SPARK+AI
SUMMIT 2019

WIFI SSID:Spark+AISummit | Password: UnifiedDataAnalytics

End-to-End ML Pipelines with Databricks Delta and Hopsworks Feature Store

Kim Hammar, Logical Clocks AB
Jim Dowling, Logical Clocks AB

KimHammar1
jim_dowling

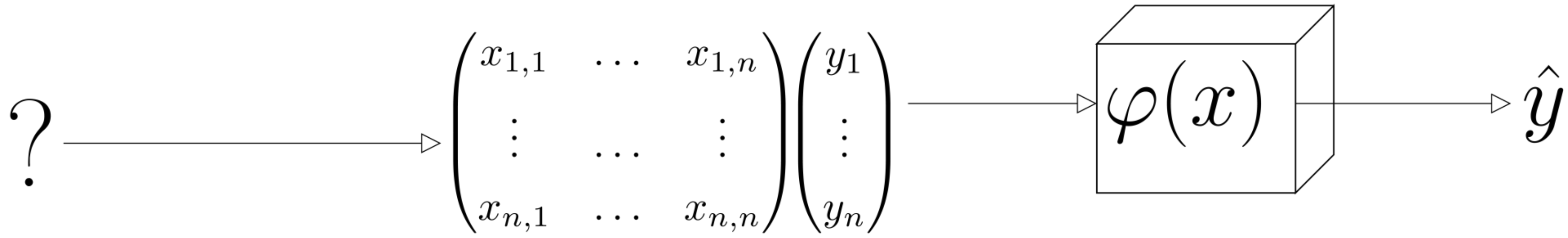
#UnifiedDataAnalytics #SparkAISummit

SPARK+AI SUMMIT 2019

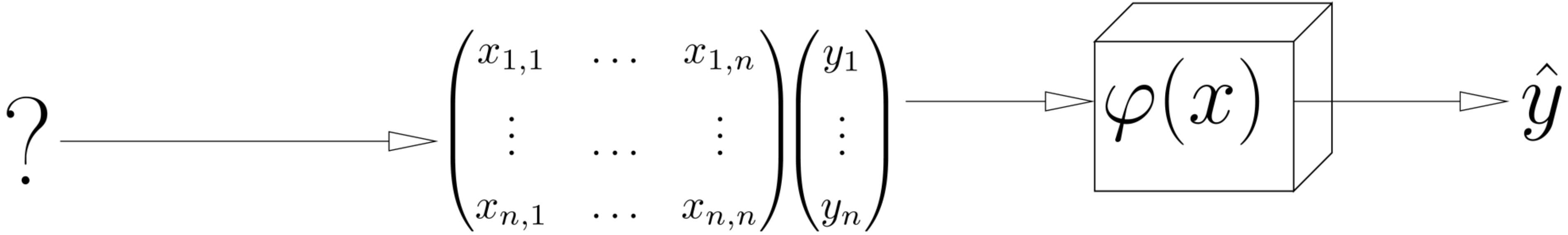
Machine Learning in the Abstract

$$\begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \longrightarrow \boxed{\varphi(x)} \longrightarrow \hat{y}$$

Where does the Data come from?

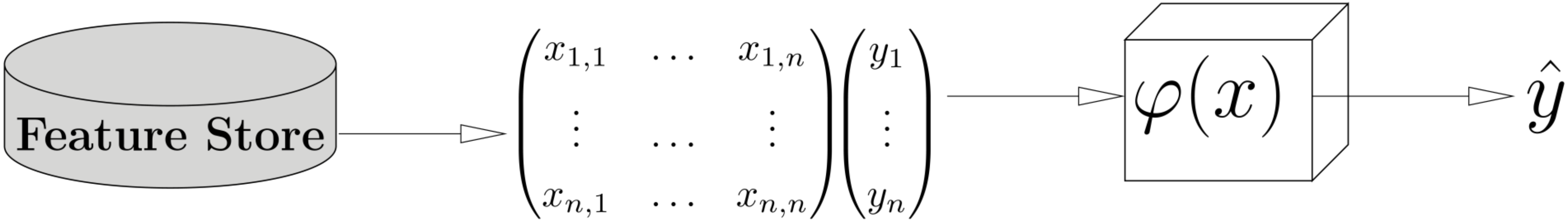


Where does the Data come from?

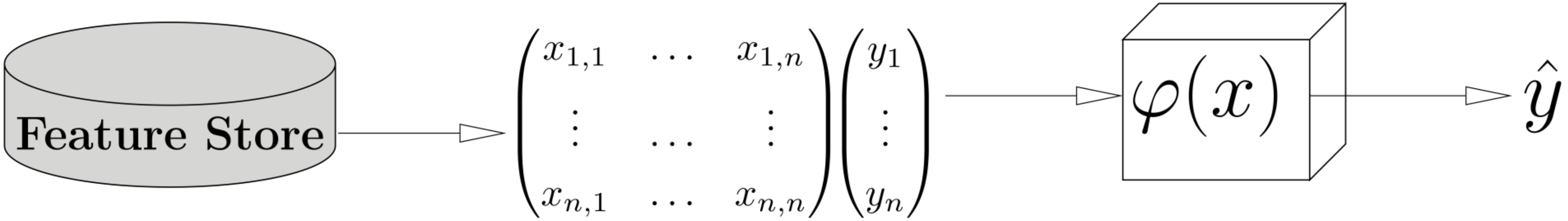


"Data is the hardest part of ML and the most important piece to get right. Modelers spend most of their time selecting and transforming features at training time and then building the pipelines to deliver those features to production models." [Uber on Michelangelo]

Data comes from the Feature Store



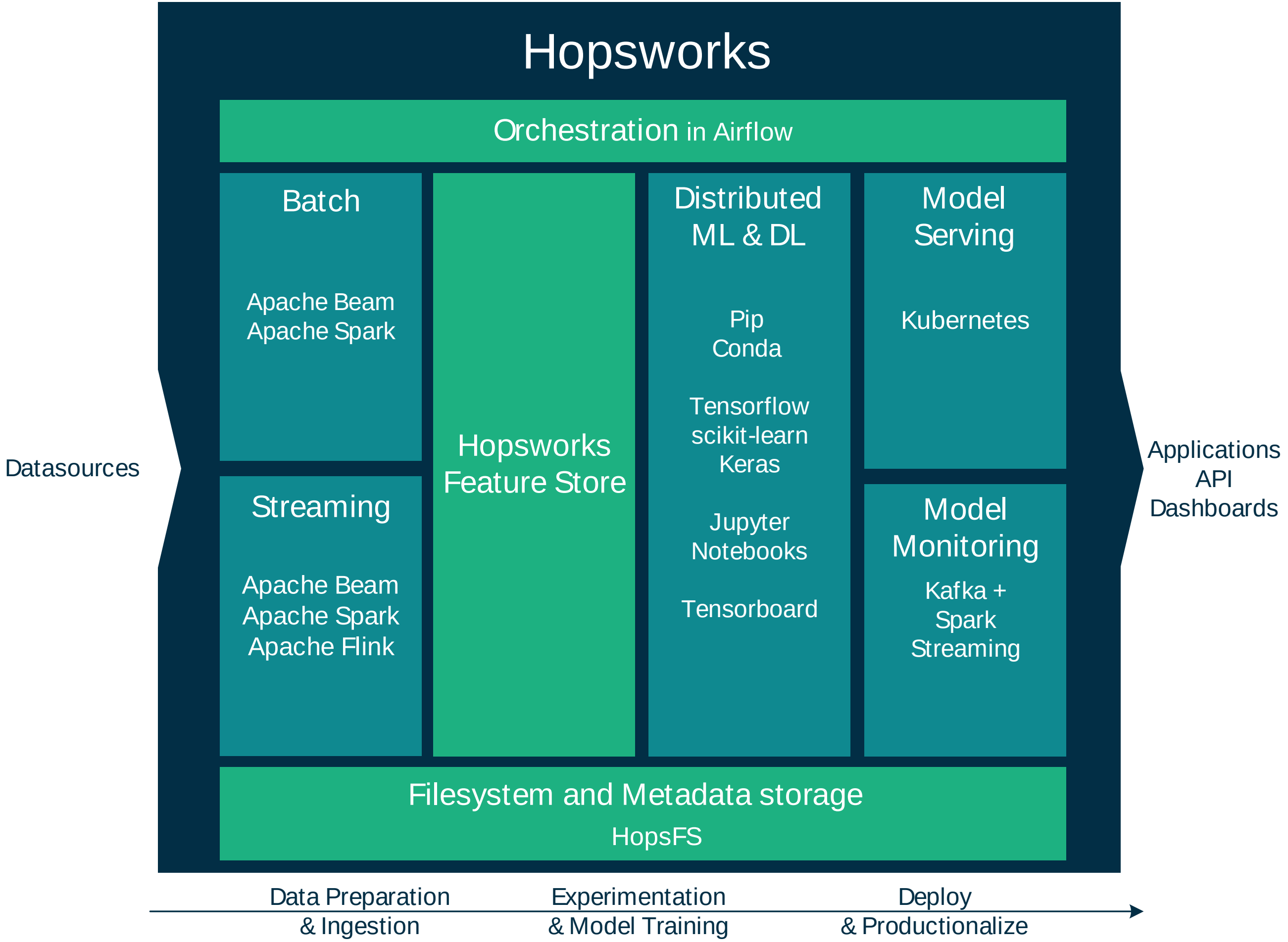
How do we feed the Feature Store?



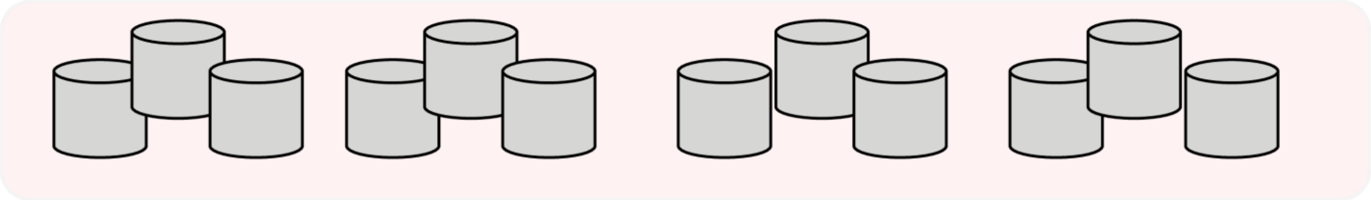
DELTA LAKE
Apache hudi
ICEBERG

Outline

1. Hopsworks
2. Databricks Delta
3. Hopsworks Feature Store
4. Demo
5. Summary

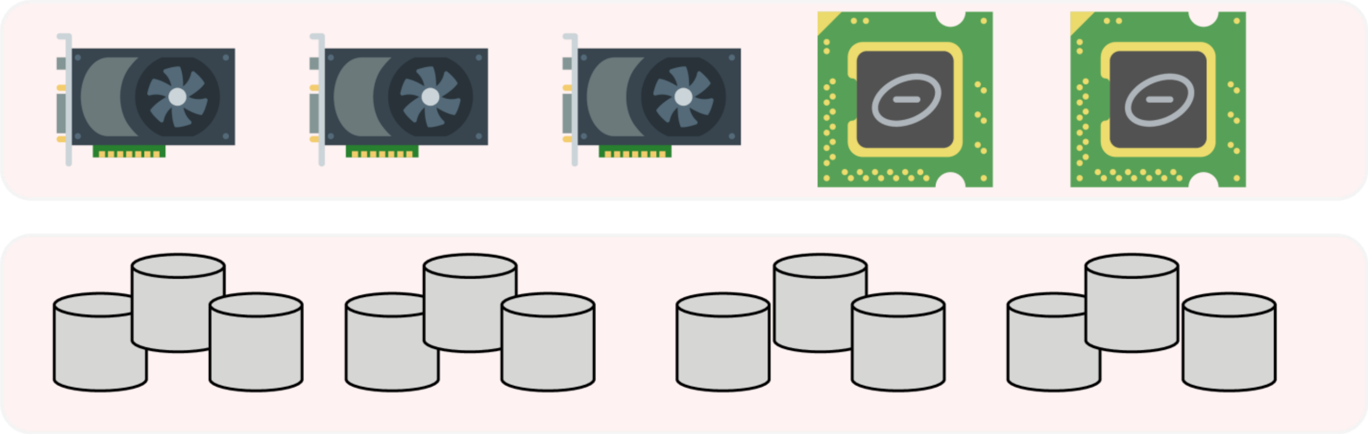

Hopsworks
Orchestration in Airflow
Batch
Apache Beam
Apache Spark
Streaming
Apache Beam
Apache Spark
Apache Flink
Hopsworks
Feature Store
Distributed
ML & DL
Pip
Conda
Tensorflow
scikit-learn
Keras
Jupyter
Notebooks
Tensorboard
Model
Serving
Kubernetes
Model
Monitoring
Kafka +
Spark
Streaming
Filesystem and Metadata storage
HopsFS
Datasources
Applications
API
Dashboards
Data Preparation
& Ingestion
Experimentation
& Model Training
Deploy
& Productionalize

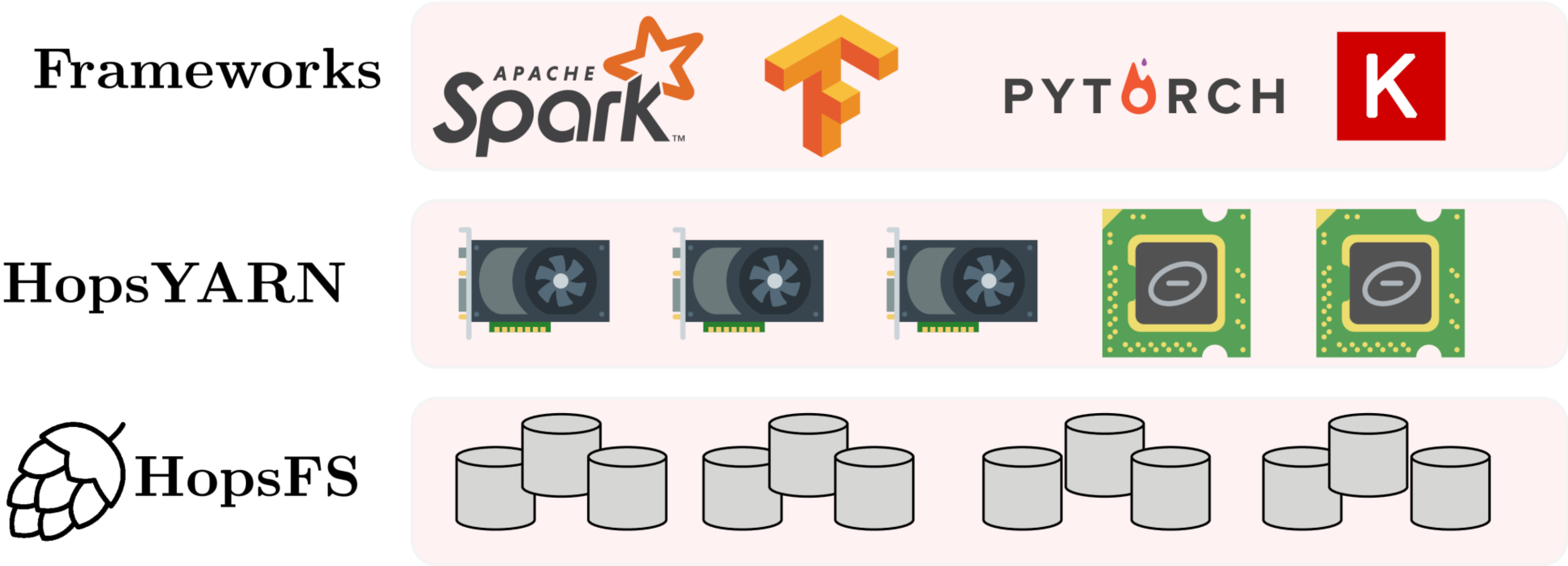
HopsFS

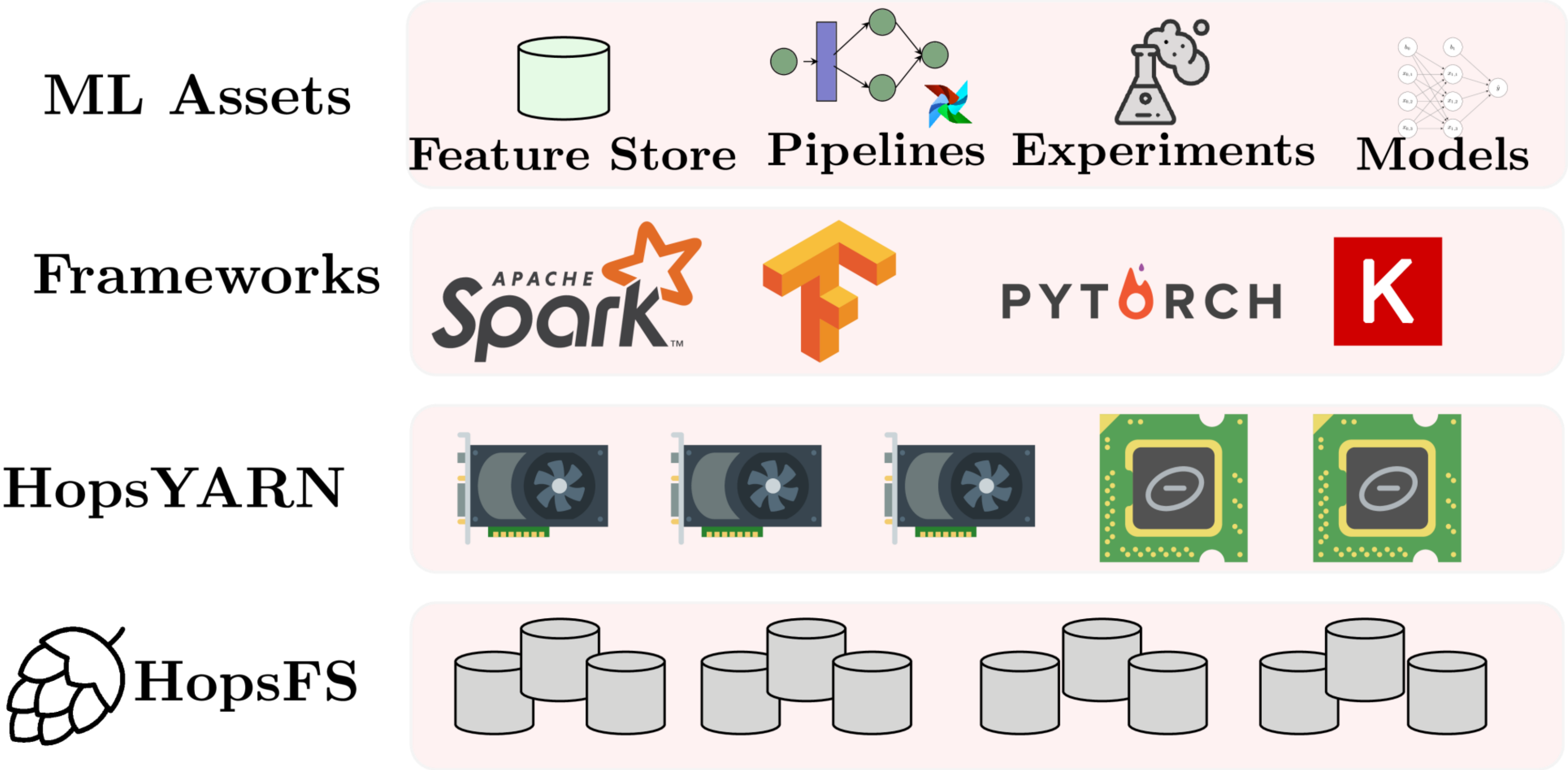


HopsYARN

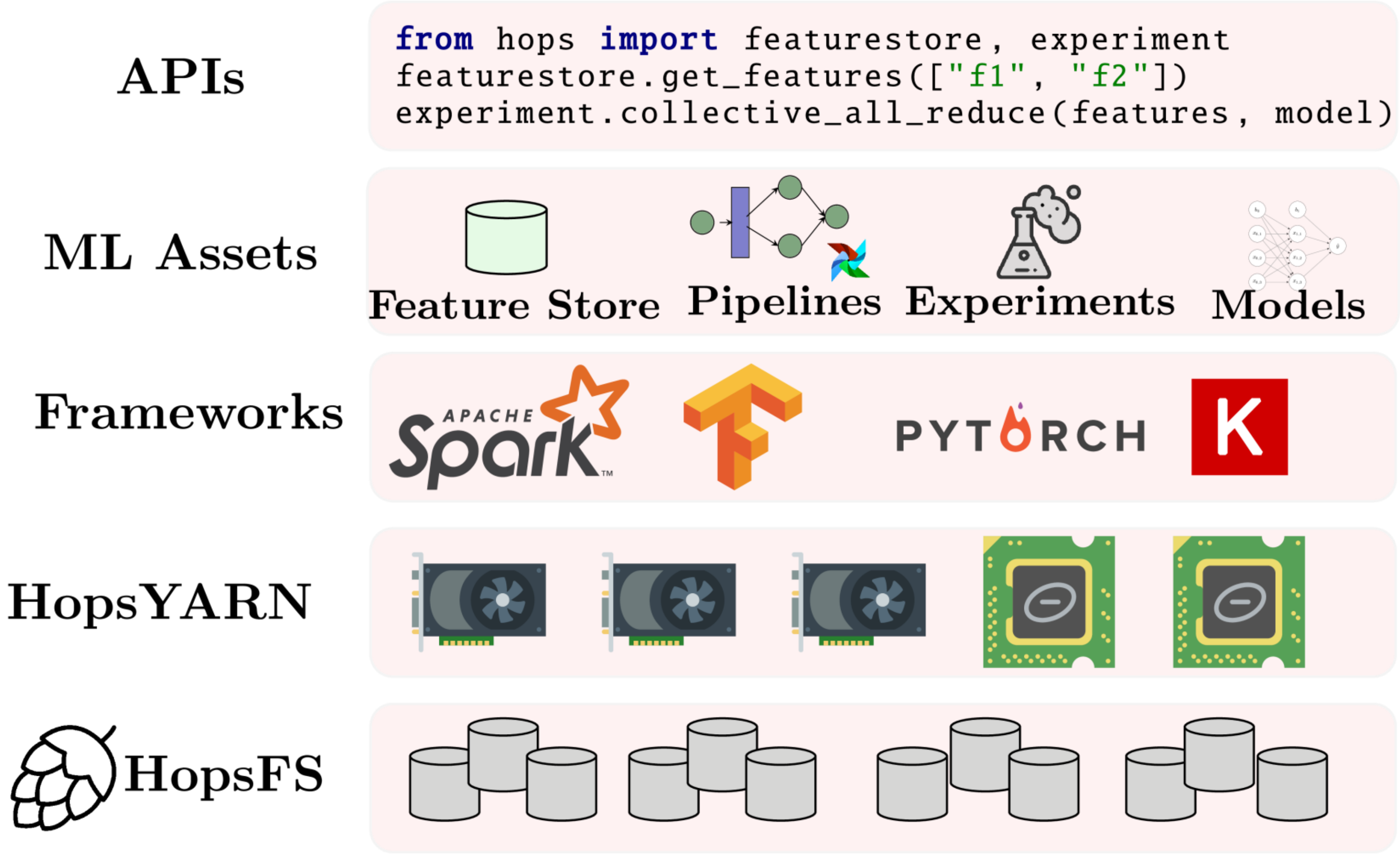
HopsFS

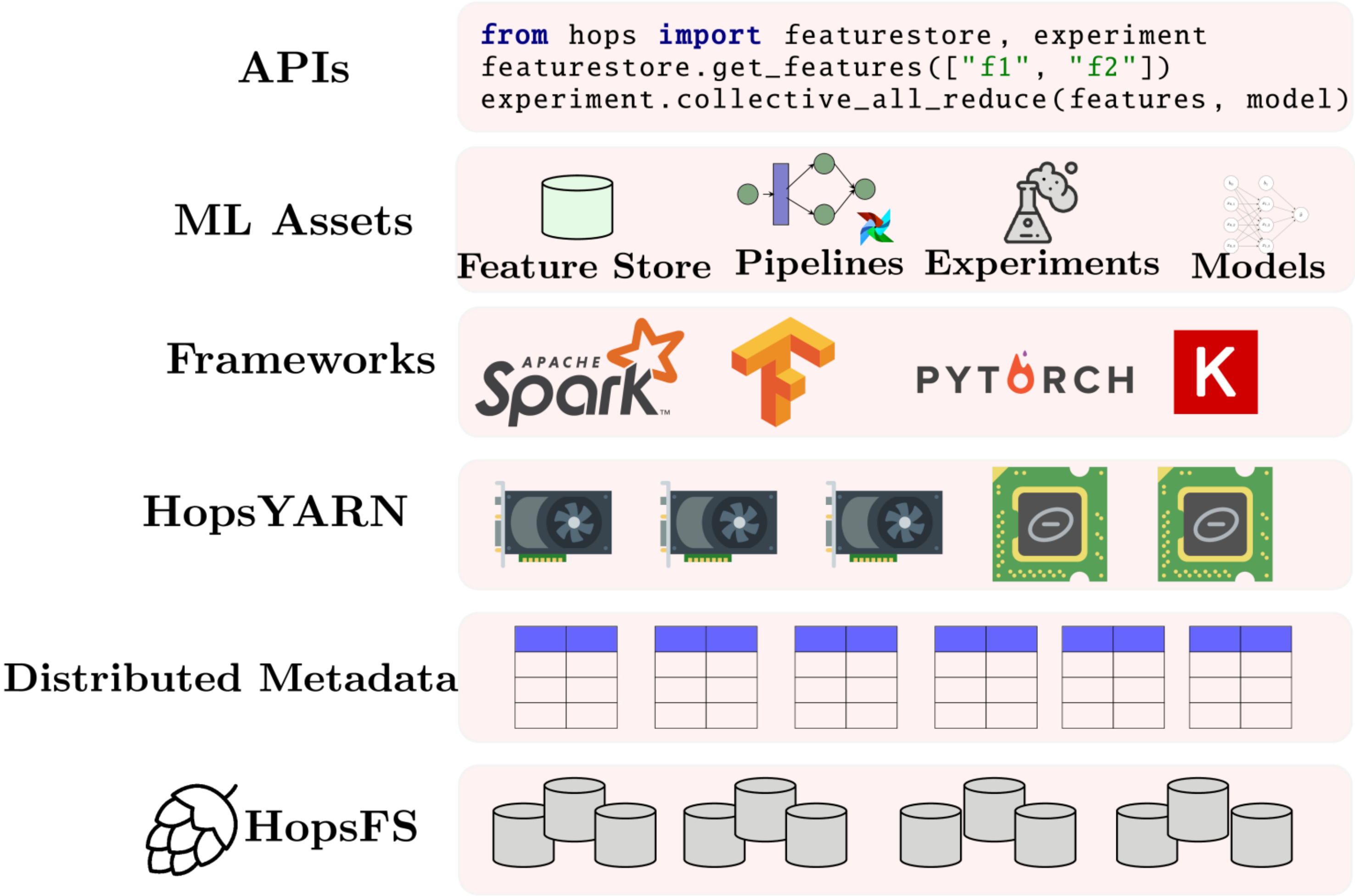



Frameworks
APACHE Spark
PYTORCH
K
HopsYARN
HopsFS



ML Assets
Feature Store
Pipelines
Experiments
Models
Frameworks
APACHE Spark
PYTORCH
K
HopsYARN
HopsFS


APIs
from hops import featurestore, experiment
featurestore.get_features(["f1", "f2"])
experiment.collective_all_reduce(features, model)
ML Assets
Feature Store
Pipelines
Experiments
Models
Frameworks
APACHE Spark
PYTORCH
K
HopsYARN
HopsFS

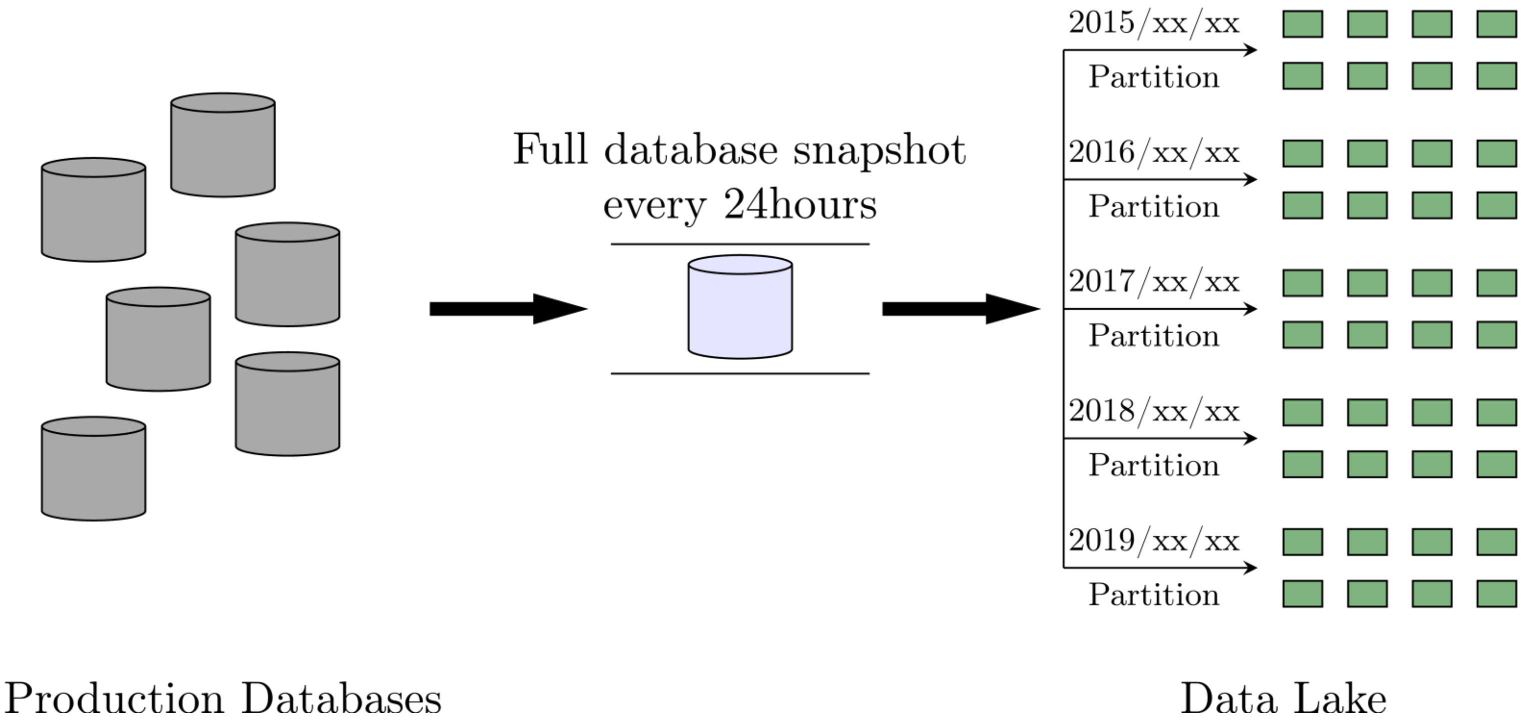

APIs
from hops import featurestore, experiment
featurestore.get_features(["f1", "f2"])
experiment.collective_all_reduce(features, model)
ML Assets
Feature Store
Pipelines
Experiments
Models
Frameworks
APACHE Spark
PYTORCH
K
HopsYARN
Distributed Metadata
HopsFS

Next-Gen Data Lakes

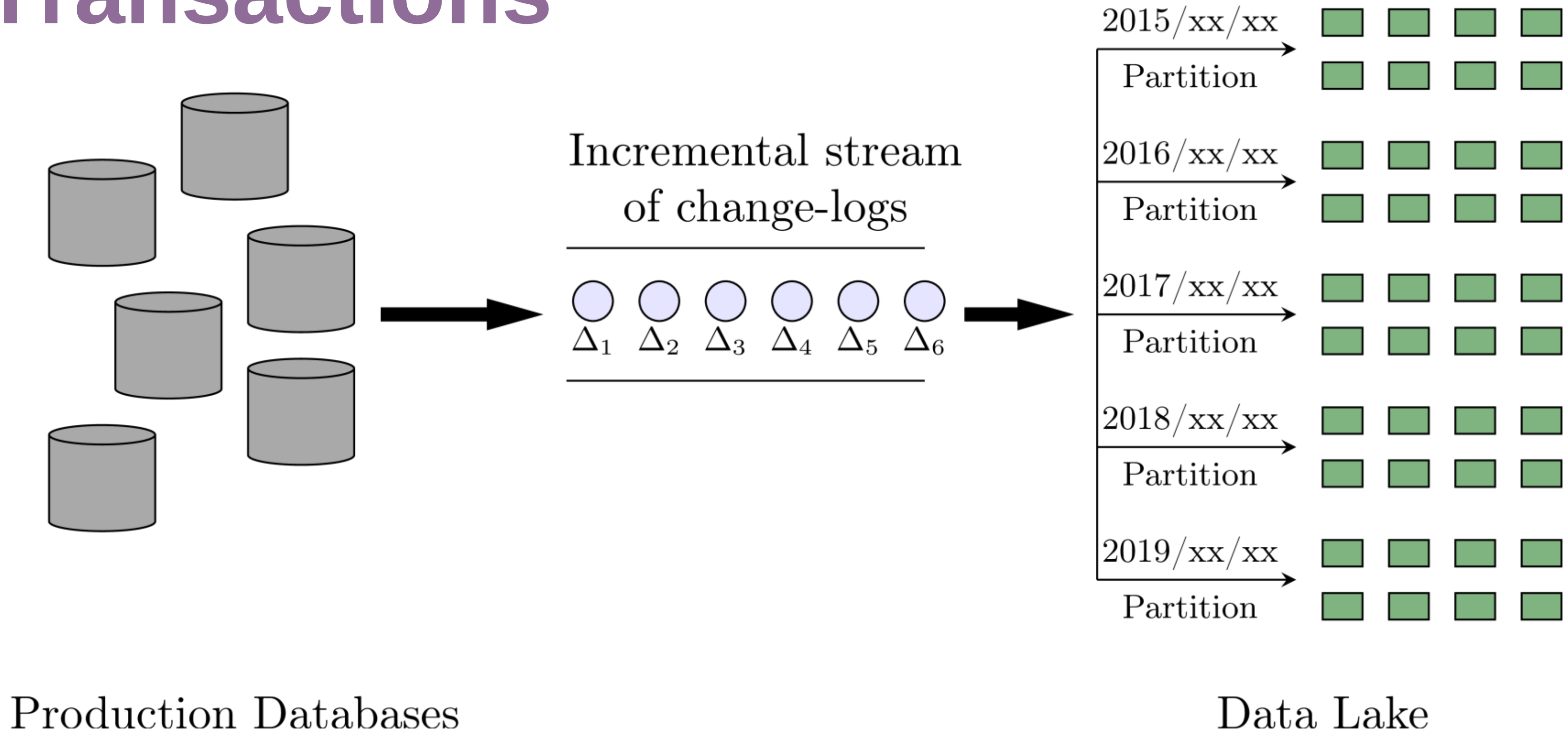
Data Lakes are starting to resemble databases:

- Apache Hudi, Delta, and Apache Iceberg add:
    - ACID transactional layers on top of the data lake
    - Indexes to speed up queries (data skipping)
    - Incremental Ingestion (late data, delete existing records)
    - Time-travel queries

Problems: No Incremental Updates, No rollback on failure, No Time-Travel, No Isolation.



Solution: Incremental ETL with ACID Transactions



Upsert & Time Travel Example

Upsert & Time Travel Example

Upsert ==Insert or Update

Mini-batches of changelogs

$$\begin{array}{c} \text{Mini batch 1} \\ time = t_1 \end{array} \begin{cases} \Delta_1 \langle id = 1, a = 1, b = 7, c = 3 \rangle \text{ insert} \\ \Delta_2 \langle id = 2, a = 3, b = 9, c = 0 \rangle \text{ insert} \end{cases}$$

$$\begin{array}{c} \text{Mini batch 2} \\ time = t_2 \end{array} \begin{cases} \Delta_3 \langle id = 3, a = 3, b = 0, c = 0 \rangle \text{ insert} \\ \Delta_4 \langle id = 1, a = 0, b = 9, c = 0 \rangle \text{ update} \end{cases}$$

Version Data By Commits

Mini-batches of changelogs

Mini batch 1
$time = t_1$

$$\begin{cases} \Delta_1\langle id = 1, a = 1, b = 7, c = 3\rangle \text{ insert} \\ \Delta_2\langle id = 2, a = 3, b = 9, c = 0\rangle \text{ insert} \end{cases}$$

Mini batch 2
$time = t_2$

$$\begin{cases} \Delta_3\langle id = 3, a = 3, b = 0, c = 0\rangle \text{ insert} \\ \Delta_4\langle id = 1, a = 0, b = 9, c = 0\rangle \text{ update} \end{cases}$$

Table Views

Table at commit time t_1

Id	a	b	c
1	1	7	3
2	3	9	0

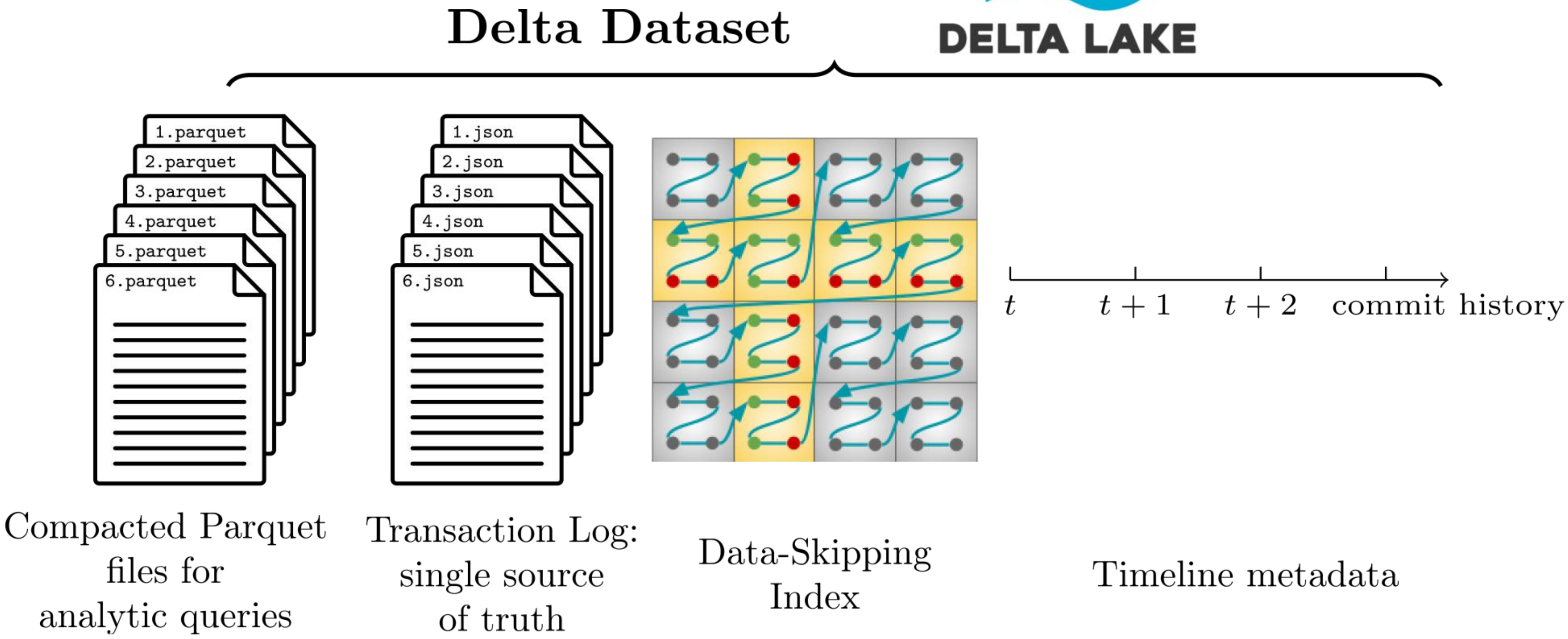
Table at commit time t_2

Id	a	b	c
1	$\cancel{1}$ 0	$\cancel{7}$ 9	$\cancel{3}$ 0
2	3	9	0
3	3	0	0

Delta Lake by Databricks

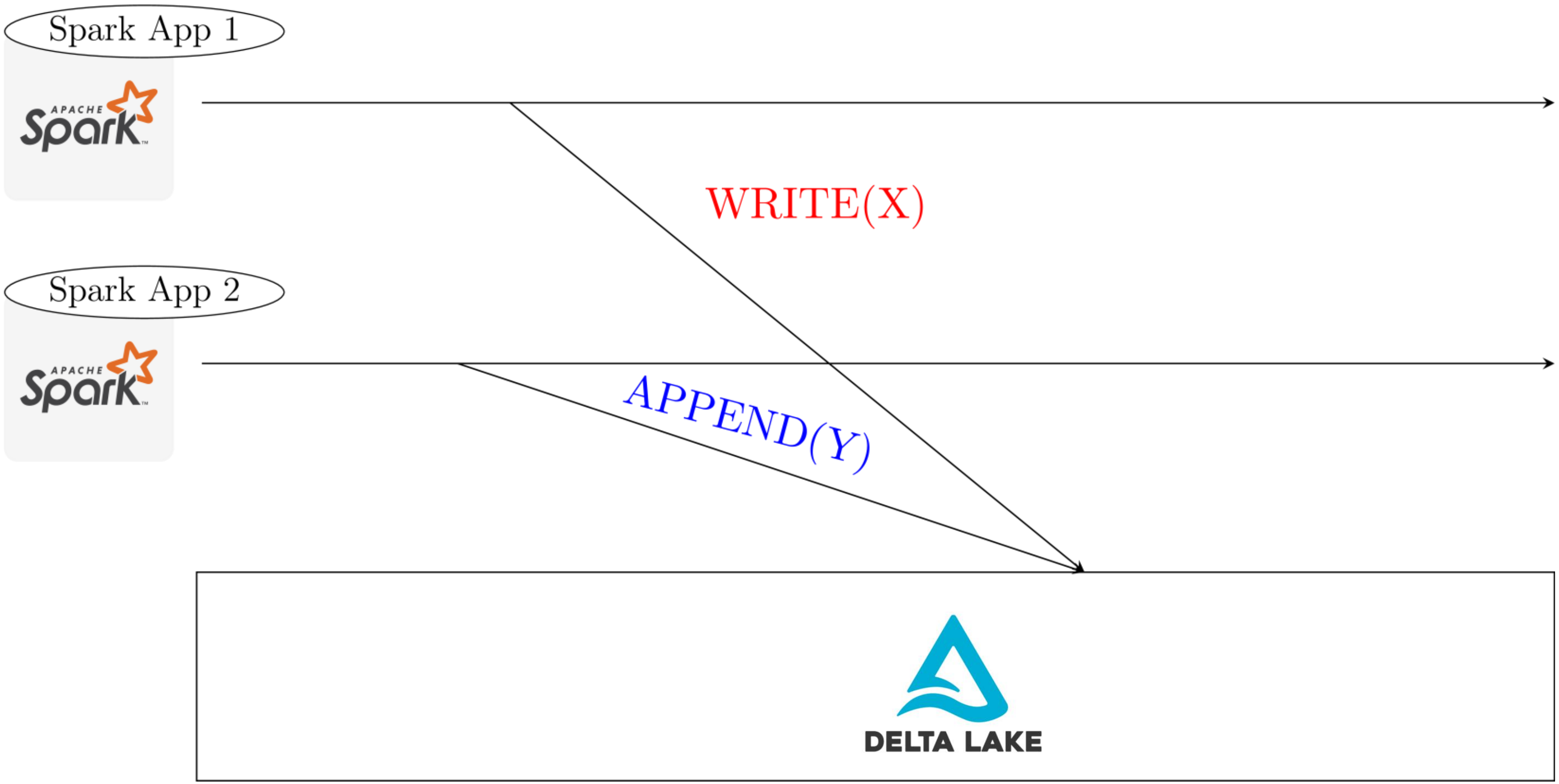
- Delta Lake is a Transactional Layer that sits on top of your Data Lake:
  - ACID Transactions with Optimistic Concurrency Control
  - Log-Structured Storage
  - Open Format (Parquet-based storage)
  - Time-travel

Delta Datasets

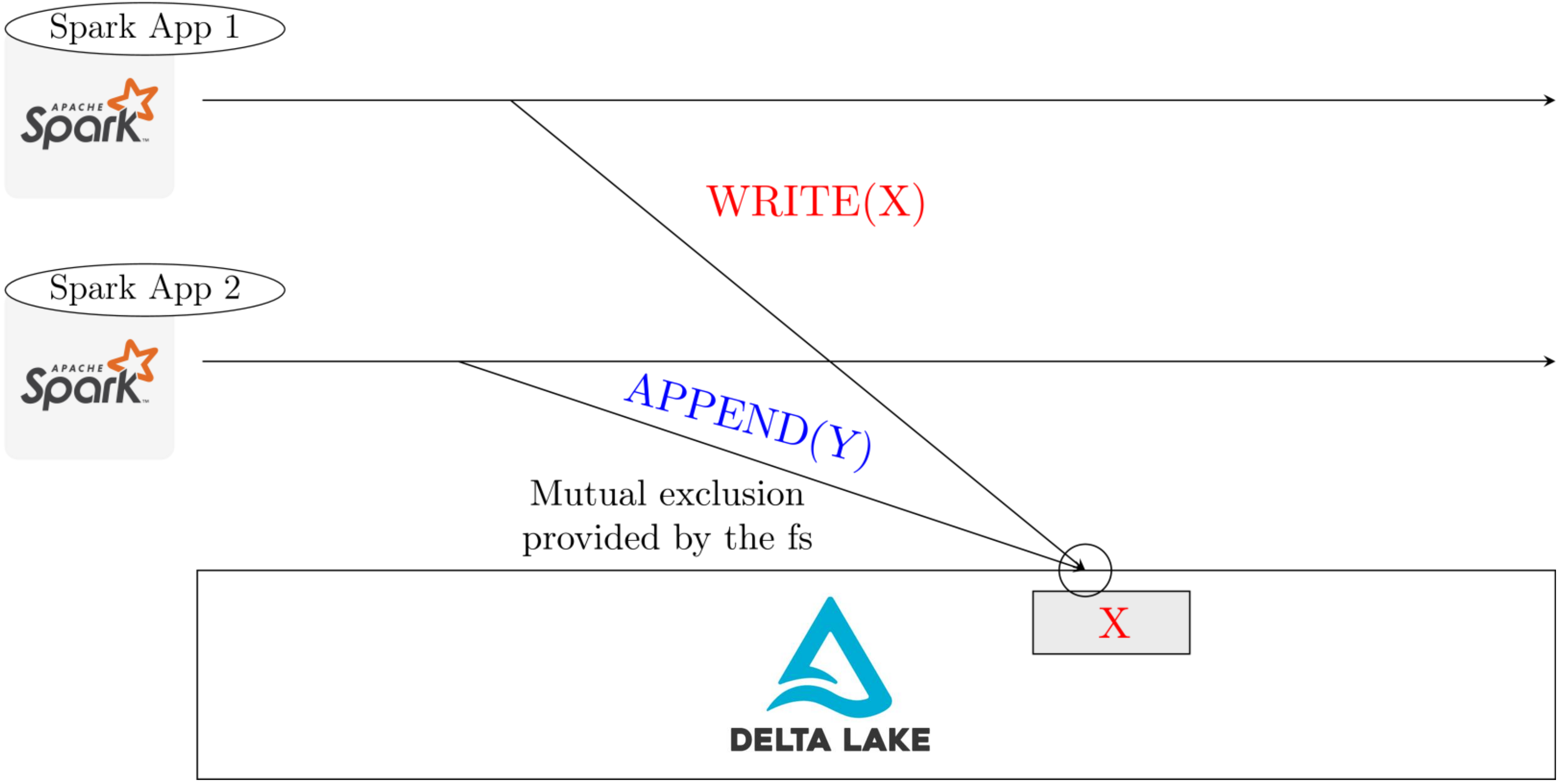


Optimistic Concurrency Control

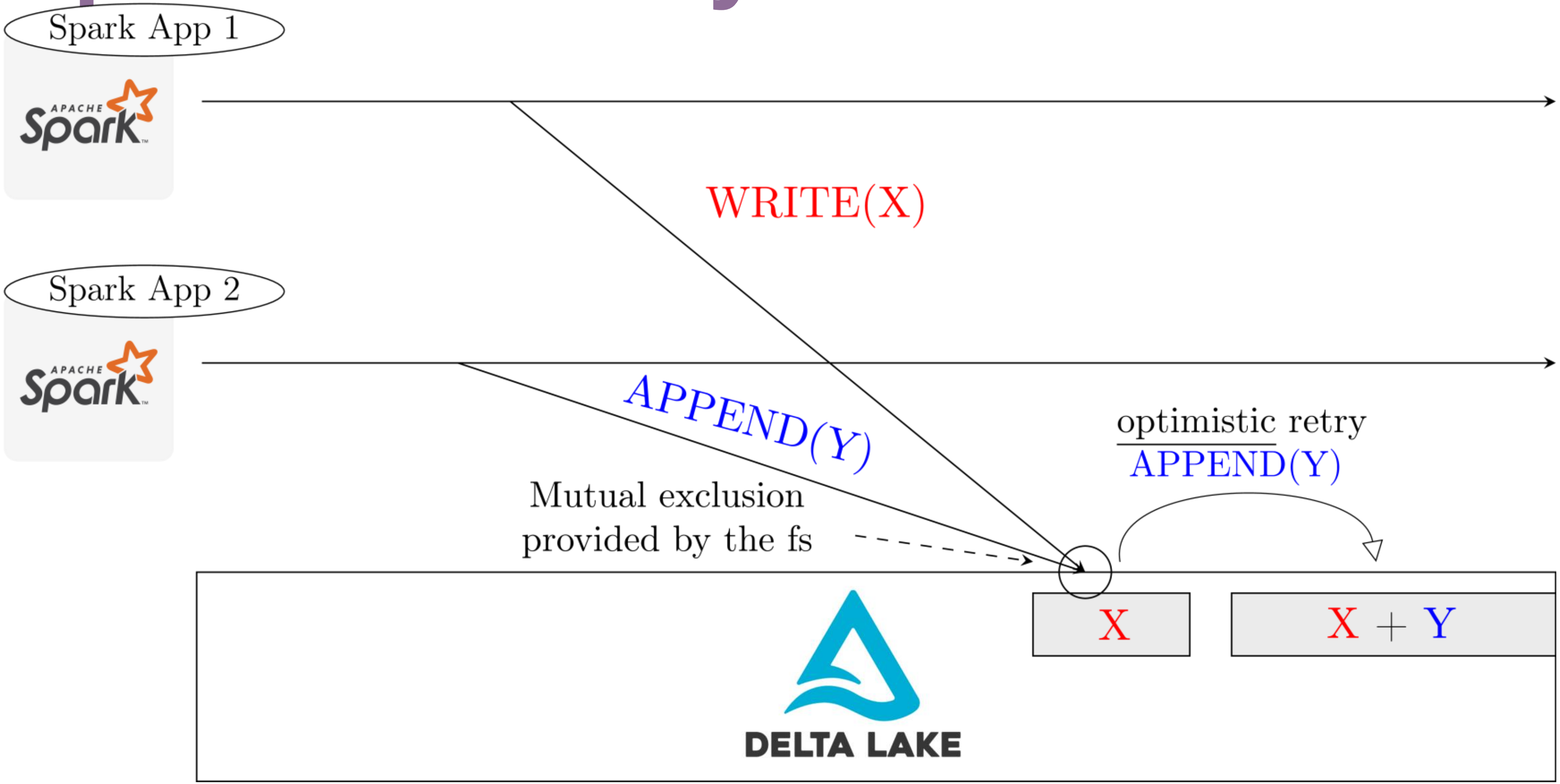
Optimistic Concurrency Control



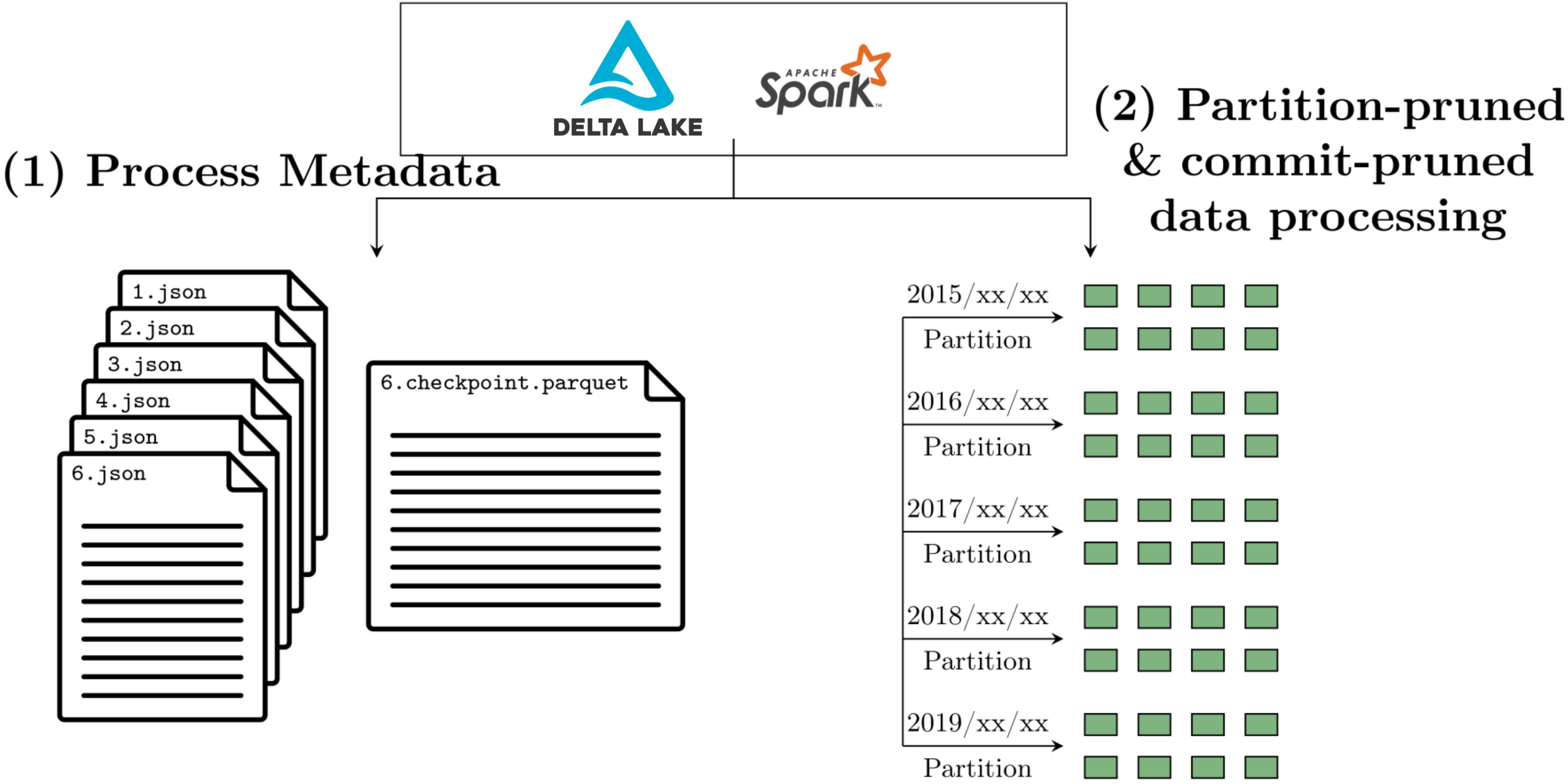
Mutual Exclusion for Writers



Optimistic Retry



Scalable Metadata Management



Other Frameworks: Apache Hudi, Apache Iceberg

- Hudi was developed by Uber for their Hadoop Data Lake (HDFS first, then S3 support)
- Iceberg was developed by Netflix with S3 as target storage layer
- All three frameworks (Delta, Hudi, Iceberg) have common goals of adding ACID updates, incremental ingestion, efficient queries.

Next-Gen Data Lakes Compared

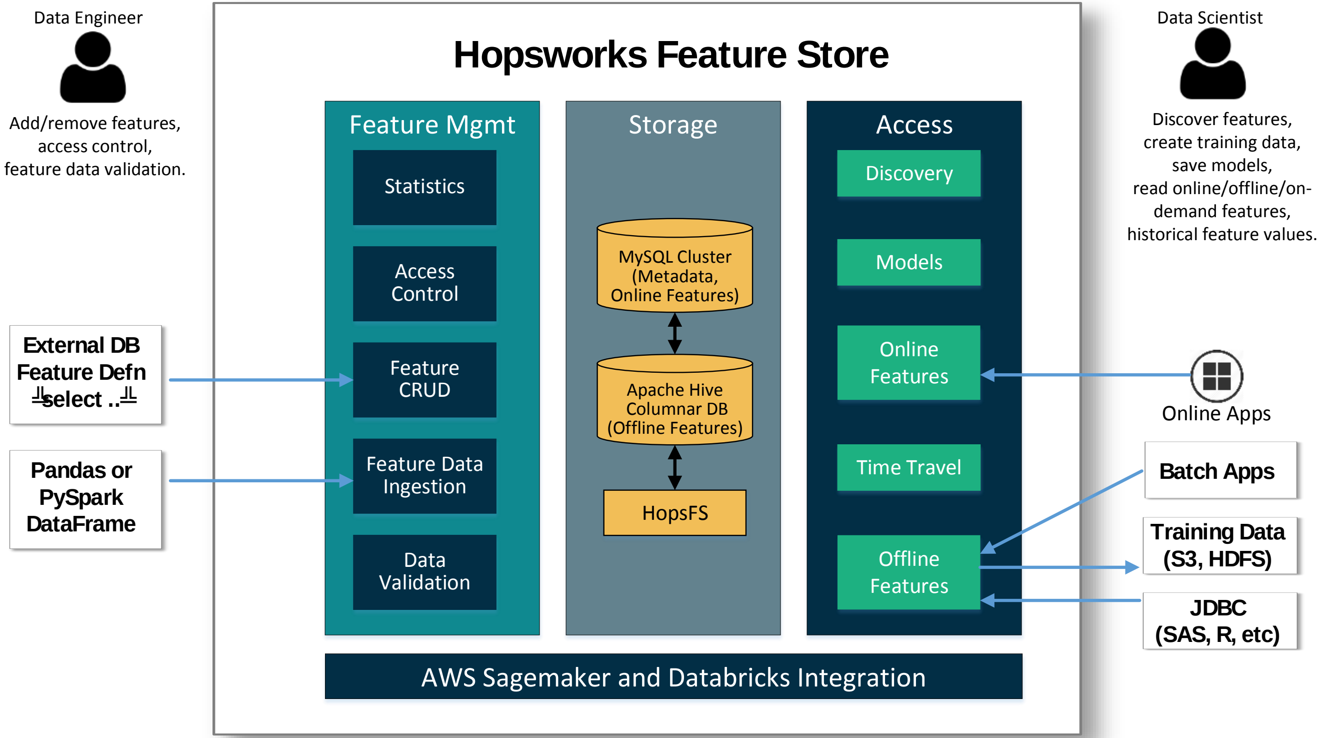
	Delta	Hudi	Iceberg
Incremental Ingestion	Spark	Spark	Spark
ACID updates	HDFS, S3*	HDFS	S3, HDFS
File Formats	Parquet	Avro, Parquet	Parquet, ORC
Data Skipping (File-Level Indexes)	Min-Max Stats+Z-Order Clustering*	File-Level Max-Min stats + Bloom Filter	File-Level Max-Min Filtering
Concurrency Control	Optimistic	Optimistic	Optimistic
Data Validation	Expectations (coming soon)	In Hopsworks	N/A
Merge-on-Read	No	Yes (coming soon)	No
Schema Evolution	Yes	Yes	Yes
File I/O Cache	Yes*	No	No
Cleanup	Manual	Automatic, Manual	No
Compaction	Manual	Automatic	No

***Databricks version only (not open-source)**

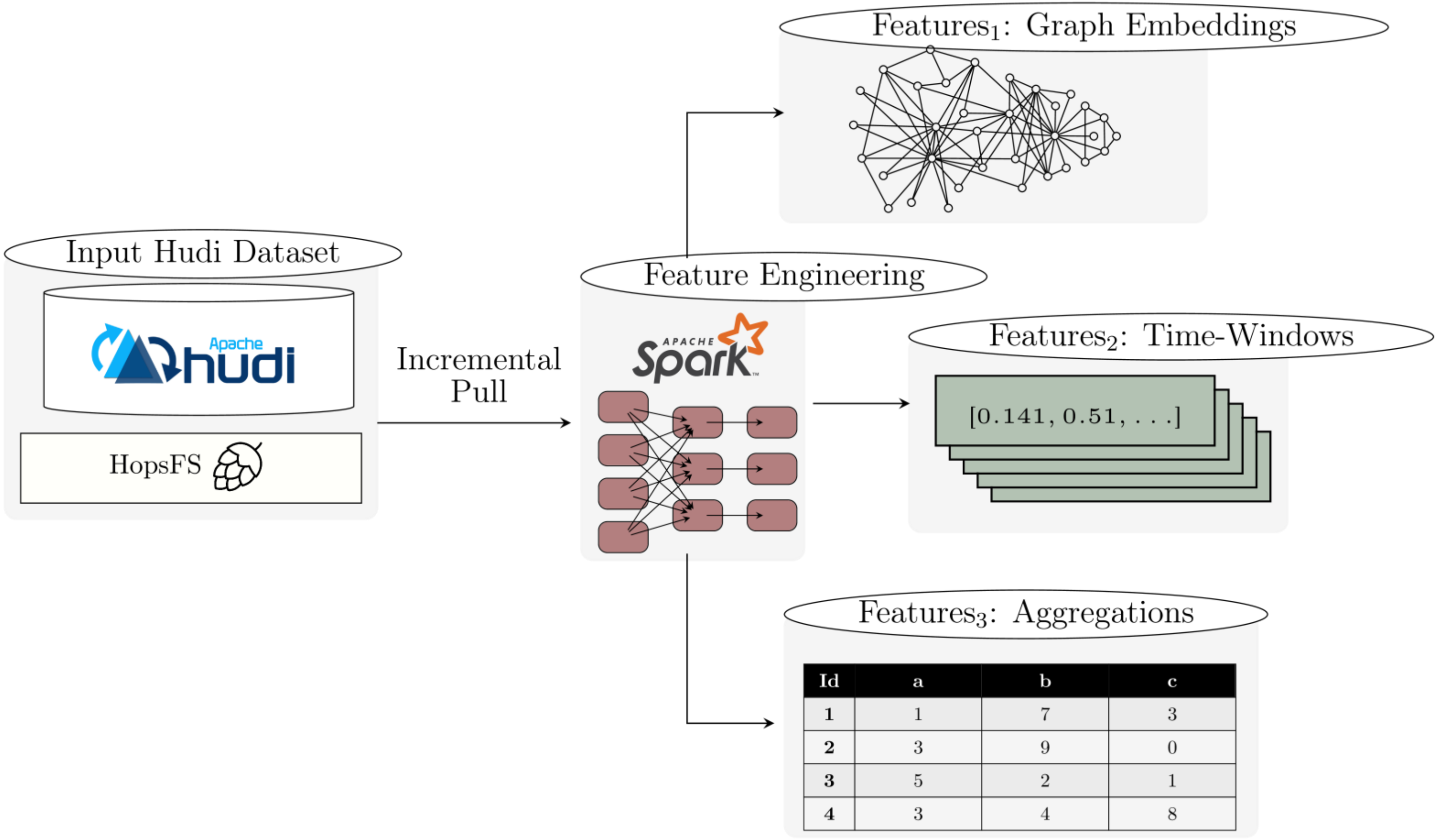
How can a Feature Store leverage Log-Structured Storage (e.g., Delta or Hudi or Iceberg)?

Hopsworks Feature Store

- Computation engine (Spark)
- Incremental ACID Ingestion
- Time-Travel
- Data Validation
- On-Demand or Cached Features
- Online or Offline Features

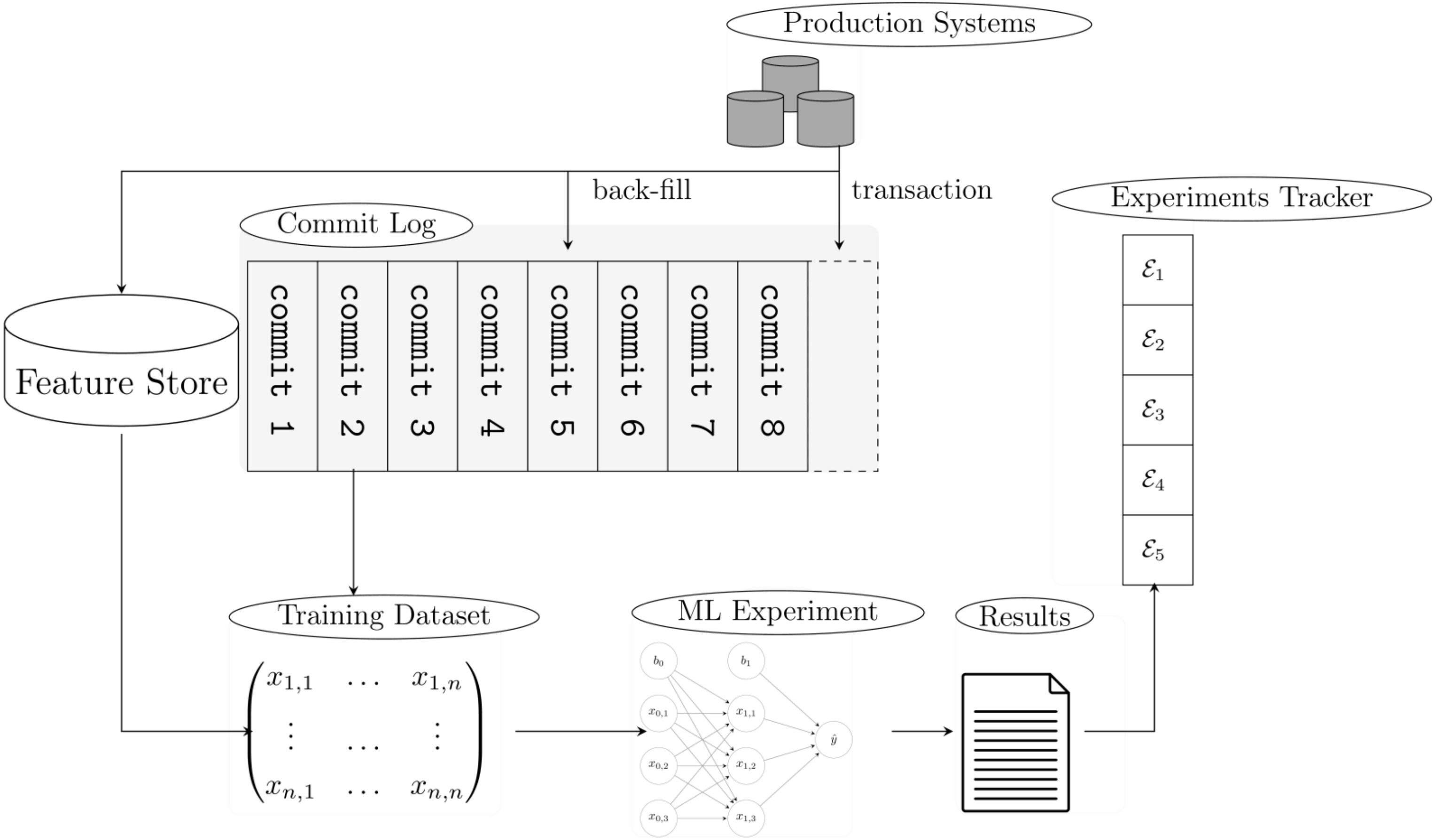


Incremental Feature Engineering with Hudi



Id	a	b	c
1	1	7	3
2	3	9	0
3	5	2	1
4	3	4	8

Point-in-Time Correct Feature Data

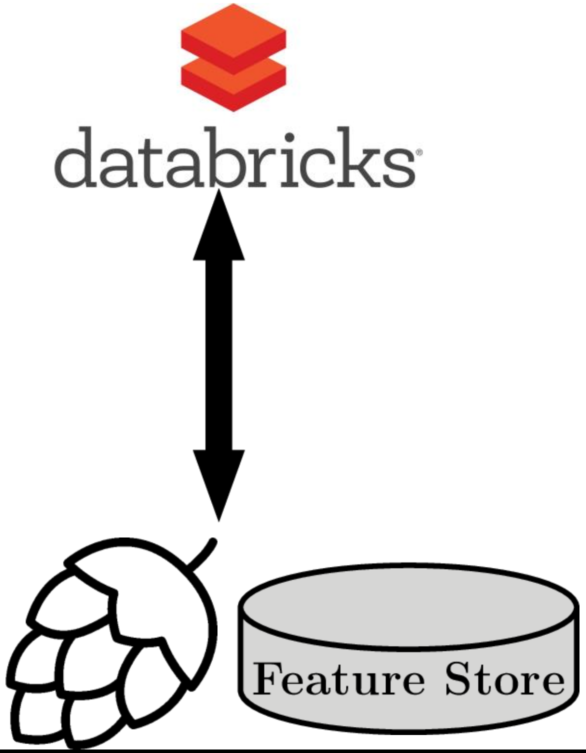


Feature Time Travel with Hudi and Hopsworks Feature Store

```
import io.hops.util.Hops
Hops.createFeaturegroup("trx_features")
                   .setHudi(true) //or setDelta(true)
                   .setPartitionBy(partitionCols)
                   .setDataframe(sparkDf1)
                   .setPrimaryKey("id").write

val sparkDf2 = Hops.getFeaturegroup("trx_features")
                   .read
                   .option("commitTime", commitTime)
```

Demo: Hopsworks Featurestore + Databricks Platform



Summary

- Delta, Hudi, Iceberg bring **Reliability**, **Upserts** & **Time-Travel** to Data Lakes
  - Functionalities that are well suited for Feature Stores

- **Hopsworks Feature Store** builds on Hudi/Hive and is the world’s first open-source Feature Store (released 2018)

- **The Hopsworks Platform** also supports End-to-End ML pipelines using the Feature Store and Spark/Beam/Flink, Tensorflow/PyTorch, and Airflow

Thank you!

470 Ramona St, Palo Alto

Kista, Stockholm

https://www.logicalclocks.com

Register for a free account at

www.hops.site

Twitter

@logicalclocks

@hopsworks

GitHub

https://github.com/logicalclocks/hopsworks

https://github.com/hopshadoop/hops

References

- Feature Store: the missing data layer in ML pipelines? https://www.logicalclocks.com/feature-store/
- Python-First ML Pipelines with Hopsworks https://hops.readthedocs.io/en/latest/hopsml/hopsML.html.
- Hopsworks white paper. https://www.logicalclocks.com/whitepapers/hopsworks
- HopsFS: Scaling Hierarchical File System Metadata Using NewSQL Databases. https://www.usenix.org/conference/fast17/technical-sessions/presentation/niazi
- Open Source:
  https://github.com/logicalclocks/hopsworks
  https://github.com/hopshadoop/hops
- Thanks to Logical Clocks Team: Jim Dowling, Seif Haridi, Theo Kakantousis, Fabio Buso, Gautier Berthou, Ermias Gebremeskel, Mahmoud Ismail, Salman Niazi, Antonios Kouzoupis, Robin Andersson, Alex Ormenisan, Rasmus Toivonen, Steffen Grohsschmiedt, and Moritz Meister

SPARK+AI SUMMIT 2019

DON'T FORGET TO RATE AND REVIEW THE SESSIONS

SEARCH SPARK + AI SUMMIT

Download on the App Store

GET IT ON Google Play

SPARK+AI SUMMIT 2019