D ONE Presents
Brick by Brick

IEEE SDS 2033 Workshop

22.06.2023

Robert Yousif
robert.yousif@done.ai

Spyros Cavadias
spyros.cavadias@done.ai

Content

- Introducing the Databricks Lakehouse
- Part 1 Hands-on:
  - Databricks Workspace
  - Delta + Unity Catalog
  - Medallion Architecture & Workflow Orchestration
- Part 2 Hands-on:
  - ML & MLops in Databricks

Introduction

What is a data platform?

A data platform is an **integrated set of technologies** that collectively meets an organization's end-to-end data needs.

MongoDB

A data analytics platform is an **ecosystem of services and technologies** that needs to perform analysis on voluminous, complex and dynamic data
-
that allows you to retrieve, combine, interact with, explore, and visualize data from the various sources a company might have.

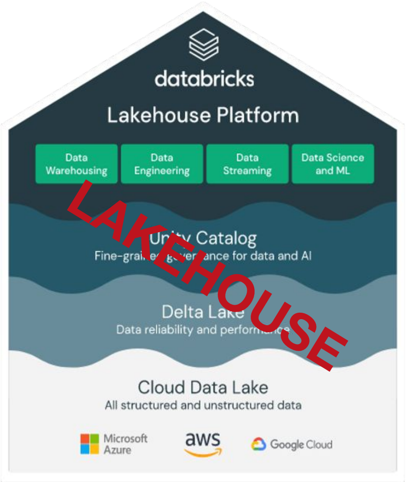
databricks

Data platforms encompass a **range of elements** required to support the data management cycle.

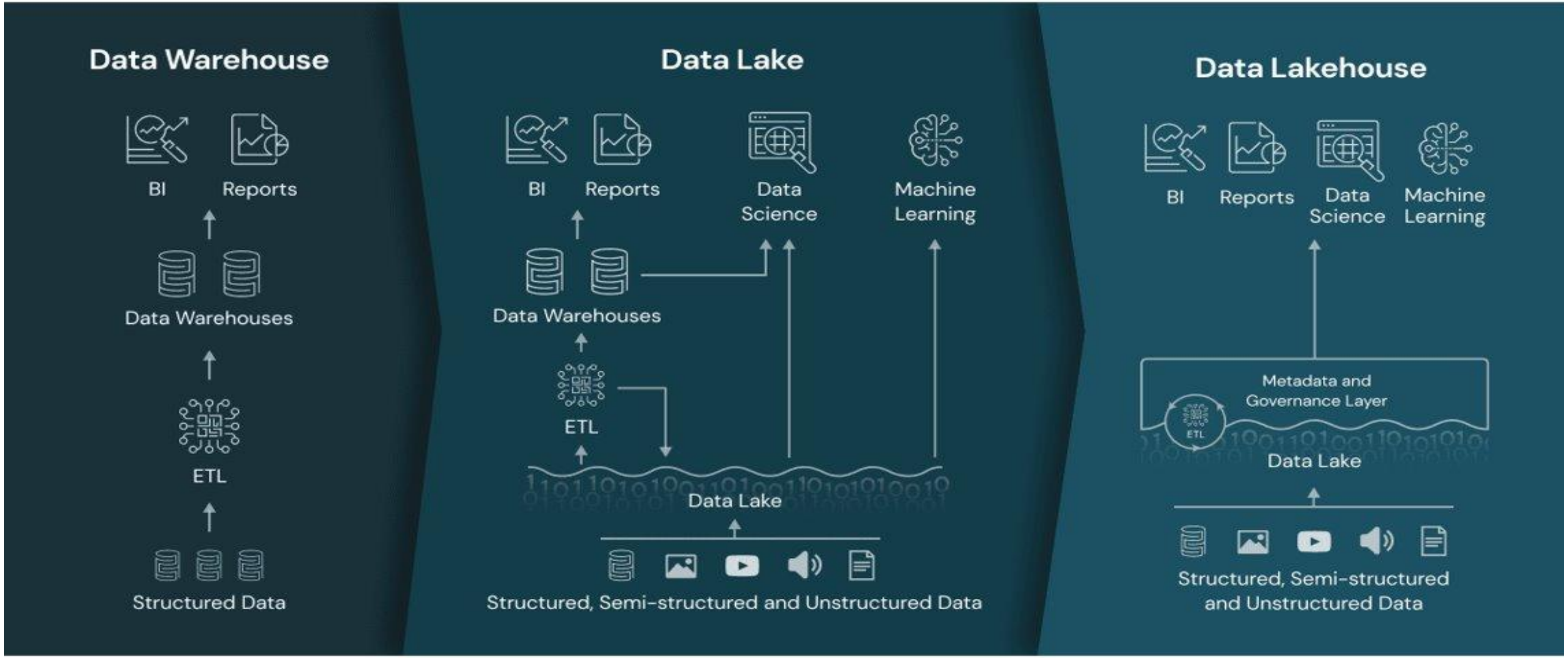
snowflake

What do you need from a data platform?

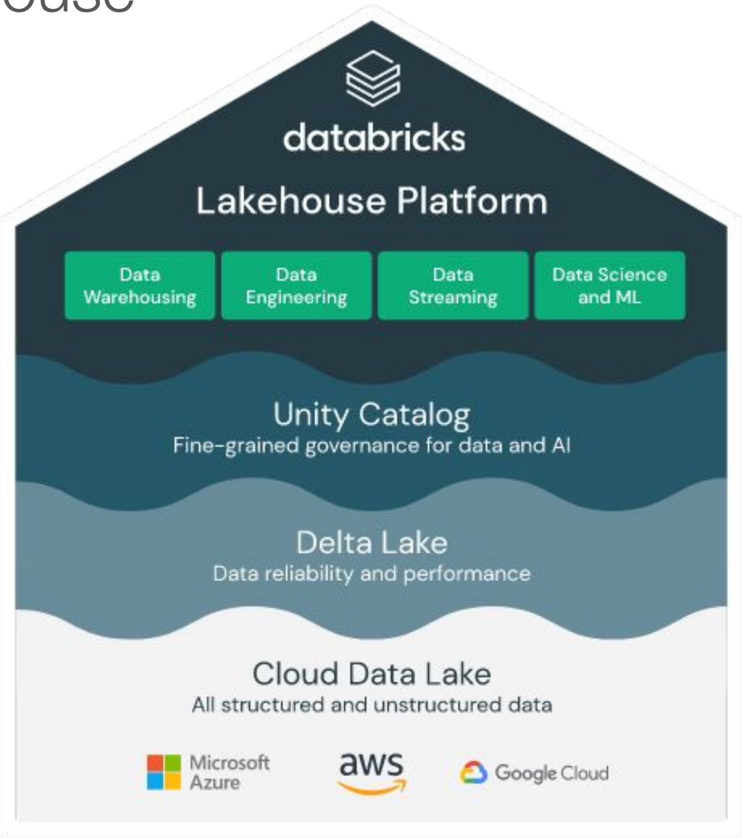
- Warehouse
  - BI & Reporting
  - Structured Data
- Data Lake
  - Data Science & Machine Learning
  - Unstructured Data
- Others
  - Infrastructure
  - Governance
  - Operations



Why the Lakehouse?



Databricks Lakehouse

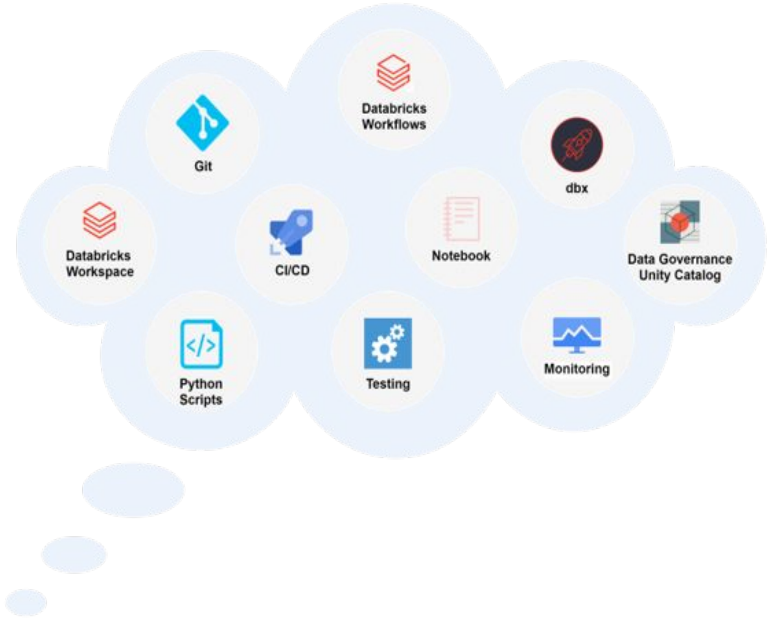


What is Databricks

A unified set of tools for building, deploying, sharing and maintaining enterprise-grade data solutions at scale.

Combining

- Data Engineering
- Machine Learning, AI & Data Science
- Data Warehousing, analytics & BI
- Data Governance and Secure Data Sharing



Databricks - Platform Integrations/services/technologies

Open Source:

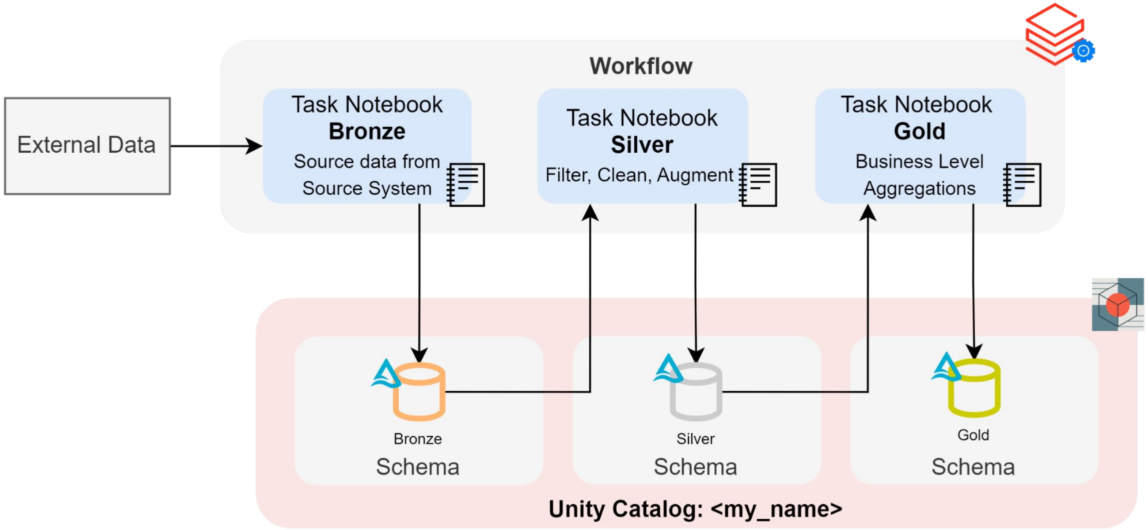
- Delta Lake
- Apache Spark
- Delta Sharing
- Mlflow

Proprietary Tools:

- Workflows
- Databricks SQL
- Delta Live Tables
- Unity Catalog

Workshop Part 1

1. Databricks Workspace
2. Delta + Unity Catalog
    a. Read and Write Tables
    b. Upload data to Unity Catalog
    c. Time Travel + Installing Libraries
3. Medallion Architecture & Workflow Orchestration
    a. 3 Notebooks - Medallion architecture
    b. Databricks Workflows

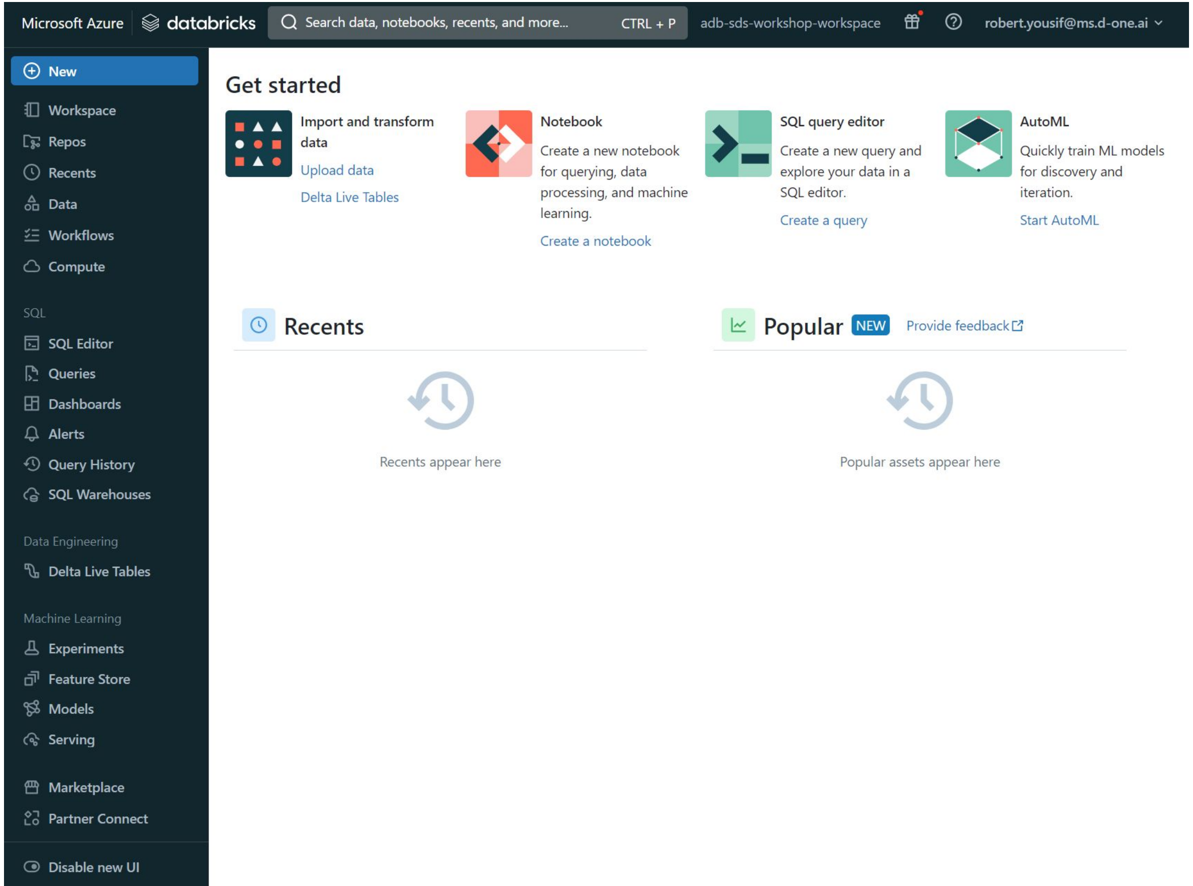


Databricks Workspace

Databricks Workspace

New Look - All in one page

- Workspace
- SQL
- Data Engineering
- Machine Learning
- Others



Demo + Exercise

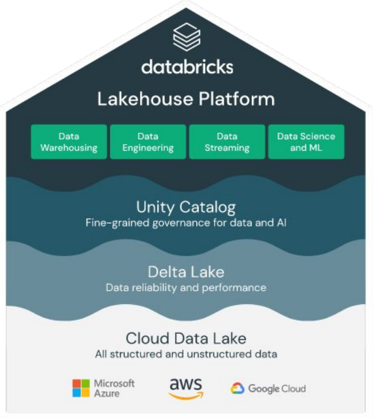
- Demo
  - Workspace
- Exercise: Setup your workspace
  - Adding the repository
  - Creating a personal cluster

LINKS:

- Github Repository for the workshop.
- Databricks Workspace

Delta + Unity Catalog

Delta Lake



ACID Transactions

Protect your data with serializability, the strongest level of isolation

Scalable Metadata

Handle petabyte-scale tables with billions of partitions and files with ease

Time Travel

Access/revert to earlier versions of data for audits, rollbacks, or reproduce

Open Source

Community driven, open standards, open protocol, open discussions

Unified Batch/Streaming

Exactly once semantics ingestion to backfill to interactive queries

Schema Evolution / Enforcement

Prevent bad data from causing data

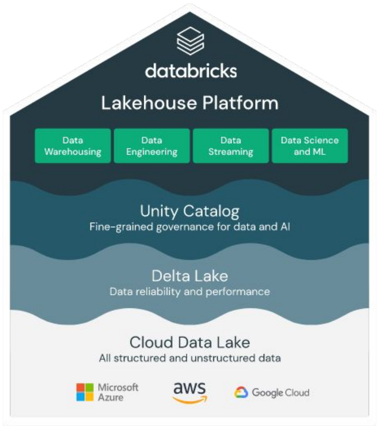
Audit History

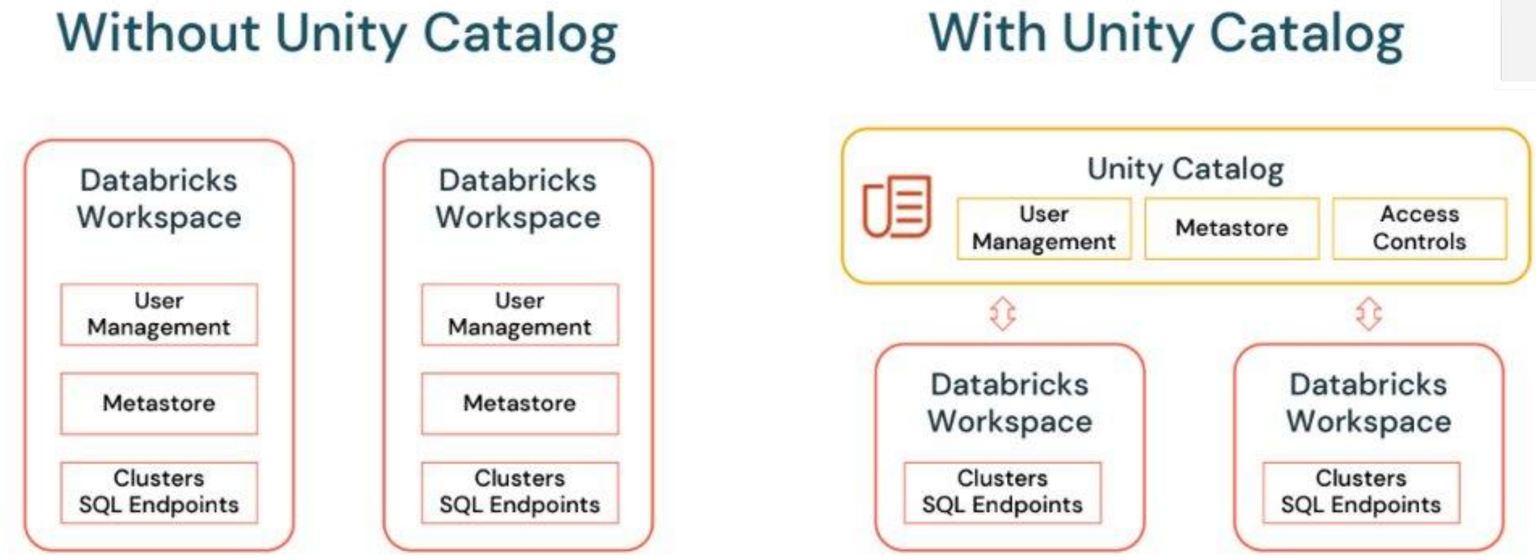
Delta Lake log all change details providing a fill audit trail

DML Operations

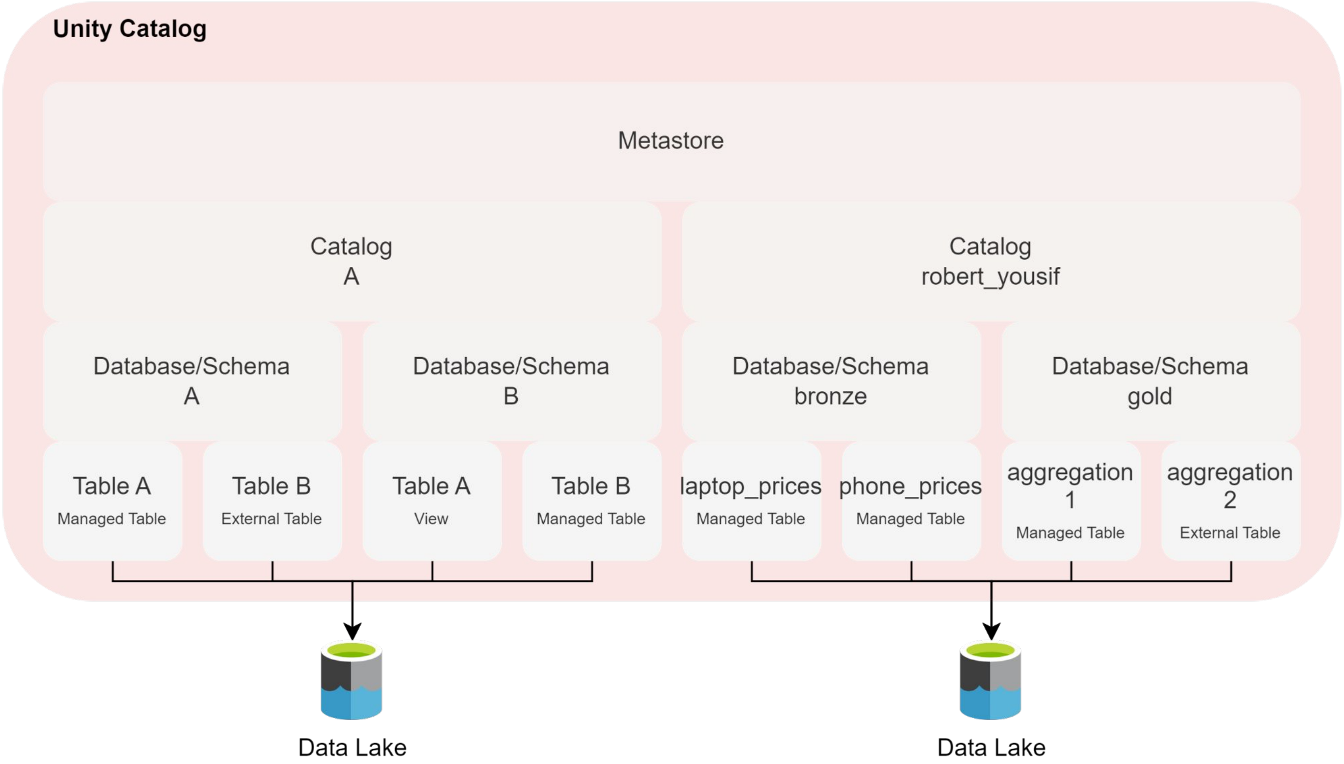
SQL, Scala/Java and Python APIs to merge, update and delete datasets

Unity Catalog





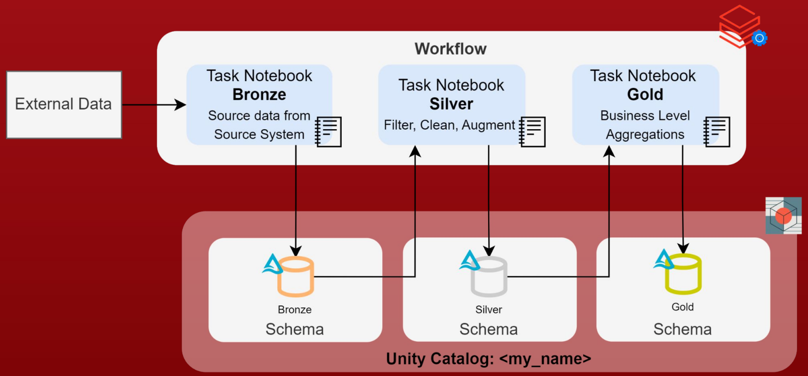
Unity Catalog - Structure



Demo + Exercise

- Demo of Delta + Unity Catalog Notebook
- Exercise
  - `1. Download the data from the repository:`
    `[laptop_data](https://github.com/d-one/sds-brick-by-brick/tree/main/data)`
  - `2. Upload the data as a workspace object to your personal directory.`
  - `3. Read the data and display it`
  - `4. Write to your own table inside your own catalog and schema`
  - ``5. See what other `DESCRIBE` commands you can run on your table to get more information``
  - `6. Share the table with the person sitting next to you.`

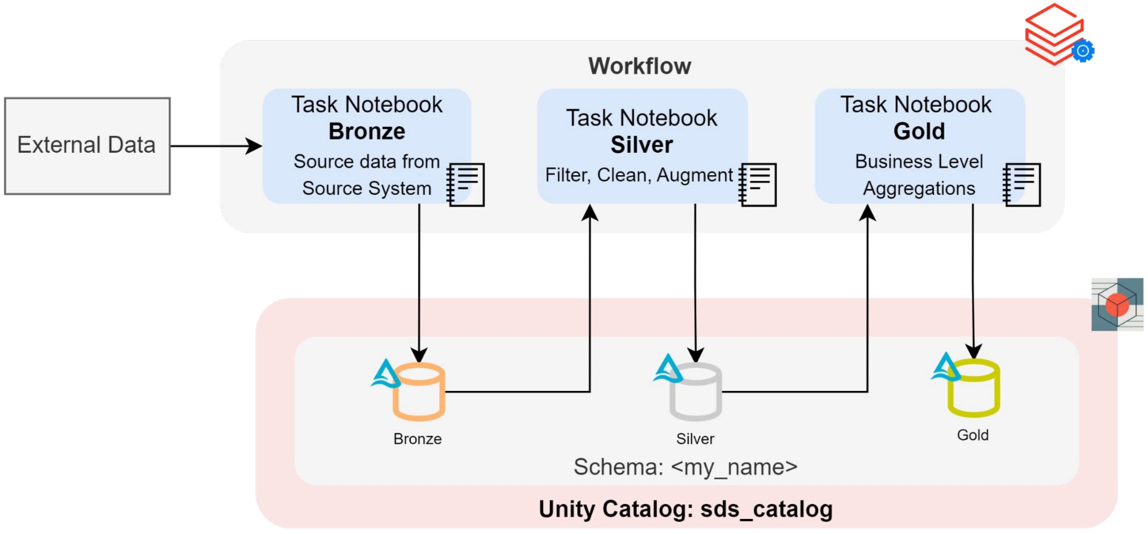
Medallion Architecture & Workflow Orchestration



Databricks Workflows

- Orchestrate & Automate end-to-end data pipelines
- GUI & API for defining and managing complex workflows
- Supports multiple task types
  - Python Script/Wheel file
  - Notebooks
  - dbt & dlt
  - Databricks SQL Queries

Demo of Workflows



Exercise

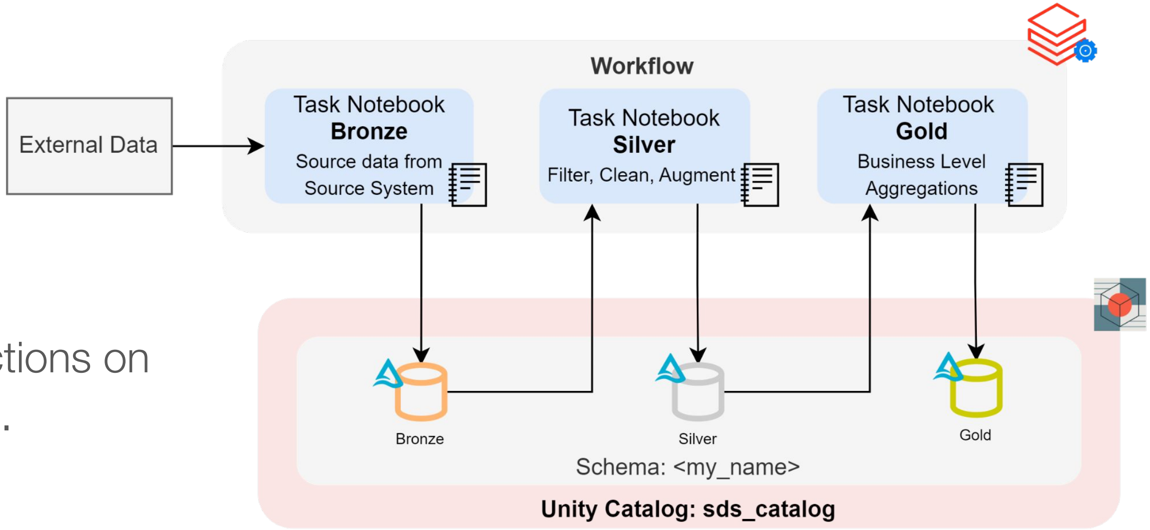
- 3 notebooks - the medallion architecture
  - Create Bronze, Silver & Gold table inside your own Unity Catalog Schema

LINKS:

- Github Repository for the workshop.
- Databricks Workspace

Exercise 2

- Create & run your own workflow job
- Monitor your workflow



LINKS:

- Github Repository with instructions on how to create a Workflow Job.
- Databricks Workspace

ML and MLOps in Databricks

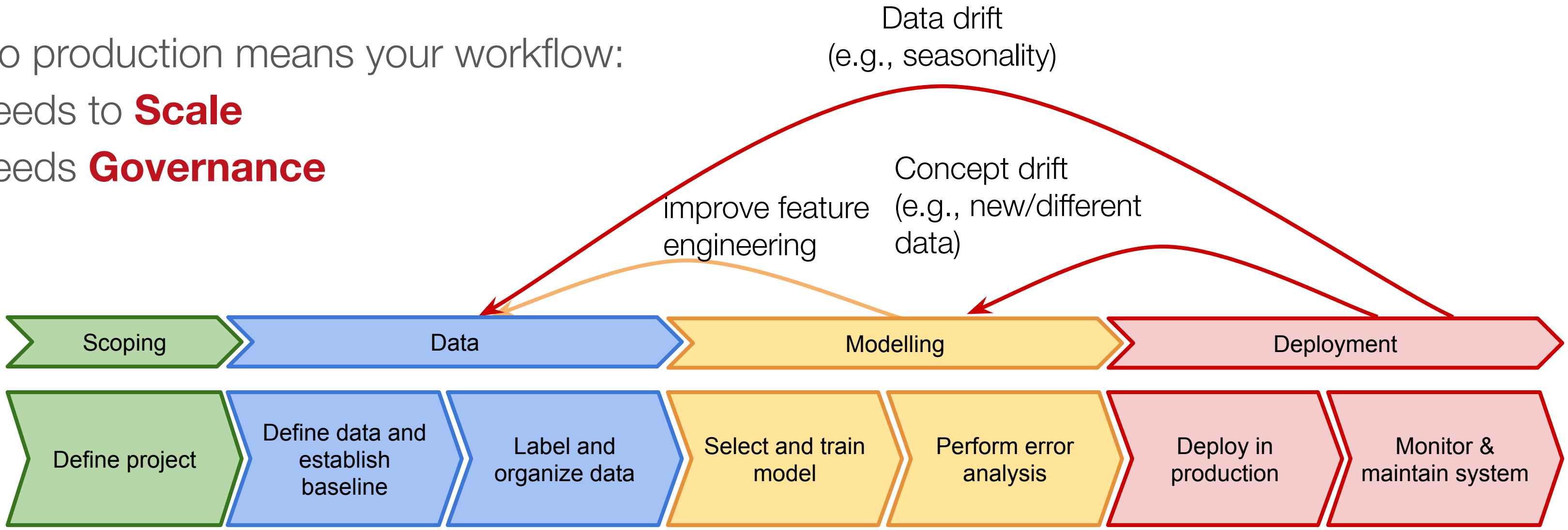
Workshop Part 2

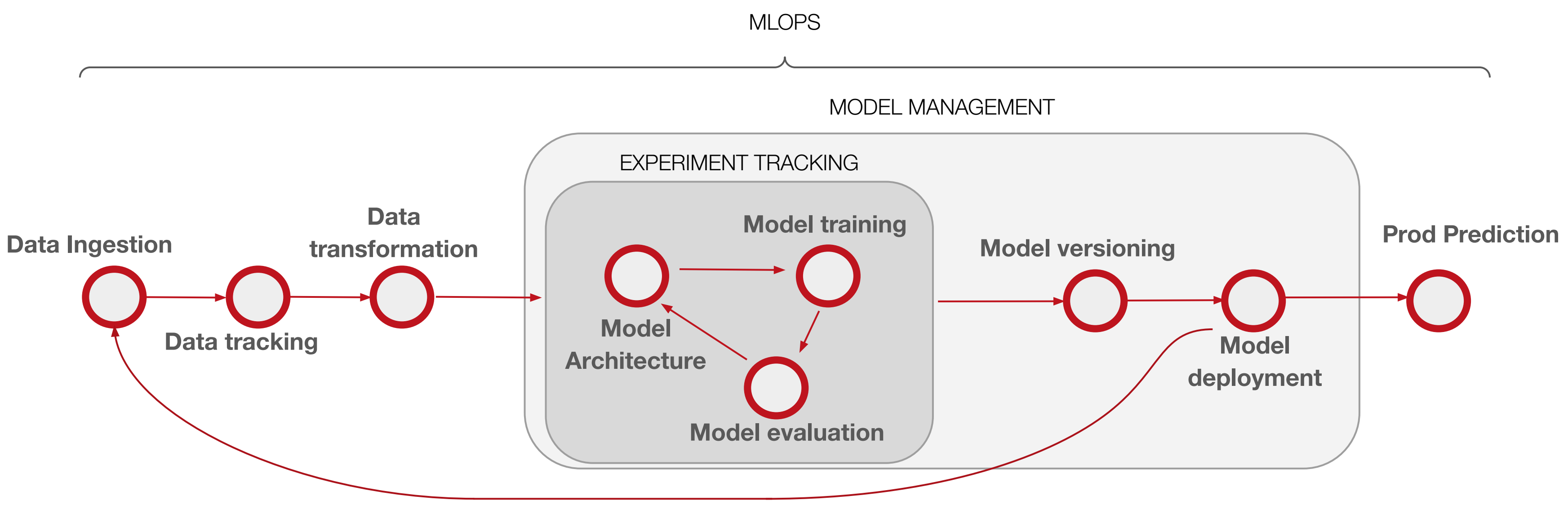
1. Tracking Experiments
2. Model Registry
3. Model Serving
4. Hands-on

Machine Learning Pipeline

Going to production means your workflow:

- Needs to **Scale**
- Needs **Governance**




MLOPS
MODEL MANAGEMENT
EXPERIMENT TRACKING
Data Ingestion
Data tracking
Data transformation
Model training
Model Architecture
Model evaluation
Model versioning
Model deployment
Prod Prediction

mlflow and databricks

- Open source
- Runs the same way everywhere (locally or in the cloud)
- Useful from 1 developer to 100+ developers

- Design philosophy:
  1. API-First
  2. Integration with popular libraries
  3. Modular design (can use DISTINCT components separately)

MLflow Components

MLflow Tracking	MLflow Projects	MLflow Models	Model Registry
Record and query experiments: code, data, config, and results	Package data science code in a format to reproduce runs on any platform	Deploy machine learning models in diverse serving environments	Store, annotate, discover, and manage models in a central repository
Read more	Read more	Read more	Read more

Useful links:

- www.mlflow.org
- www.github.com/mlflow
- www.databricks.com/mlflow

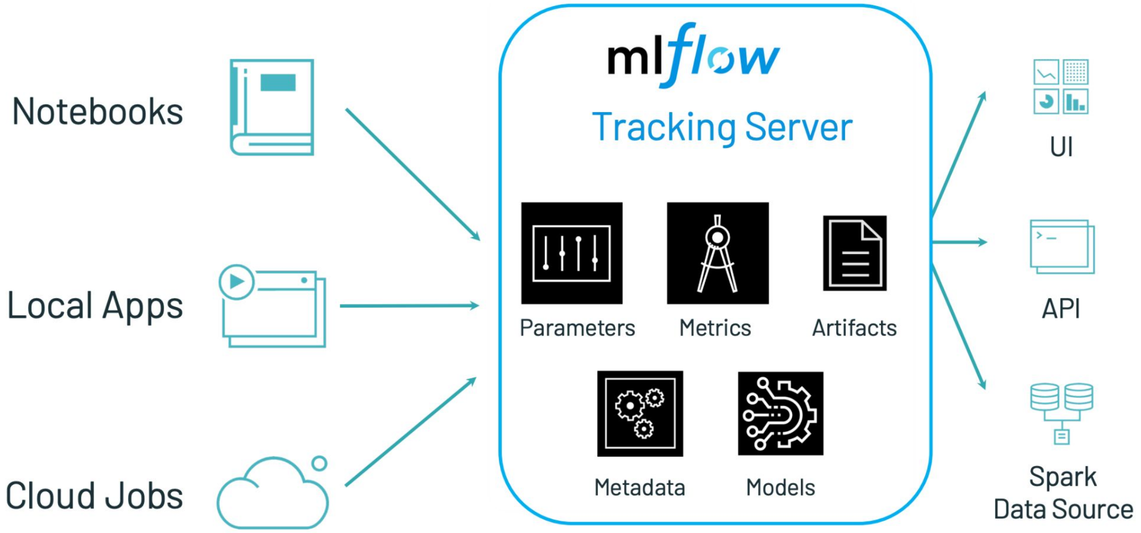
MLflow Tracking

What do we track?

- Parameters : inputs to our code `mlflow.log_param()..`
- Metrics : numeric values to access our models `mlflow.log_metric()..`
- Tags/Notes: info about the run `mlflow.set_tag()..`
- Artifacts: files,data and models produced `mlflow.log_artifact(), mlflow.log_artifacts()..`
- Source: what code run
- Version: what version of the code run (github)
- Run: the particular code instance (id) captured by MLflow `mlflow.start_run()..`
- Experiment: the set of runs `mlflow.create_experiment(), mlflow.set_experiment()..`

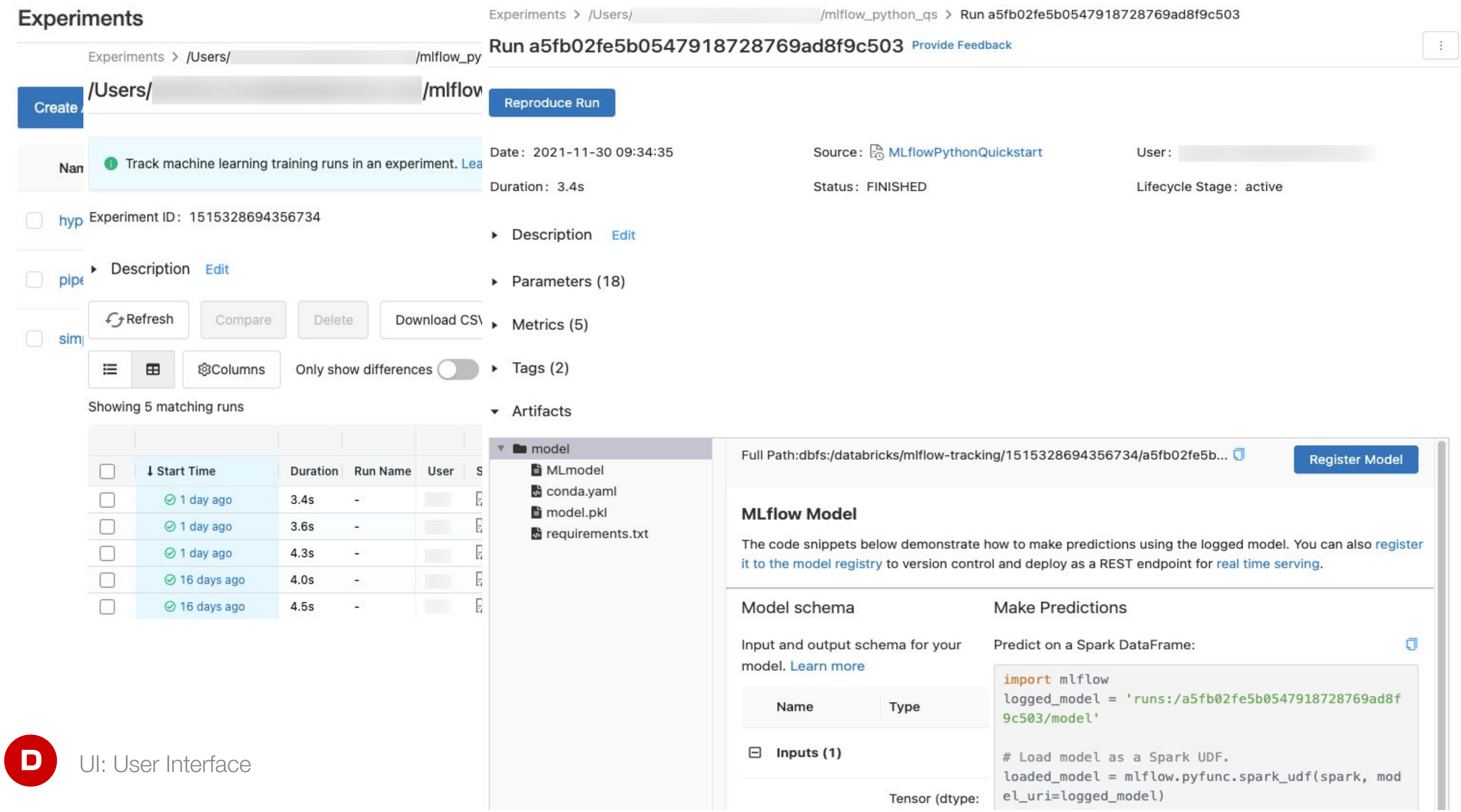
More on : https://www.mlflow.org/docs/latest/tracking.html

MLflow Tracking



URI: Uniform Resource Identifier

MLflow UI - Databricks experiments



UI: User Interface

MLflow Model

- Standard format for packaging machine learning models in MLflow
- Defines a convention that lets you save a model in different “flavors” that can be understood by different downstream tools

```
# Directory written by
mlflow.sklearn.save_model(model,
"my_model")

my_model/
├── MLmodel
├── model.pkl
├── conda.yaml
└── requirements.txt
```

MLflow Model

- Standard format for packaging machine learning models in MLflow
- Defines a convention that lets you save a model in different “flavors” that can be understood by different downstream tools

```
# Directory written by
mlflow.sklearn.save_model(model,
"my_model")

my_model/
├── MLmodel
├── model.pkl
├── conda.yaml
└── requirements.txt
```

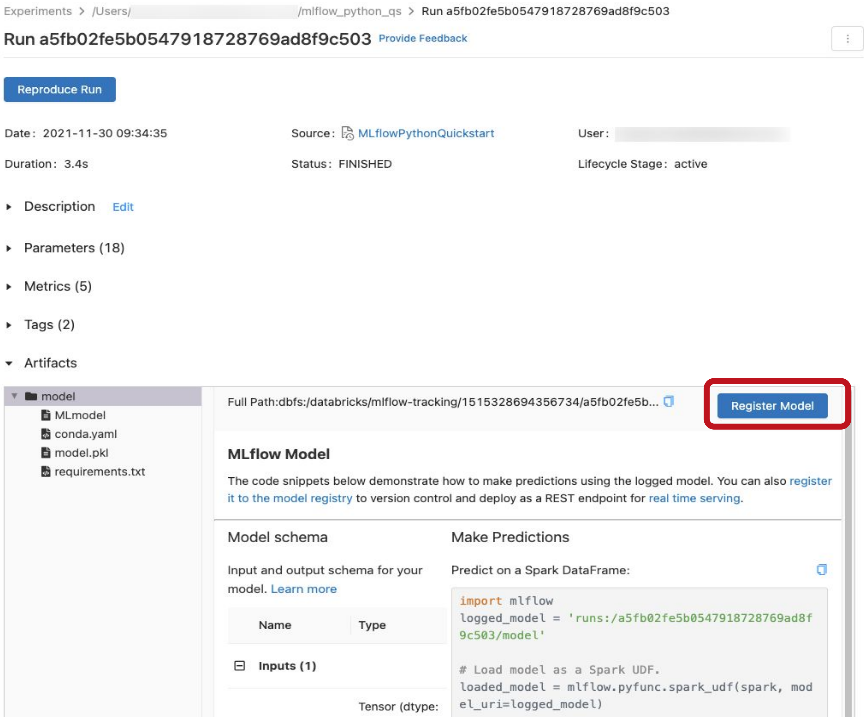
```
# in MLmodel file

time_created:
2021-10-25T17:28:53.35

flavors:
  sklearn:
    sklearn_version: 0.24.1
    pickled_model: model.pkl
  python_function:
    loader_module: mlflow.sklearn
```

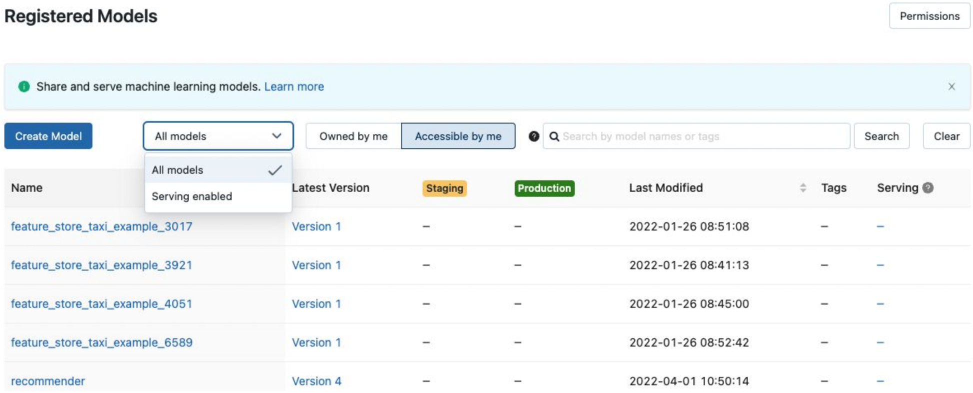
MLflow Model Registry

- It is a centralized model store, set of APIs, and UI, to collaboratively manage the full lifecycle of an MLflow Model.
- Provides **model lineage** (which MLflow experiment and run produced the model), **model versioning**, **stage transitions** (for example from staging to production), and **annotations**.
- You register a model through:
  1. API Workflow
  2. UI Workflow

```
# register model
res = mlflow.register_model(my_model_uri, "my_model")
```

MLflow Model Registry



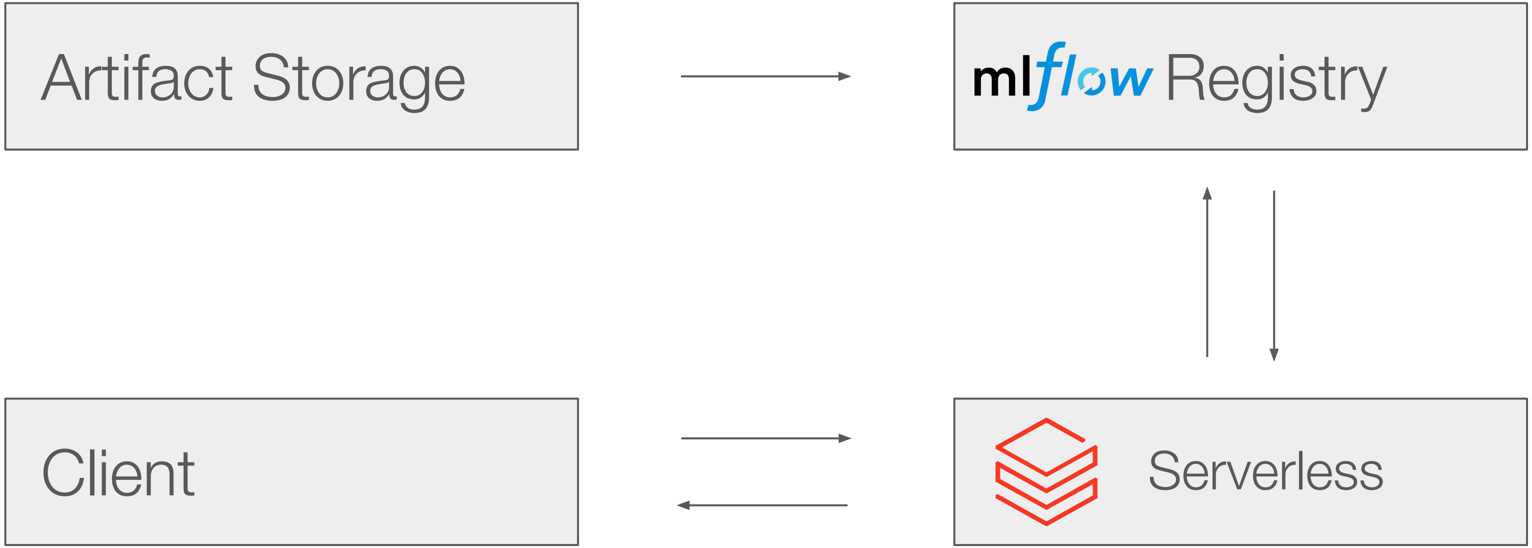


Deployment

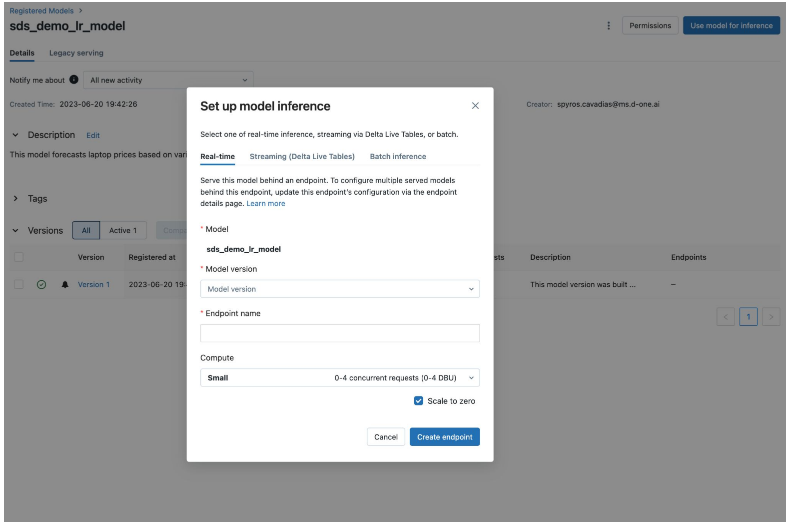
Batch Prediction

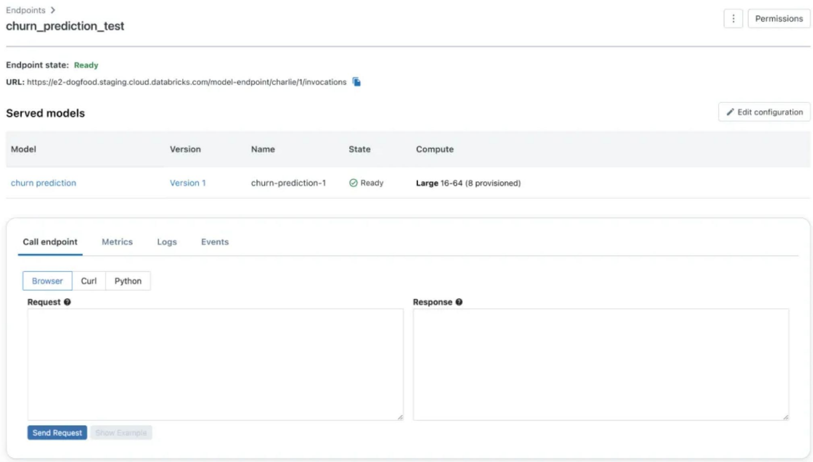
Online Prediction

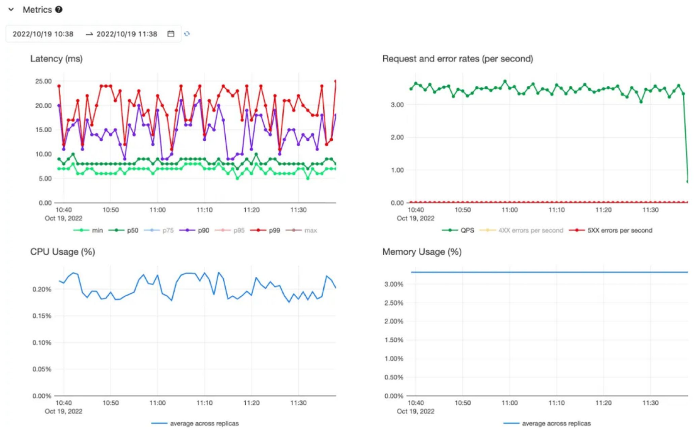
Model Serving



Model Serving



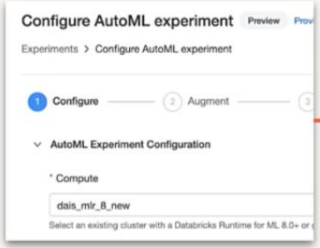


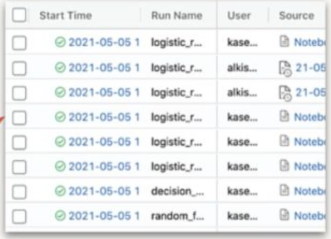


Databricks AutoML

A glass-box solution that empowers data teams without taking away control

UI and API to start AutoML training

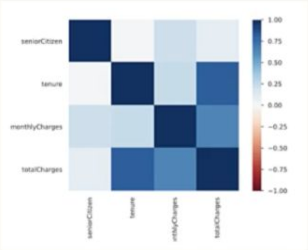




MLflow experiment

Auto-created MLflow Experiment to track models and metrics

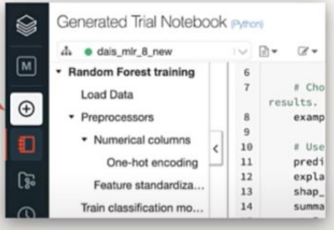
→ Easily deploy to Model Registry



Data exploration notebook

Generated notebook with feature summary statistics and distributions

→ Understand and debug data quality and preprocessing



Reproducible trial notebooks

Generated notebooks with source code for every model

→ Iterate further on models from AutoML, adding your expertise

Feature Store

The first Feature Store codesigned with a Data and MLOps Platform

Feature Registry

- Discoverability and Reusability
- Versioning
- Upstream and downstream Lineage

Feature Provider

- Batch and online access to Features
- Feature lookup packaged with Models
- Simplified deployment process

Co-designed with DELTA LAKE

- Open format
- Built-in data versioning and governance
- Native access through PySpark, SQL, etc.

Co-designed with mlflow

- Open model format that supports all ML frameworks
- Feature version and lookup logic hermetically logged with Model

Exercise 3

- Demo of Model Serving Notebook
- Exercise
  - 1. Run the ML Preprocessing notebook in your catalog to create the features table
  - 2. Move on to the ML MLflow Tracking notebook and walk through the steps to understand how to interact with MLflow experiments inside the Databricks workspace
  - 3. Move on to the ML Model Registry notebook and walk through the steps to understand how to interact with the model registry via python APIs or via the directly using the UI
  - 4. (Optional)Tie steps 1-3 together by creating a new ML workflow! See the results of the workflow run in the UI.
  - 5. (Optional)Finally move on to the AutoML notebook and see for yourself how easy it is to use databricks AutoML as a quick way to create baseline models.

Recap

Recap

- Introduction the Databricks Lakehouse
- Part 1 - Data:
  - Databricks Workspace
  - Delta + Unity Catalog
  - Medallion Architecture & Workflow Orchestration
- Part 2 - ML:
  - Experiment Tracking
  - Model Registry
  - Model Serving