Learning Lifecycle

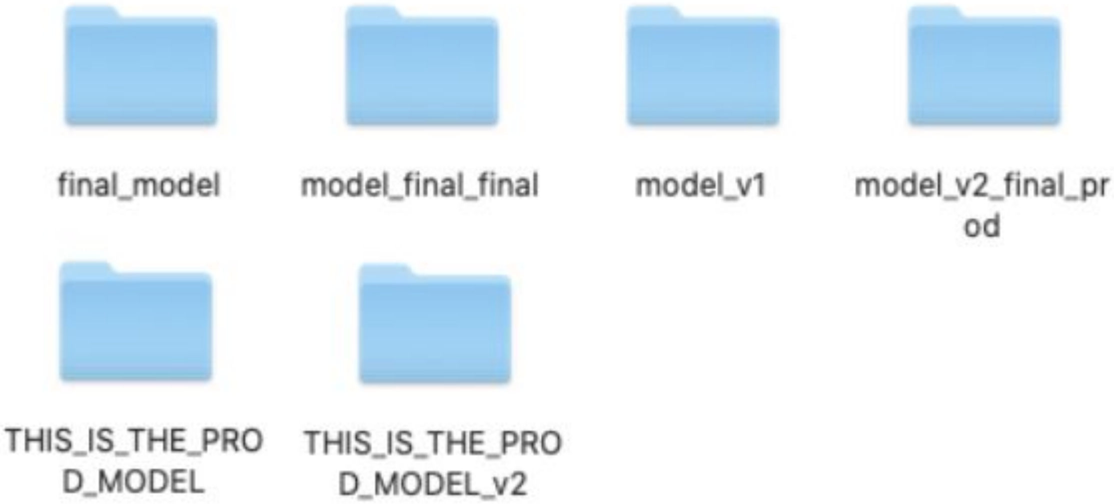


Source: https://neptune.ai/blog/ml-experiment-tracking

Model management

What’s wrong with this?

- Error prone
- No versioning
- No model lineage



Logging models in MLflow

Two options:

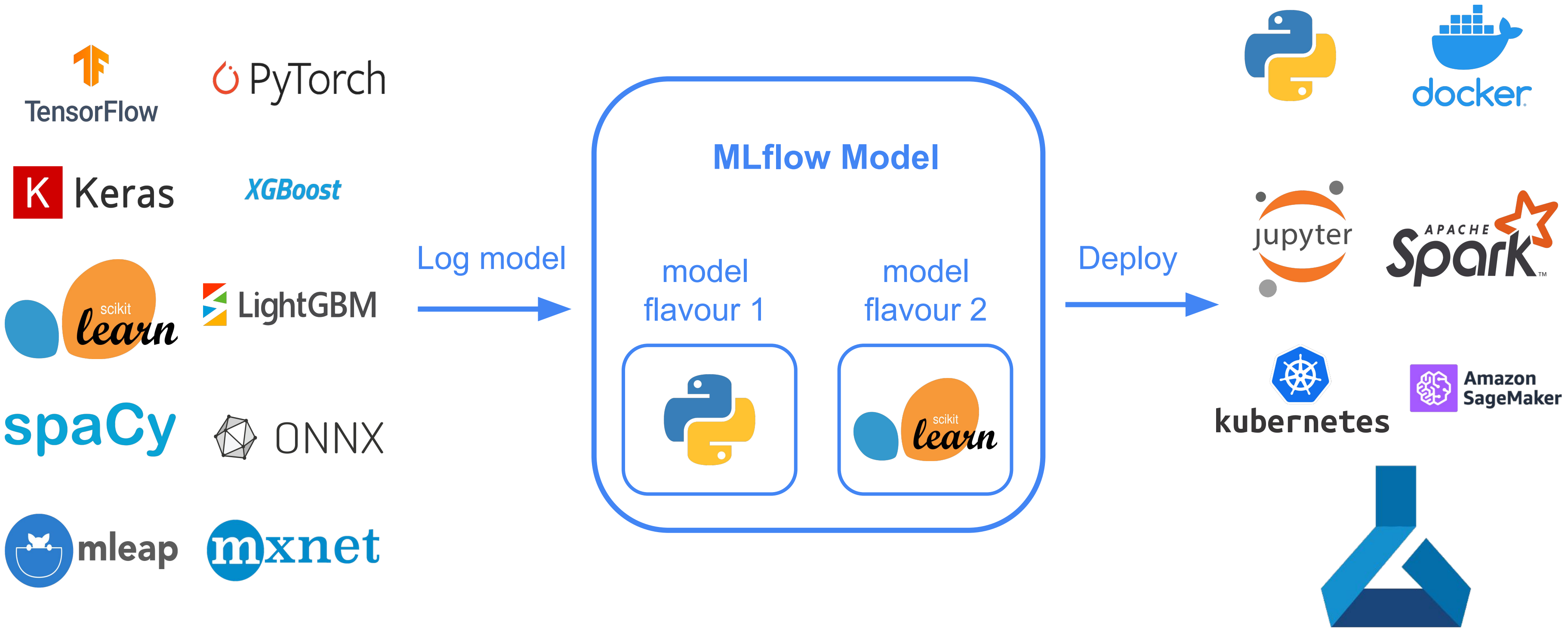
- Log model as an artifact

```
mlflow.log_artifact("mymodel", artifact_path="models/")
```

- Log model using the method `log_model`

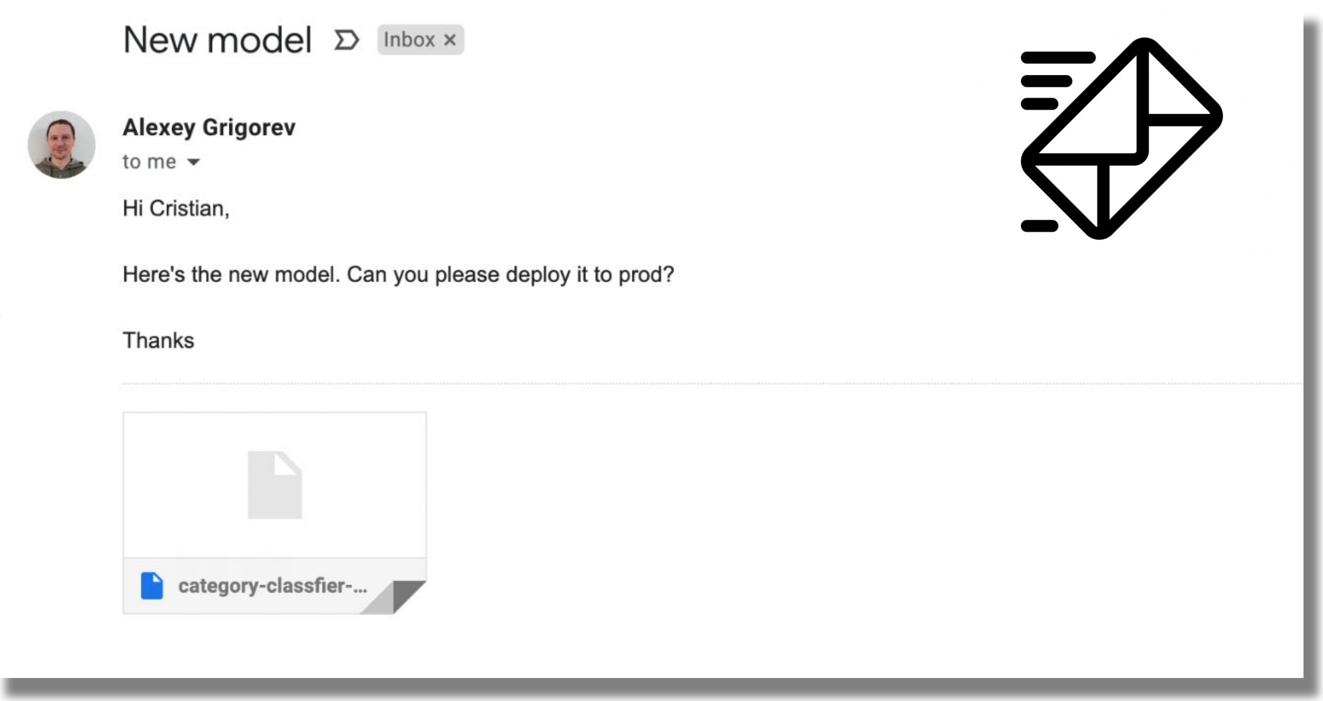
```
mlflow.<framework>.log_model(model, artifact_path="models/")
```

MLflow Model Format

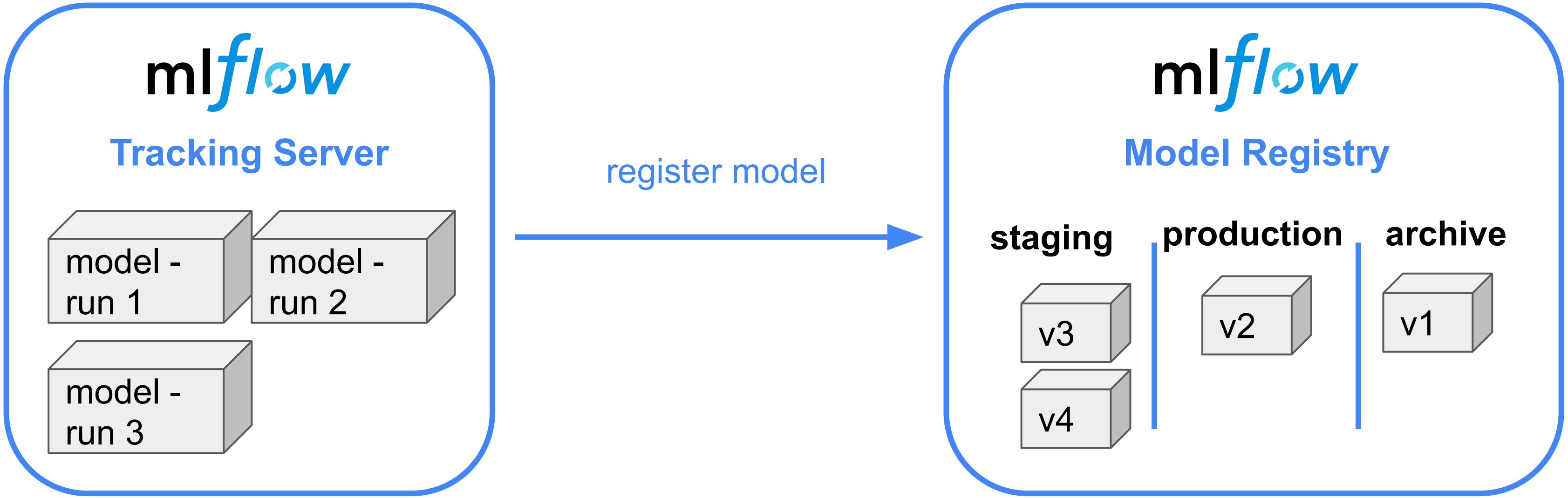


Model Registry

Motivation


New model
Inbox ×
Alexey Grigorev
to me
Hi Cristian,
Here's the new model. Can you please deploy it to prod?
Thanks
category-classfier-...

Model Registry



`MlflowClient` Class

- A client of …
  - an MLflow Tracking Server that creates and manages experiments and runs.
  - an MLflow Registry Server that creates and manages registered models and model versions.
- To instantiate it we need to pass a tracking URI and/or a registry URI:

```
from mlflow.tracking import MlflowClient

client = MlflowClient(tracking_uri="sqlite:///mlflow.db")
```

Model management in MLflow

The Model Registry component is a centralized model store, set of APIs, and a UI, to collaboratively manage the full lifecycle of an MLflow Model.

It provides:

- Model lineage,
- Model versioning,
- Stage transitions, and
- Annotations

MLflow in Practice

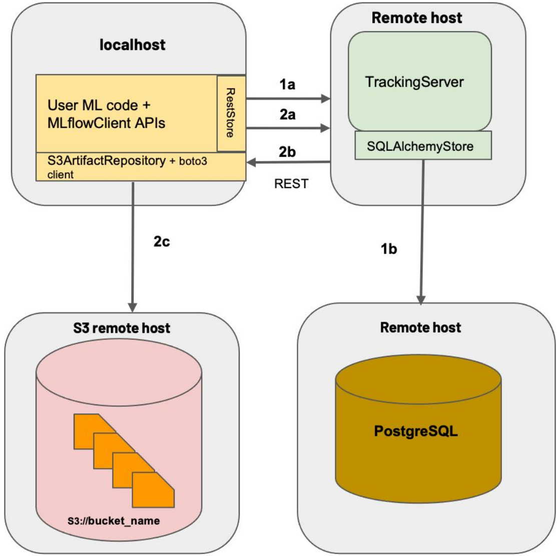
Different scenarios for running MLflow

Let’s consider these three scenarios:

- A single data scientist participating in an ML competition
- A cross-functional team with one data scientist working on an ML model
- Multiple data scientists working on multiple ML models

Configuring MLflow

- Backend store
  - local filesystem
  - SQLAlchemy compatible DB (e.g. SQLite)
- Artifacts store
  - local filesystem
  - remote (e.g. s3 bucket)
- Tracking server
  - no tracking server
  - localhost
  - remote

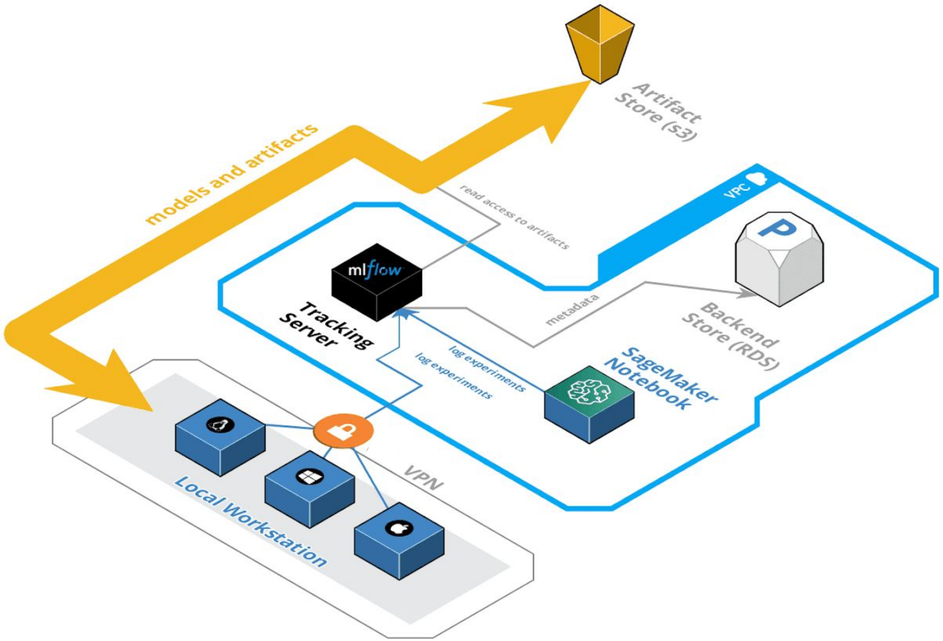


Remote tracking server

The tracking server can be easily deployed to the cloud

Some benefits:

- Share experiments with other data scientists
- Collaborate with others to build and deploy models
- Give more visibility of the data science efforts



Issues with running a remote (shared) MLflow server

- Security
  - Restrict access to the server (e.g. access through VPN)
- Scalability
  - Check Deploy MLflow on AWS Fargate
  - Check MLflow at Company Scale by Jean-Denis Lesage
- Isolation
  - Define standard for naming experiments, models and a set of default tags
  - Restrict access to artifacts (e.g. use s3 buckets living in different AWS accounts)

MLflow limitations (and when not to use it)

- **Authentication & Users**: The open source version of MLflow doesn’t provide any sort of authentication
- **Data versioning**: to ensure full reproducibility we need to version the data used to train the model. MLflow doesn’t provide a built-in solution for that but there are a few ways to deal with this limitation
- **Model/Data Monitoring & Alerting**: this is outside of the scope of MLflow and currently there are more suitable tools for doing this

MLflow alternatives

There are some paid alternatives to MLflow:

- Neptune
- Comet
- Weights & Biases
- and many more