**https://www.youtube.com/watch?v=vqigwhYyJ7M&t=299s**

**https://www.youtube.com/watch?v=OWJHHAtnAwY --use**

**https://docs.databricks.com/applications/mlflow/quick-start-python.html#install-mlflow**

**MLflow** Tracking is an API and UI for logging parameters, code versions, metrics, and artifacts when running your machine learning code and for later visualizing the results. Teams can also use it to compare results from different users. **MLflow** Projects are a standard format for packaging reusable data science code.

MLflow Tracking lets log and query experiments using Python, REST, R API, and Java API APIs.

MLflow is organized into four components:

[Tracking](https://www.mlflow.org/docs/latest/tracking.html" \l "tracking), [Projects](https://www.mlflow.org/docs/latest/projects.html" \l "projects), [Models](https://www.mlflow.org/docs/latest/models.html" \l "models), and [Model Registry](https://www.mlflow.org/docs/latest/model-registry.html" \l "registry)

<https://www.mlflow.org/docs/latest/concepts.html>

<https://mlflow.org/>

https://www.mlflow.org/docs/latest/quickstart.html

MLflow is an open source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow currently offers four components:

**MLflow Tracking**--Record and query experiments: code, data, config, and results

**MLflow Projects**--Package data science code in a format to reproduce runs on any platform

**MLflow Models**--Deploy machine learning models in diverse serving environments

**Model Registry**--Store, annotate, discover, and manage models in a central repository

MLFlow has integration for lot of open source projects like Tensorflow, Pytorch, Keras, Apache Spark, sklearn, h2o.ai, python among others

Machine learning requires experimenting with a wide range of datasets, data preparation steps, and algorithms to build a model that maximizes some target metric.

Once we have built a model, you also need to deploy it to a production system, monitor its performance, and continuously retrain it on new data and compare with alternative models.

**MLflow Components**

MLflow provides four components to help manage the ML workflow:

**MLflow Tracking** is an API and UI for logging parameters, code versions, metrics, and artifacts when running your machine learning code and for later visualizing the results. You can use MLflow Tracking in any environment (for example, a standalone script or a notebook) to log results to local files or to a server, then compare multiple runs. Teams can also use it to compare results from different users.

**MLflow Projects** are a standard format for packaging reusable data science code. Each project is simply a directory with code or a Git repository, and uses a descriptor file or simply convention to specify its dependencies and how to run the code. For example, projects can contain a conda.yaml file for specifying a Python Conda environment. When you use the MLflow Tracking API in a Project, MLflow automatically remembers the project version (for example, Git commit) and any parameters. You can easily run existing MLflow Projects from GitHub or your own Git repository, and chain them into multi-step workflows.

**MLflow Models** offer a convention for packaging machine learning models in multiple flavors, and a variety of tools to help you deploy them. Each Model is saved as a directory containing arbitrary files and a descriptor file that lists several “flavors” the model can be used in. For example, a TensorFlow model can be loaded as a TensorFlow DAG, or as a Python function to apply to input data. MLflow provides tools to deploy many common model types to diverse platforms: for example, any model supporting the “Python function” flavor can be deployed to a Docker-based REST server, to cloud platforms such as Azure ML and AWS SageMaker, and as a user-defined function in Apache Spark for batch and streaming inference. If you output MLflow Models using the Tracking API, MLflow also automatically remembers which Project and run they came from.

**MLflow Registry** offers a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model. It provides model lineage (which MLflow experiment and run produced the model), model versioning, stage transitions (for example from staging to production or archiving), and annotations.

Concepts:

**MLflow Tracking is organized around the concept of runs**, which are executions of some piece of data science code. **Each run records the following information**:

**Code Version**-Git commit hash used for the run, if it was run from an MLflow Project.

**Start & End Time**--Start and end time of the run

**Source**--Name of the file to launch the run, or the project name and entry point for the run if run from an MLflow Project.

**Parameters**--Key-value input parameters of your choice. Both keys and values are strings.

**Metrics**--Key-value metrics, where the value is numeric. Each metric can be updated throughout the course of the run (for example, to track how your model’s loss function is converging), and MLflow records and lets you visualize the metric’s full history.

**Artifacts**--Output files in any format. For example, you can record images (for example, PNGs), models (for example, a pickled scikit-learn model), and data files (for example, a Parquet file) as artifacts.

############

**SQLALchemy is a database** manipulation tool for python which can be used as standalone library to manipulate relational databases. SQLalchemy provide core python based sql expressions and object oriented python based ORM. it also provide high level declarative sysntax for ORM for simplicity.

SQLalchemy follow data mapper pattern and **inspired from java hibernate**.

SQLAlchemy is a well-regarded database toolkit and object-relational-manner (ORM)implementation written in Python. SQLAlchemy provides a generalized interface for creating and executing database-agnostic code without needing to write SQL statements.

<https://github.com/zzzeek/sqlalchemy>