

D ONE Presents Brick by Brick



Robert Yousif
robert.yousif@done.ai



Spyros Cavadias
spyros.cavadias@done.ai

IEEE SDS 2033 Workshop



22.06.2023

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.



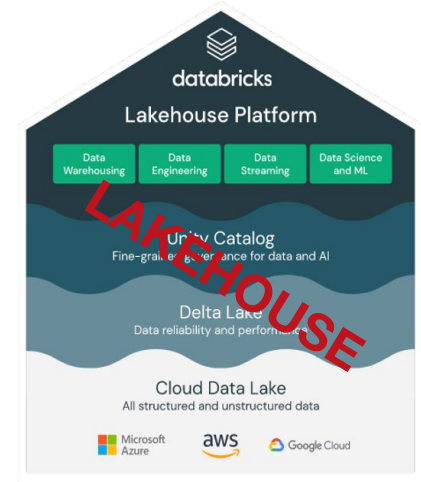
Data platforms encompass a **range of elements** required to support the data management cycle.

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.

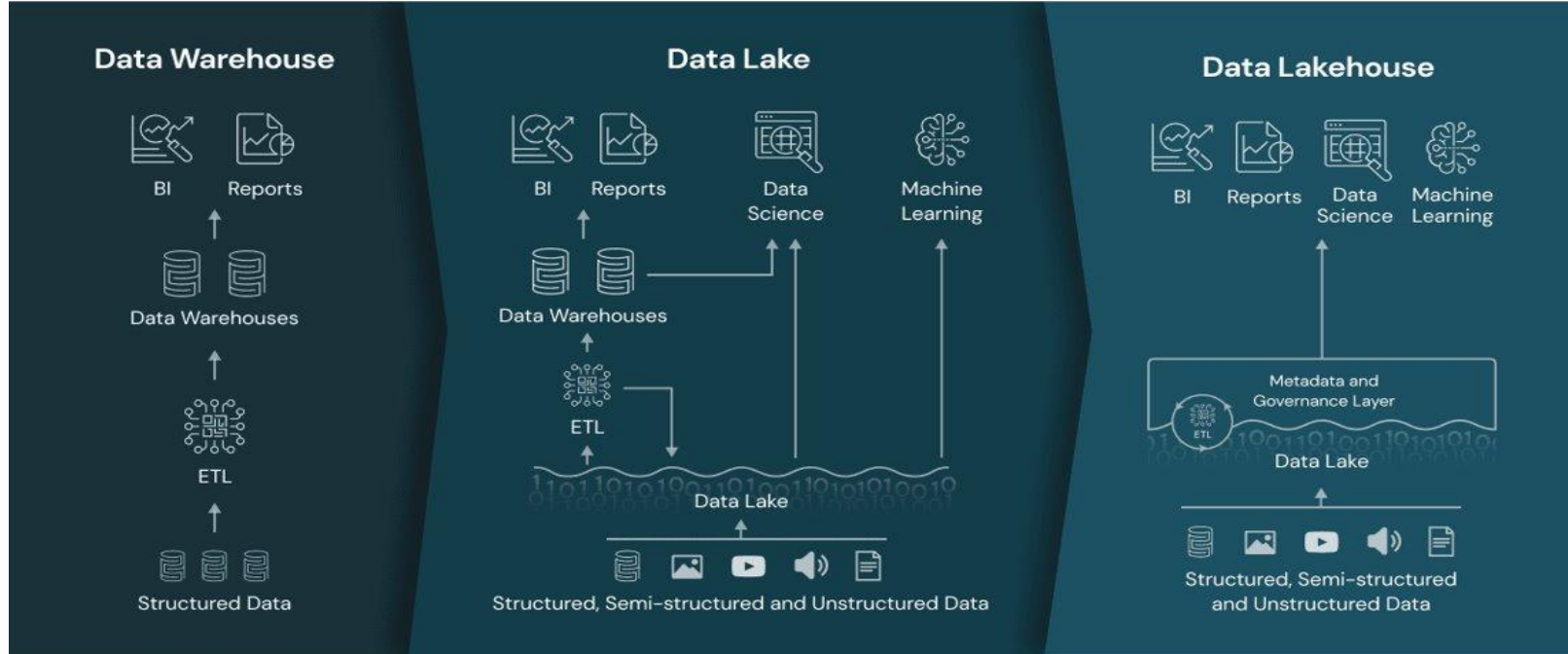


What do you need from a data platform?

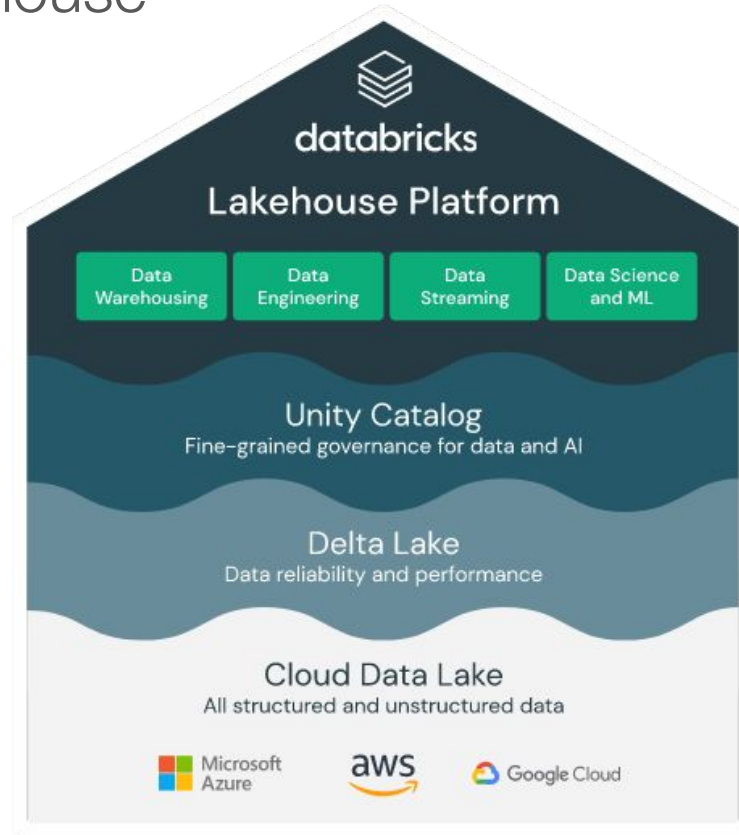
- Warehouse
 - BI & Reporting
 - Structured Data
- Data Lake
 - Data Science & Machine Learning
 - Unstructured Data
- Others
 - Infrastructure
 - Governance
 - Operations



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 

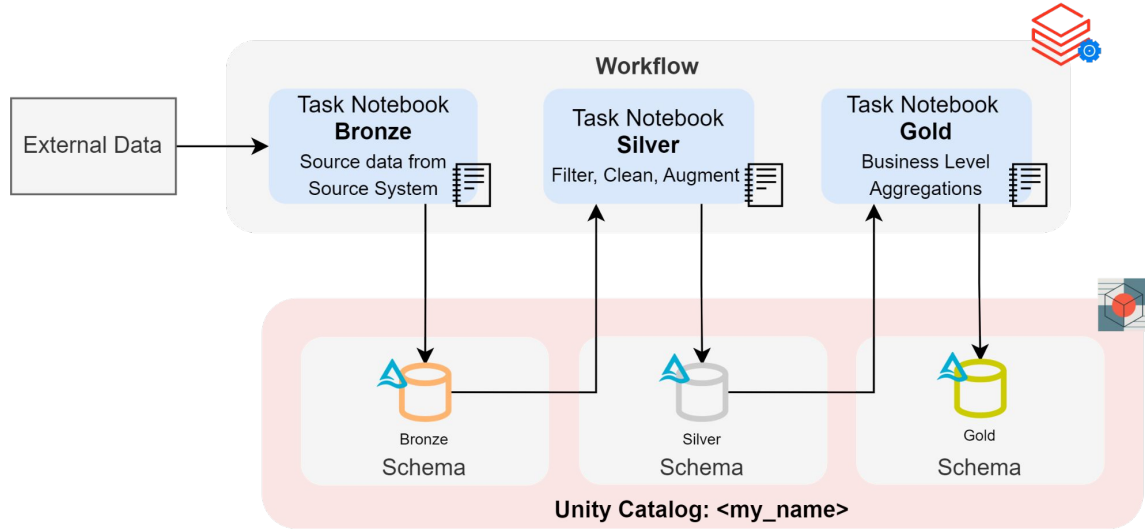
Proprietary Tools:

- 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

The screenshot displays the Databricks Workspace interface. At the top, there's a header bar with the Microsoft Azure logo, the Databricks logo, a search bar, and user information. Below the header, the interface is divided into a left sidebar and a main content area. The sidebar contains a 'New' button and a list of navigation items: Workspace, Repos, Recents, Data, Workflows, Compute, SQL (with sub-items like SQL Editor, Queries, Dashboards, Alerts, Query History, and SQL Warehouses), Data Engineering (with Delta Live Tables), Machine Learning (with Experiments, Feature Store, Models, and Serving), Marketplace, and Partner Connect. The main content area is titled 'Get started' and features four cards: 'Import and transform data' (with 'Upload data' and 'Delta Live Tables' links), 'Notebook' (with 'Create a notebook' link), 'SQL query editor' (with 'Create a query' link), and 'AutoML' (with 'Start AutoML' link). Below these, there are two sections: 'Recents' and 'Popular'. Both sections have a clock icon and a message indicating where items will appear. The 'Popular' section also includes a 'NEW' badge and a 'Provide feedback' link.



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



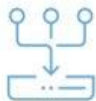
Time Travel

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



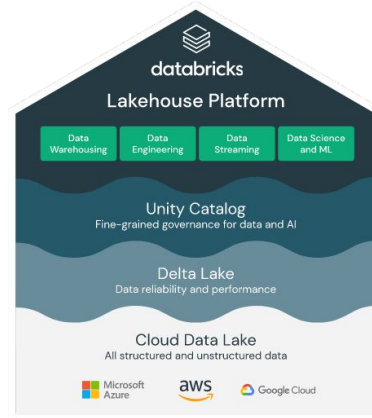
Audit History

Delta Lake log all change details providing a full audit trail



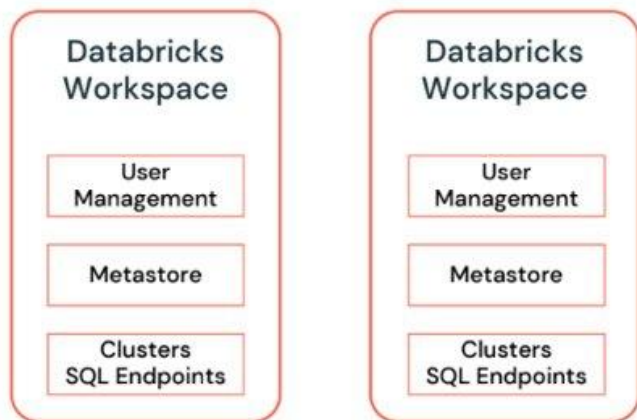
DML Operations

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

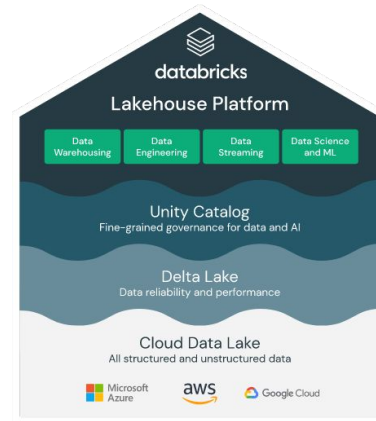
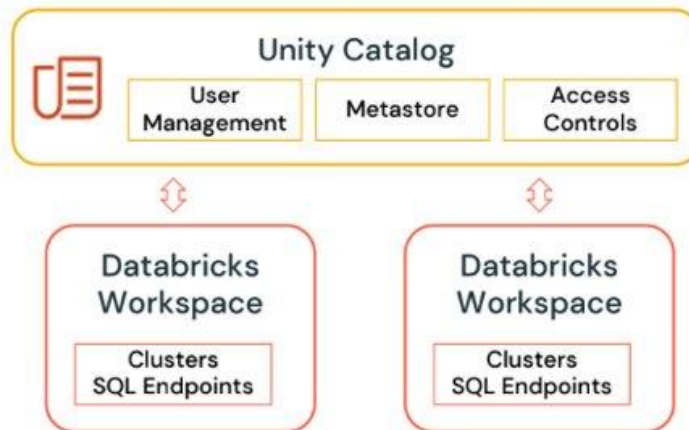


Unity Catalog

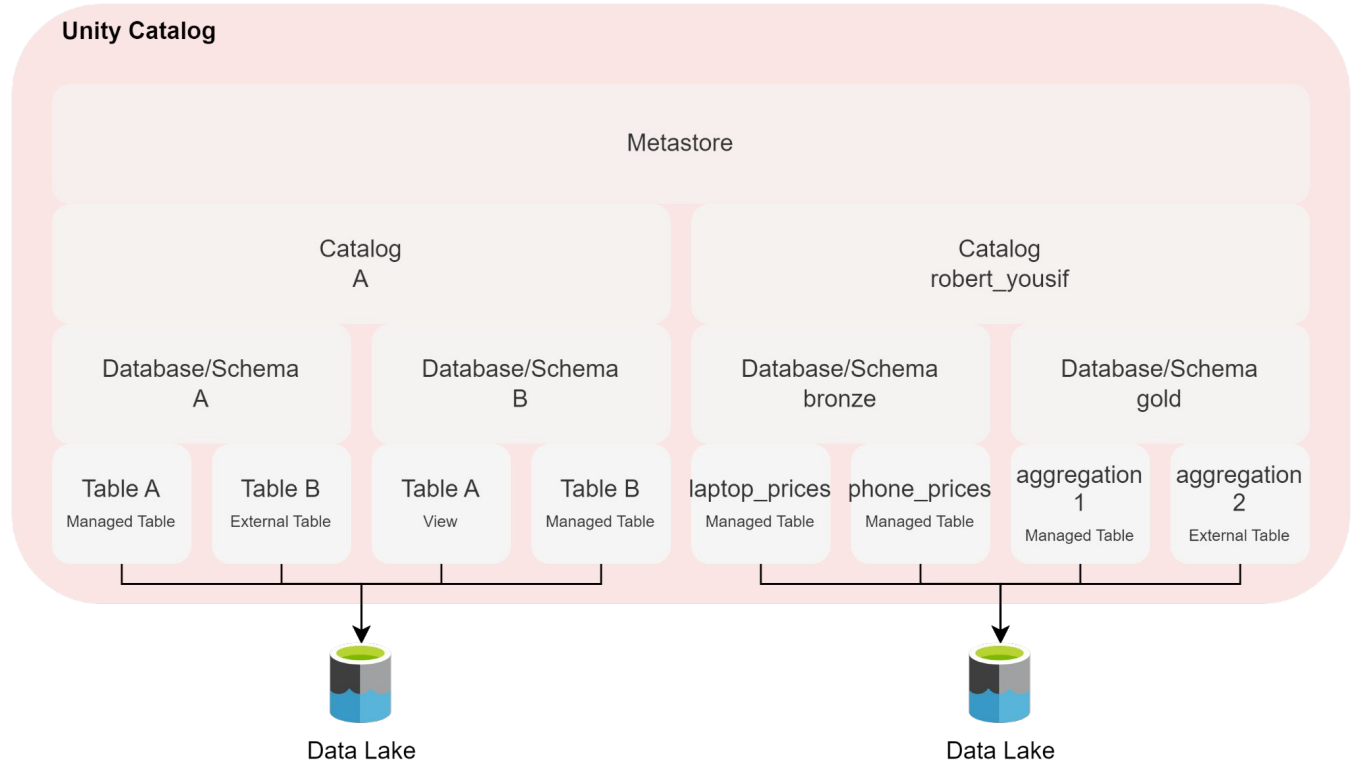
Without Unity Catalog



With Unity Catalog



Unity Catalog - Structure

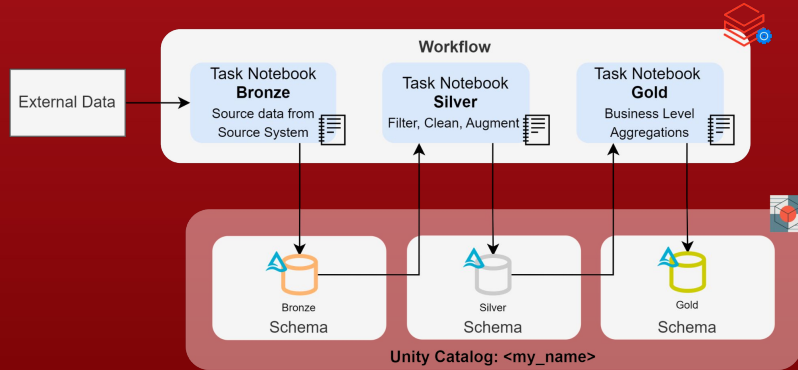


Demo + Exercise

- 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.



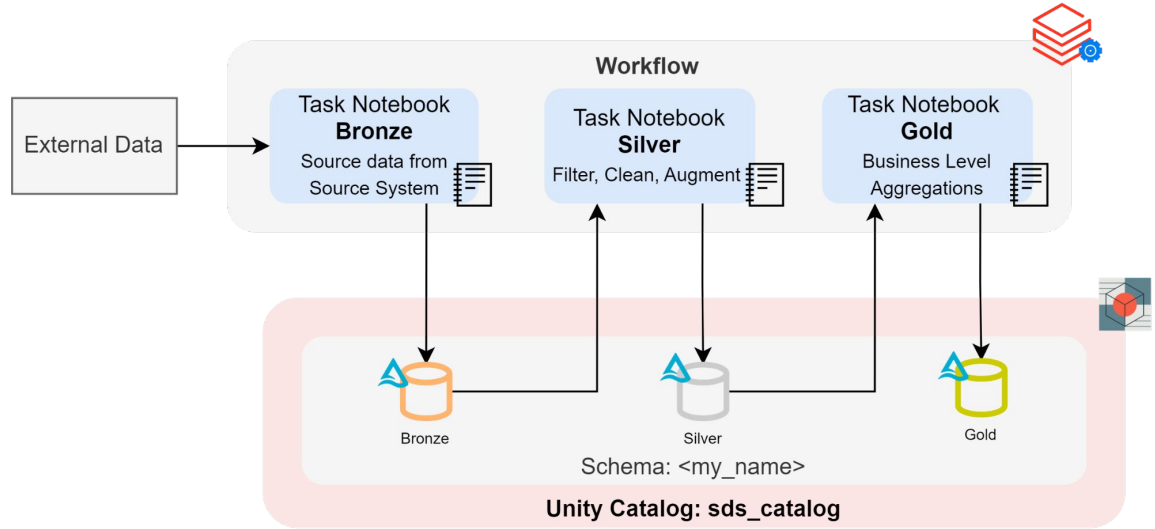
Medallion Architecture & Workflow Orchestration



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](#)

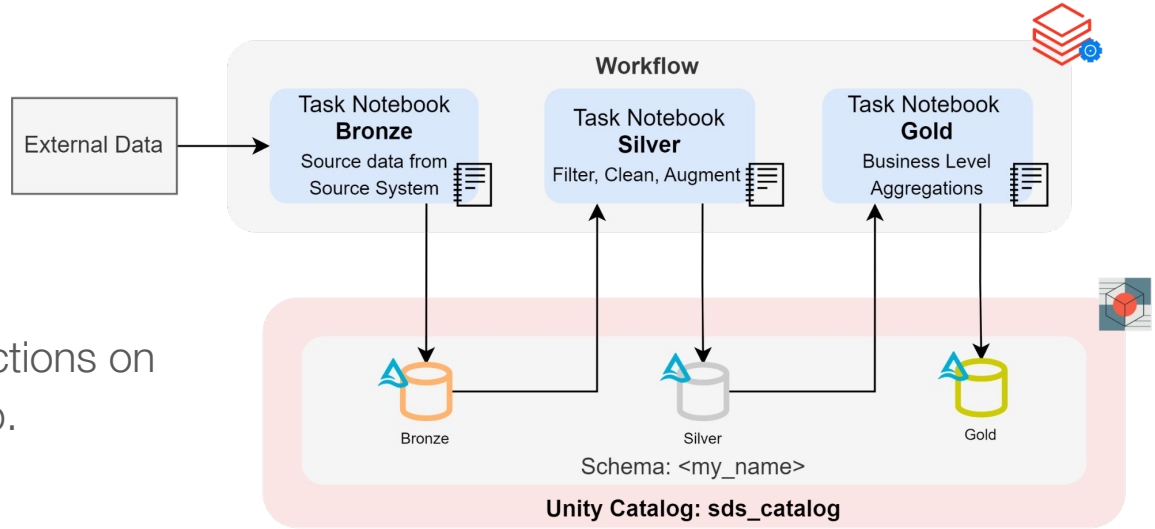


Exercise 2

- 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

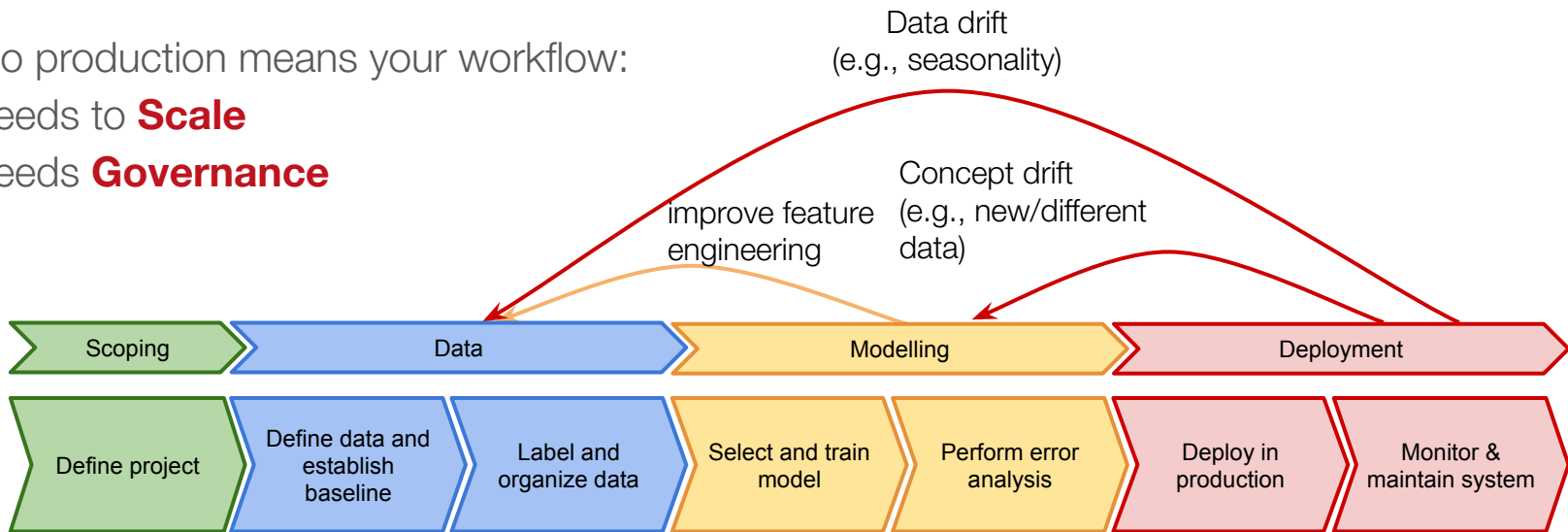
1. Tracking Experiments
2. Model Registry
3. Model Serving
4. Hands-on



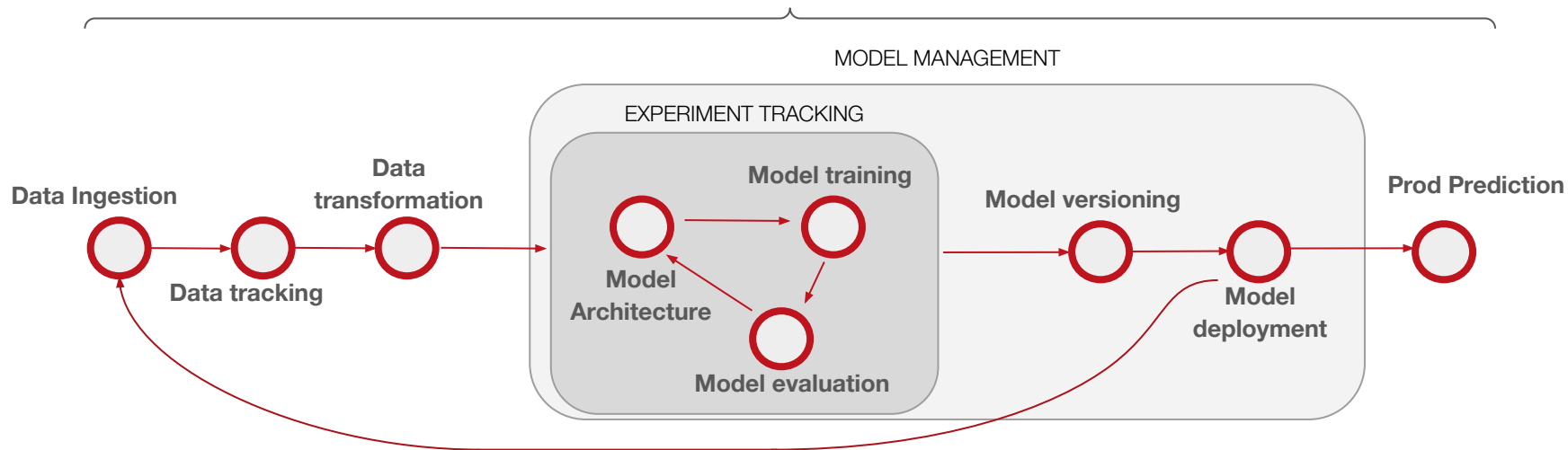
Machine Learning Pipeline

Going to production means your workflow:

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



MLOPS





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

Record and query experiments: code, data, config, and results

[Read more](#)

MLflow Projects

Package data science code in a format to reproduce runs on any platform

[Read more](#)

MLflow Models

Deploy machine learning models in diverse serving environments

[Read more](#)

Model Registry

Store, annotate, discover, and manage models in a central repository

[Read more](#)

Useful links:

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



MLflow Tracking

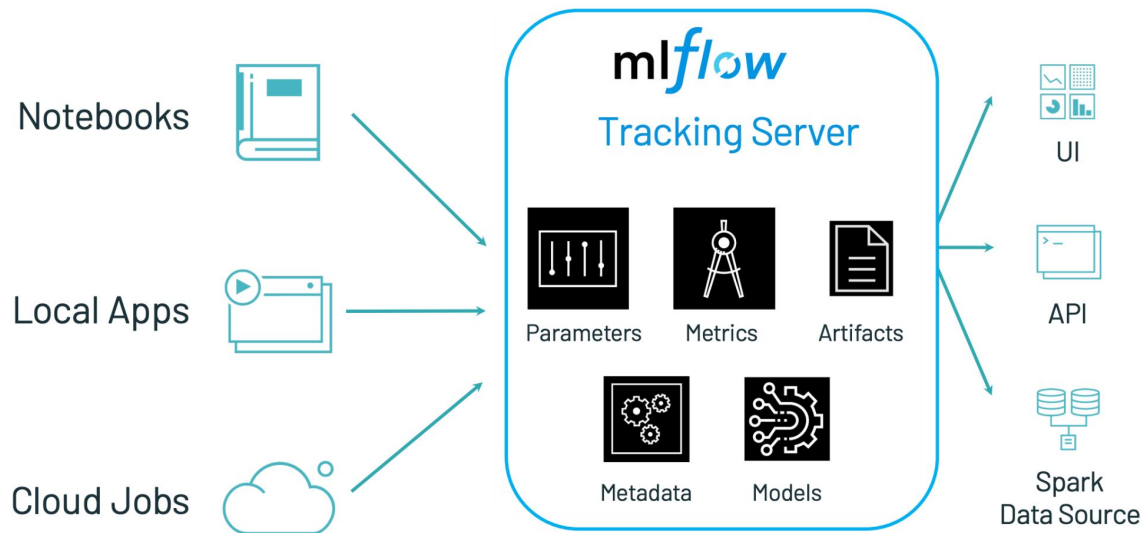
What do we track?

- 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



MLflow UI - Databricks experiments

Experiments

Experiments > /Users/ /mlflow_python_qs > Run a5fb02fe5b0547918728769ad8f9c503

Experiments > /Users/ /mlflow_py

Run a5fb02fe5b0547918728769ad8f9c503 [Provide Feedback](#)



Create

Reproduce Run

Nan

Track machine learning training runs in an experiment. [Learn](#)

Date: 2021-11-30 09:34:35

Source: [MLflowPythonQuickstart](#)

User:

Duration: 3.4s

Status: FINISHED

Lifecycle Stage: active

☐ hyp Experiment ID: 1515328694356734

▶ Description [Edit](#)

☐ pip ▶ Description [Edit](#)

▶ Parameters (18)

☐ sim [Refresh](#) [Compare](#) [Delete](#) [Download CSV](#)

▶ Metrics (5)

[Columns](#) Only show differences ☐

▶ Tags (2)

Showing 5 matching runs

▼ Artifacts

<input type="checkbox"/>	Start Time	Duration	Run Name	User	S
<input type="checkbox"/>	1 day ago	3.4s	-		
<input type="checkbox"/>	1 day ago	3.6s	-		
<input type="checkbox"/>	1 day ago	4.3s	-		
<input type="checkbox"/>	16 days ago	4.0s	-		
<input type="checkbox"/>	16 days ago	4.5s	-		

model

- MLmodel
- conda.yaml
- model.pkl
- requirements.txt

Full Path: dbfs:/databricks/mlflow-tracking/1515328694356734/a5fb02fe5b...

[Register Model](#)

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register it to the model registry](#) to version control and deploy as a REST endpoint for real time serving.

Model schema

Input and output schema for your model. [Learn more](#)

Name Type

Inputs (1)

Tensor (dtype:

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/a5fb02fe5b0547918728769ad8f9c503/model'

# Load model as a Spark UDF.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)
```



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

Experiments > /Users/ /mlflow_python_qs > Run a5fb02fe5b0547918728769ad8f9c503

Run a5fb02fe5b0547918728769ad8f9c503 [Provide Feedback](#)

Reproduce Run

Date: 2021-11-30 09:34:35

Source:  MLflowPythonQuickstart

User:

Duration: 3.4s

Status: FINISHED

Lifecycle Stage: active

▸ Description [Edit](#)

▸ Parameters (18)

▸ Metrics (5)

▸ Tags (2)

▾ Artifacts

model
MLmodel
conda.yaml
model.pkl
requirements.txt

Full Path: dbfs:/databricks/mlflow-tracking/1515328694356734/a5fb02fe5b...

Register Model

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also [register it to the model registry](#) to version control and deploy as a REST endpoint for [real time serving](#).

Model schema

Input and output schema for your model. [Learn more](#)

Name Type

Inputs (1)

Tensor (dtype:

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
logged_model = 'runs:/a5fb02fe5b0547918728769ad8f9c503/model'

# Load model as a Spark UDF.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)
```



Registered Models

Permissions

Share and serve machine learning models. [Learn more](#)

Create Model

All models

Owned by me

Accessible by me

Search by model names or tags

Search

Clear

Name	Latest Version	Staging	Production	Last Modified	Tags	Serving
feature_store_taxi_example_3017	Version 1	—	—	2022-01-26 08:51:08	—	—
feature_store_taxi_example_3921	Version 1	—	—	2022-01-26 08:41:13	—	—
feature_store_taxi_example_4051	Version 1	—	—	2022-01-26 08:45:00	—	—
feature_store_taxi_example_6589	Version 1	—	—	2022-01-26 08:52:42	—	—
recommender	Version 4	—	—	2022-04-01 10:50:14	—	—

Deployment

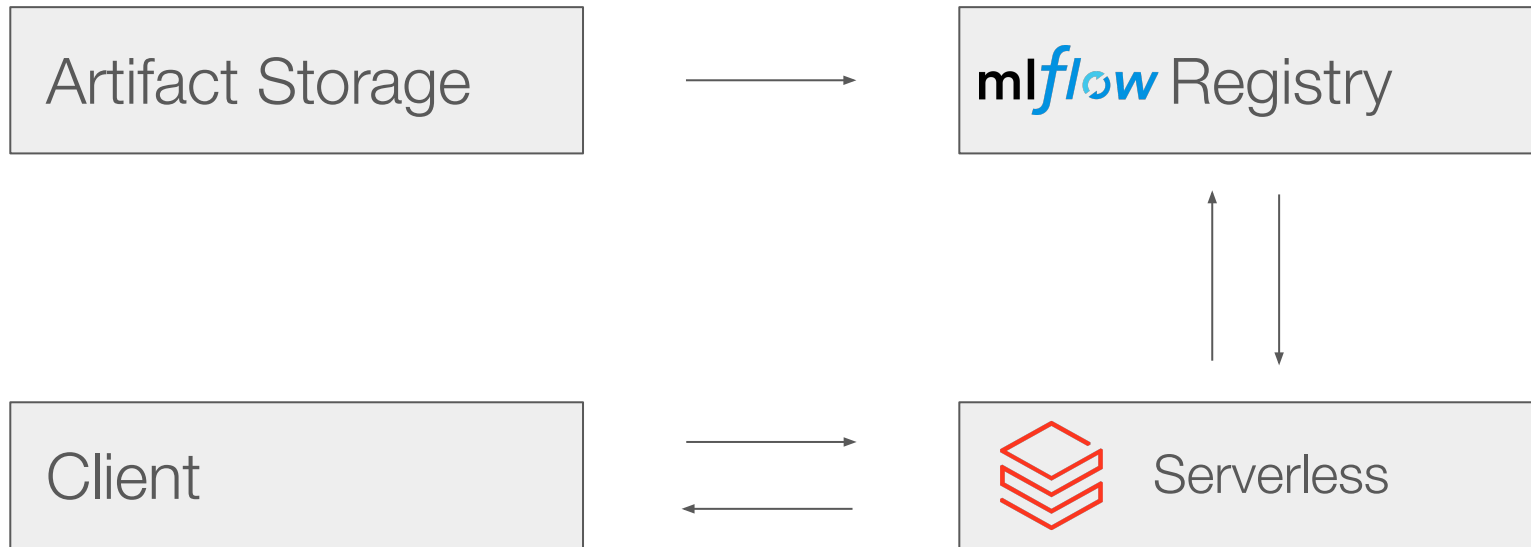


Batch Prediction

The diagram consists of two parallelogram shapes placed side-by-side. The left shape is solid blue and contains the text 'Batch Prediction'. The right shape is green with a thick yellow border and contains the text 'Online Prediction'.

Online Prediction

Model Serving



Model Serving

The screenshot shows the Databricks Model Serving interface. A modal dialog titled "Set up model inference" is open, allowing configuration for the "sds_demo_lr_model". The dialog has tabs for "Real-time", "Streaming (Delta Live Tables)", and "Batch inference". The "Real-time" tab is selected. It includes fields for "Model" (sds_demo_lr_model), "Model version" (selected from a dropdown), and "Endpoint name" (empty). Under "Compute", "Small" is selected with a note "0-4 concurrent requests (0-4 DBU)". The "Scale to zero" checkbox is checked. "Cancel" and "Create endpoint" buttons are at the bottom.



This screenshot shows the details for the "churn_prediction_test" endpoint. The "Endpoint state" is "Ready" with a URL: <https://e2-dogfood.staging.cloud.databricks.com/model-endpoint/charlie/invocations>. Below, a table lists "Served models":

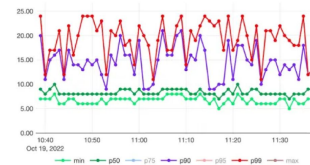
Model	Version	Name	State	Compute
churn_prediction	Version 1	churn-prediction-1	Ready	Large 16-64 (8 provisioned)

The "Call endpoint" tab is active, showing input fields for "Request" and "Response" with "Send Request" and "Show Examples" buttons.

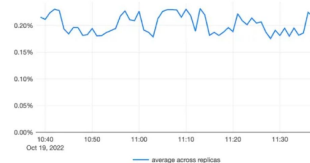
Metrics

2022/10/19 10:38 → 2022/10/19 11:38

Latency (ms)



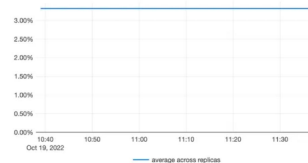
CPU Usage (%)



Request and error rates (per second)



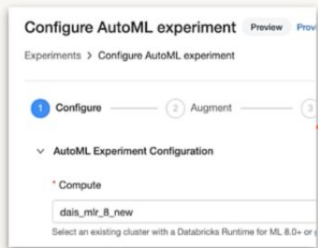
Memory Usage (%)



Databricks AutoML

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

UI and API to
start AutoML
training

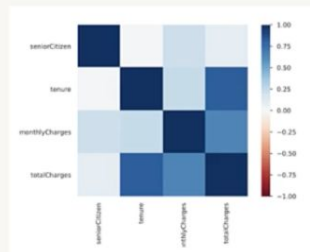


<input type="checkbox"/>	Start Time	Run Name	User	Source
<input type="checkbox"/>	2021-05-05 1	logistic_r...	kase...	Noteb...
<input type="checkbox"/>	2021-05-05 1	logistic_r...	alkis...	21-05...
<input type="checkbox"/>	2021-05-05 1	logistic_r...	alkis...	21-05...
<input type="checkbox"/>	2021-05-05 1	logistic_r...	kase...	Noteb...
<input type="checkbox"/>	2021-05-05 1	logistic_r...	kase...	Noteb...
<input type="checkbox"/>	2021-05-05 1	logistic_r...	kase...	Noteb...
<input type="checkbox"/>	2021-05-05 1	decision_...	kase...	Noteb...
<input type="checkbox"/>	2021-05-05 1	random_f...	kase...	Noteb...

MLflow experiment

Auto-created MLflow
Experiment to track models and
metrics

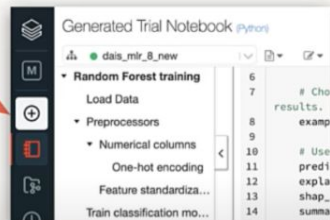
→ Easily deploy
to Model
Registry



Data exploration notebook

Generated notebook with
feature summary statistics and
distributions

→ Understand and
debug data
quality and
preprocessing



Reproducible trial notebooks

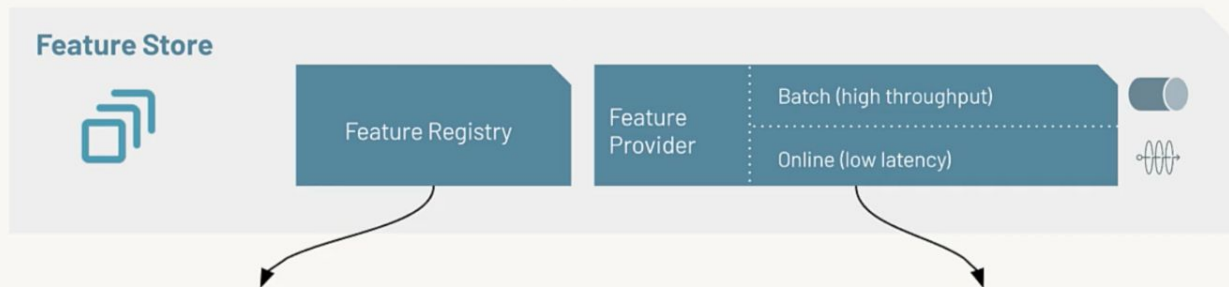
Generated notebooks with source
code for every model

→ Iterate further on
models from
AutoML, adding
your expertise



Feature Store

The first Feature Store codedesigned 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



- 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



