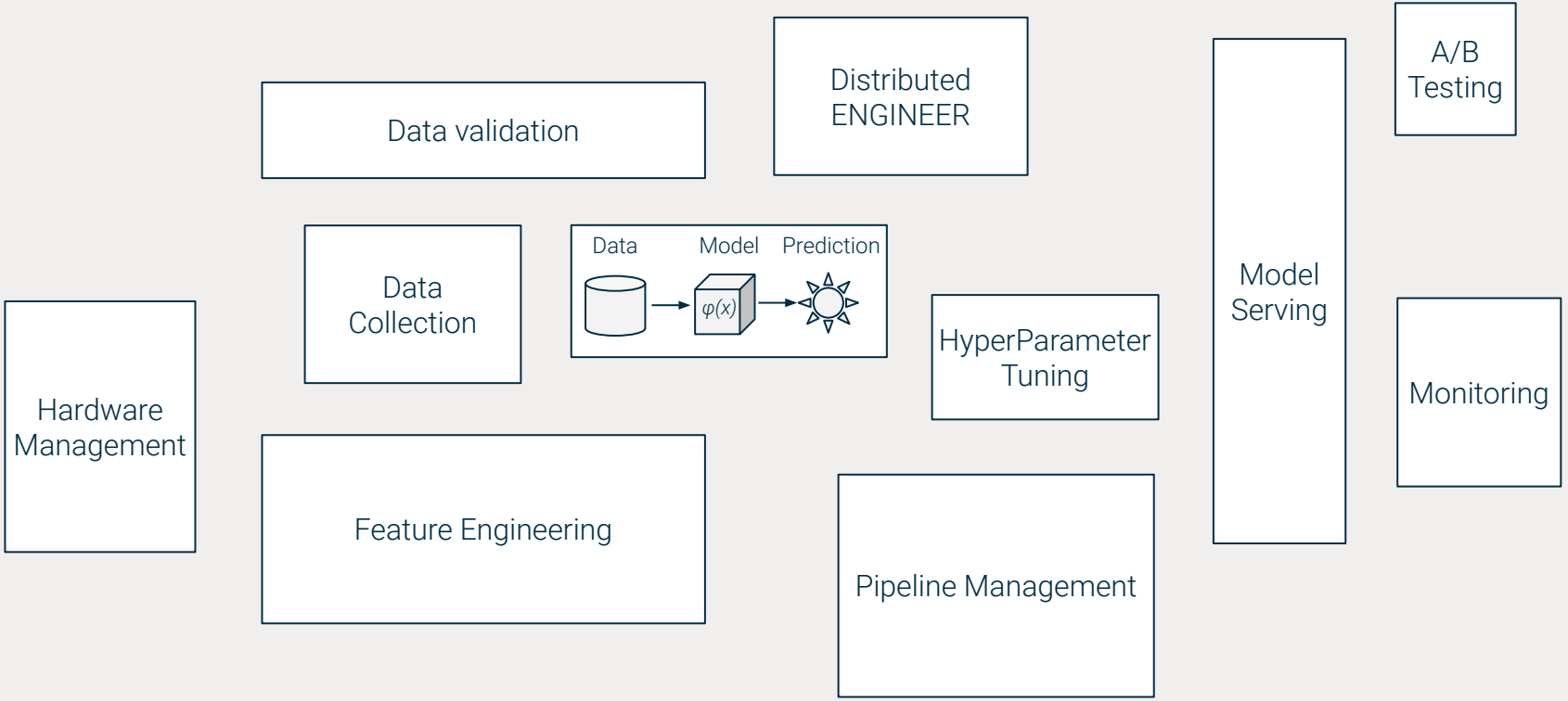
HOPSWORKS

KTH Lecture for ID2223

Dr. Jim Dowling[1,2]
Slides together with Alexandru A. Ormenisan[1,2], Mahmoud Ismail[1,2]

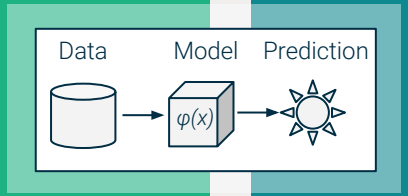
KTH - Royal Institute of Technology (1)
Logical Clocks AB (2)

Growing Consensus on how to manage complexity of AI



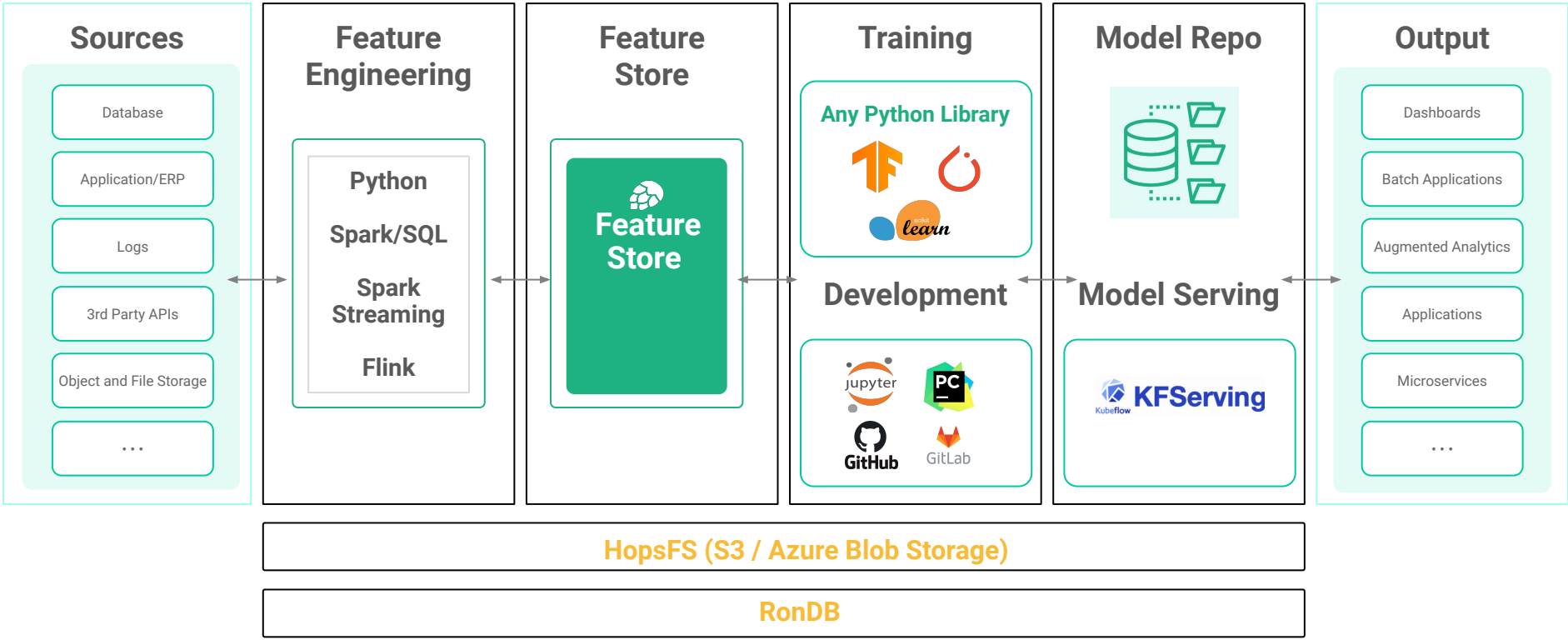
Growing Consensus on how to manage complexity of AI

FEATURE STORE



ML PLATFORM
TRAIN and SERVE

Hopsworks - Design and Operate AI Applications at Scale



Hopsworks is an Open, Modular Feature Store

Data Science

Azure Machine Learning

Amazon SageMaker

Google AI

databricks

DOMINO

data iku

Data Engineering

python

Spark

Apache Airflow

Microsoft SQL Server

mongoDB.

snowflake

Teams use the tools of their choice, integrated with the Hopsworks Feature Store

HOPSWORKS Feature Store

Model Serving

Azure Machine Learning

Amazon SageMaker

Google AI

databricks

Kubeflow

SELDON

Compliance & Regulatory

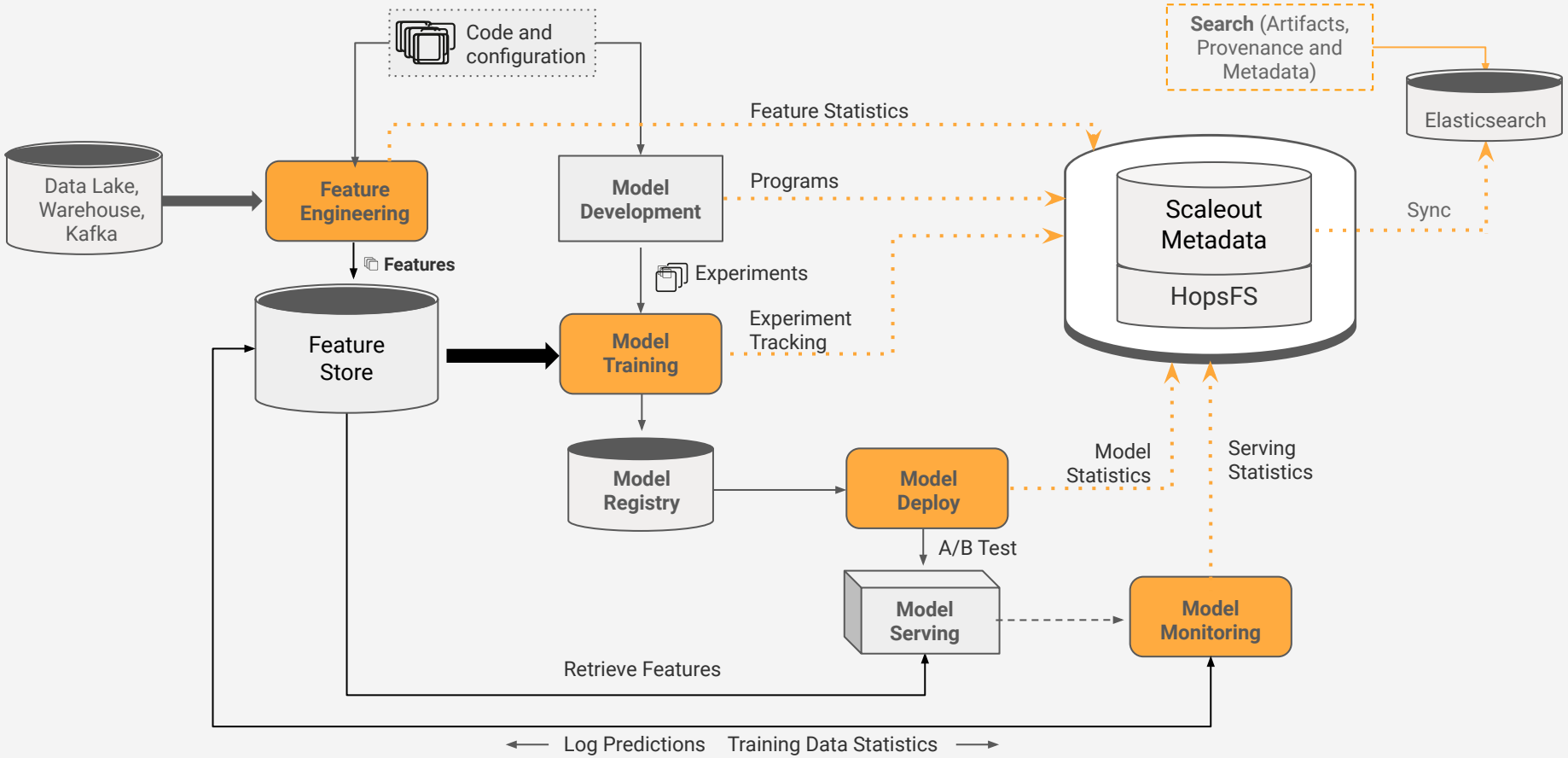
Informatica

collibra

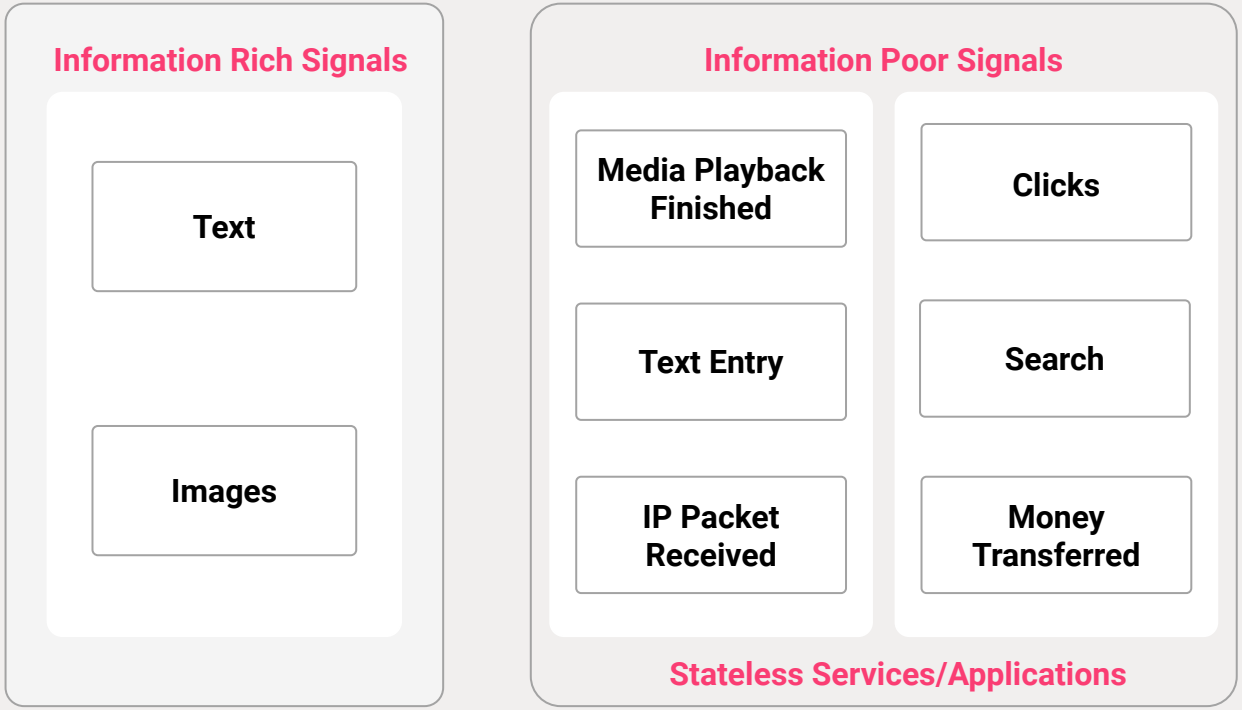
Alation

Apache Atlas

Hopsworks End-to-End Machine Learning (ML) Pipelines



The Feature Store - From Information Poor Signals to Information Rich Signals



Commercial Feature Stores for Machine Learning

databricks

HOPSWORKS

Google Cloud

Amazon SageMaker

iguazio

RASGO

KASKADA

tecton

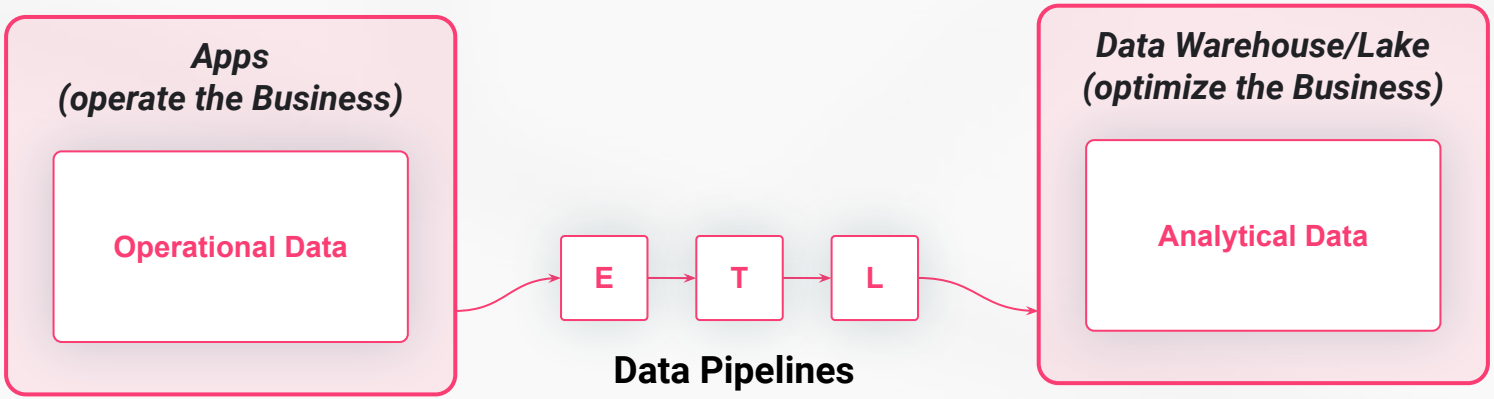
ABACUS.AI

{featureform

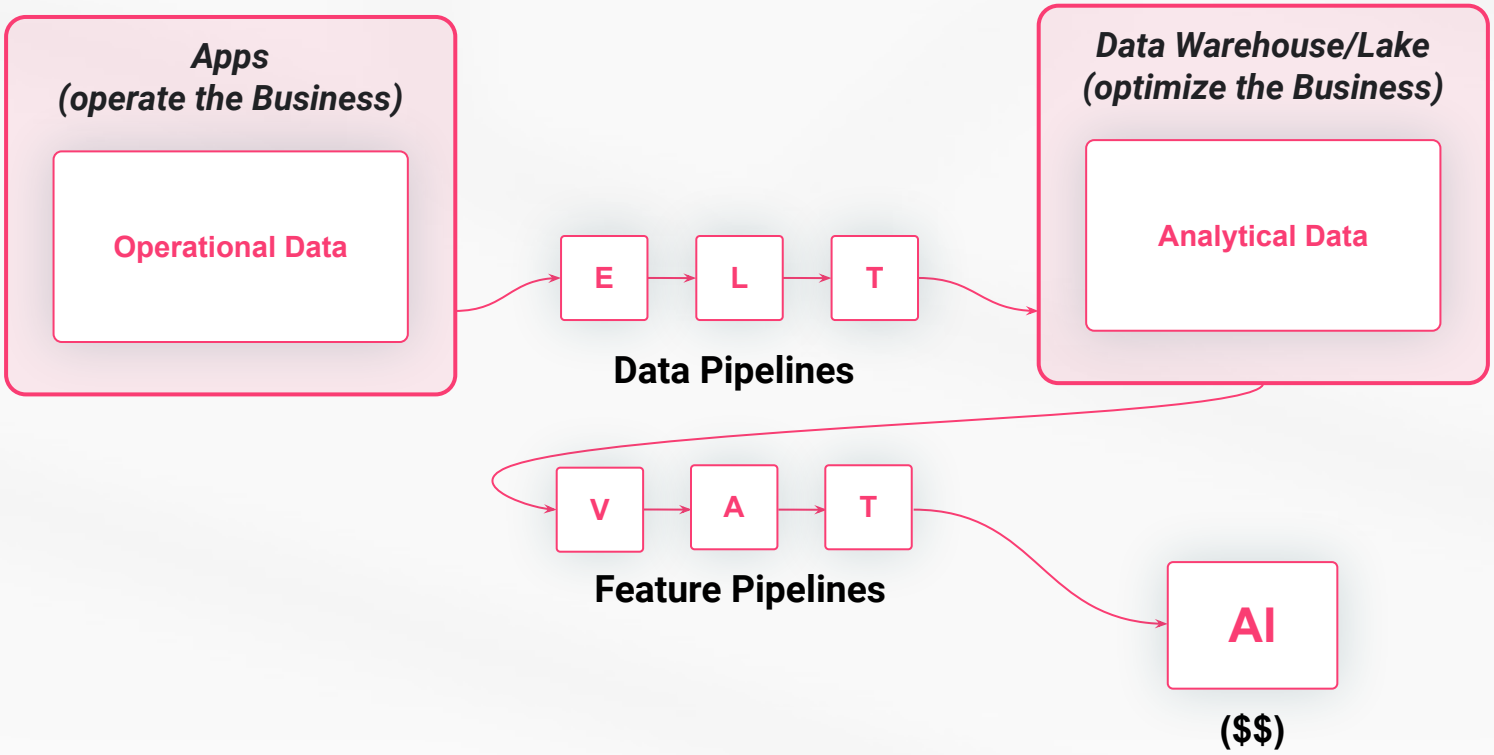
MOLECULA

scribble Data

How the feature store fits in your data infrastructure

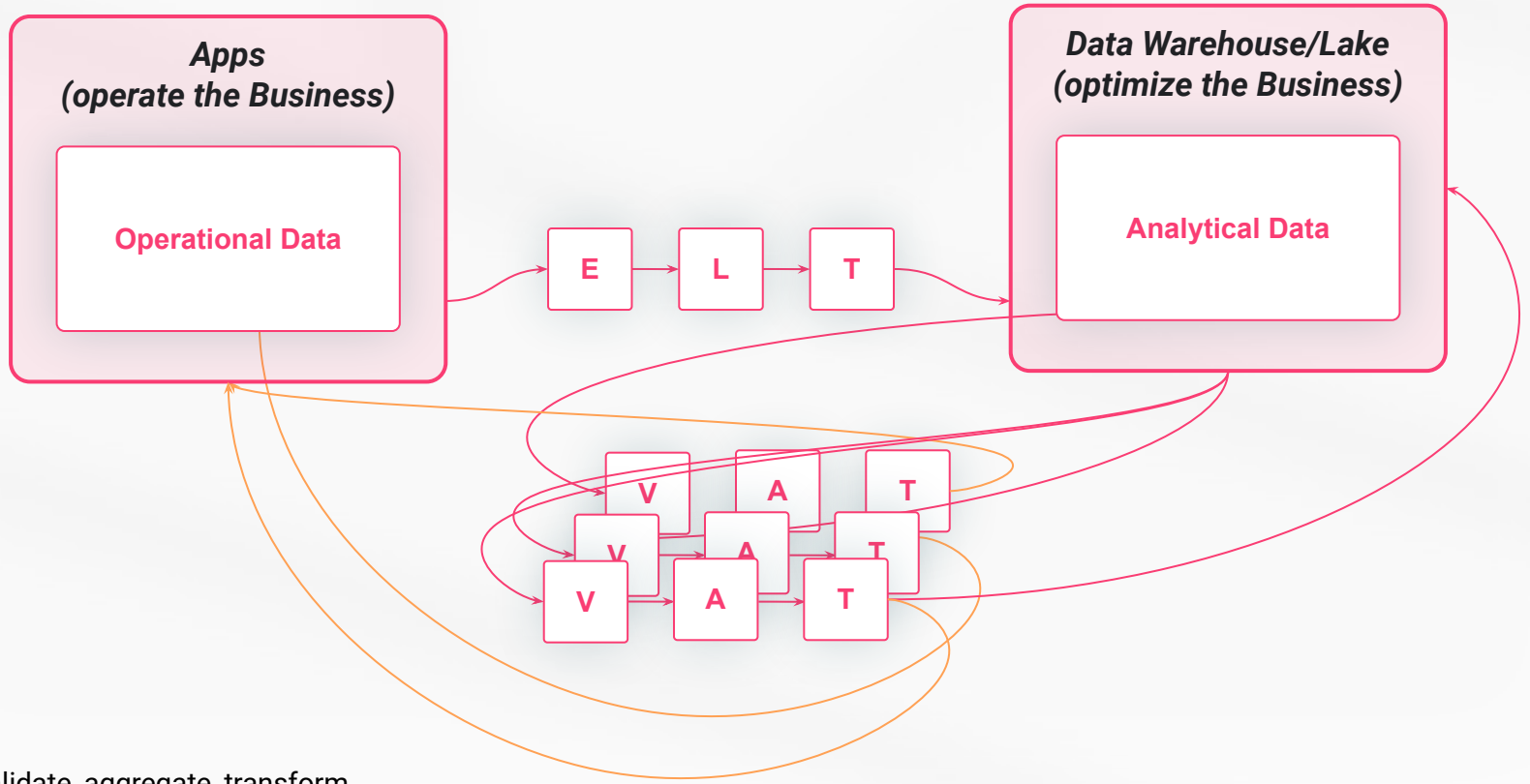


How the feature store fits in your data infrastructure



VAT = validate, aggregate, transform

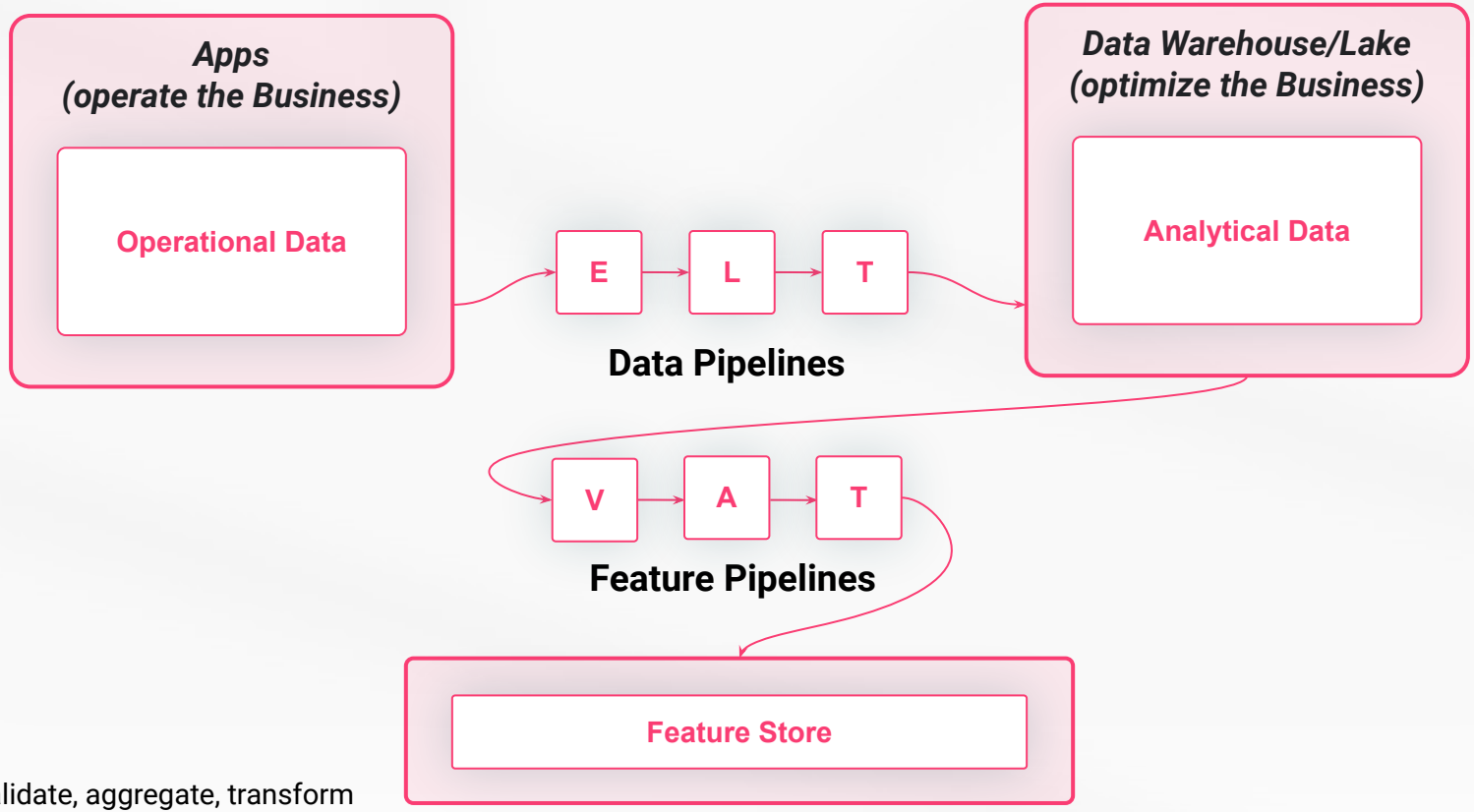
How the feature store fits in your data infrastructure



VAT = validate, aggregate, transform

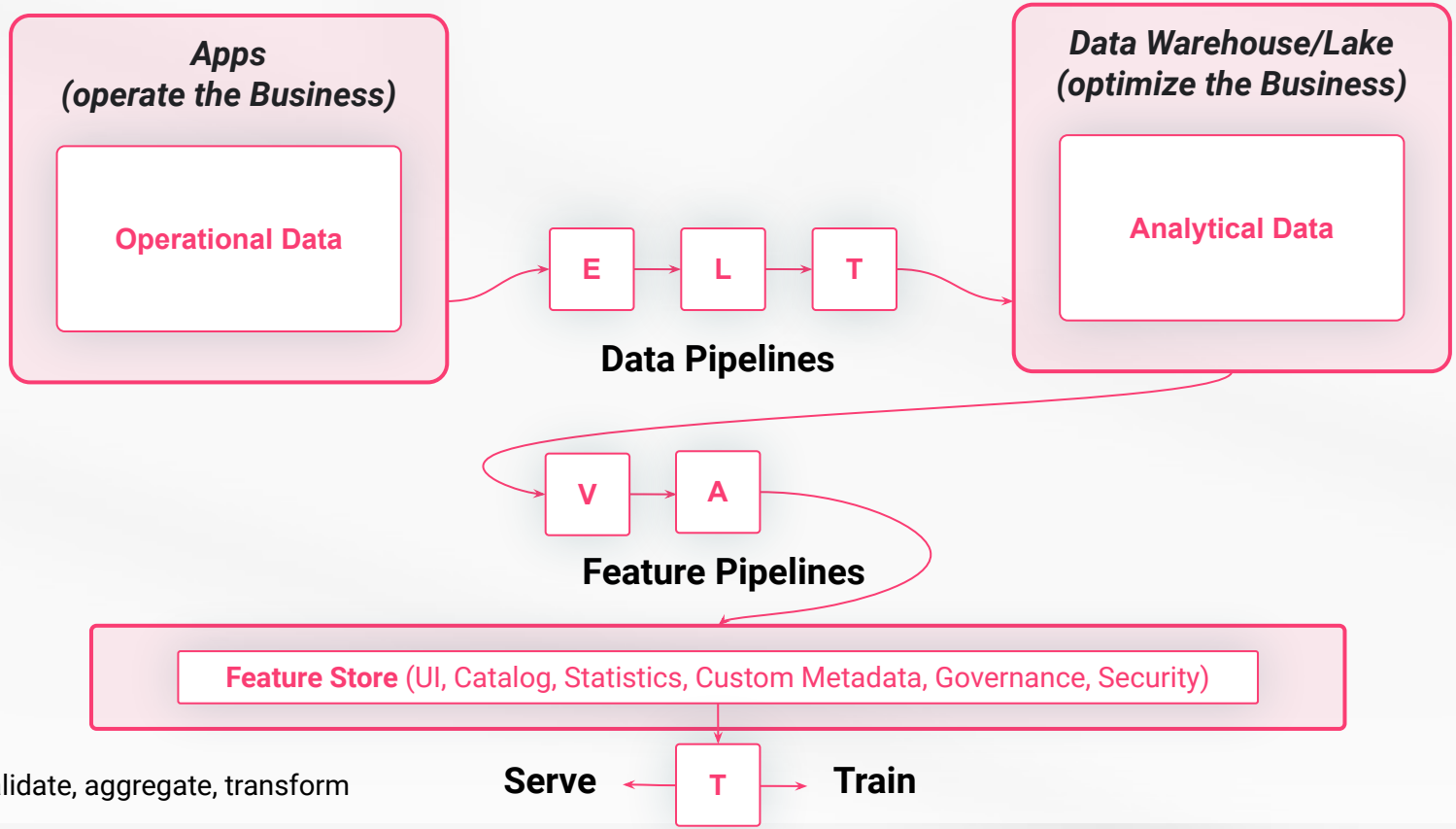
Feature Pipelines Jungle

How the feature store fits in your data infrastructure



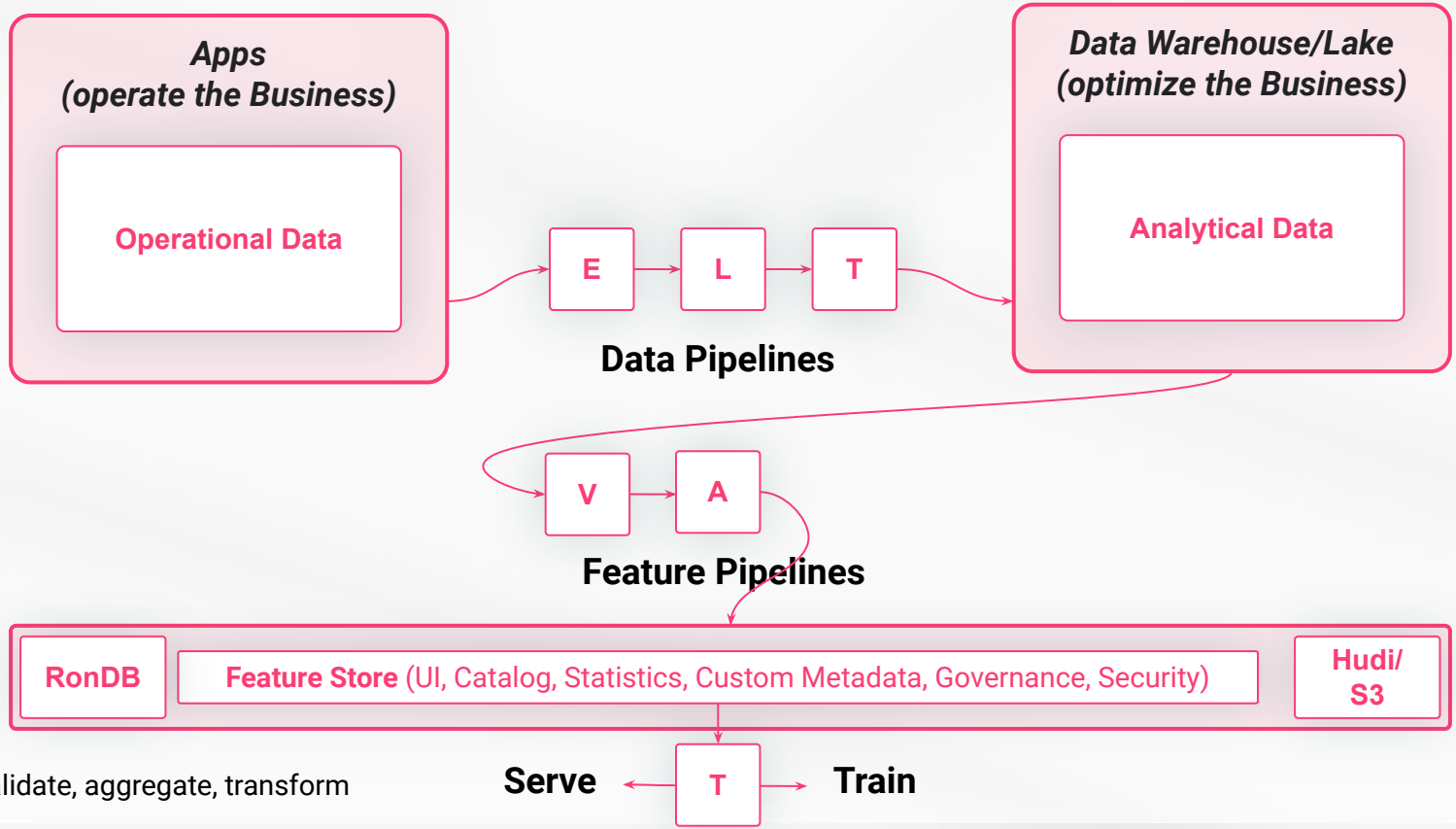
VAT = validate, aggregate, transform

How the feature store fits in your data infrastructure



VAT = validate, aggregate, transform

How the feature store fits in your data infrastructure



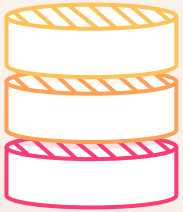
VAT = validate, aggregate, transform

Enables Collaboration between folks who speak different languages

Data Scientist
Python

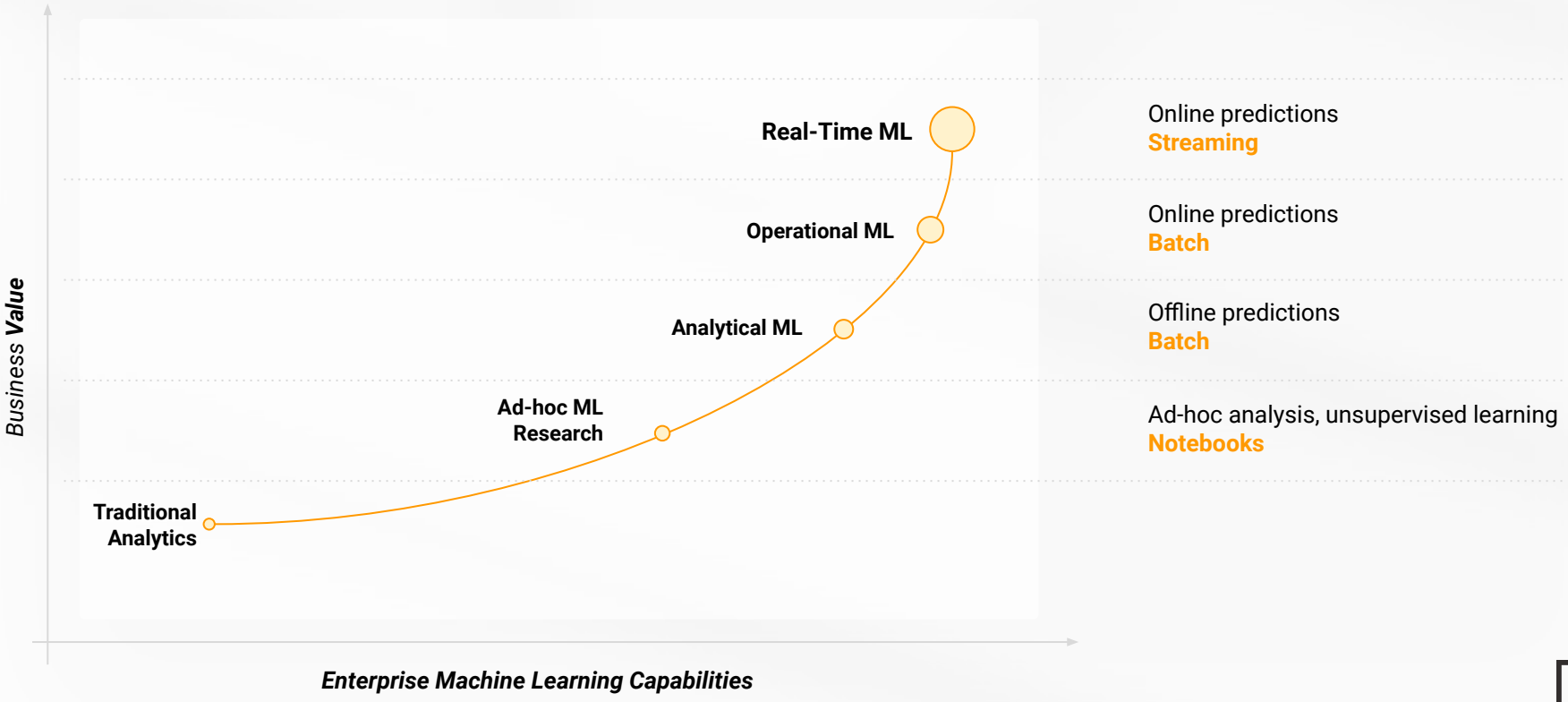
Data Engineer
SQL, Spark, Flink, Python

Feature Store

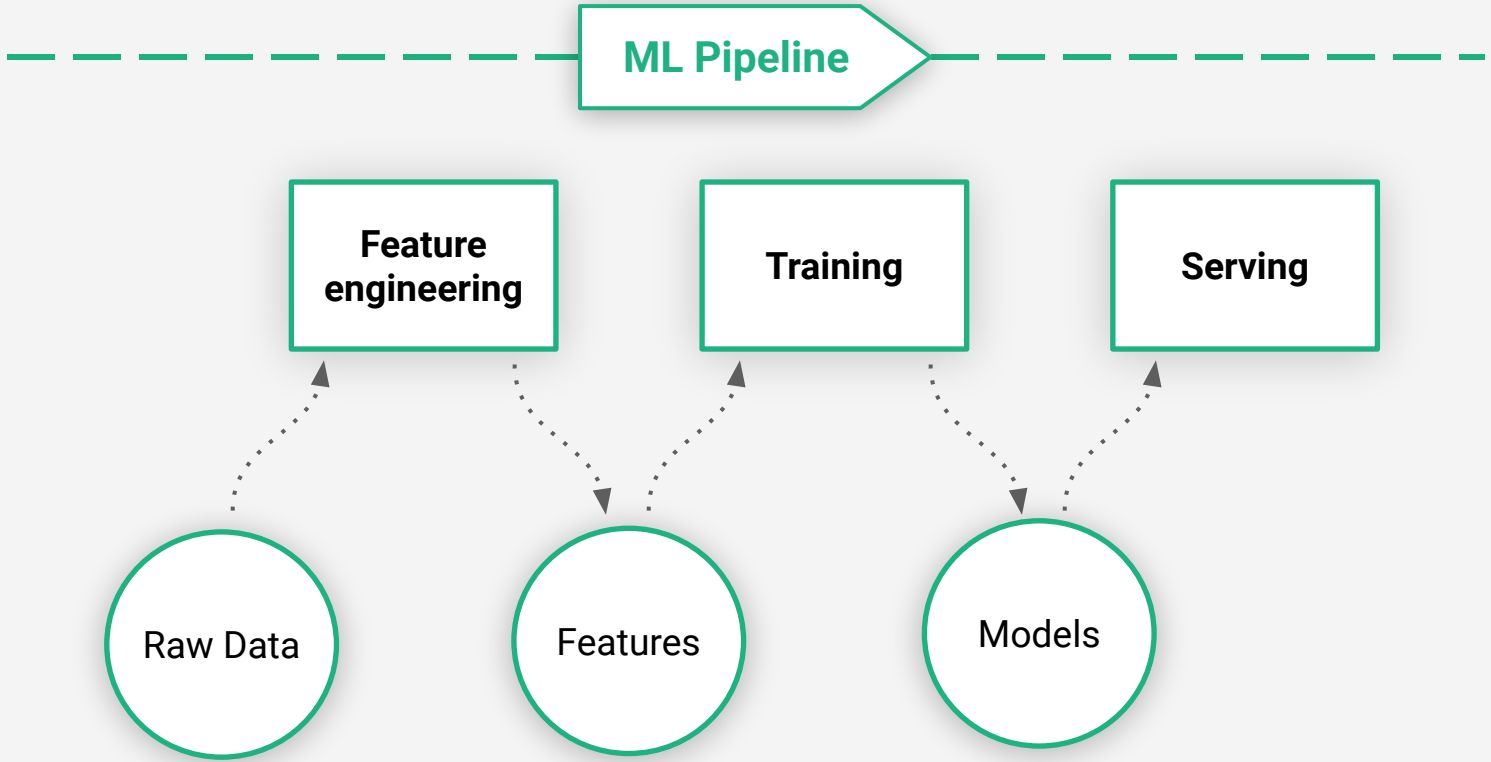


ML Engineer
Kubernetes, Serverless

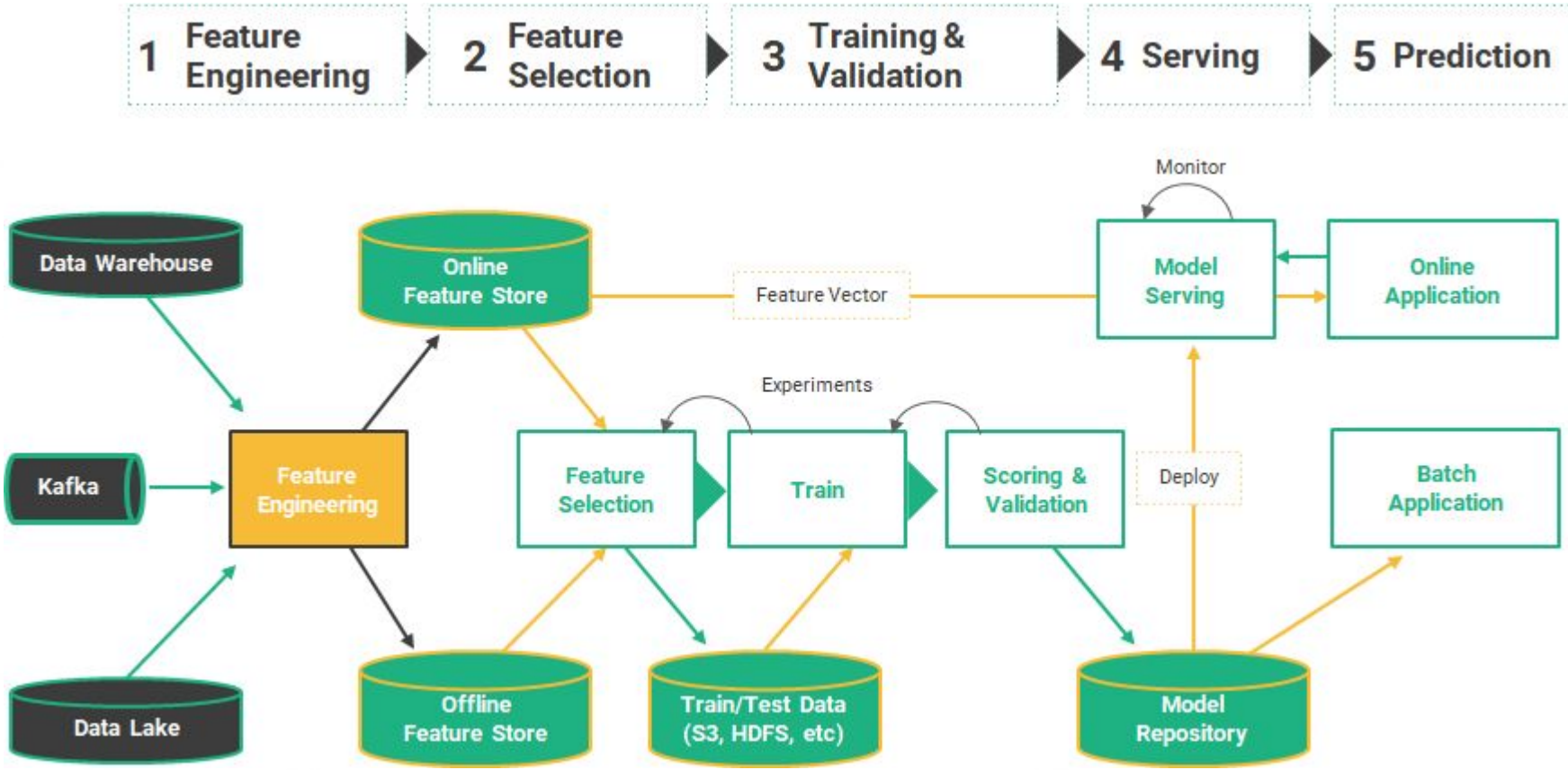
Road To AI Value



What are ML Pipelines?

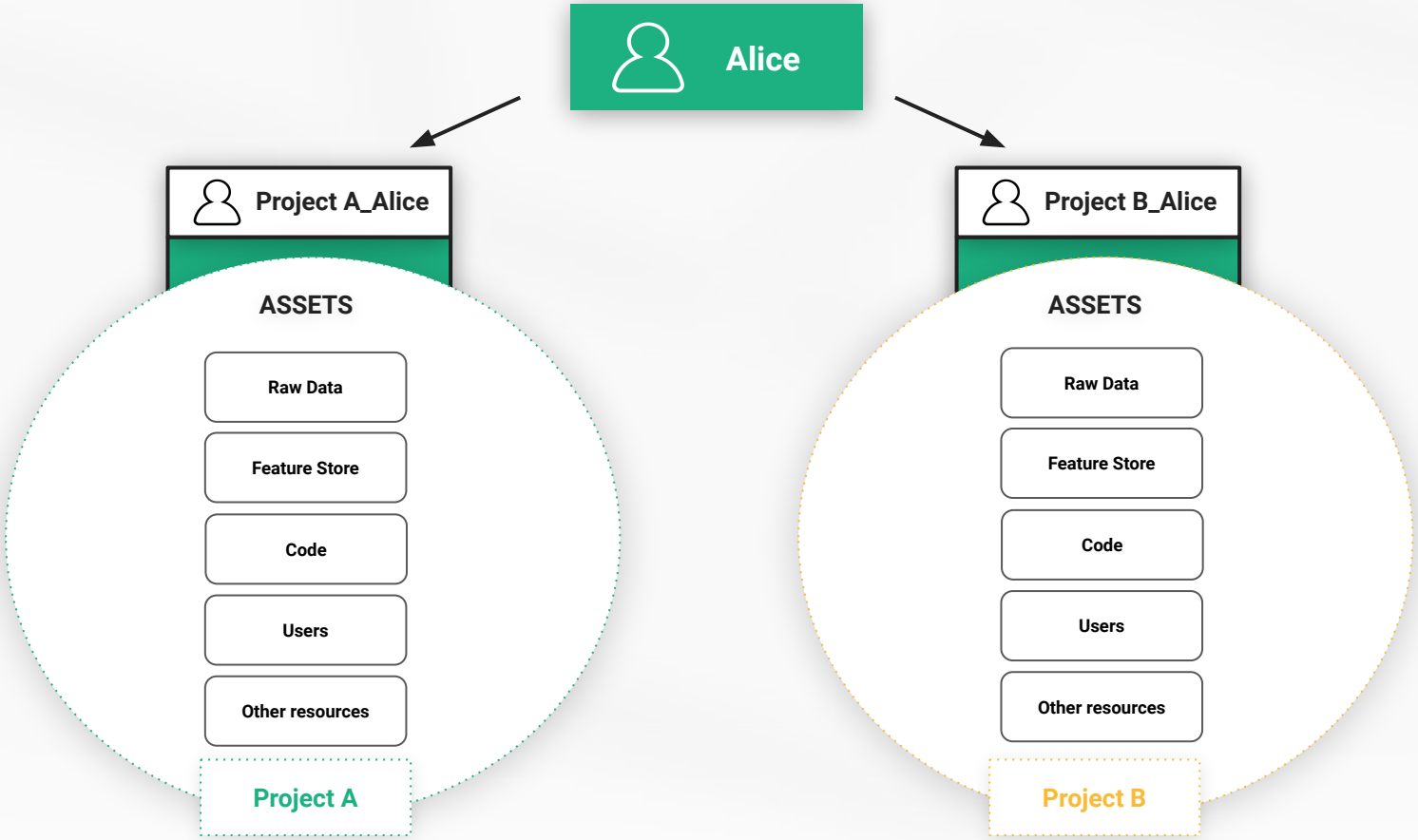


End-to-End Machine Learning (ML) Pipelines

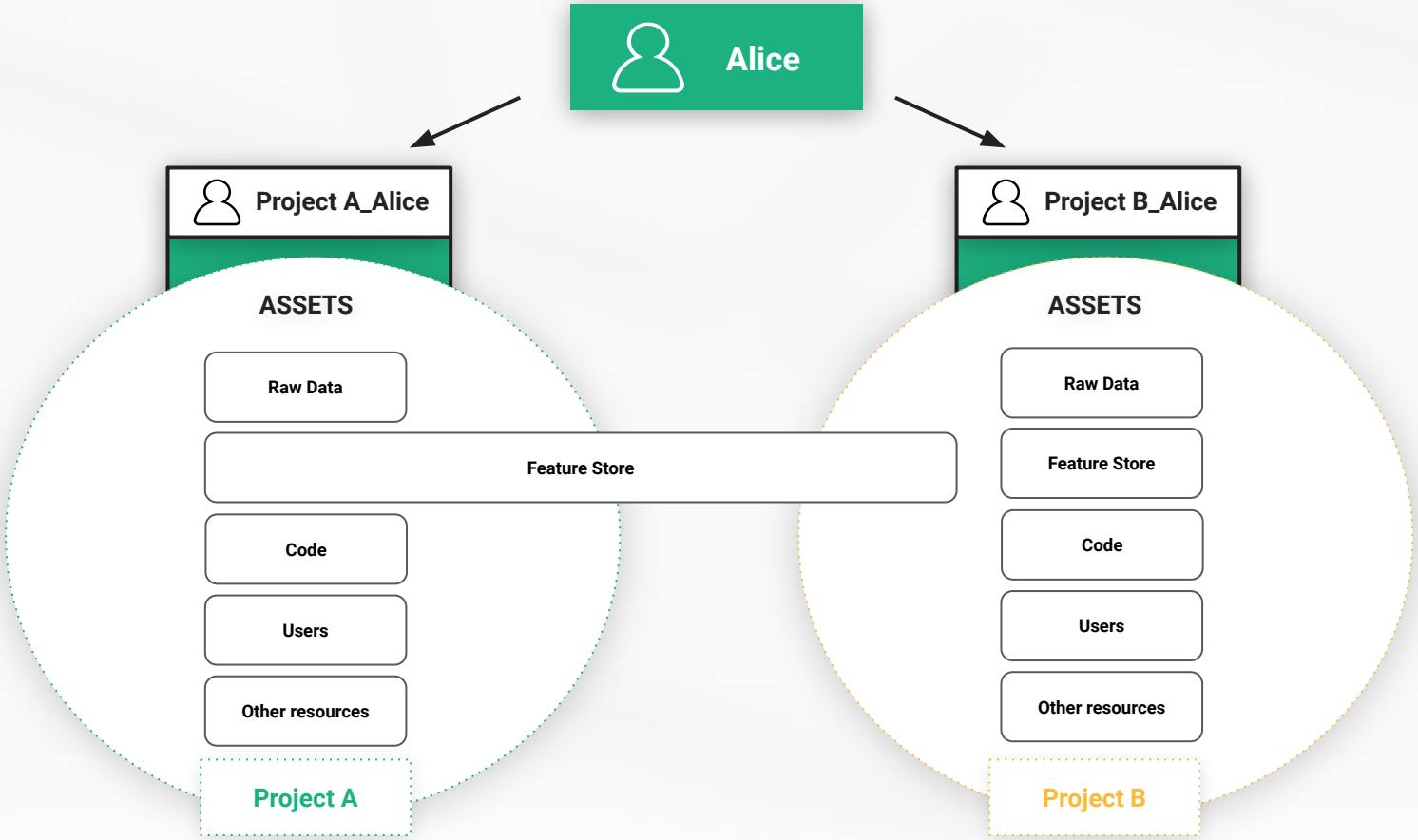


How it Started

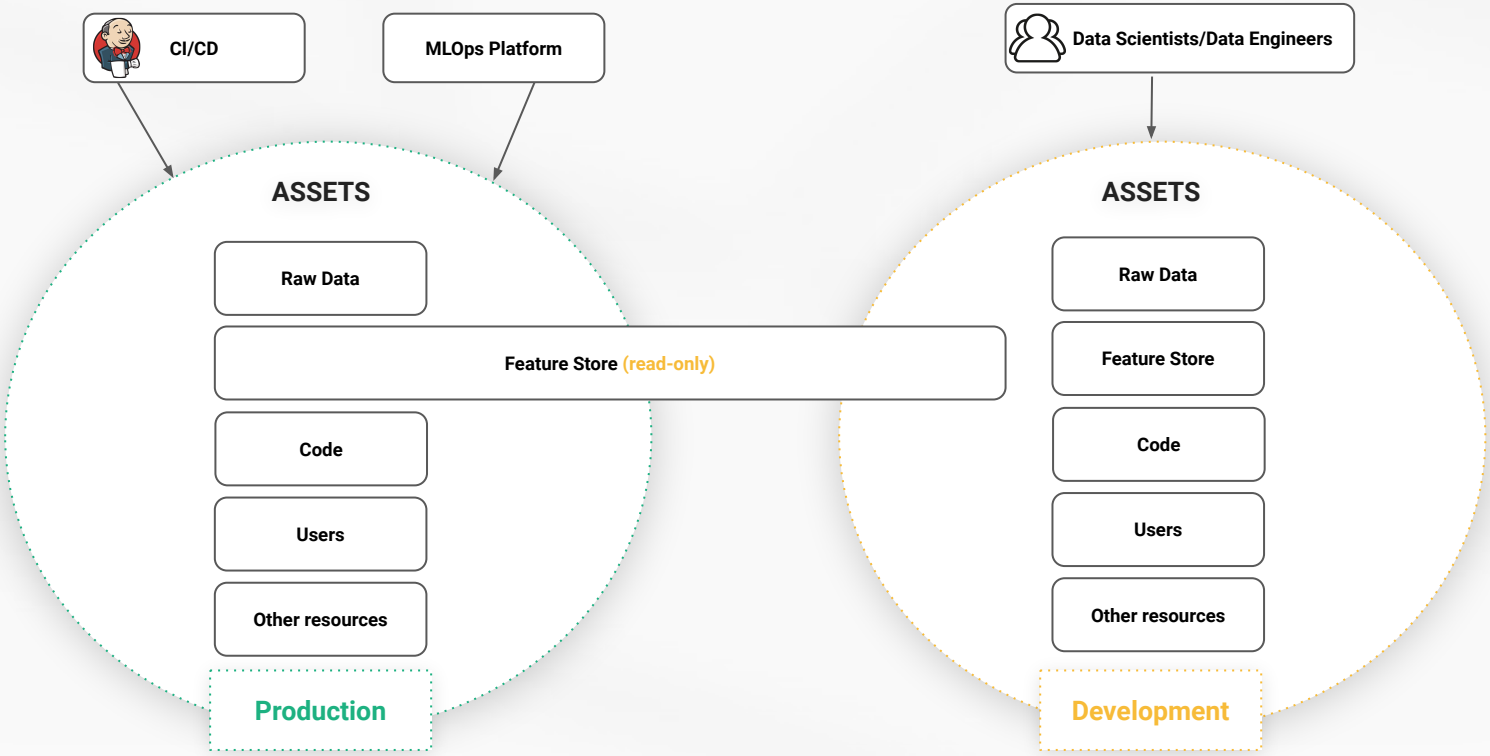
Project Based Multi Tenancy



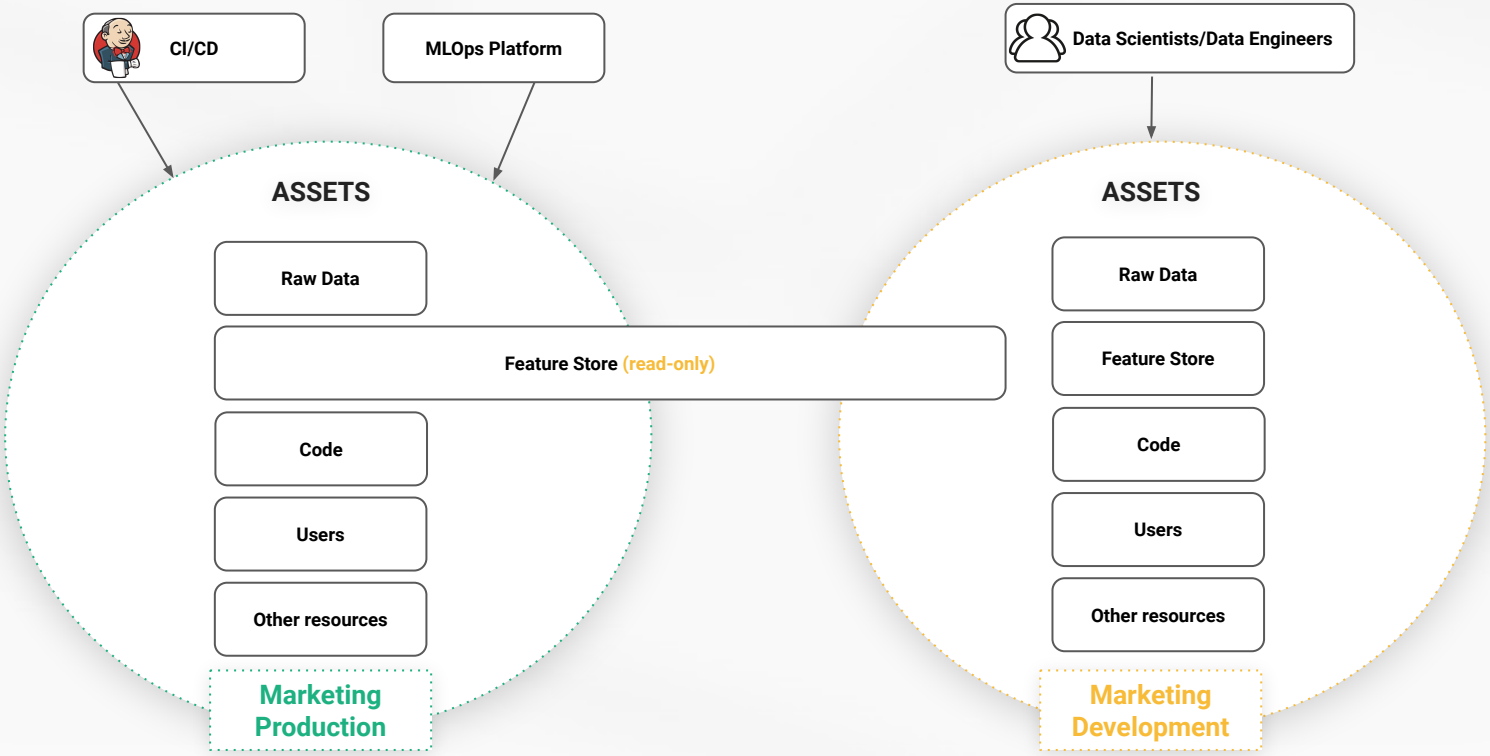
Project Based Multi Tenancy - Shared Data



Project Based Multi Tenancy - Production/Development

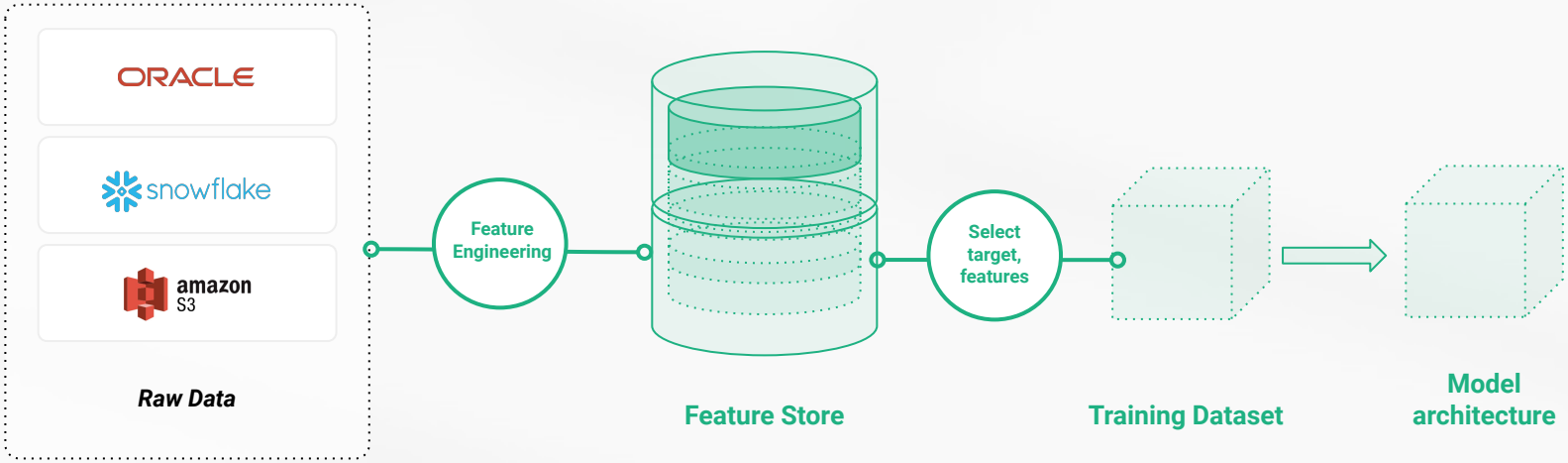


Project Based Multi Tenancy - Mix Structure

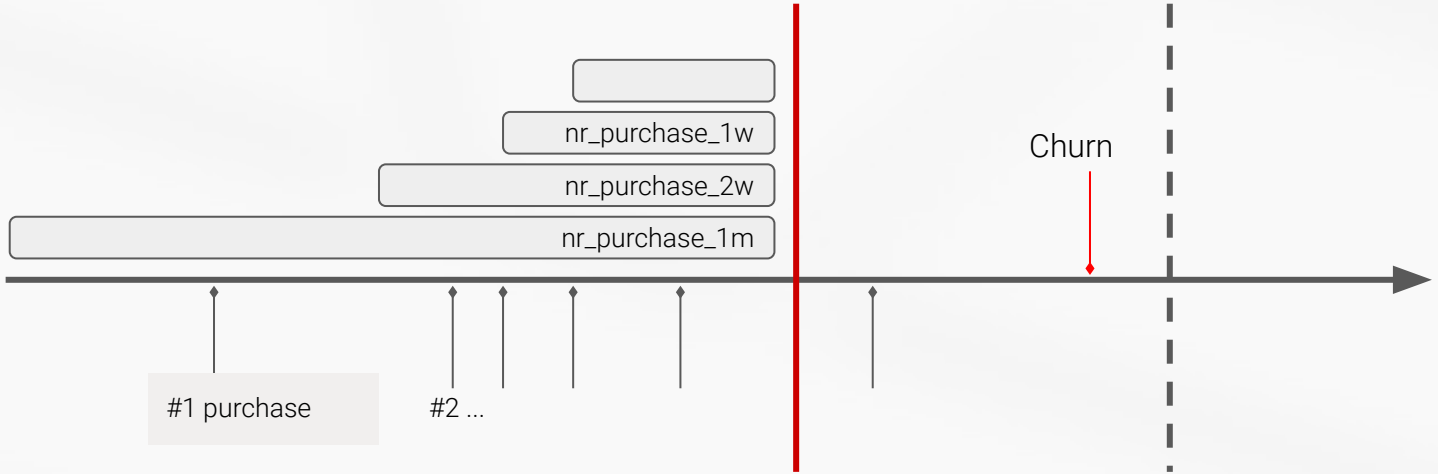


Feature Engineering

From Data to Features to Training Data to Models



Feature engineering - Aggregation example



Transaction

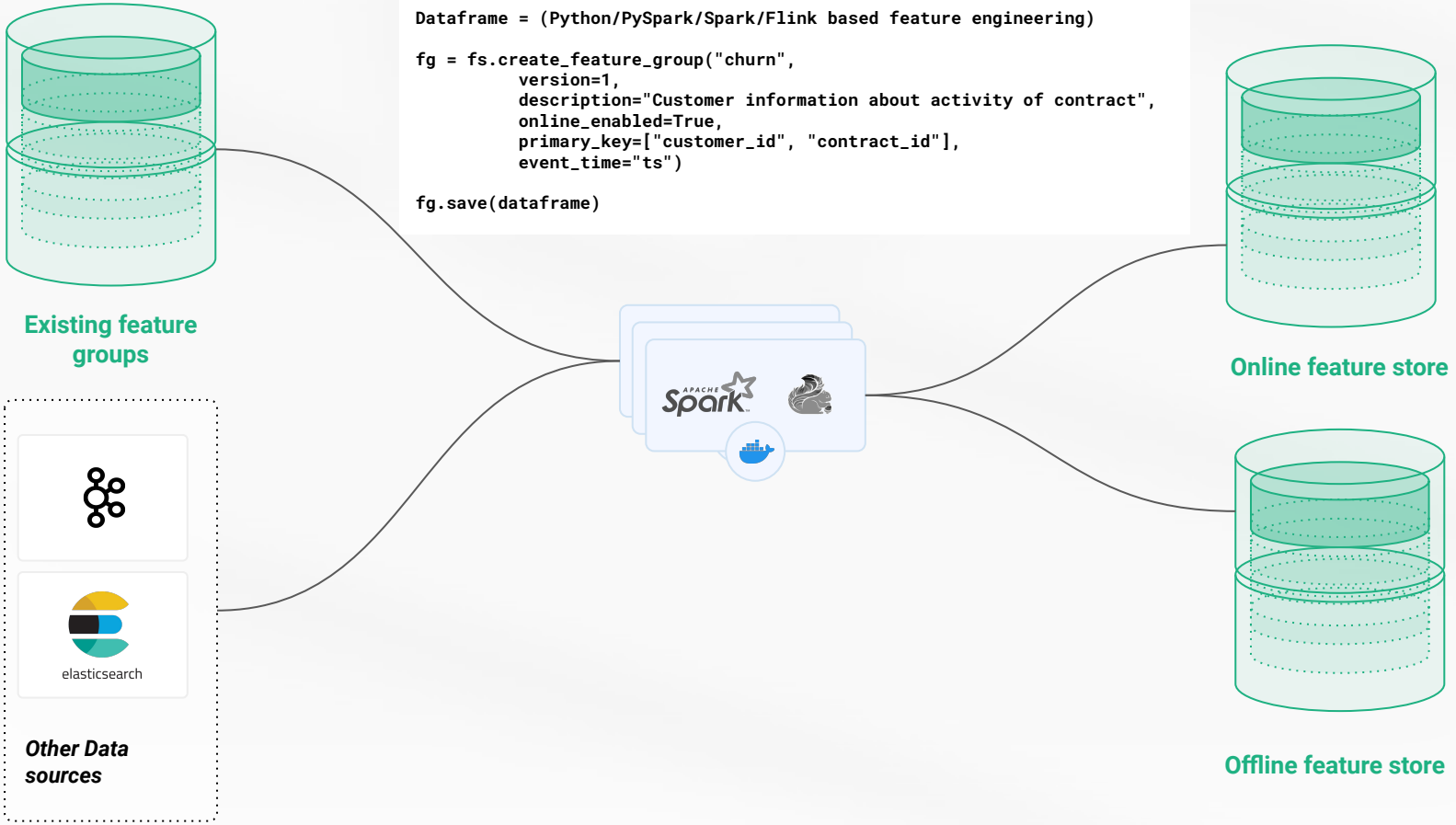
transaction_time	*id*
2021/02/10	23
2021/01/24	42
2021/01/20	47

Feature Aggregation

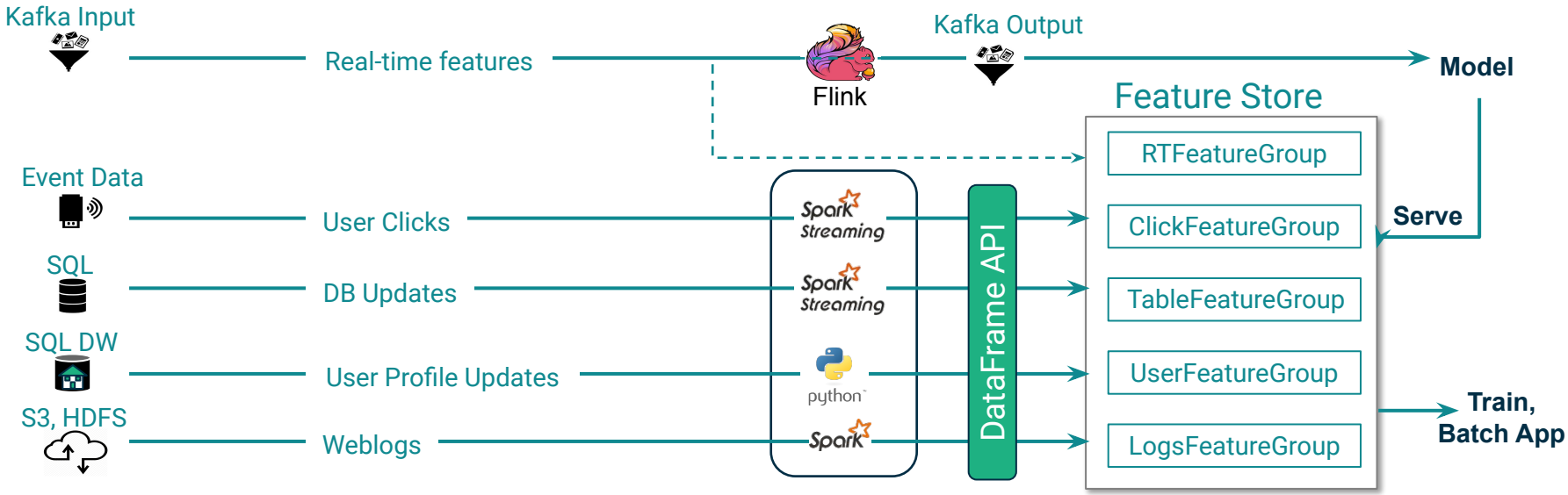
Features

Entity	***1w***	*1m*
1	1	2

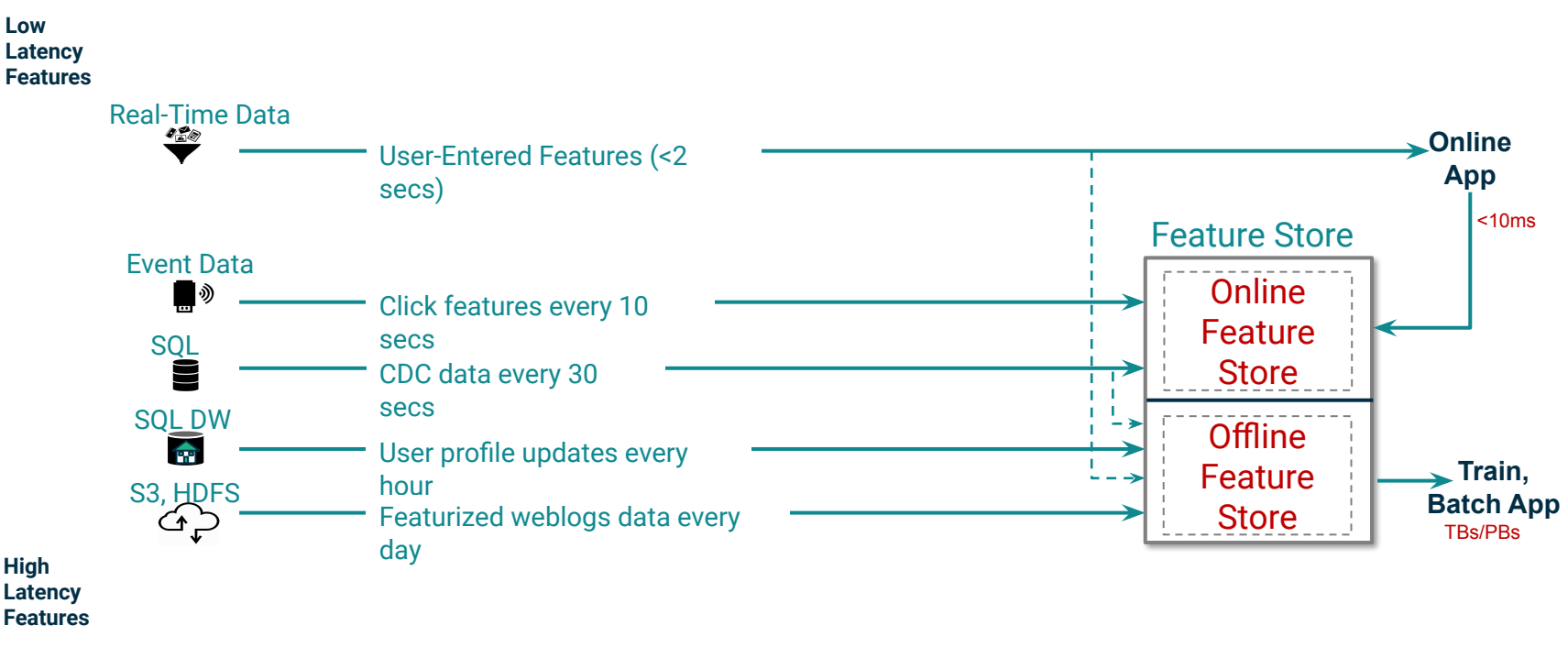
Feature engineering



Reusable Features are stored in FeatureGroups

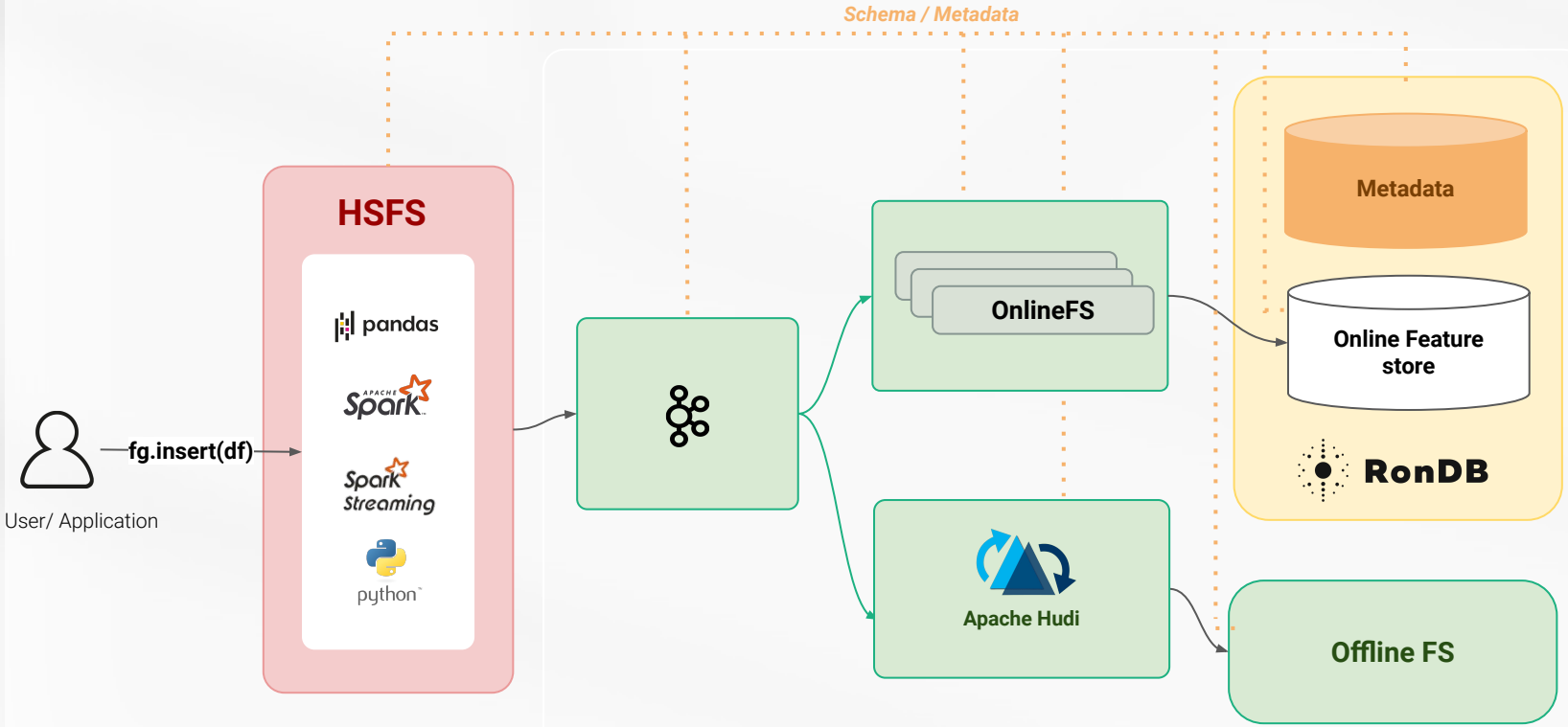


Feature Pipelines update the Feature Store (2 Databases!) with data from backend Platforms



No existing database is both scalable (PBs) and low latency (<10ms). Hence, online + offline Feature Stores.

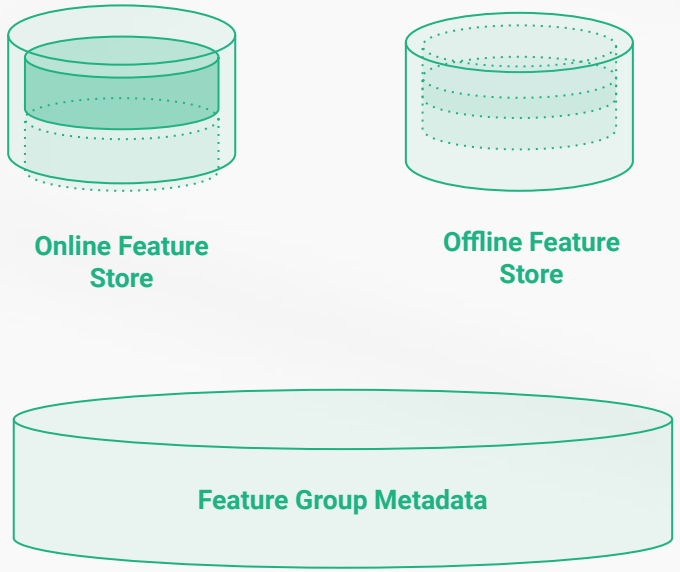
Streaming Applications can write Fresh Features



Feature Groups

Cached feature groups

- Feature groups stored on Hopsworks
- Can be available both offline and online.
- Documentation: https://docs.hopsworks.ai/feature-store-api/latest/generated/feature_group/
- Example: https://examples.hopsworks.ai/featurestore/hsfs/basics/feature_engineering/



Feature Group Versioning

Schema versioning

- Each feature group has a version number
- Version numbers allow users to identify breaking changes to the schema (feature dropped, change in the way a feature is being computed)
- Appending feature to a feature group is not considered a breaking change

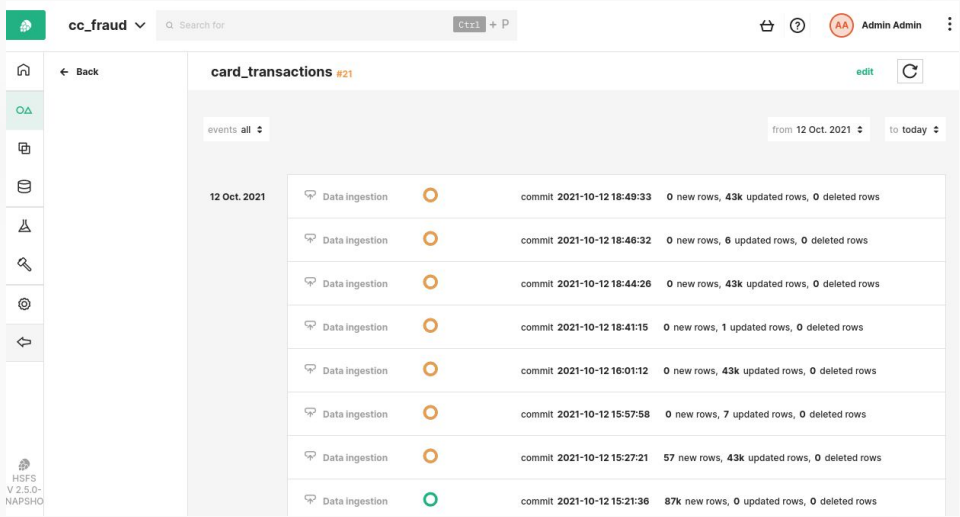
Data versioning

- Calling insert()/save() on a feature group generates a new data commit.
- Data commits allow users to track how the data changed during the lifetime of a feature group.
- Users can navigate the commit history using the Activity UI
- Using the *as_of* method, users can retrieve features from a feature group, at a specific point in time.

Metadata - Activity

List actions performed on a feature group:

- Feature group creation
- Data ingestion
- Statistics computation
- Data validation

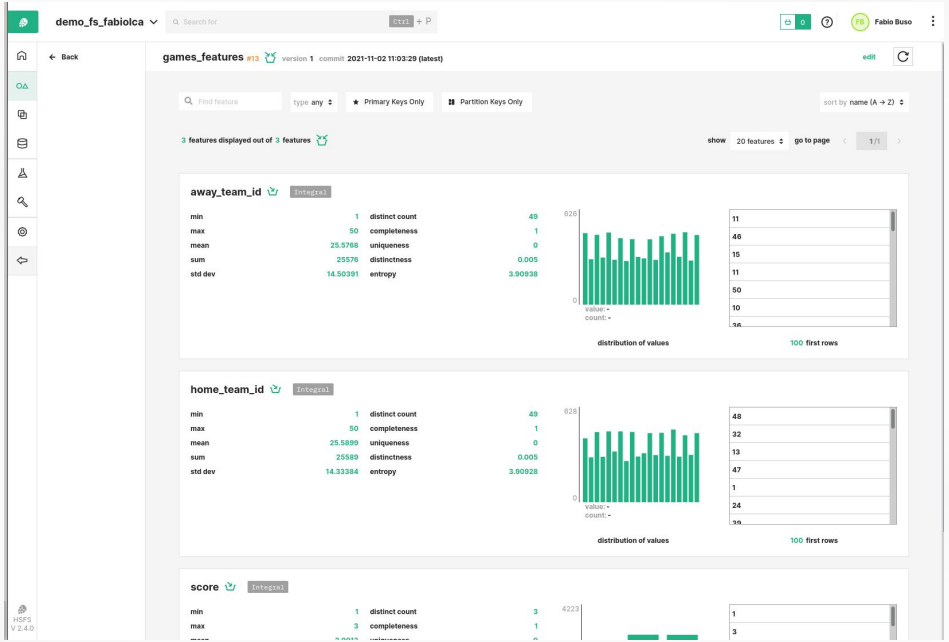


Metadata - Statistics

Statistics are computed at feature group level for each [data commit]

Hopsworks computes automatically: descriptive statistics, histograms and correlations between features

Statistics can be explored from the UI.

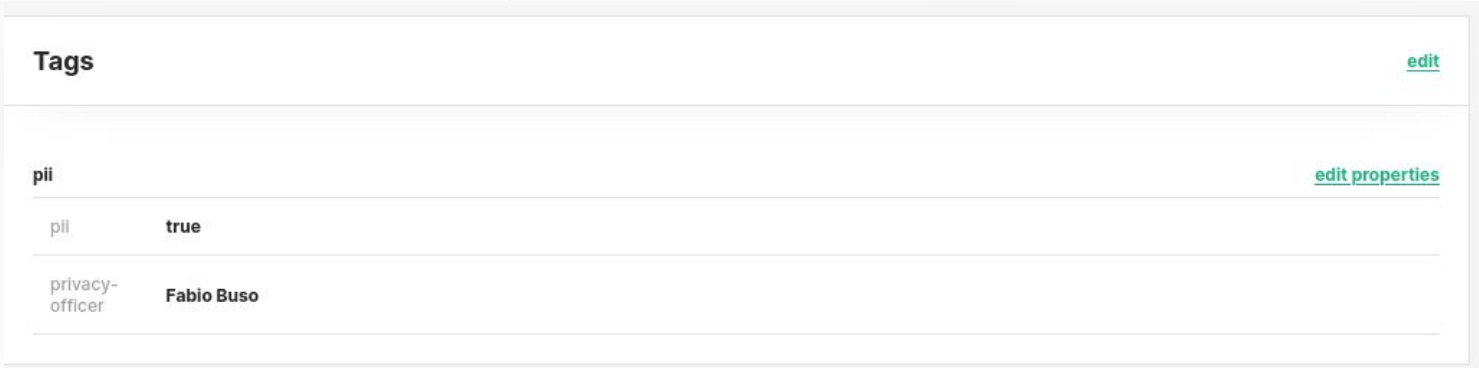


Metadata - Tags

Tags allow users to specify arbitrary metadata and make it searchable throughout the feature store.

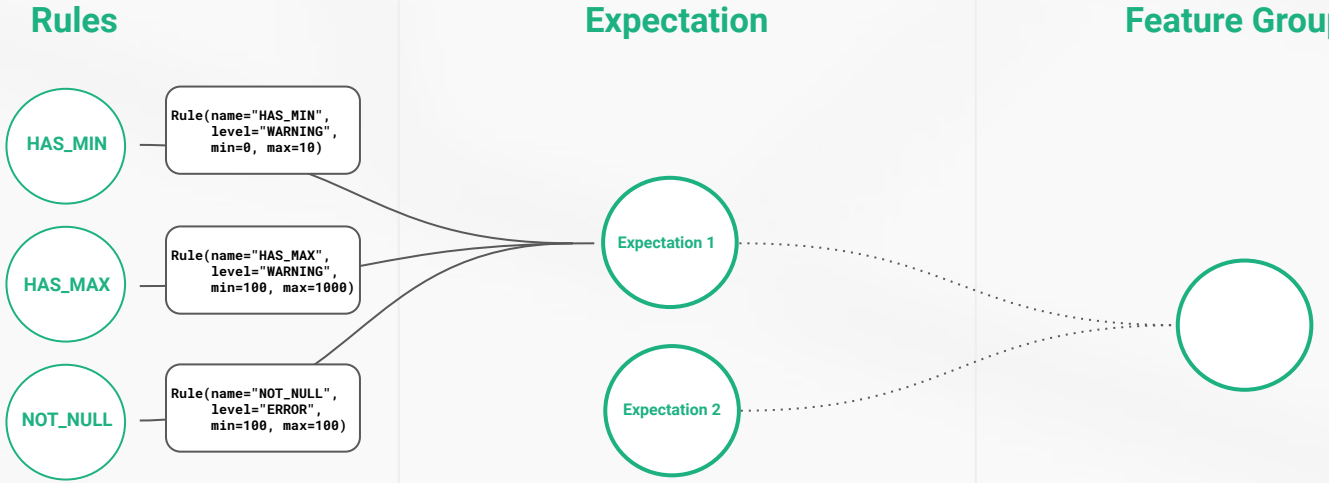
Tags require a schema that can be defined and enforced at platform level.

Tags can be manipulated through the UI or through the APIs



Data Validation

Data validation



These are the rules supported by the platform:
https://docs.hopsworks.ai/feature-store-api/latest/generated/feature_validation/#rule-definitions

An expectation is a set of rules with specific values and rule severity severity

Expectations can be applied to specific features on a feature group, or to the entire set of features of a feature group

https://docs.hopsworks.ai/feature-store-api/latest/generated/feature_validation/#rule-definitions

Validation types

Validation Type	Success	Warning	Failure
Strict	Insertion	Reject	Reject
Warning	Insertion	Insertion	Reject
All	Insertion	Insertion	Insertion
None	No data validation performed		

An expectation is a set of rules with specific values and rule severity severity

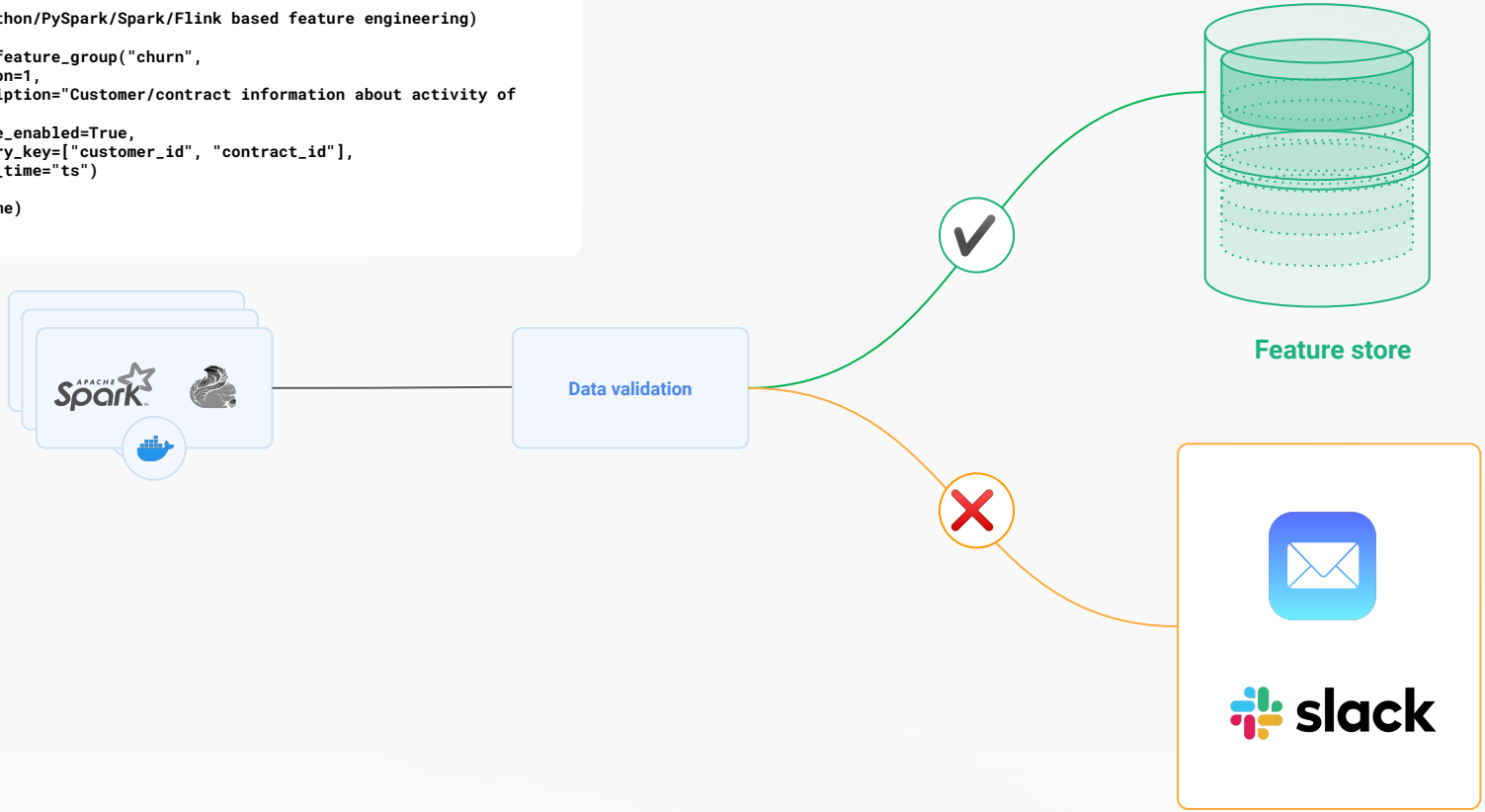
Expectations can be applied to specific features on a feature group, or to the entire set of features of a feature group

https://docs.hopsworks.ai/feature-store-api/latest/generated/feature_validation/#rule-definitions

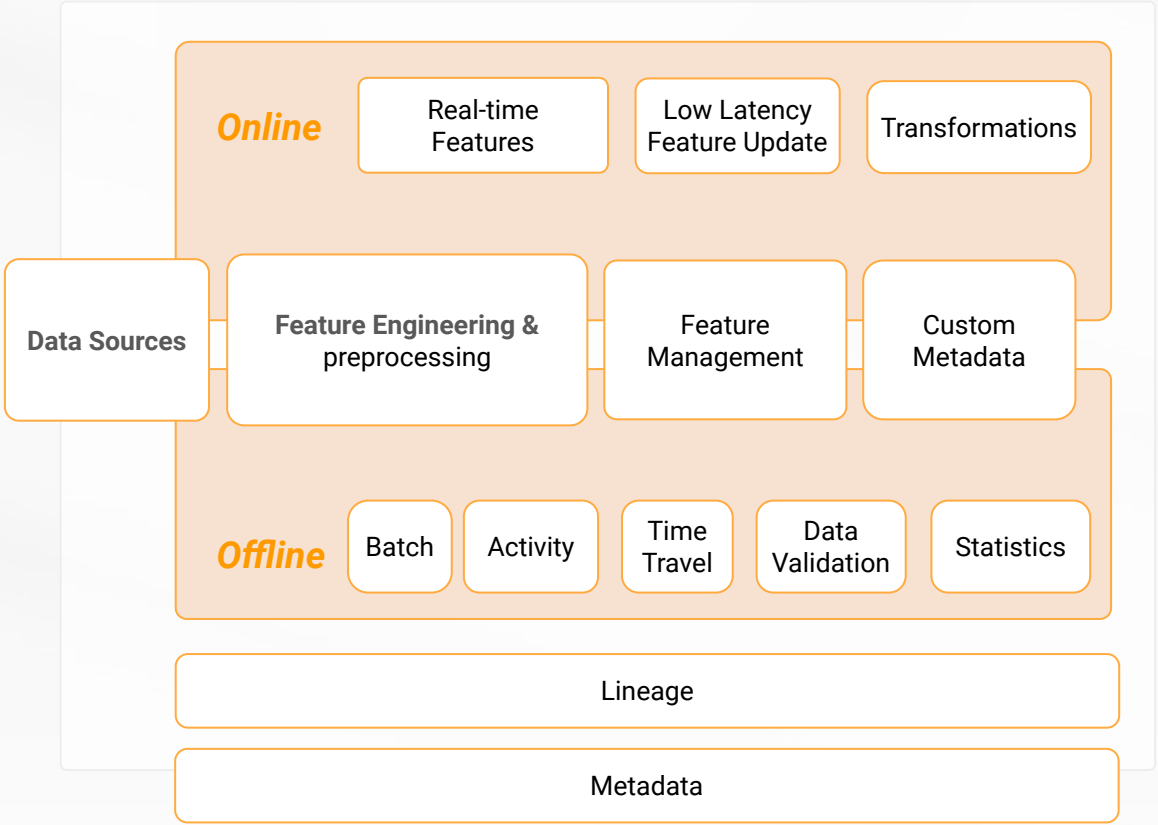
```
Dataframe = (Python/PySpark/Spark/Flink based feature engineering)

fg = fs.create_feature_group("churn",
          version=1,
          description="Customer/contract information about activity of
contract",
          validation_type="STRICT",
          primary_key=["customer_id", "contract_id"])

fg.save(dataframe)
```

Data Validation

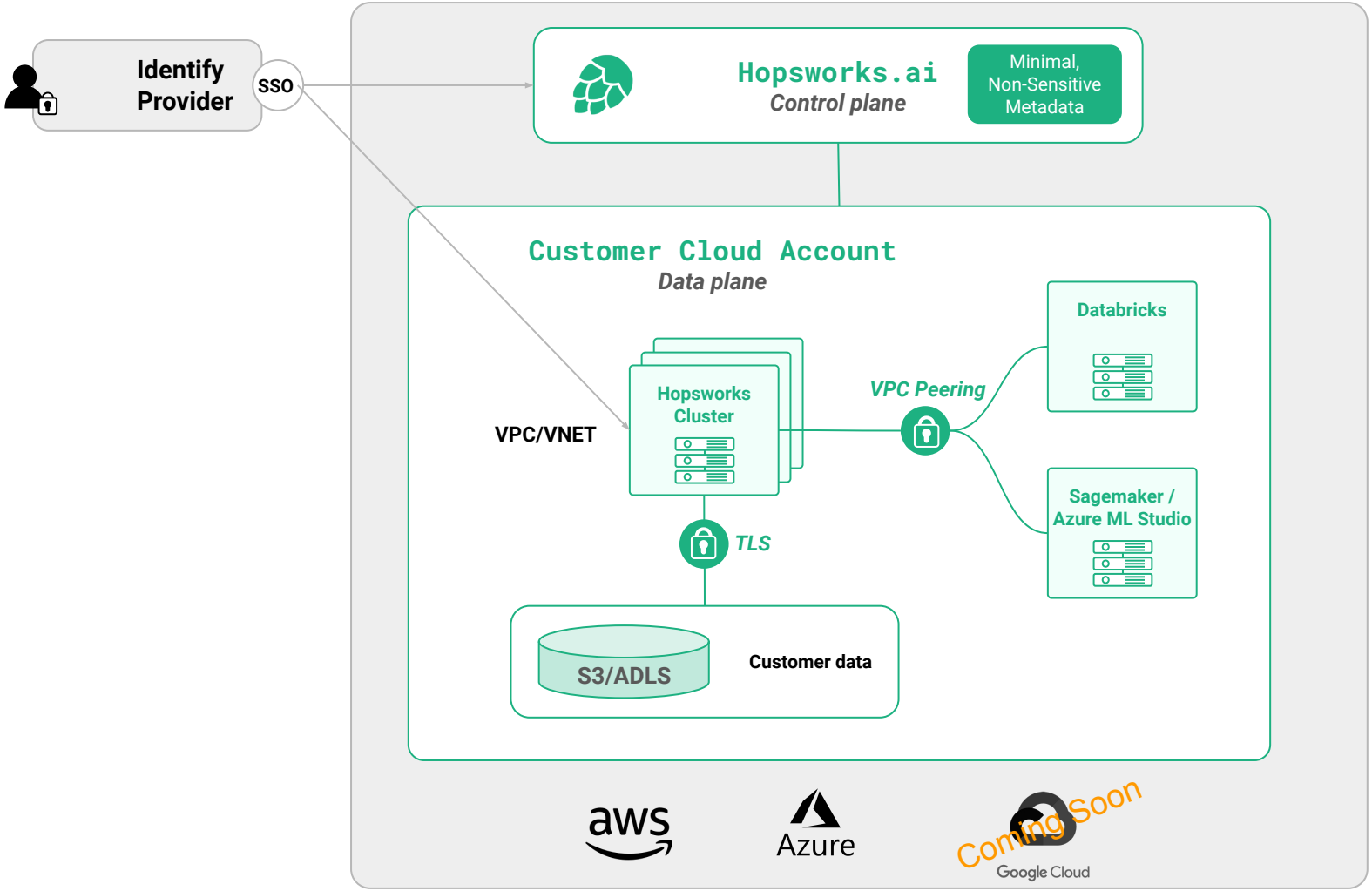
```
Dataframe = (Python/PySpark/Spark/Flink based feature engineering)

fg = fs.create_feature_group("churn",
          version=1,
          description="Customer/contract information about activity of
contract",
          online_enabled=True,
          primary_key=["customer_id", "contract_id"],
          event_time="ts")

fg.save(dataframe)
```



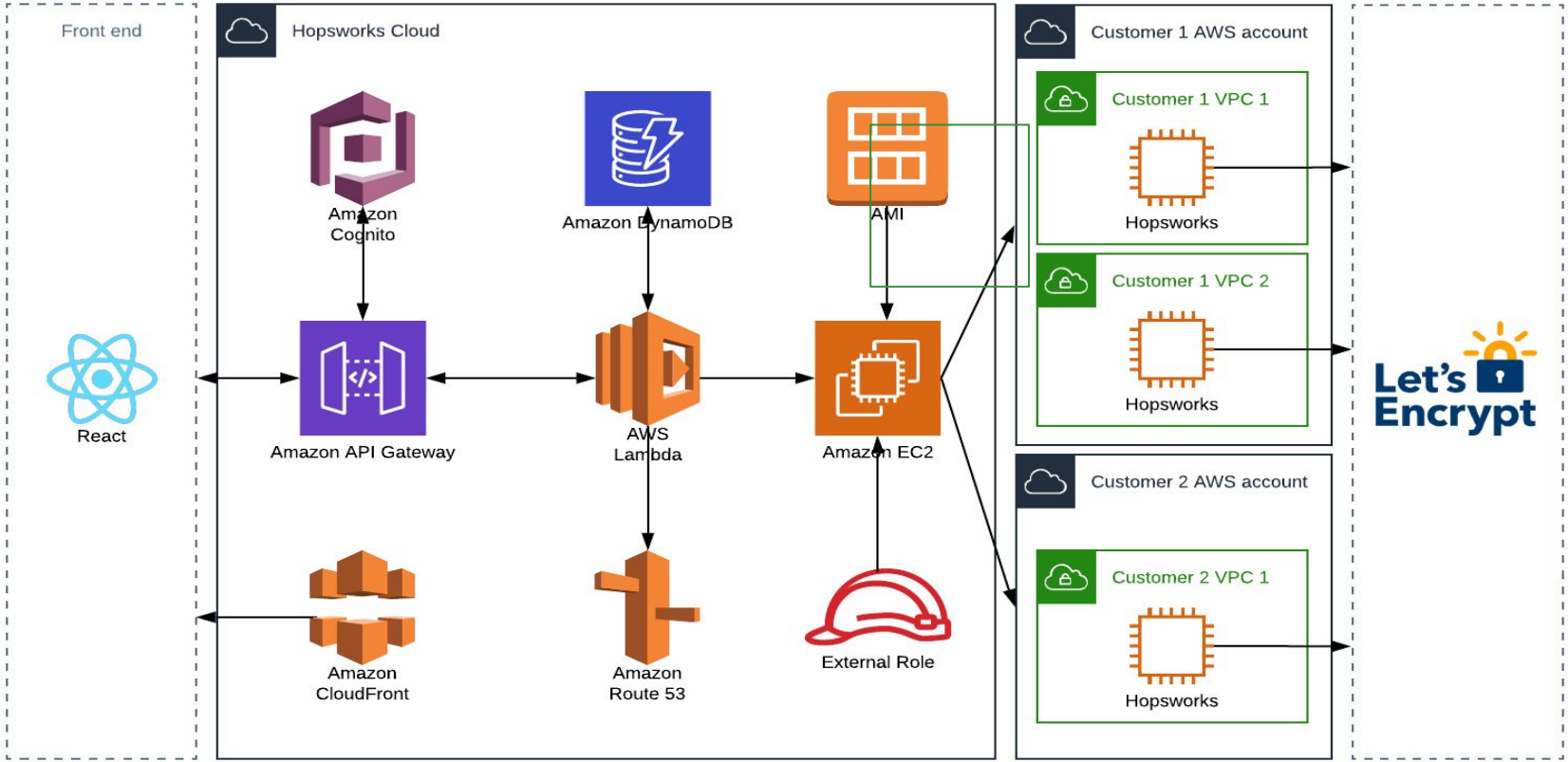
Hopsworks Feature Store



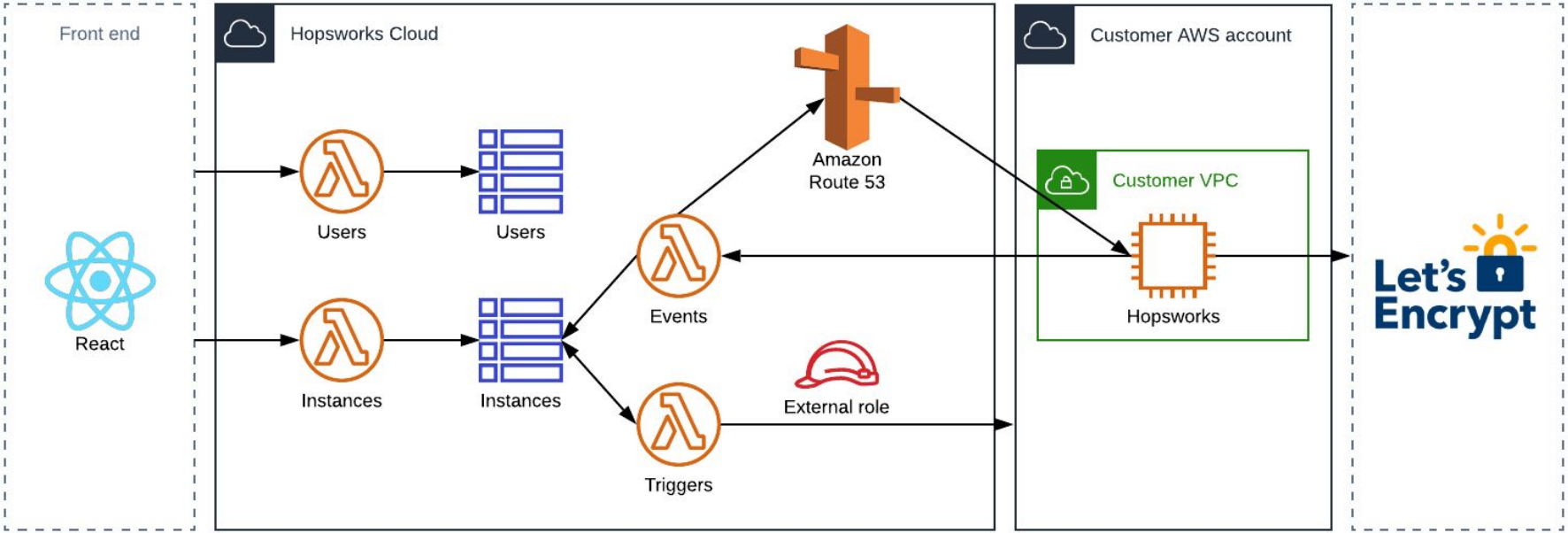
Hopsworks.ai
Deployment / Security


Identify Provider
SSO
Hopsworks.ai
Control plane
Minimal, Non-Sensitive Metadata
Customer Cloud Account
Data plane
Databricks
Hopsworks Cluster
VPC Peering
VPC/VNET
Sagemaker / Azure ML Studio
TLS
S3/ADLS
Customer data
aws
Azure
Coming Soon
Google Cloud

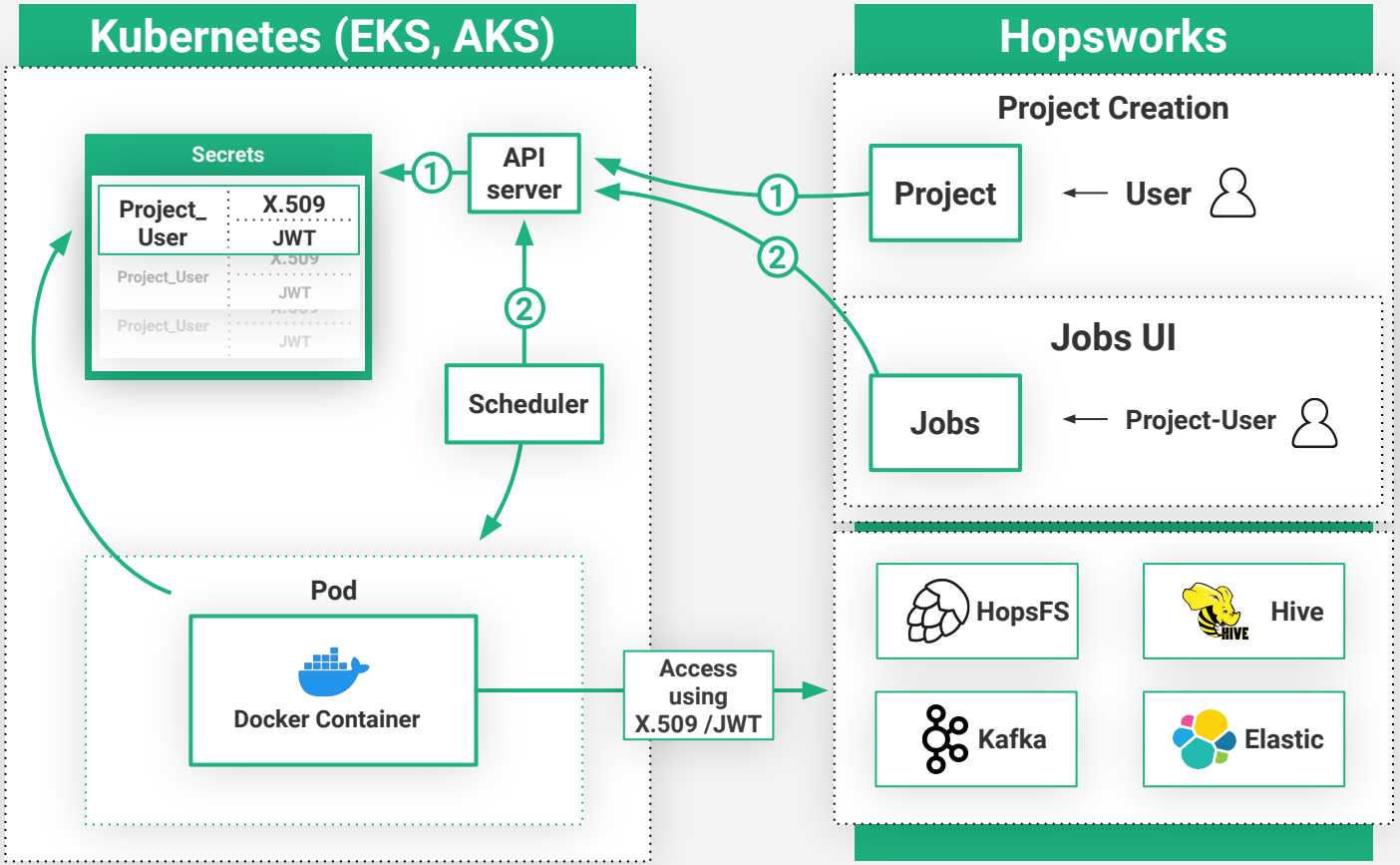
Serverless Platform on AWS - Amplify, Cognito, CloudFront, Lambdas, Route 53, DynamoDB



Lambdas

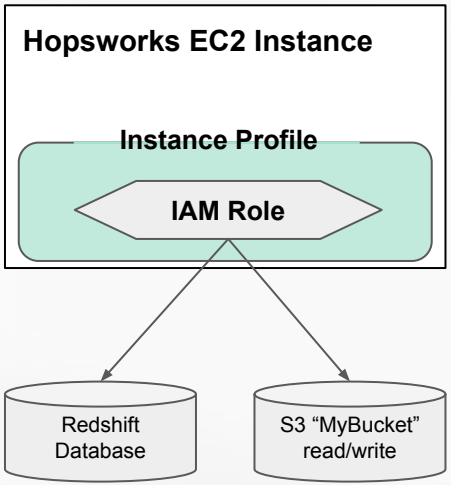

Front end
React
Hopsworks Cloud
Users
Users
Instances
Instances
Events
Triggers
Amazon
Route 53
External role
Customer AWS account
Customer VPC
Hopsworks
Let's
Encrypt

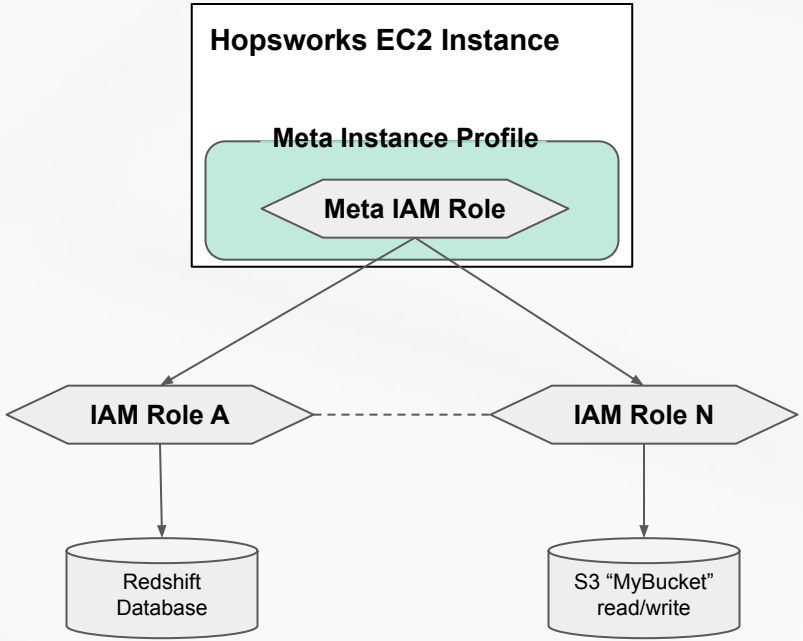
Cloud-Native Kubernetes Integration



https://www.logicalclocks.com/blog/how-we-secure-your-data-with-hopsworks

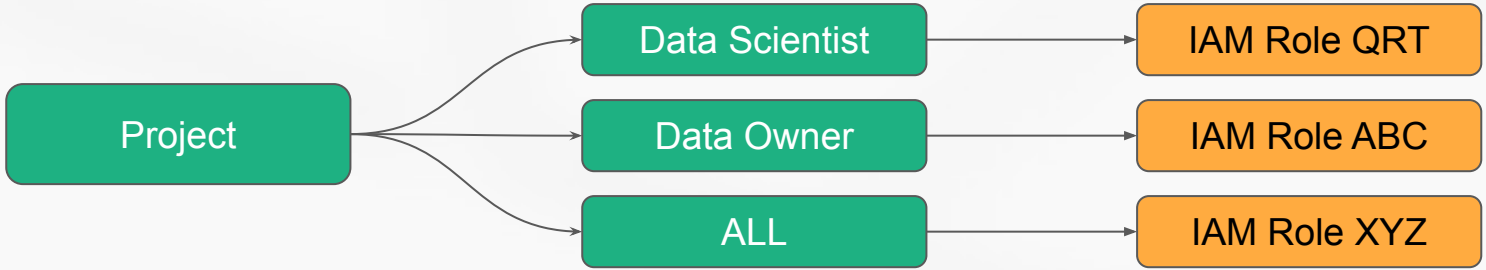
IAM Instance Profile OR IAM Role Chaining



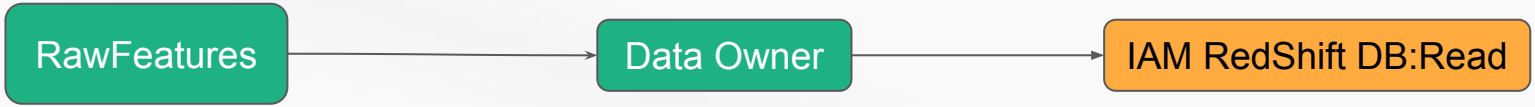


https://hopsworks.readthedocs.io/en/stable/admin_guide/cloud_role_mapping.html?highlight=iam

Configuring AWS IAM Role Chaining

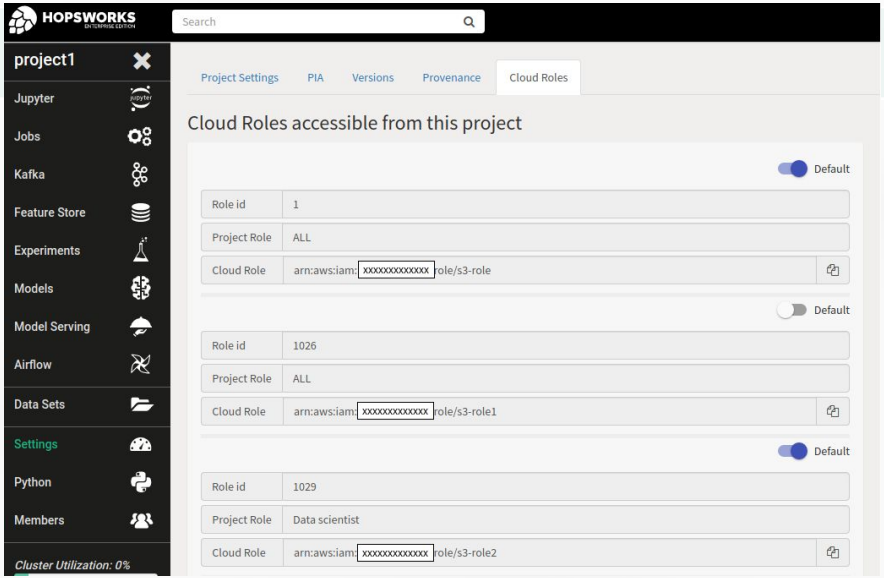


Example: Only allow admins of Project ‘RawFeatures’ to read from Redshift



https://hopsworks.readthedocs.io/en/stable/admin_guide/cloud_role_mapping.html?highlight=iam#aws-role-chaining

Assuming AWS IAM Roles




```
from hops.credentials_provider import get_role, assume_role
credentials = assume_role(role_arn=get_role(1))
spark.read.csv("s3a://resource/test.csv").show()
```

```
import io.hops.util.CredentialsProvider
val creds = CredentialsProvider.assumeRole(CredentialsProvider.getRole(1))
spark.read.csv("s3a://resource/test.csv").show()
```

https://hopsworks.readthedocs.io/en/stable/user_guide/hopsworks/assume-role.html

Metadata is **data** that describes other **data**.

Artifacts and Metadata in End-to-End ML Pipelines

Metastore (Database)

Provenance queries

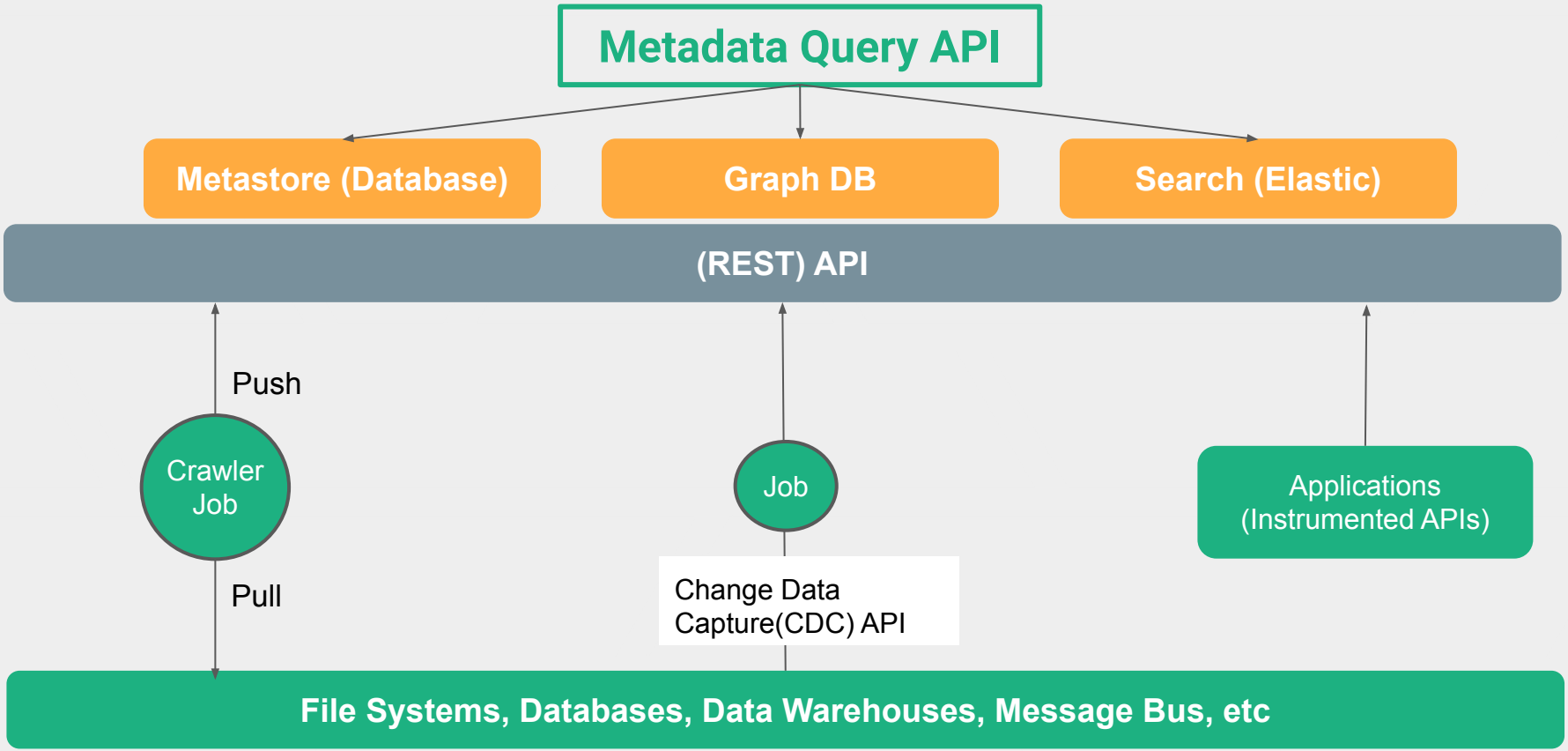
- SQL or Free-Text or Graph?
- Update Throughput?
- Latency of queries?
- Size of Metadata?

File System (S3, HopsFS, etc)

Metadata Cataloging Systems - a whole industry

https://www.dataplatformschool.com/blog/w0y8g0-the-data-governance-zoo

3 Mechanisms for Metadata Collection. Polyglot Metadata Storage for Efficient Querying.



Artifacts and Metadata in End-to-End ML Pipelines

Metastore

?

Consistency issues
Synchronization

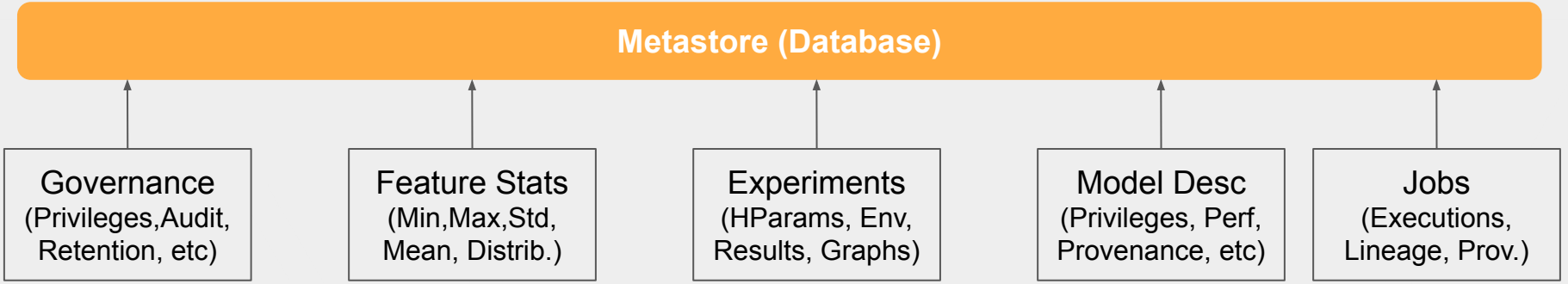
File System (S3, HopsFS, etc)

What is Metadata Revisited?

Metadata is **data** that describes other **data**.

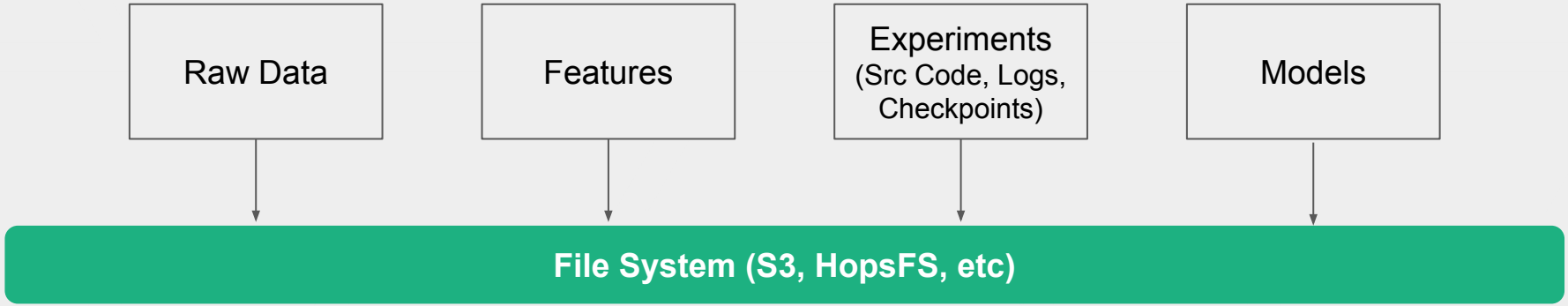
Unspoken Assumption:
Why are Data and Metadata always **separate stores**?

Artifacts and Metadata in End-to-End ML Pipelines

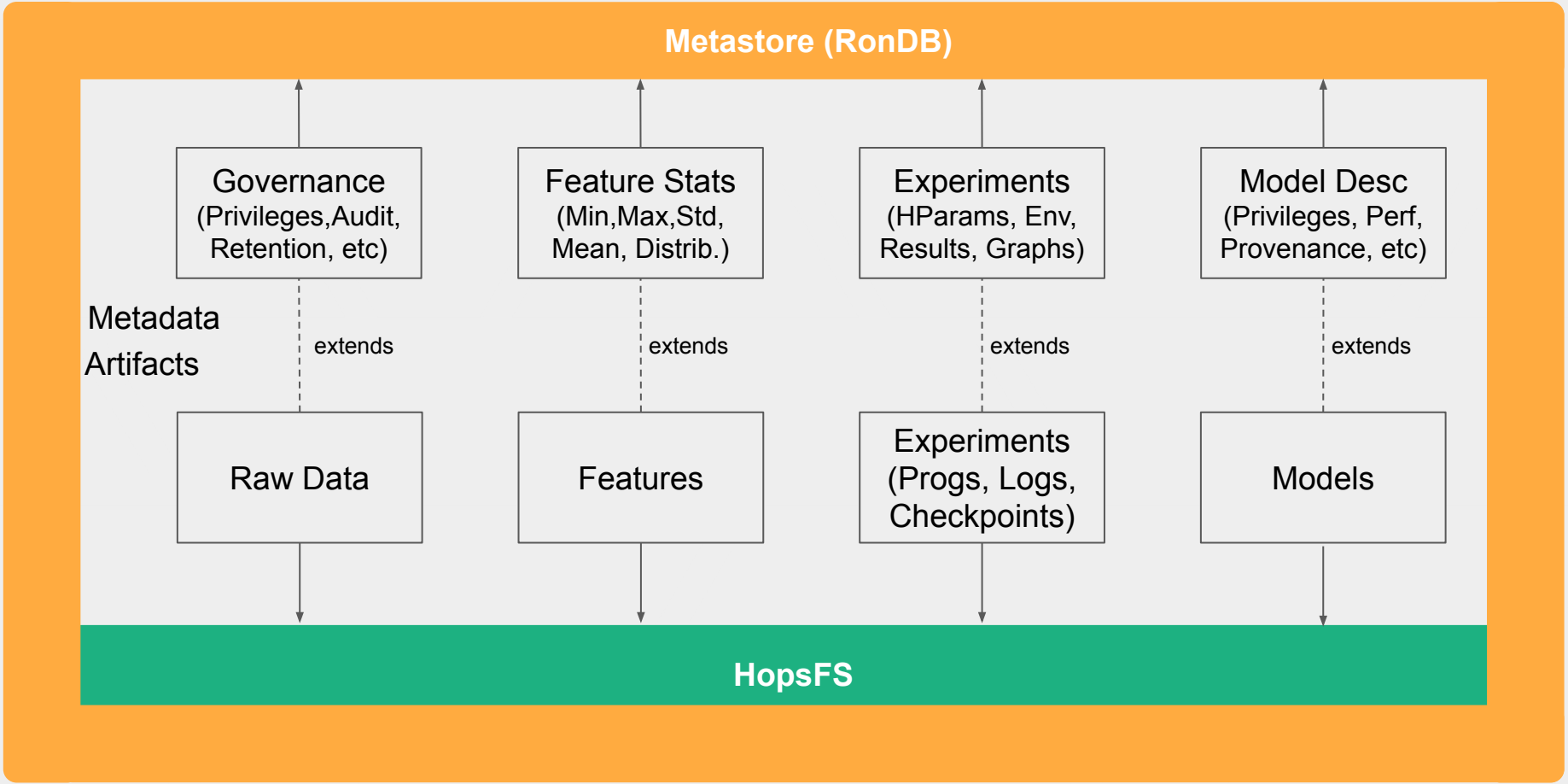


Metadata

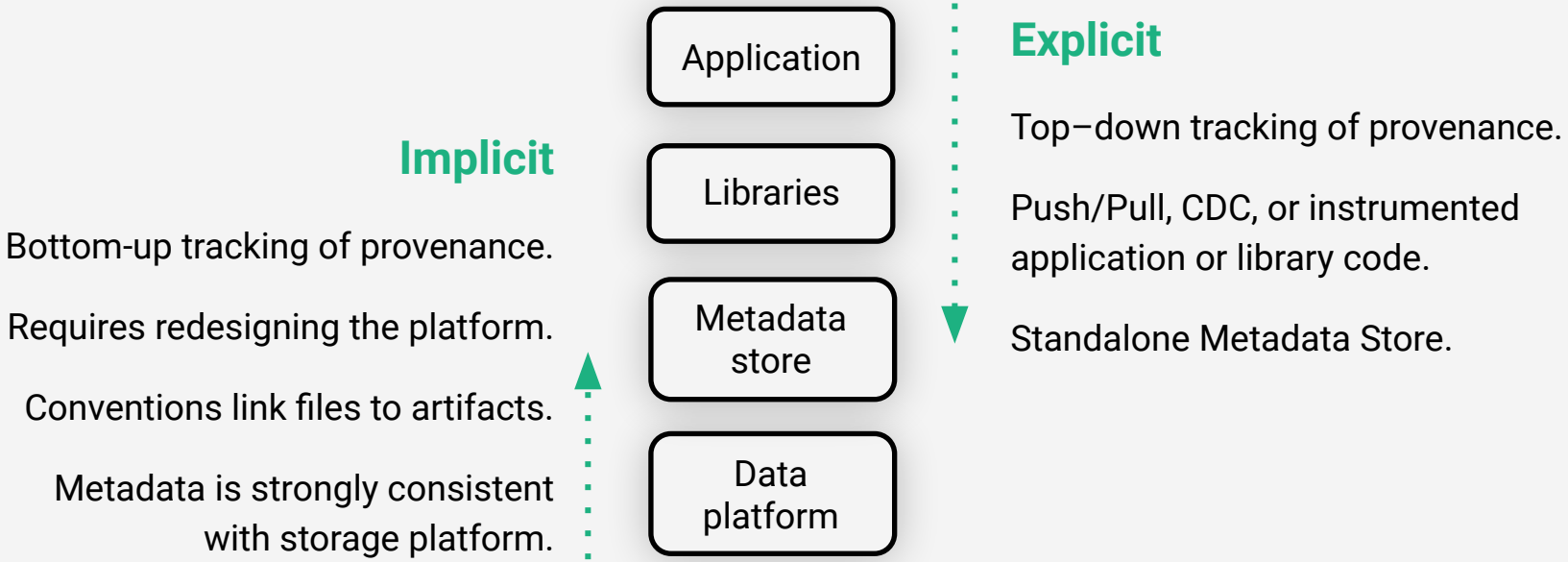
Artifacts



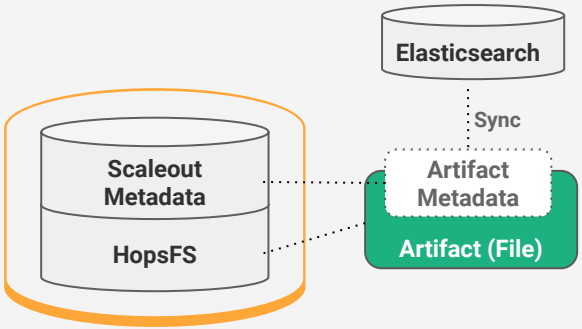
Mechanism 4: Artifacts and Metadata in the same system - a Unified Metadata Layer (Hopsworks)



Mechanism 4: Implicit Provenance



Tightly coupled Metadata and Data - replicating Metadata to External Systems



ePipe: Near Real-Time Polyglot Persistence of HopsFS Metadata

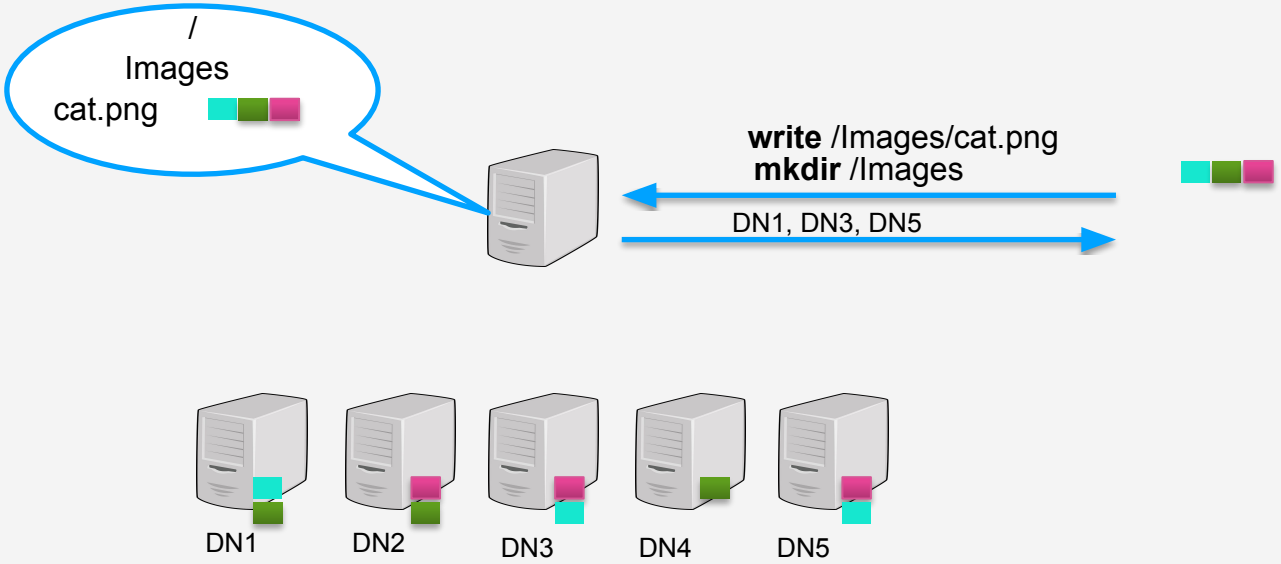
Mahmoud Ismail[1], Mikael Ronström[2], Seif Haridi[1], Jim Dowling[1]

[1] KTH - Royal Institute of Technology [2] Oracle

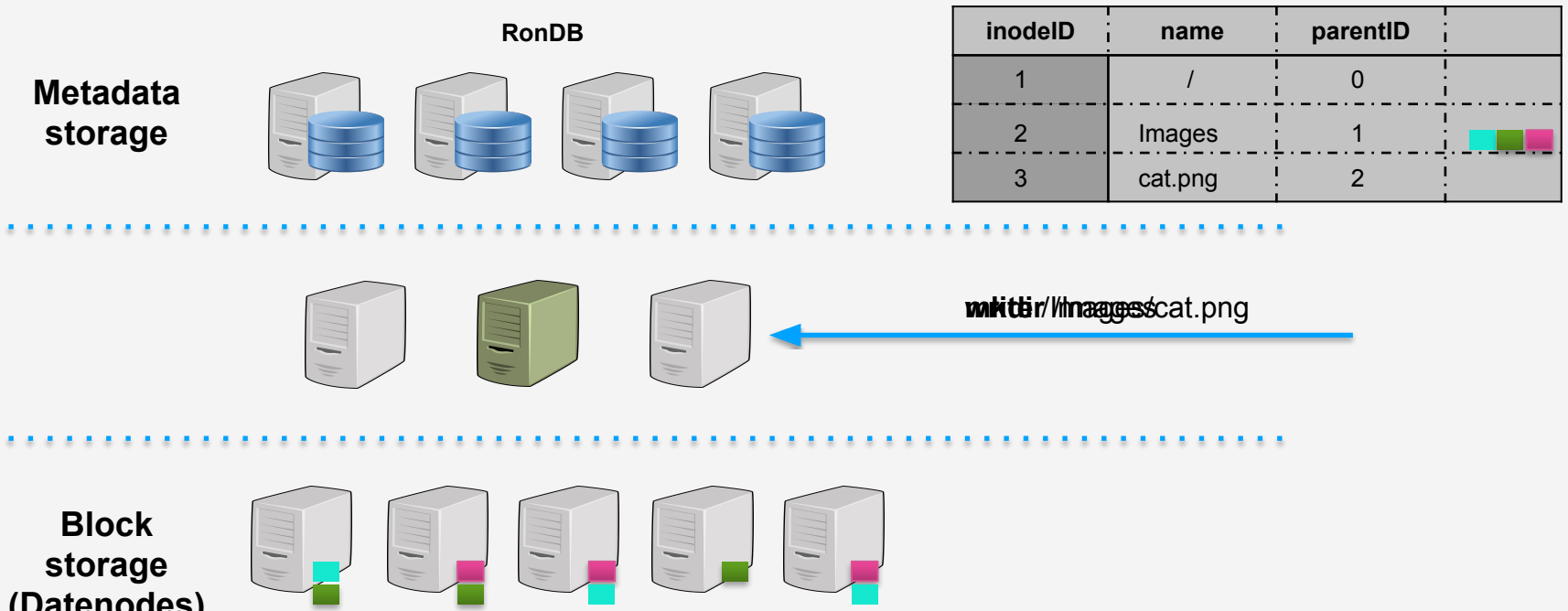
19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (IEEE/ACM CCGrid 2019), May 15th

What is HopsFS?

- Highly scalable next-generation distribution of HDFS



What is HopsFS?



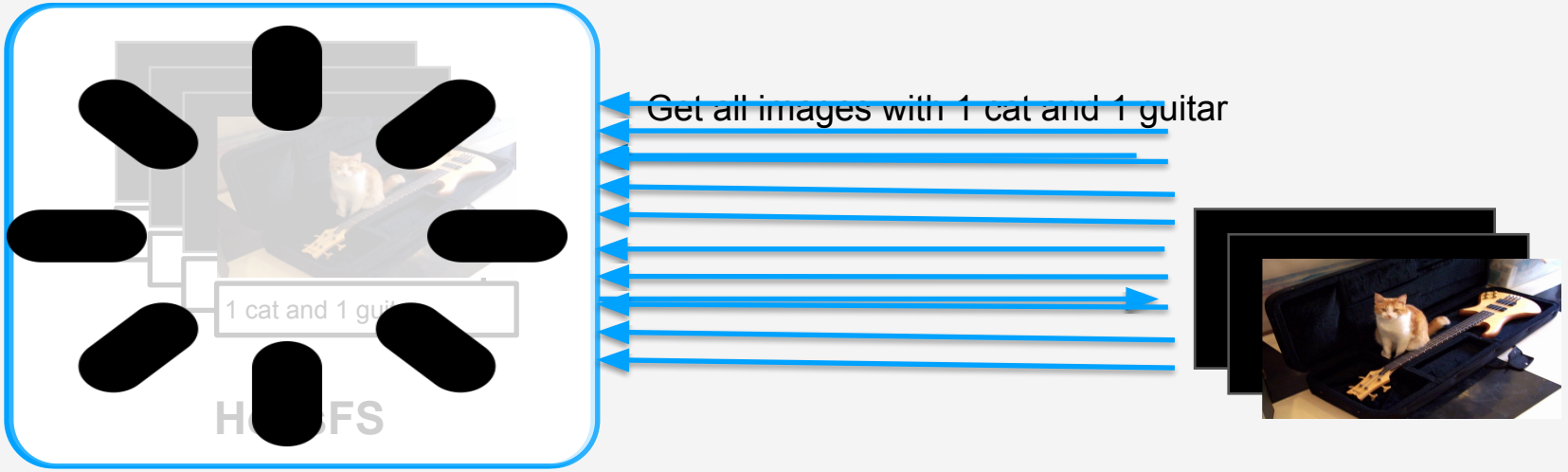
What is HopsFS?

- Drop-in replacement distribution of HDFS
- **16X** - **37X** the throughput of HDFS
- **37** larger clusters than HDFS
- **10** times lower latency

Scale Challenge Winner (2017)

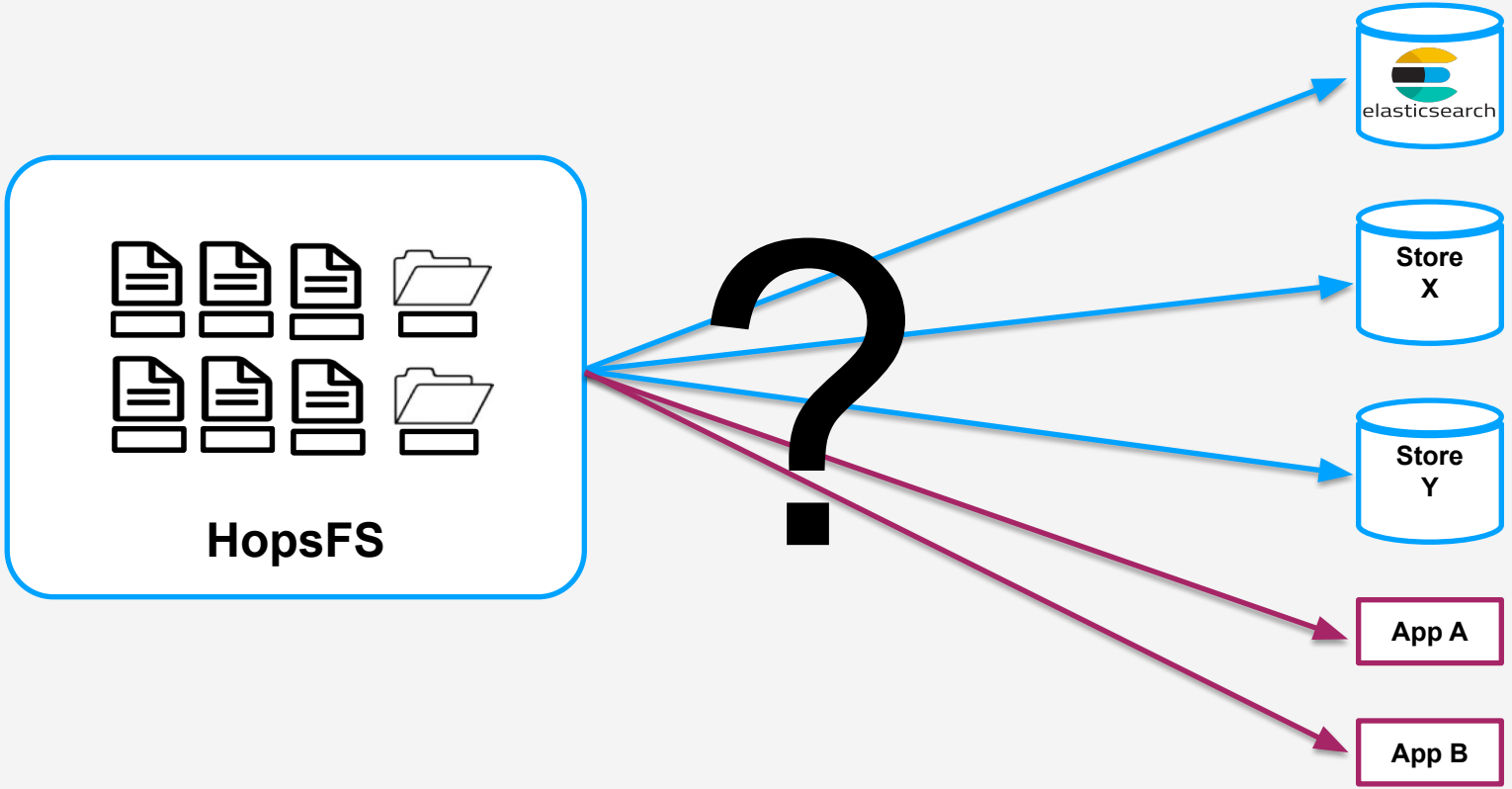
ePipe: Near Real-Time Polyglot Persistence of ~~HopsFS~~ Metadata

Search in HopsFS



Full-text search is not supported by RonDB

Polyglot Persistence - Replicating Metadata to External Systems for Efficient Querying



ePipe: Near Real-Time ~~Polyglot Persistence~~ of ~~HopsFS~~ Metadata

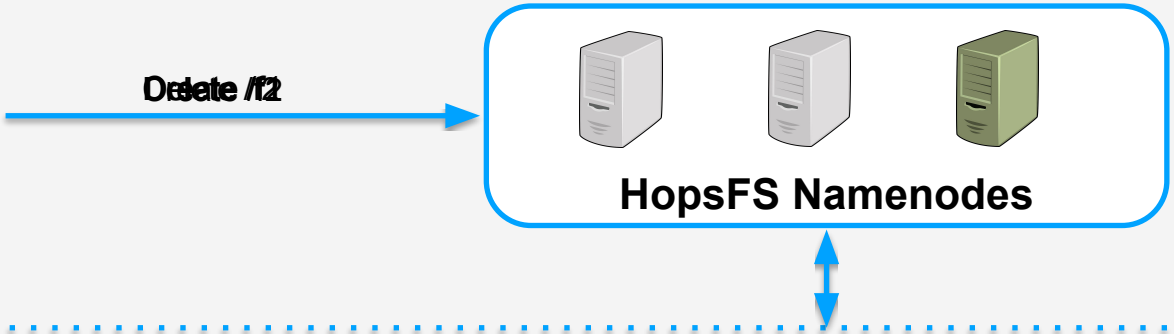
ePipe

- ePipe is a databus that provides replicated metadata as a service for HopsFS
- ePipe internally
  - creates a consistent and correctly ordered change stream for HopsFS metadata
  - and eventually delivers the change stream with low latency (sub second) (Near Real-time) to consumers

ePipe: Design Decisions

- Extend HopsFS with a logging table to log file system changes
- Leverage the RonDB event API to stream changes on the logging table to ePipe
- ePipe enriches the file system events with appropriate data and publish the enriched events to the consumers

Inodes table and logging table updated in the same Transaction to ensure Consistency/Integrity



RonDB

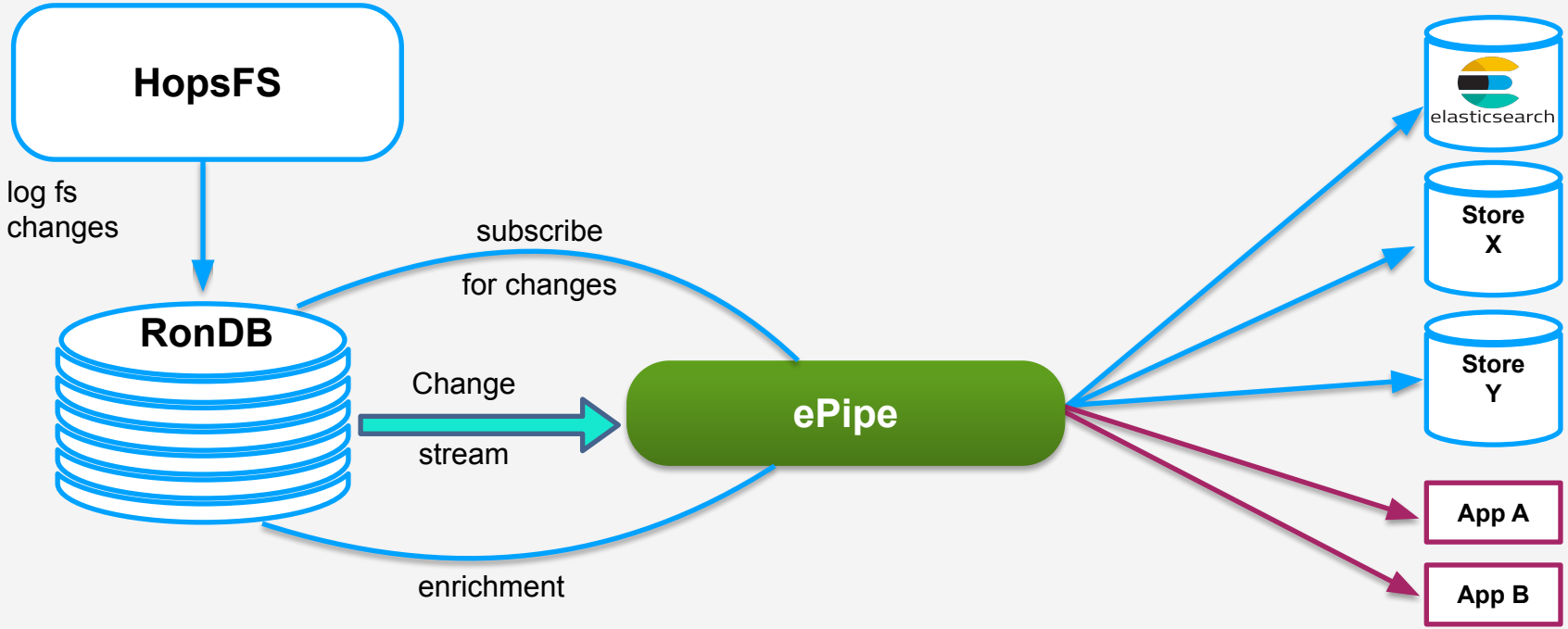
Inodes table

inodeID	name	parentID	
1	/	0	
2	f1	1	
3	f2	1	

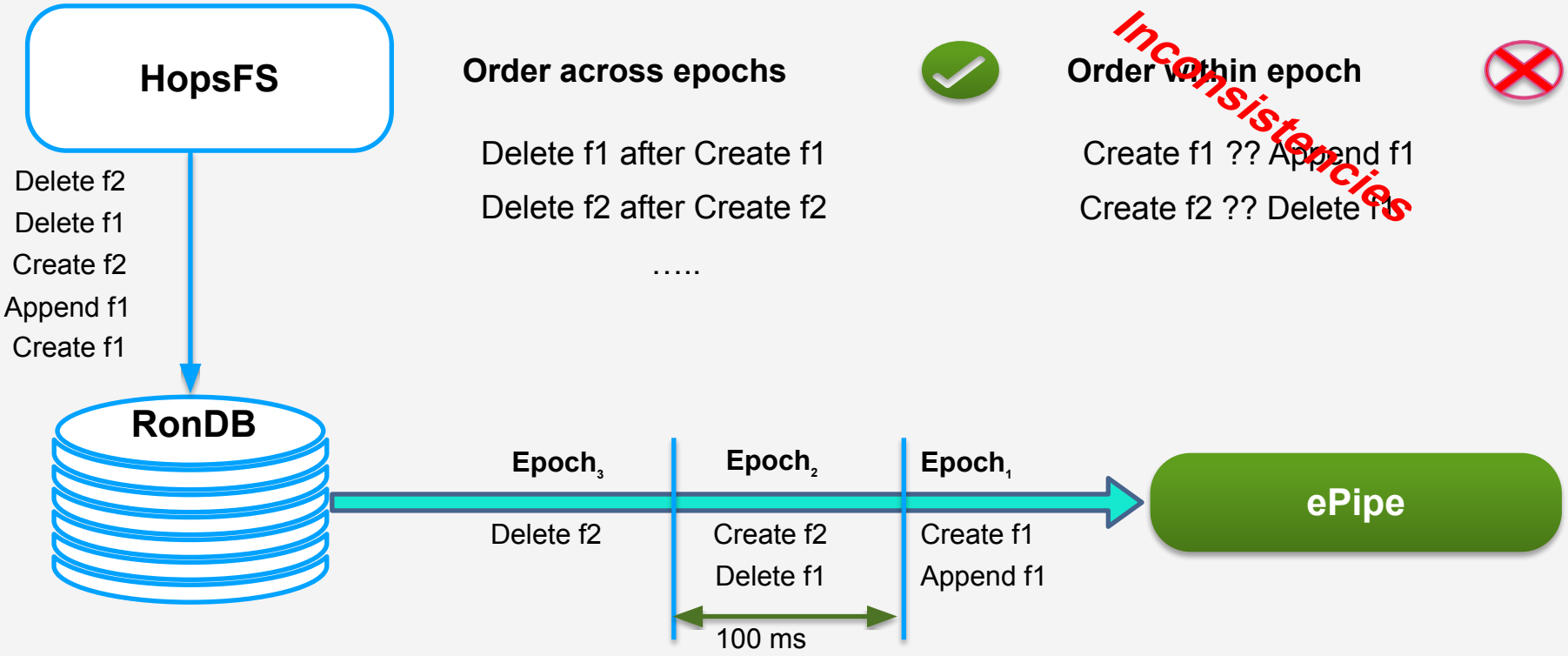
logging table

name	operation
f1	CREATE
f2	CREATE
f2	DELETE
f1	DELETE

ePipe


HopsFS
log fs changes
RonDB
subscribe
for changes
Change
stream
ePipe
enrichment
elasticsearch
Store X
Store Y
App A
App B

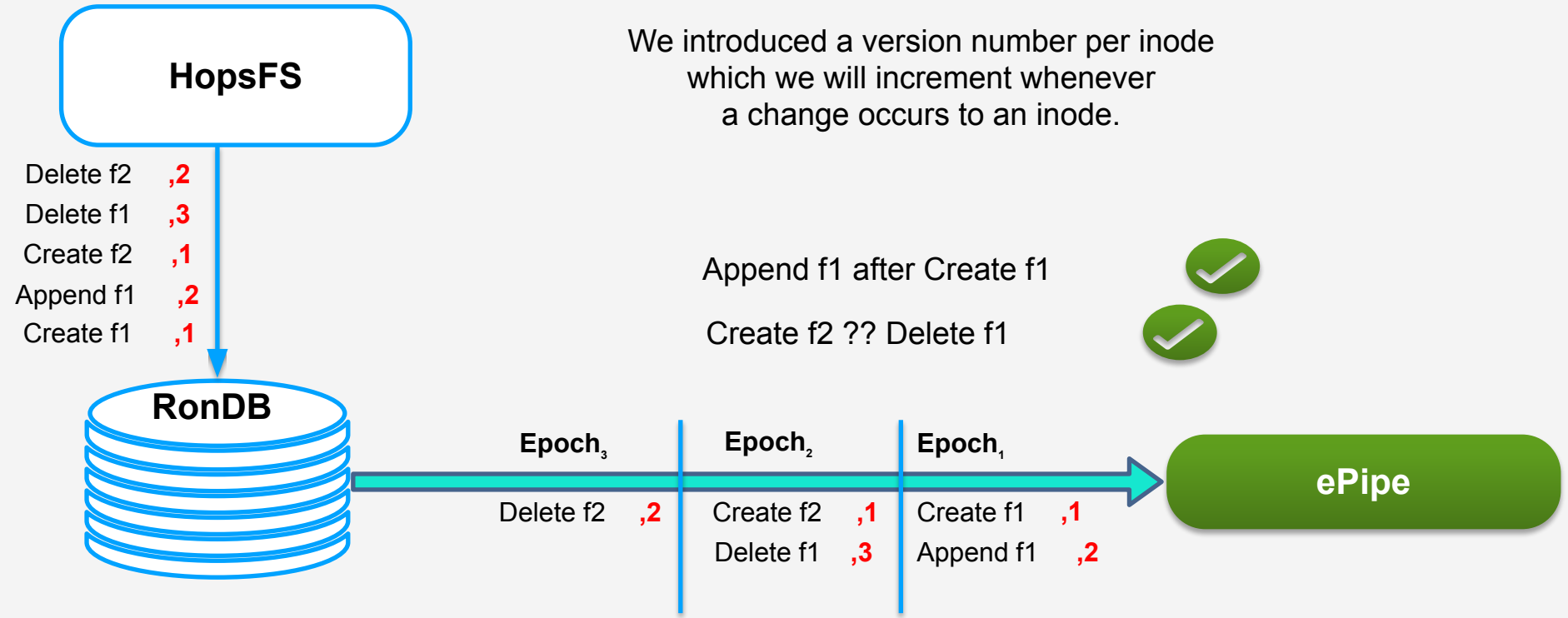
Ordering of Log Entries



NDB Ordering Properties

- **Property 1:** The epochs are totally ordered.
- **Property 2:** The changes within the same transaction happen in the same epoch.
- **Property 3:** The changes on files are ordered only if they are in different epochs, that is, no ordering is guaranteed within the same epoch.

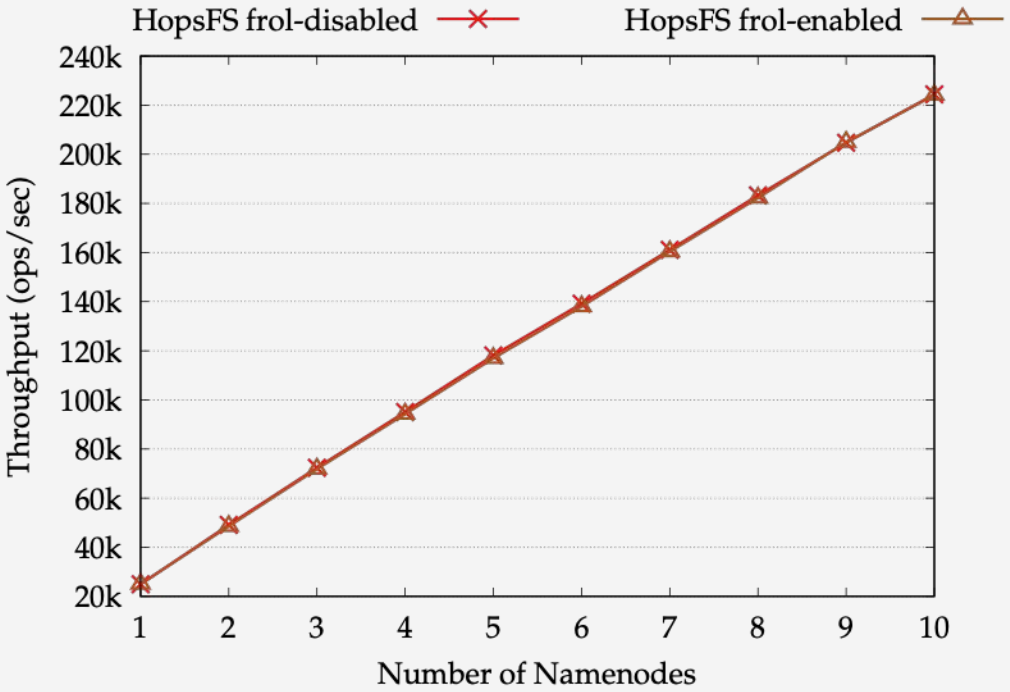
Strengthening NDB Ordering Properties



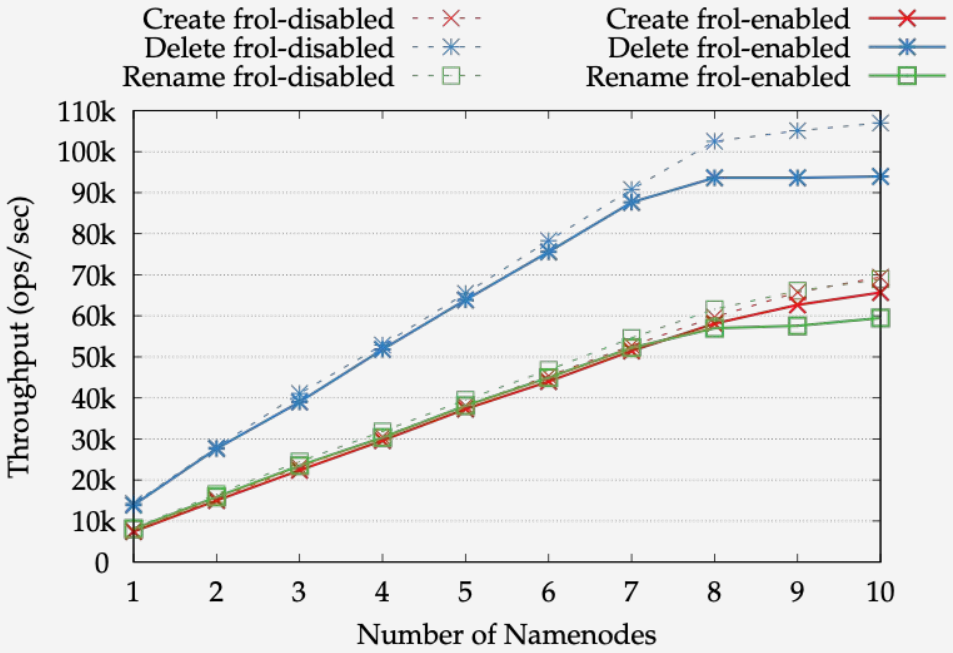
ePipe ordering Properties

- **Property 1 & 2 & 3**
- **Property 4 & 5:** The version number ensures the serializability of the changes on the same file/directory within epochs.
- **Property 6:** The order of changes for different files/directories within the same epoch doesn't matter.

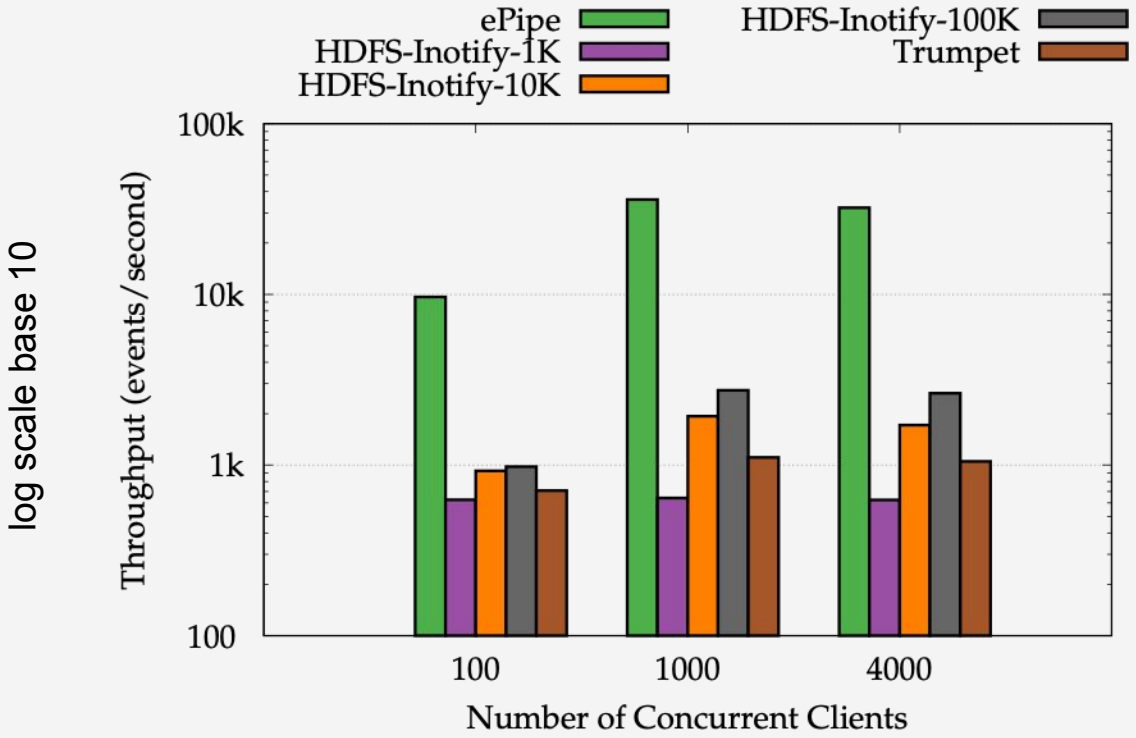
Logging overhead on HopsFS



Logging overhead on HopsFS

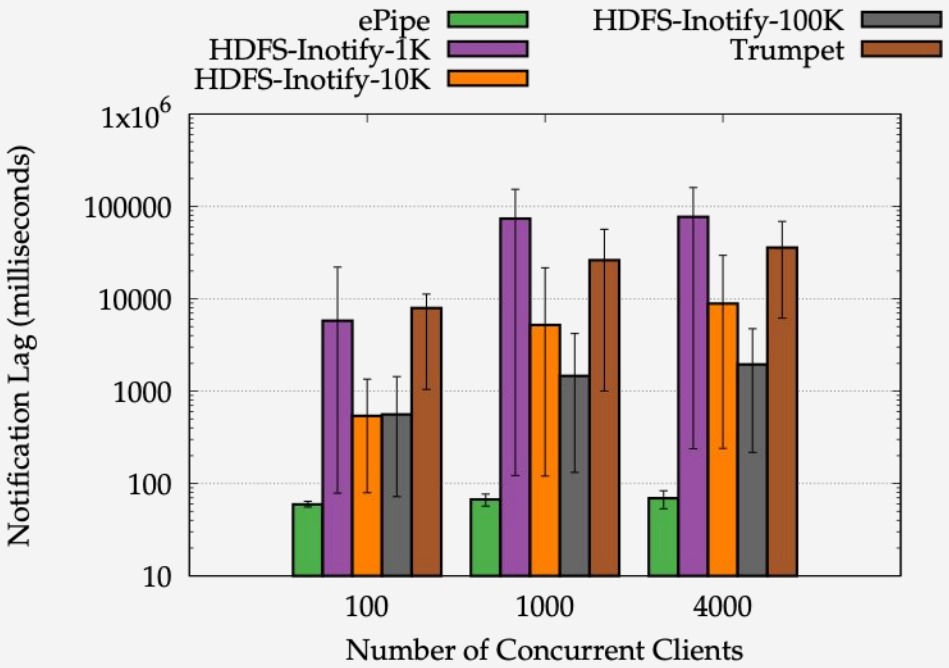


Notifications Throughput



Latency: average Lag Time

log scale base 10



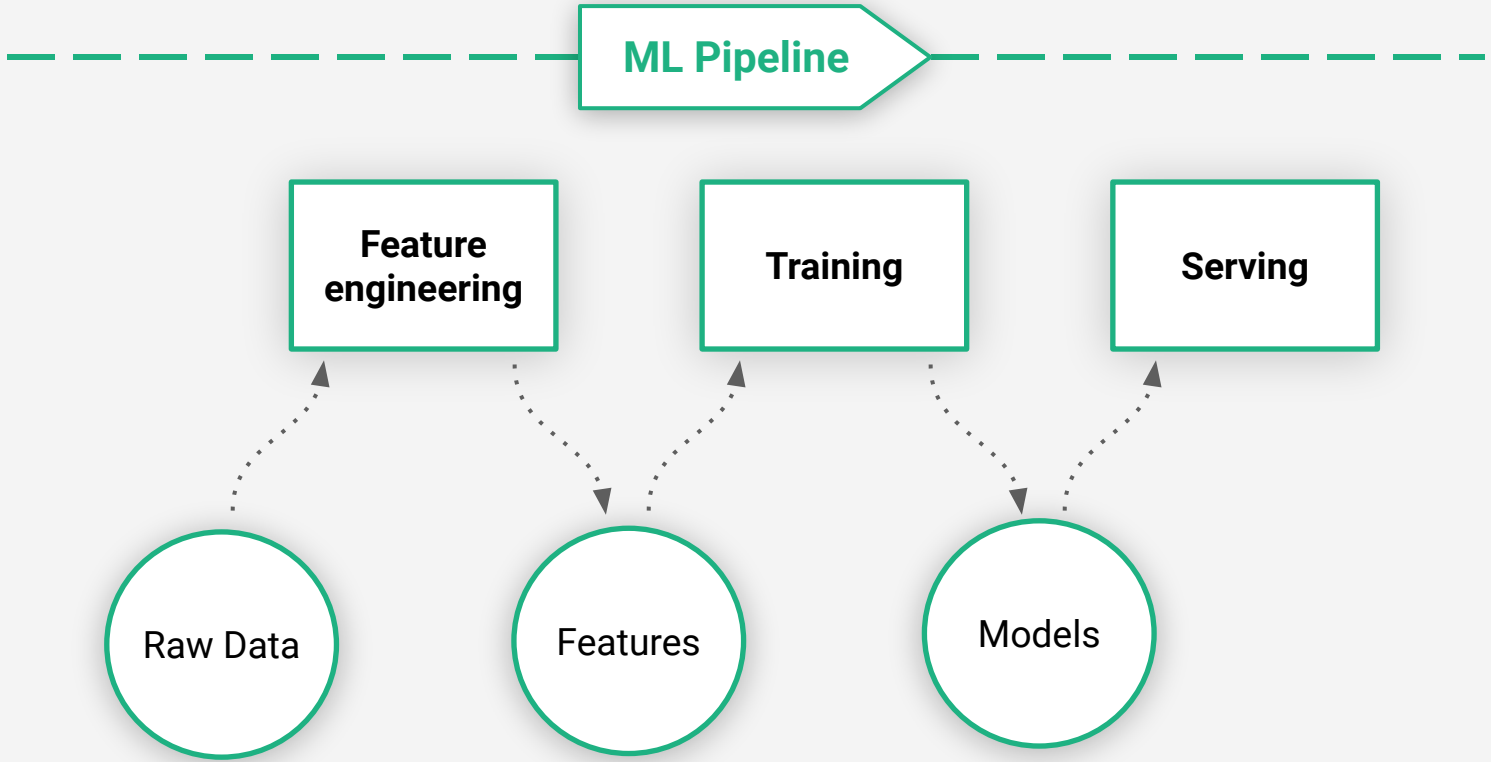
More about ePipe

- Supports failure recovery thanks to the persistent logging table
  - The log entries are deleted only once the associated events are successfully replicated to the downstream consumers.
  - At least once delivery semantics.
- Pluggable architecture
  - For example, filter events based on file name or any other attribute.
- Not Limited to HopsFS
  - Can be extended to watch for other logging tables for different purposes.

ePipe Properties

- A databus that provides replicated metadata as a service for HopsFS
- Low overhead on HopsFS
- Low replication lag (sub-second)
- High throughput
- Pluggable architecture

What is provenance - **ML Pipeline**



Provenance in ML Pipelines

- Provenance improves understanding of complex ML Pipelines.
- Provenance should not change the core ML pipeline code.
- Provenance facilitates Debugging, Analyzing, Automating and Cleaning of ML Pipelines.
- Provenance and Time Travel facilitate reproducibility of experiments.
- In Hopsworks, we introduced a new mechanism for provenance based on embedded metadata in a scale-out consistent metadata layer.

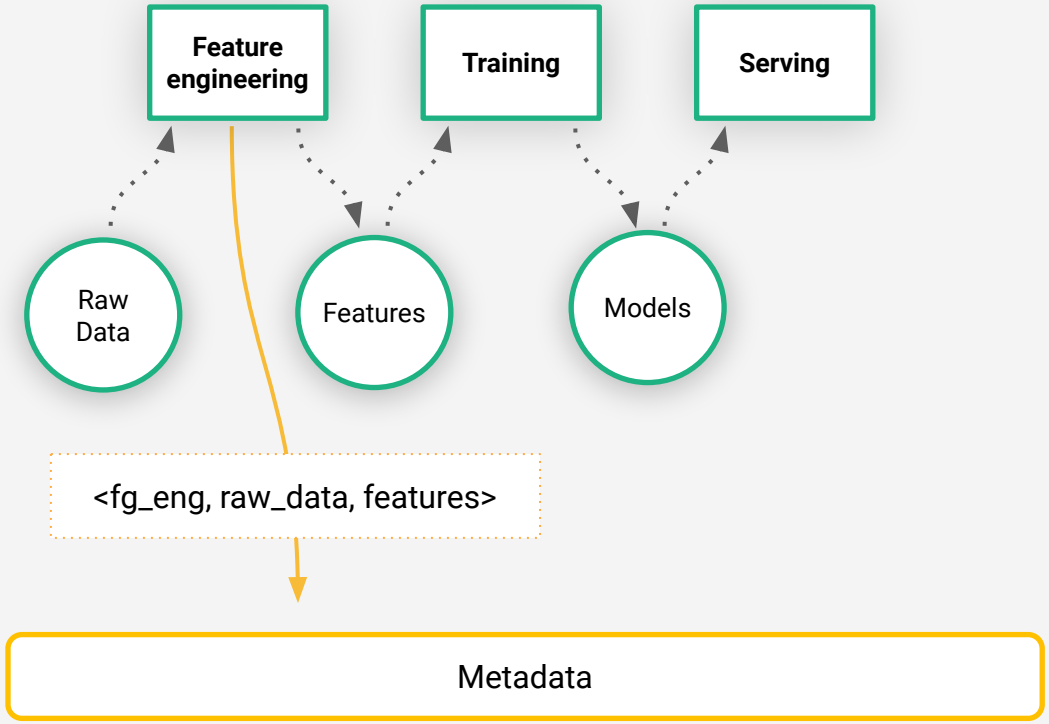
MLFlow Metadata - Explicit API calls

```
def train(data_path, max_depth, min_child_weight, estimators, model_name):
     X_train, X_test, y_train, y_test = build_data(..)
     mlflow.set_tracking_uri("jdbc:mysql://username:password@host:3306/database")
     mlflow.set_experiment("My Experiment")
     with mlflow.start_run() as run:
         ...
         mlflow.log_param("max_depth", max_depth)
         mlflow.log_param("min_child_weight", min_child_weight)
         mlflow.log_param("estimators", estimators)
         with open("test.txt", "w") as f:
             f.write("hello world!")
         mlflow.log_artifacts("/full/path/to/test.txt")
         ...
         model.fit(X_train, y_train) # auto-logging
         ...
         mlflow.tensorflow.log_model(model, "tensorflow-model",
              registered_model_name=model_name)
```

Hopsworks Metadata - Implicit Metadata

```
def train(data_path, max_depth, min_child_weight, estimators):
    X_train, X_test, y_train, y_test = build_data(..)
    ...
    print("hello world") # monkeypatched - prints in notebook
    ...
    model.fit(X_train, y_train) # auto-logging
    …
    #Saves model to ”hopsfs://Projects/myProj/models/..”
    hops.export_model(model, "tensorflow",..,model_name)
    ...
    # maggy makes an API call to track this dict
    return {'accuracy': accuracy, 'loss': loss, 'diagram': 'diagram.png'}

from maggy import experiment
experiment.lagom(train, name="My Experiment", ...)
```

What is provenance - **Metadata**

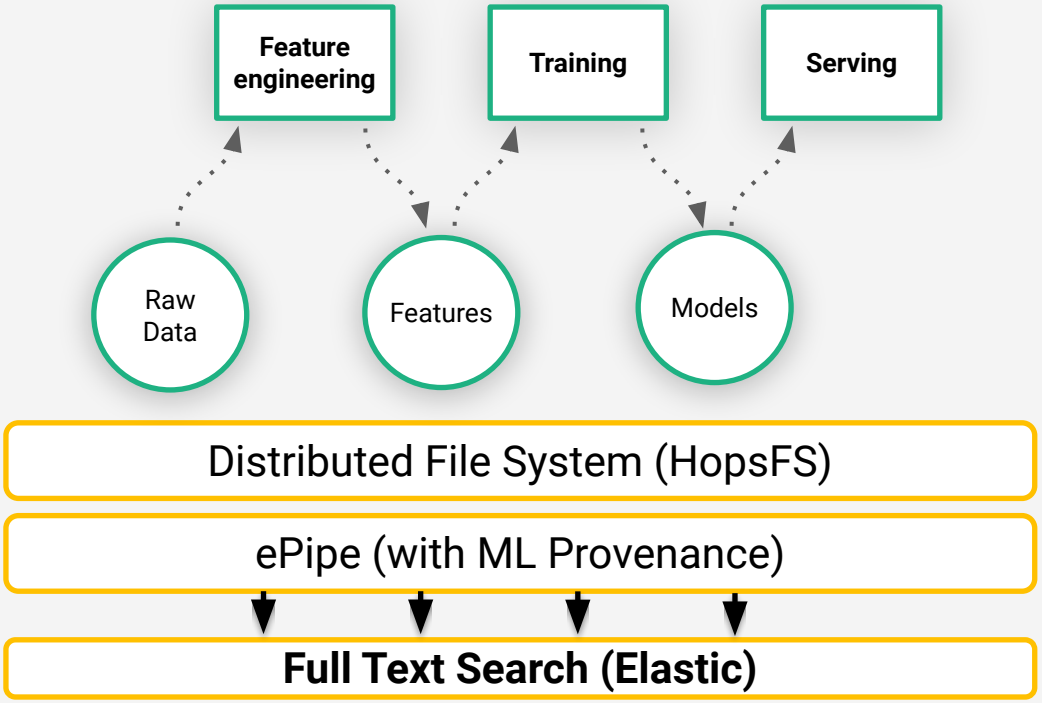


Pipeline code

In [ ]:

```
add(fg_eng, raw_data, features)
...
add(training, features, model)
```

Let the platform manage the metadata!



ML Artifacts

Features, Feature Metadata,
Train/Test Datasets
Models, Model Metadata

Possibly thousands of files

Distributed File System

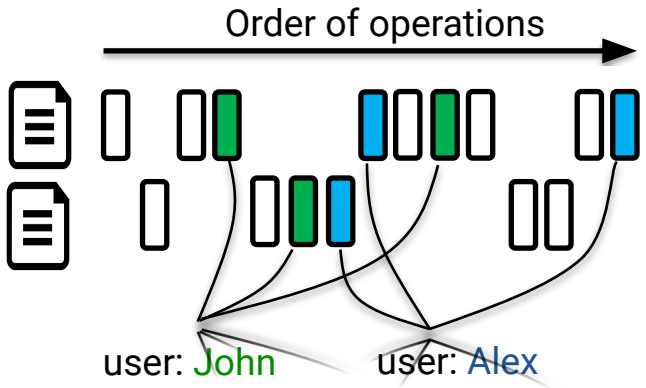
Generate thousands of operations

Change Data Capture (CDC)

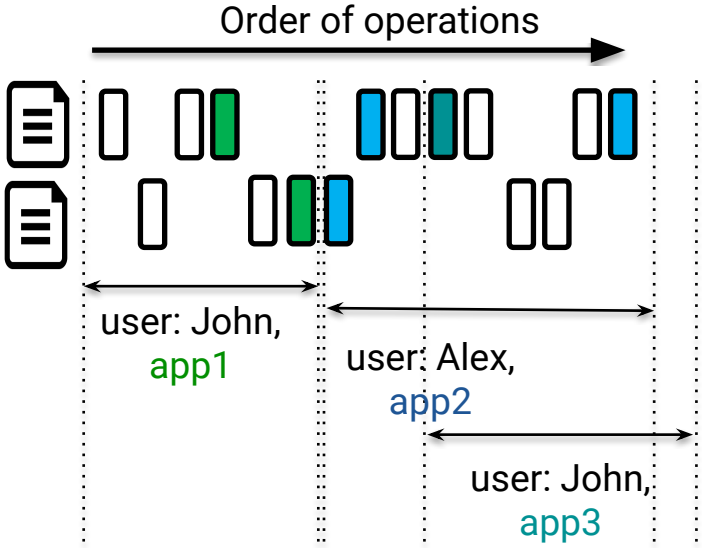
Capture only relevant operations

Richer provenance information

More context for file system operations?



Are any of these operations related?

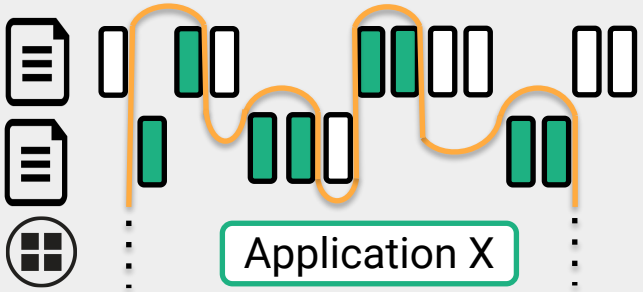


Certificates (with AppId) enabled FS Operation

Richer provenance information

Distributed File System

Read/Write/Create/Delete/XAttr/Metadata



Resource Manager - Yarn (Application Context)

Job Manager - Hopsworks (Job Context)

Workflow Manager - Airflow (Pipeline Context)

Additional Context

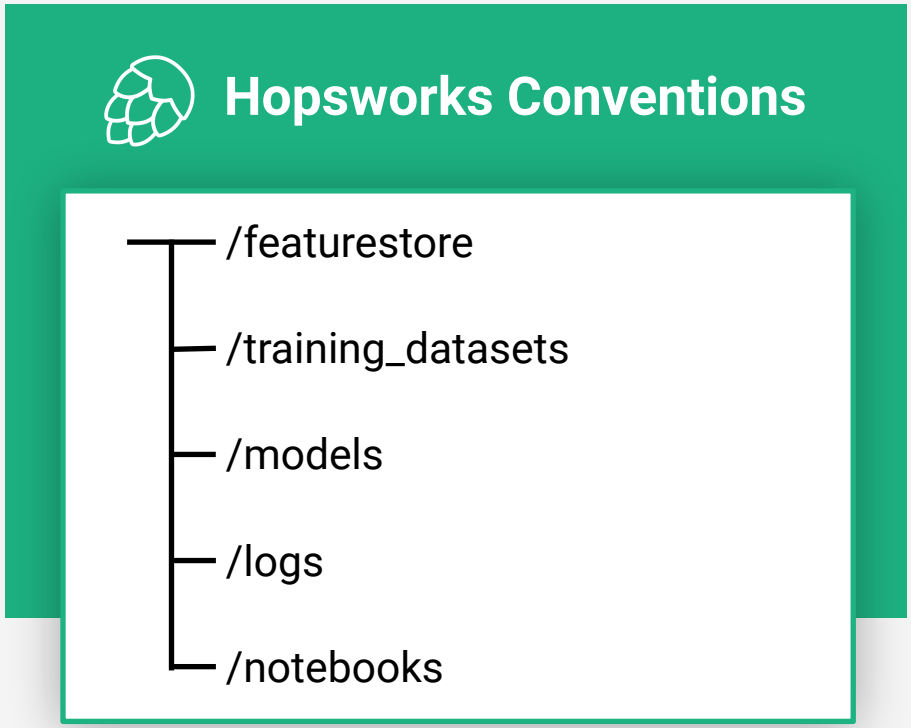
<file, op, user_id, app_id, job_id, pipeline_id>

Link input/output files via Apps

Different Executions of the same Job

Jobs as Stages of the same Pipeline

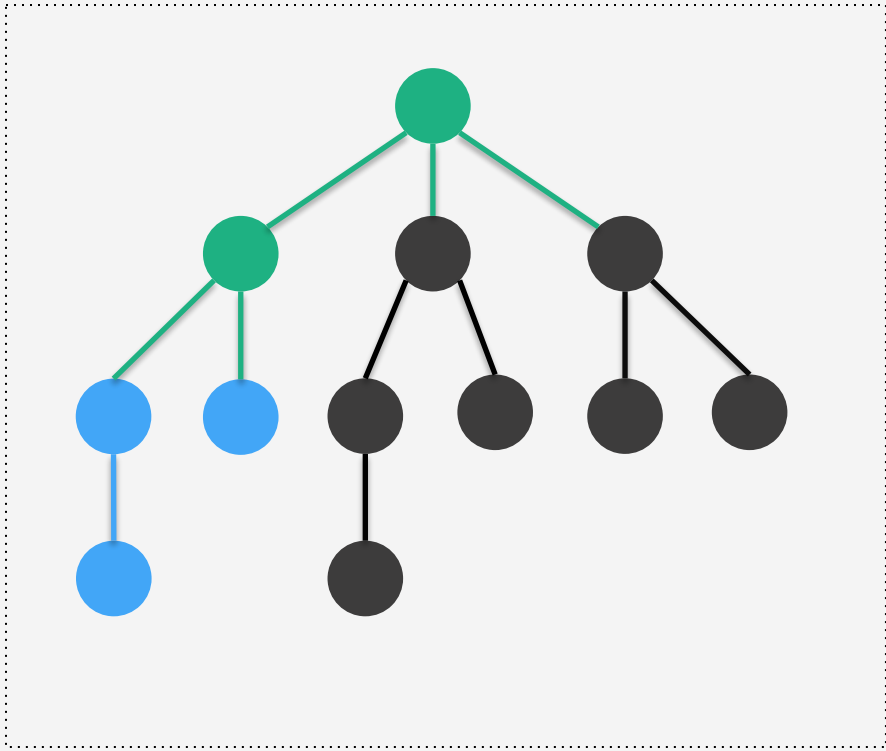
CDC API - **Filtering Mechanisms**



CDC API - **Filtering Mechanisms**

- **Path based filtering**

Example Project
- /featurestore
- /training_datasets
- /models

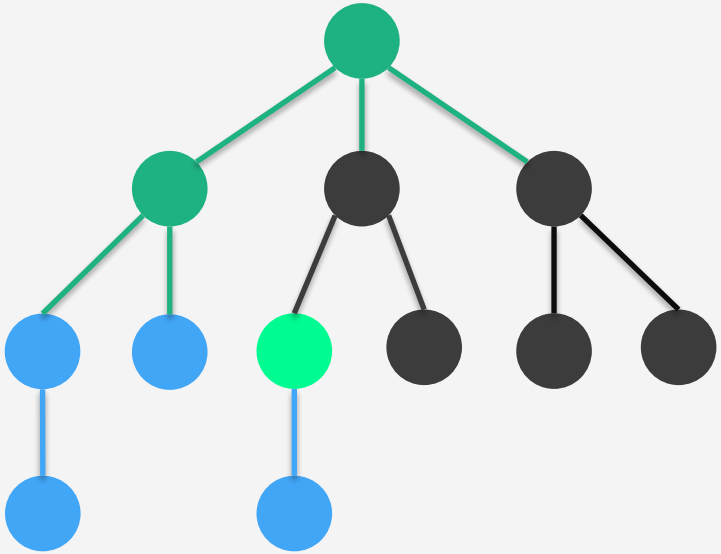


CDC API - **Filtering Mechanisms**

Path based filtering

- **Tag based filtering**

Example:
Custom metadata based on HDFS XAttr.
Tag: <tutorial>, <debug>
Tags can enable logging of all operations, if path based filtering is not easy to set



CDC API - **Filtering Mechanisms**

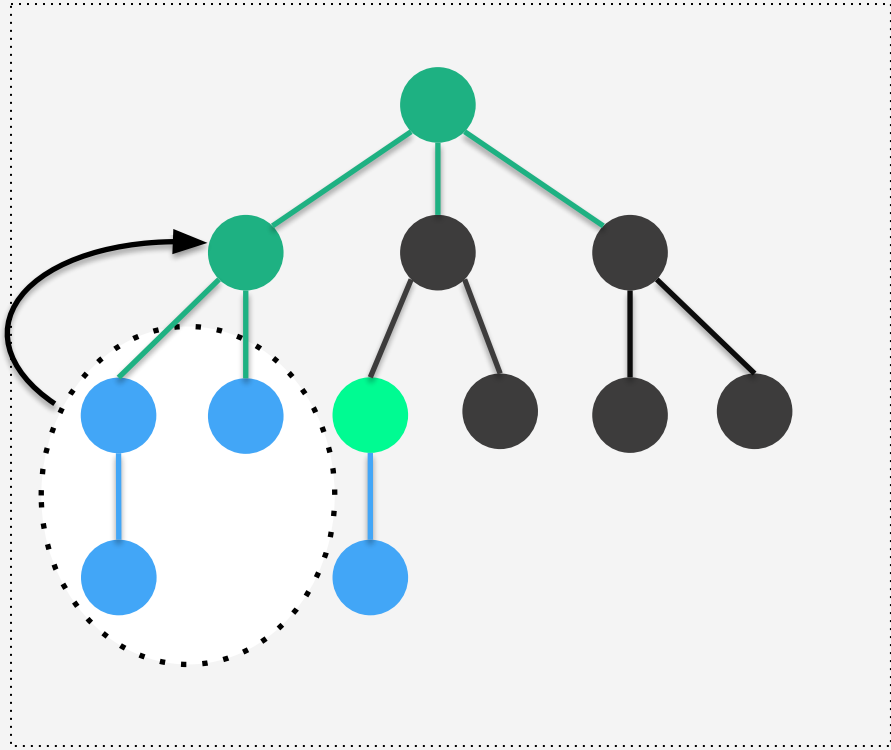
Path based filtering

Tag based filtering

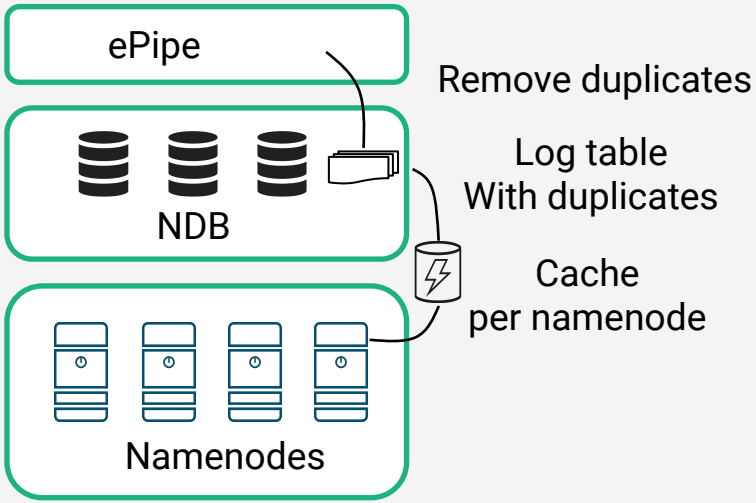
- **Coalesce FS Operations**

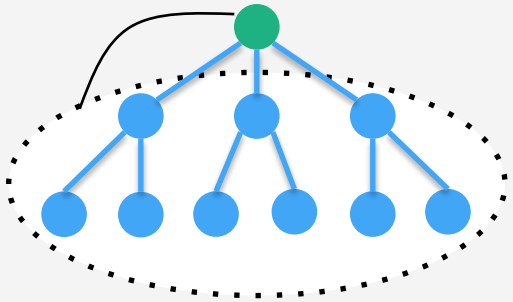
Example:
Read $file_1$
Read $file_2$
...
Read $file_n$

→ $Access_1$
Training Dataset



Optimization - **FS Operation Coalesce**





In [ ]:

```
hops.load_training_dataset(
“/Projects/LC/Training_Datasets/ImageNet”)
…
hops.save_model(“/Projects/LC/Models/ResNet”)
```

Parent Create	Artifact Create
Parent Delete	Artifact Delete
Children Read	Artifact Access
Children Create/Delete/ Append/Truncate	Artifact Mutation

CDC API - **Filtering Mechanisms**

Path based filtering

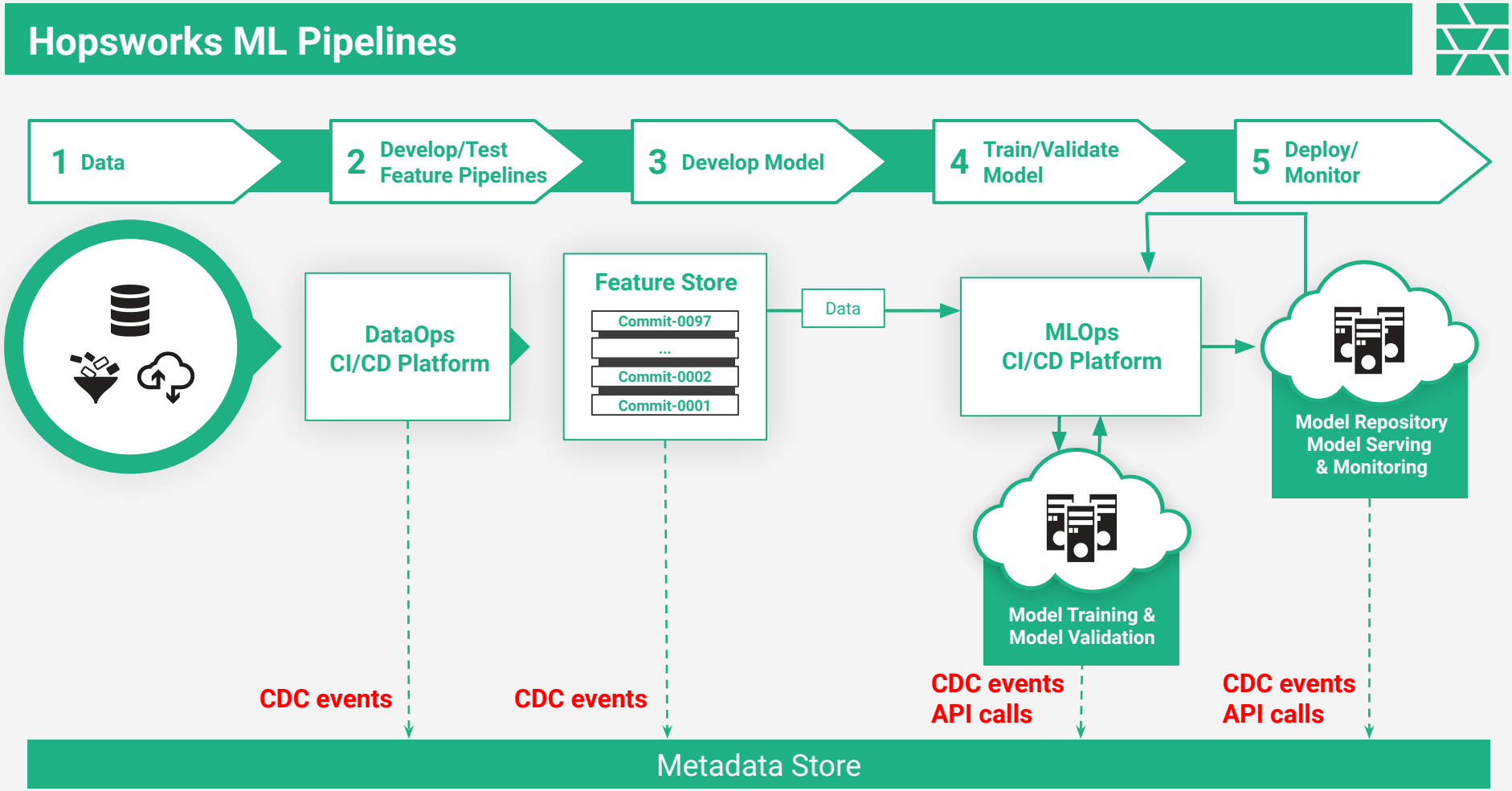
Tag based filtering

Coalesce FS Operations

- **Filtered Operations**

Filesystem Op	Metadata Stored
Create/Delete	Artifact existence
XAttr	Add metadata to artifact
Read	Artifact used by ..
Children Files Create/Delete	Artifact mutation
Append/Truncate	Artifact mutation
Permissions/ACL	Artifact metadata mutation

Hopsworks ML Pipelines



Provenance example - Helping Debug Problems in ML Pipelines

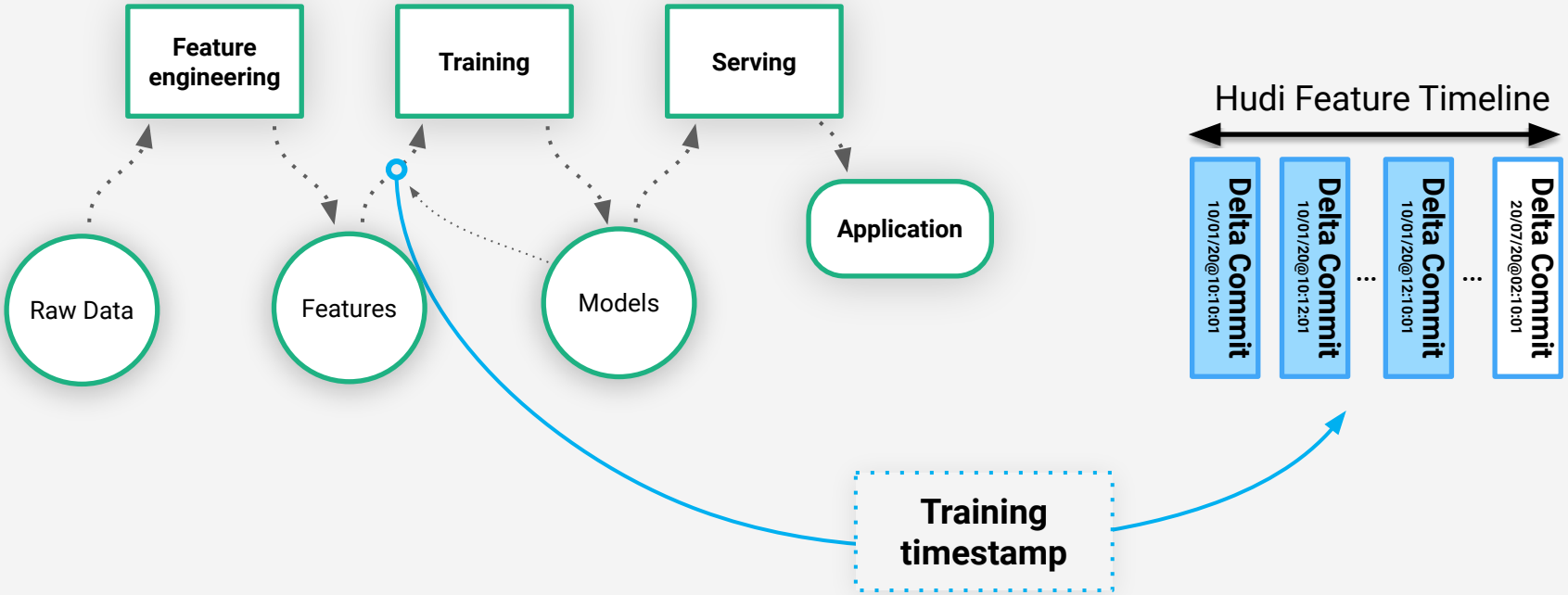
Bias Detected !

What do I do ?

Provenance + Time travel

Claim of Model Bias!

Can we determine the exact features used?



HOPSWORKS.ai

BY LOGICAL CLOCKS

@hopsworks

http://github.com/logicalclocks/hopsworks

References

- Ormenisan et al, Time-travel and Provenance for ML Pipelines, Usenix OpML 2020
- Niazi et al, HopsFS, Usenix Fast 2017
- Ismail et al, ePipe, CCGrid 2019
- Small Files in HopsFS, ACM Middleware 2018
- Ismail et al, HopsFS-S3, ACM Middleware 2020
- Meister et al, Oblivious Training Functions, 2020
- Hopsworks