Databricks mlflow Object Relationships

Andre Mesarovic
Sr. Specialist Solutions Architect
12 September 2022

databricks

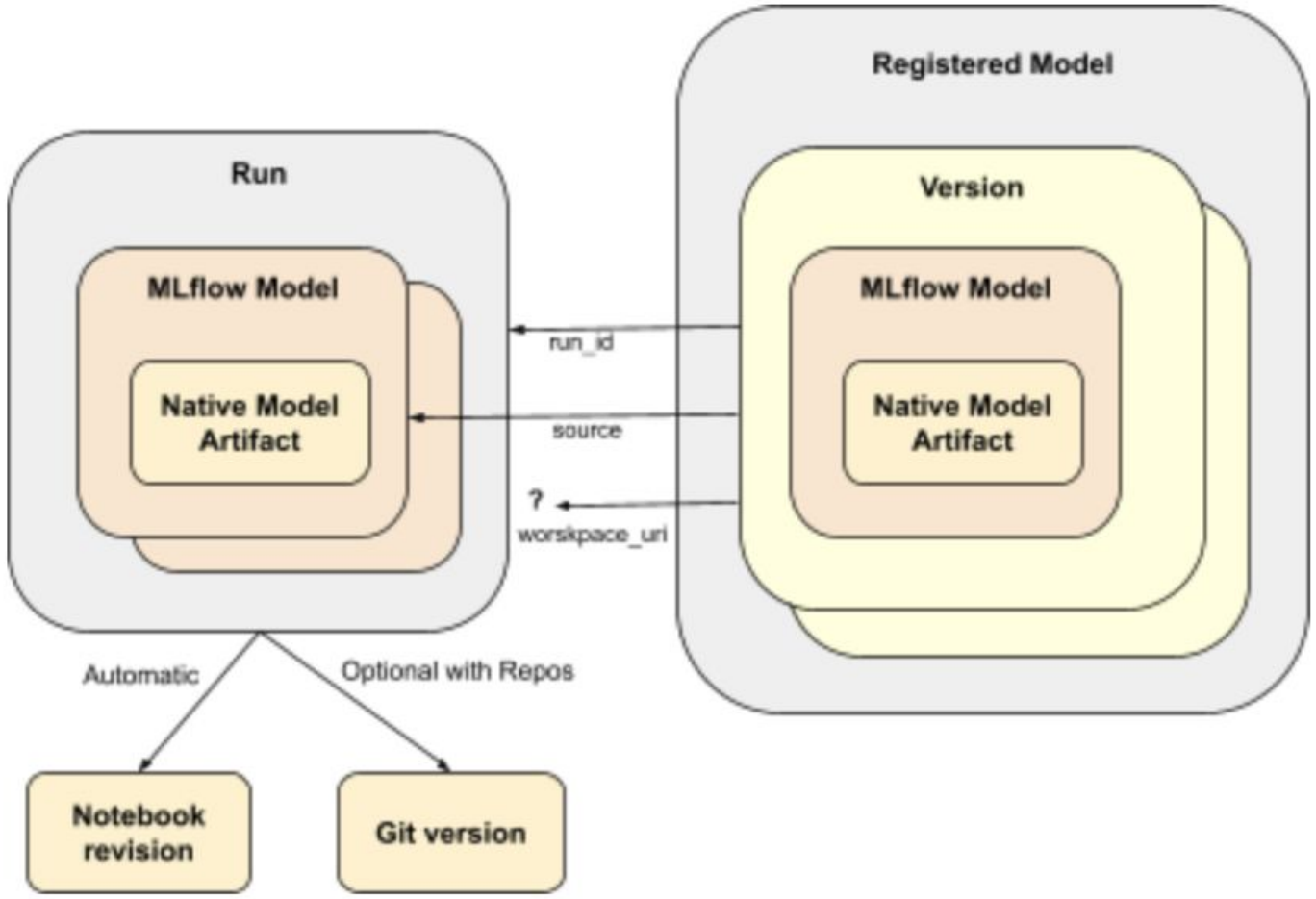
Overview

- Databricks MLflow objects (runs, experiments, registered models and their versions, notebooks) form a complex web of relationships.
- Objects live in different places: workspace objects, DBFS (cloud) and MySQL.
  - A run's metadata lives in MySQL, its artifacts in cloud and its notebook in the workspace and/or git.
- Experiments have zero or more runs.
- Registered models have 0 or more versions that point to a run's MLflow model.
- Code that generated a run's MLflow model:
  - MLflow runs have pointers to a notebook revision that generated the model.
  - Runs will/should have pointers to the git version of a notebook that generated the model.

Model terminology

- Model is an overloaded term with three meanings:
  - **Native model artifact** - this is the lowest level and is simply the native flavor's serialized format. For sklearn it's a pickle file, for Keras it's a directory with TensorFlow's native SaveModel format files.
  - **MLflow model** - a wrapper around the native model artifact with metadata in the MLmodel file and environment information in conda.yaml and requirements.txt files.
  - **Registered model** - a bucket of model versions. A model version contains one MLflow model that is cached in the model repository. A version has the following links (expressed as tags):
    - run_id - points to the run that generated the version's model.
    - source - points to the path of MLflow model in the run that corresponds to the version's model.
    - workspace_uri - currently missing. Needed if using shared model registry. ML-19472.

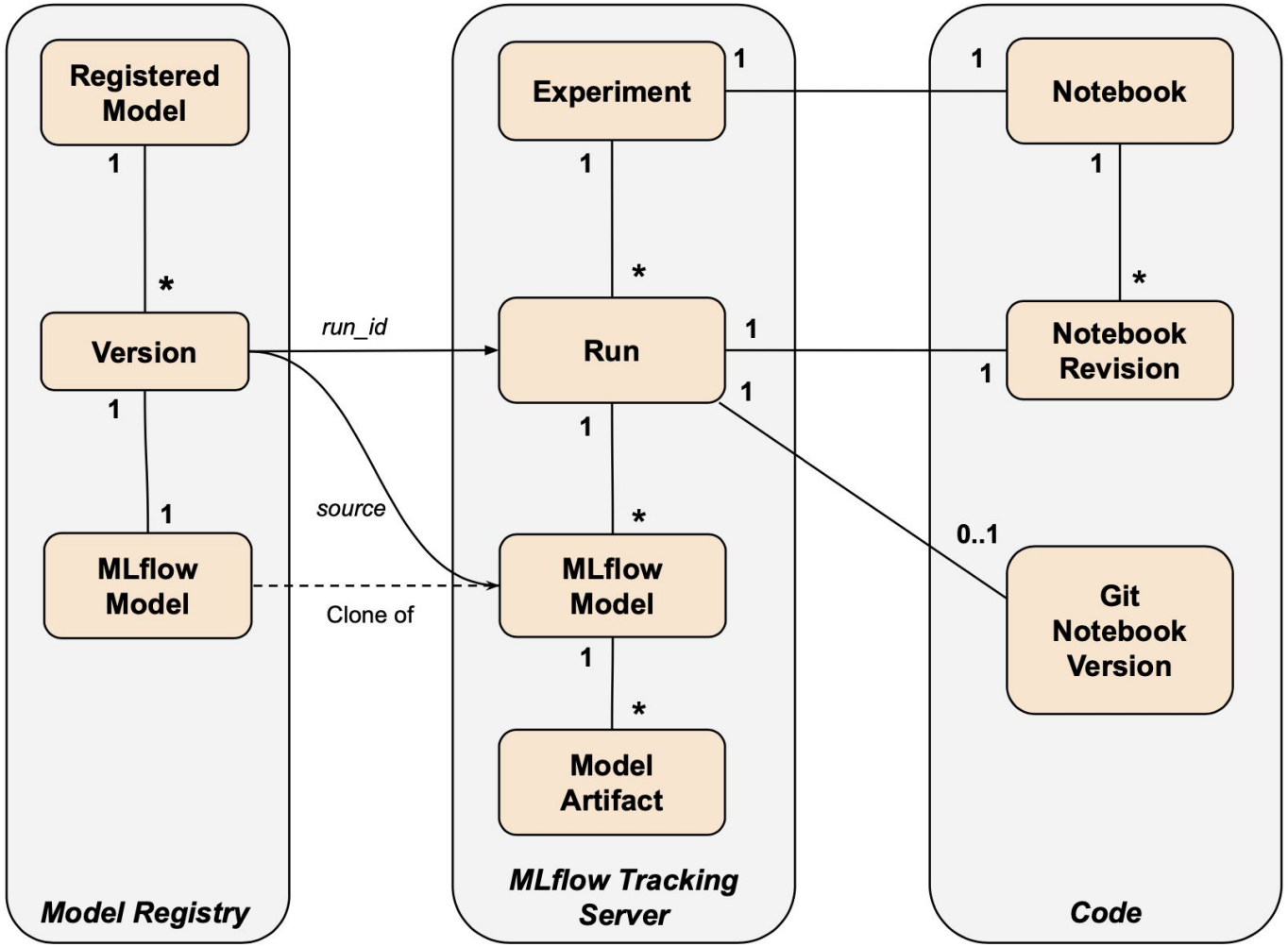
Model relationships



Databricks MLflow object relationships

- Runs
  - Contains one or more MLflow models
- Experiments
  - Notebook experiments
  - Workspace experiments
- Registered models
  - A registered model contains versions
  - A version points to one run's MLflow model
  - Native model artifacts - the actual bits that execute predictions that are part of the MLflow model
- Notebooks

Databricks MLflow objects relationships



databricks

Diagram legend

- Diagram uses the UML modeling language.
  - *: indicates a many relationship
  - 1: indicates a required one relationship.
  - 0..1: indicates an optional one relationship.
- This is a logical diagram. Not all nuances are captured for simplification.
- The diagram represents a notebook experiment.
- A workspace experiment is not represented in the diagram.

Registered models

- A registered model is a bucket for model versions.
- A version has one MLflow model which is linked to the run that generated it.
- The *production* and *staging* stage have one "latest" version.
- Registered model versions are cached in the model registry.
- This is a clone of the run's MLflow model that the version points to.
- If source run is in a different workspace we have a lineage reachability problem.

  See ML-19472 - *Add workspace URI field in ModelVersion for a registered model to make run reachable*.

Experiments

- An experiment has zero or more runs.
- Two types of experiments:
  - Notebook experiment
    - Relationship of experiment to notebook is one-to-one.
    - Workspace path of the experiment is the same as its notebook.
  - Workspace experiment
    - Relationship of experiment to notebook is one-to-many.
    - Explicitly specify the experiment path with *set_experiment* method.
    - Different notebooks can create runs in the same experiment.

Runs

- A run belongs to only one experiment.
- A run is linked to one notebook revision. MLflow notebook tags:
  - mlflow.databricks.notebookRevisionID
  - mlflow.databricks.notebookID
  - mlflow.databricks.notebookPath
- Optionally a run's notebook can be linked to a git reference.
  - See discussion on Notebook below for details.
- A run can have one or more MLflow models (flavors) such as Sklearn and ONNX.
- Every run has a default Pyfunc flavor which is wrapper around the native model.

Notebooks

- A notebook has many revisions.
- Optionally, a notebook revision can be checked into git with Databricks Repos.
- Need to capture git reference analogous to the MLflow open source tags:
  - mlflow.source.git.commit
  - mlflow.source.git.repoURL
  - mlflow.gitRepoURL
- See ML-19473 - *Add git reference tags to Databricks run if its notebook is synced with Repos*
- Two sources of truth for a notebook snapshot that can be confusing:
  - Databricks notebook revision
  - Git version

Happy mlflow journey!