DATA+AI SUMMIT
BY databricks

FABRICATOR: A DECLARATIVE FEATURE PLATFORM

DOORDASH

Hebo Yang, Kunal Shah
6/12/2024

FABRICATOR: A DECLARATIVE FEATURE PLATFORM AT DOORDASH

Agenda

- Machine Learning at Doordash
- Feature Platform Journey
- Fabricator: overview
- Architecture deep dives
- Results and learnings

MACHINE LEARNING AT DOORDASH

Restaurant

Recommendation
s

Personalized Marketing

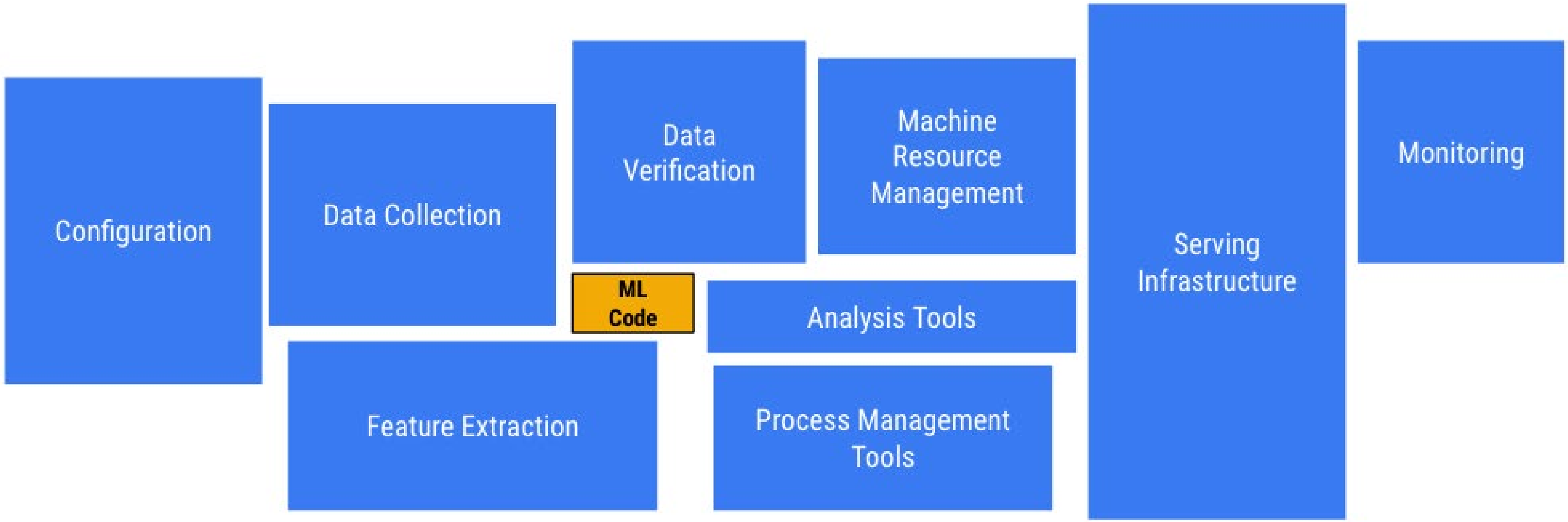
Estimated Delivery Time

Fraud Detection

Virtual Assistants

Robot and Drone Deliveries

MACHINE LEARNING SYSTEM

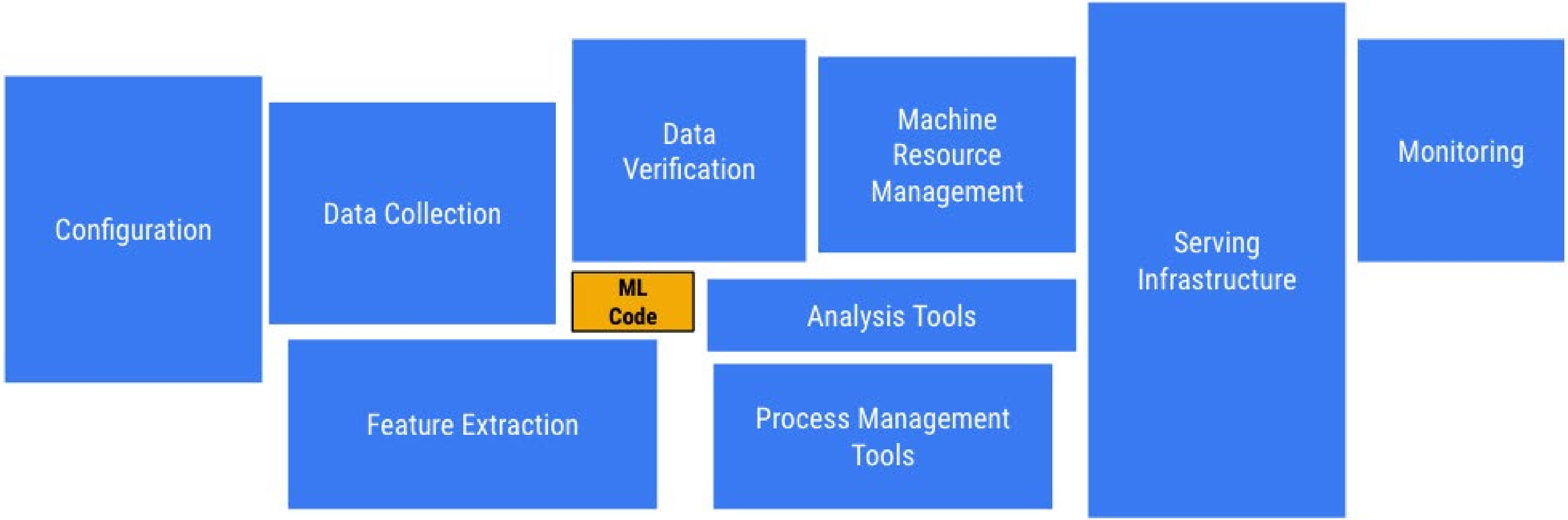


Hidden Technical Debt in Machine Learning Systems - Google 2015

MACHINE LEARNING PLATFORM

DOORDASH

Centralized team to accelerate the ML development velocity



MACHINE LEARNING PLATFORM

DOORDASH

Centralized team to accelerate the ML development velocity

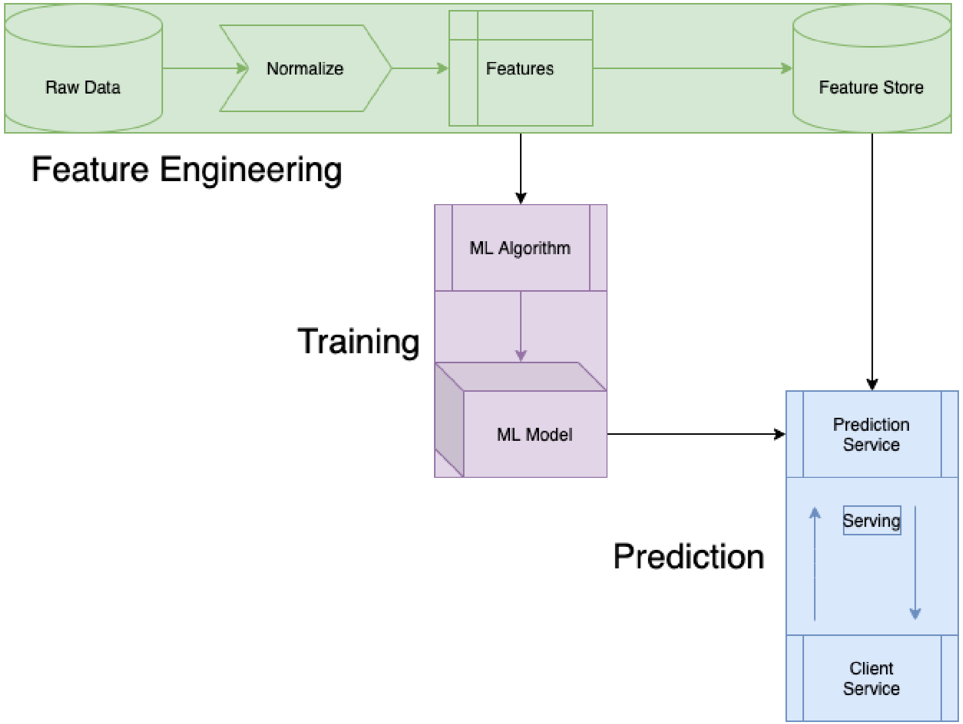
- Simplify & reduce complexity in ML development process
- Provide needed infrastructure
- Built once and leveraged by multiple ML & DS teams within the company

MACHINE LEARNING PLATFORM

DOORDASH

Centralized team to accelerate the ML development velocity

- Simplify & reduce complexity in ML development process
- Provide needed infrastructure
- Built once and leveraged by multiple ML & DS teams within the company



FABRICATOR:DECLARATIVE FEATURE PLATFORM AT DOORDASH

Agenda

- Machine Learning at Doordash
- **Feature Platform Journey**
- Fabricator: overview
- Architecture deep dives
- Results and learnings

FEATURE PLATFORM

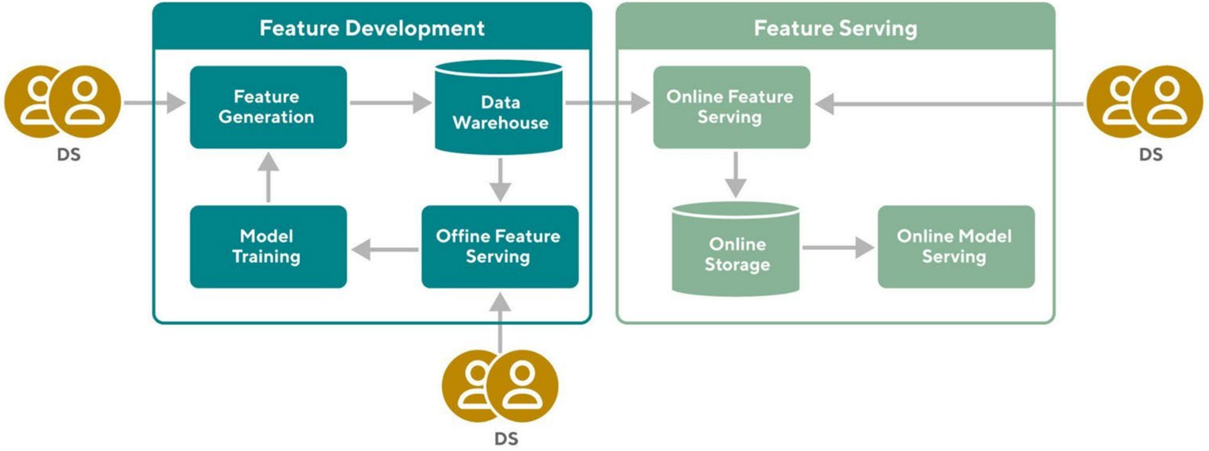
Looking back when we started in 2021

Unique Features	Batch jobs	Backfils Jobs
400	60	0

Feature Values	Realtime jobs	Users
100 B	11	20

OUR LEGACY SYSTEM

- Efficient feature store
- ETL framework with a robust warehouse
- Manual steps for everything else



PAIN POINTS

Fragmentation hampers velocity

- Data Scientists have to interface with many loosely coupled systems

Infrastructure evolution is slow

- Improving best practices and integrations takes way too long

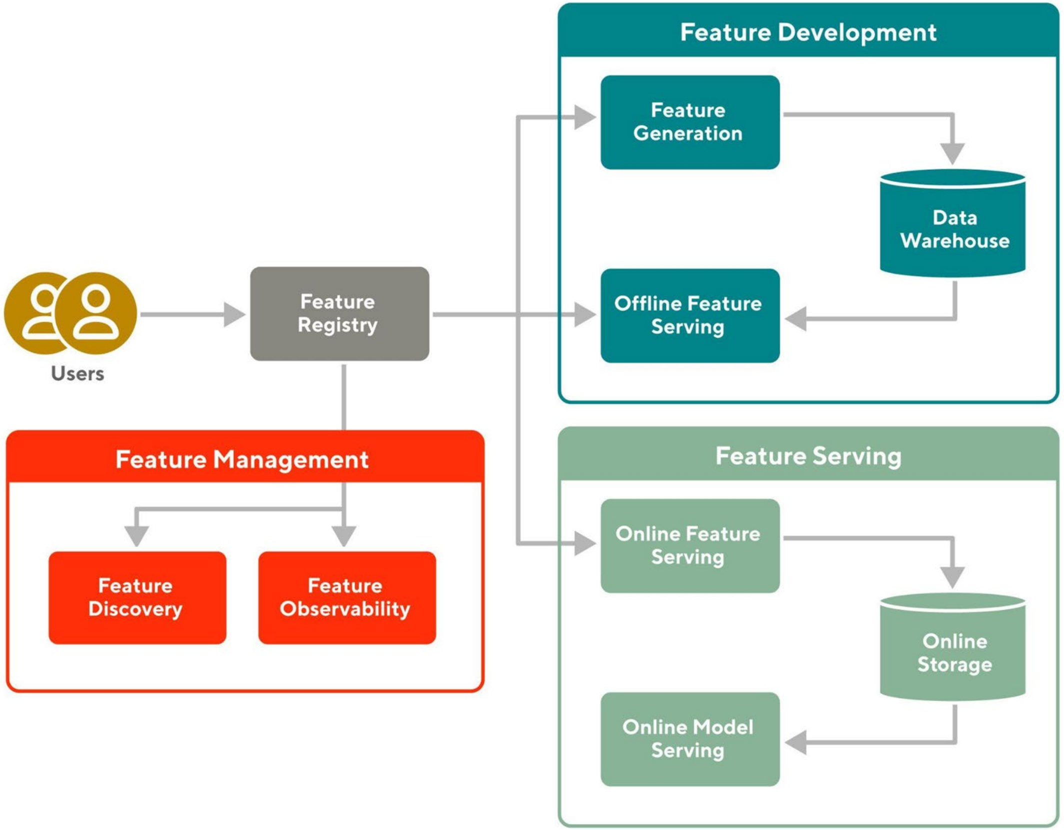
No control plane

- Maintaining features requires more than just code

WHAT DOES AN IDEAL PLATFORM LOOK LIKE?

- Single entrypoint
- Semantic feature representation
- Simplified abstractions
- High iteration velocity
- Automatic feature lifecycle management

ARCHITECTURE OF AN IDEAL PLATFORM



FABRICATOR: DECLARATIVE FEATURE PLATFORM AT DOORDASH

Agenda

- Machine Learning at Doordash
- Feature Platform journey
- **Fabricator: overview**
- Architecture deep dives
- Results and learnings

FABRICATOR VISION

Enable Data Scientists to **declaratively** define **efficient end-to-end** feature pipelines and automate the operational life cycle of features

Centralized Declarative Registry

An entrypoint that allows ML practitioners to define E2E feature semantics in simple abstractions

Unified Execution Environment

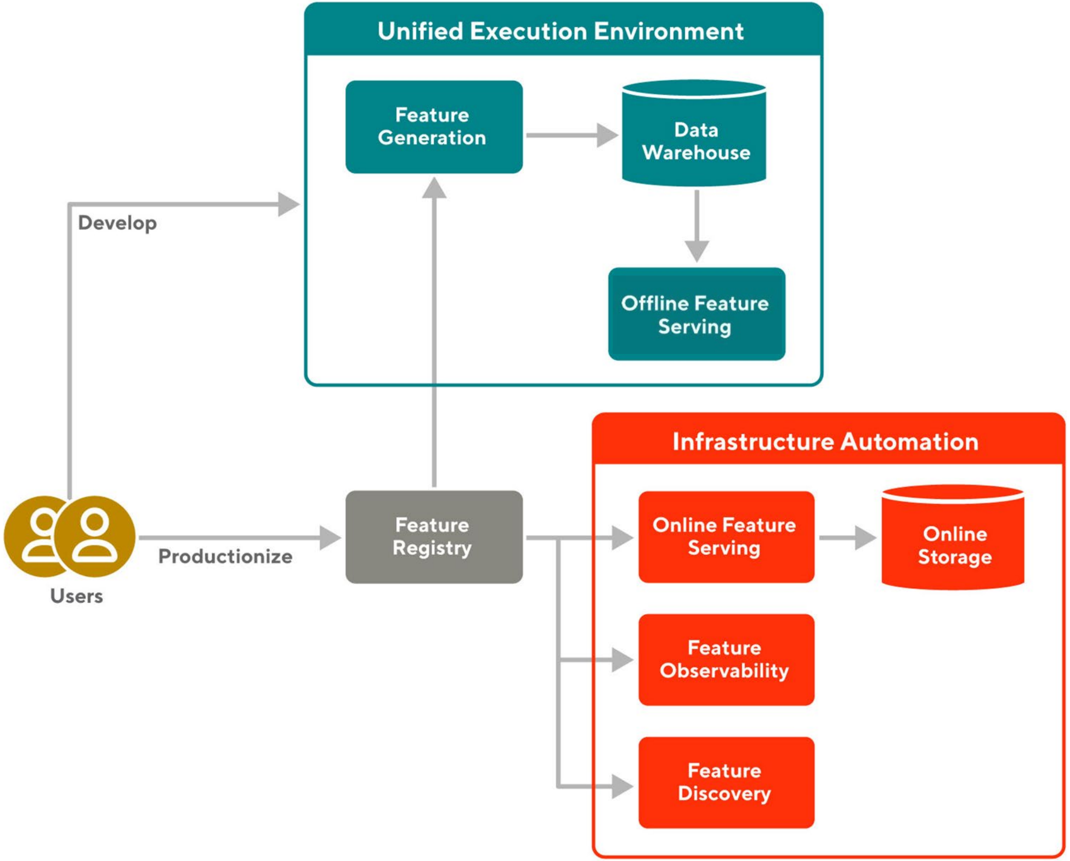
An execution environment with simple APIs for high iteration velocity

Infrastructure Automation

An automated integration for all other downstream operations

FABRICATOR ARCHITECTURE

- Registry as central entrypoint
- Unified execution env for dev and prod
- Infra automation for downstreams



FABRICATOR: DECLARATIVE FEATURE PLATFORM AT DOORDASH

Agenda

- Machine Learning at Doordash
- Feature Platform journey
- Fabricator: overview
- **Architecture deep dives**
- Results and learnings

FEATURE REGISTRY

- Simple YAML definitions for feature semantics
- Protobuf backed schema for YAML objects
- DB backed service for global access for definitions
- Continuously deployed for every change

FEATURE REGISTRY

Feature Semantics

An E2E pipeline requires only a few YAML definitions.

- Source
- Sink
- Feature

```yaml
sources:
  - name: consumer_engagement_features
    storage_spec:
      type: DELTA_LAKE
      table_name: consumer_engagement_features
    compute_spec:
      spark_spec:
        file: consumer_engagement_features.py
        resource_overrides: ...
      trigger_spec:
        upstreams:
            table_name: consumer_clicks_raw
    metadata_spec:
      user: ads-ml
      description: ...

feature_groups:
  - source: consumer_engagement_features
    features:
      - name: caf_consumre_clicks_p30d
        description: ...
      - name: caf_consumre_conversion_p30d
    materialize_spec:
      sink: ads_redis
```

FEATURE REGISTRY

Datasets

The registry also supports generating intermediate, training and validation datasets

```yaml
sources:
  - name: consumer_engagement_base
    storage_spec:
      table_name: consumer_engagement_base
      timestamp_column: active_date
      compute_spec:
        sql: >-
            select
              '{active_date}'::date as active_date
            , consumer_id
            from ...
            where ...
      trigger_spec:
        schedule: "0 6 * * *"
```

FEATURE REGISTRY

Benefits of the Design

Evolution is easy

Protobuf based backend makes our definitions robust to extension

Support for infrastructure flexibility

New storage and compute paradigms can be adopted without significant shifts

Global availability

Every downstream has immediate access to definitions

UNIFIED EXECUTION ENVIRONMENT

- Library suite that bridges registry and infrastructure
- Enables contextual executions of registry definitions
- Provides black box optimizations

UNIFIED EXECUTION ENVIRONMENT

Contextual Executions

Pythonic wrappers around YAML definitions designed to “execute” the YAMLs efficiently

```python
class FeatureContext(BaseContext):
    def __init__(
        self,
        features,
        entities,
        table_name,
        storage_type="datalake",
        env="staging",
    ):
```

```python
class SparkFeatureUpload:
    def __init__(self, context: FeatureContext):
        self.context = context
        self.df = None
```

```python
context = FeatureContext.from_source("consumer_engagement_features")
job = SparkFeatureUpload(context)
job.run()
```

UNIFIED EXECUTION ENVIRONMENT

Benefits of the Design

Most jobs are no-code

Unless you need customizations, same code executes multiple YAMLs

High fidelity testing

Notebook clusters mimic production job setup.

Efficient execution

Users don’t have to optimize for different storage or compute choices

INFRASTRUCTURE AUTOMATION

A central registry and a unified library suite to provide every downstream integration to a feature definition for free

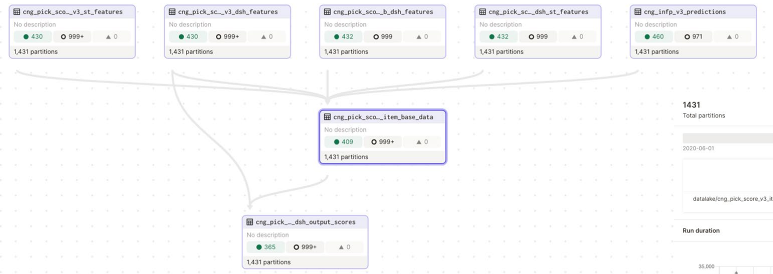
- Orchestration
- Online Serving
- Feature Discovery

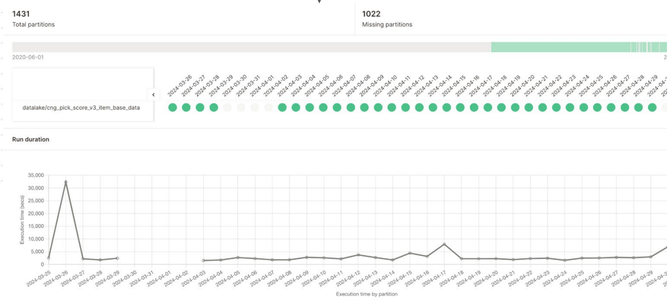
INFRASTRUCTURE AUTOMATION

Orchestration

- Automated DAG construction
- Date partitioning
- Scalable and parallelized backfilling

```yaml
sources:
  - name: consumer_engagement_features
    storage_spec:
      type: DELTA_LAKE
      table_name: consumer_engagement_features
    compute_spec:
      spark_spec:
        file: consumer_engagement_features.py
        resource_overrides: ...
      trigger_spec:
        upstreams:
          table_name: consumer_engagement_base
```



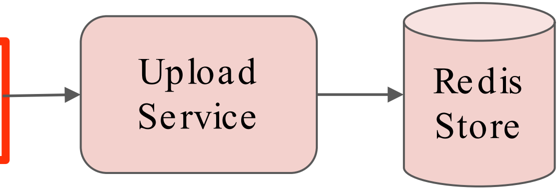


INFRASTRUCTURE AUTOMATION

Online Serving

- Automate materialization of features to our scalable feature store

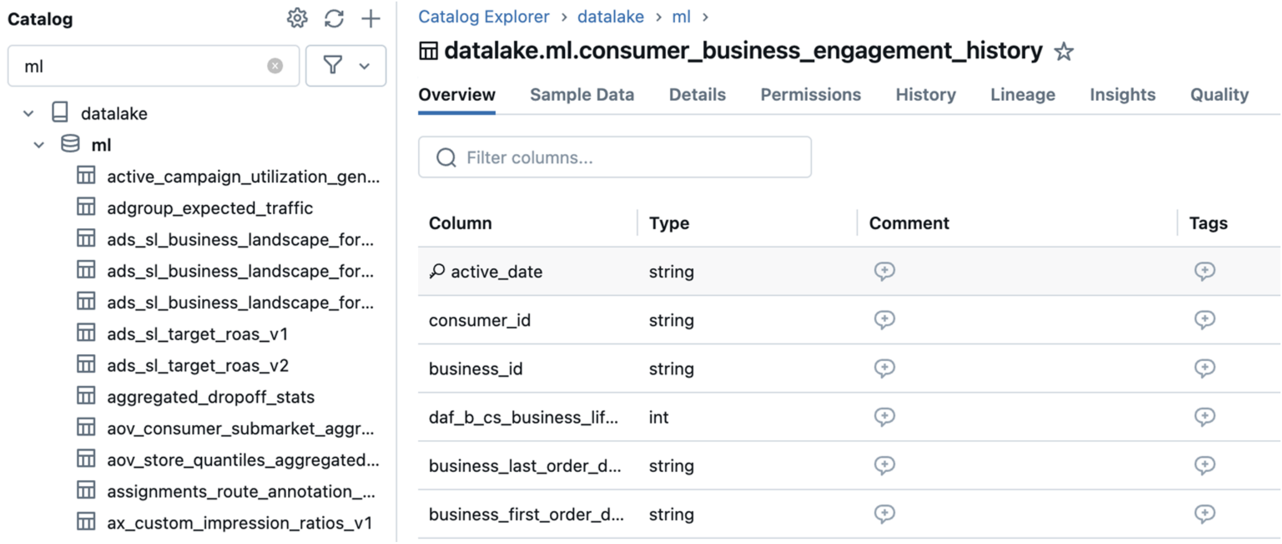
```yaml
sources:
  - name: consumer_engagement_features
    storage_spec:
      type: DELTA_LAKE
      table_name: consumer_engagement_features
    compute_spec:
      spark_spec:
        file: consumer_engagement_features.py
        resource_overrides: ...
      trigger_spec:
        upstreams:
          table_name: consumer_engagement_base
    metadata_spec:
      user: ads-ml
      description: ...

sinks:
  - name: ads_redis
    redis_spec:
      cluster_node: ads.doordash.clustercfg.usw2.cache.amazonaws.com

feature_groups:
  - source: consumer_engagement_features
    features:
      - name: caf_consumre_clicks_p30d
        description: ...
      - name: caf_consumre_conversion_p30d
    materialize_spec:
      sink: ads_redis
```



INFRASTRUCTURE AUTOMATION

Feature Discovery

- Automate registry synchronization with data catalogs
- Registry enables metadata extractors



FABRICATOR:DECLARATIVE FEATURE PLATFORM AT DOORDASH

Agenda

- Machine Learning at Doordash
- Feature Platform journey
- Fabricator: overview
- Architecture deep dives
- **Results and learnings**

FEATURE PLATFORM

Flashing back when we started in 2021

Unique Features	Batch jobs	Backfils Jobs
400	60	0

Feature Values	Realtime jobs	Users
100 B	11	20

FEATURE PLATFORM

Current daily scale on Fabricator

Unique Features	Batch jobs	Backfils Jobs
7K	1.2K	500

Feature Values	Realtime jobs	Users
2.4T	80	400

LEARNINGS

Build products, not systems

Adoption was slower when users interfaced with systems, rather than a single product

Make it easy to do the right thing

Simplify the most common patterns, and leave room for customization

Build for Integrations

Scale beyond one team by actively integrate with other services

LEARNINGS

Declaratively feature pipelines

- Very easy to author feature jobs

LEARNINGS

Declaratively feature pipelines

- Very easy to author feature jobs
- Warehouse cost and compute contention
- Spark job performance & cost
- Abandoned jobs

LEARNINGS

Declaratively feature pipelines

- Very easy to author feature jobs
- Warehouse cost and compute contention
    - team warehouses & queue
- Spark job performance & cost

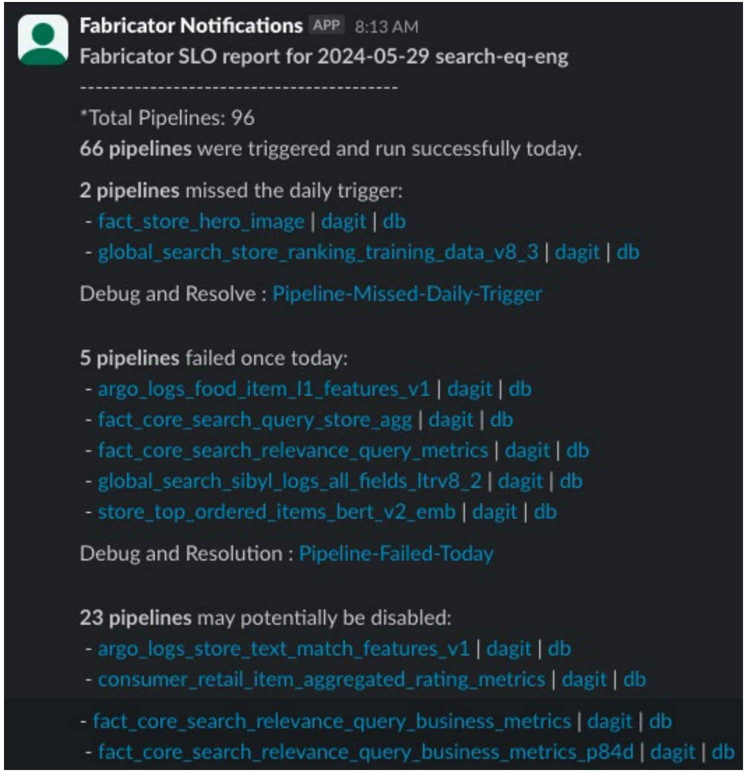
- Abandoned jobs

```yaml
groups:
  - name: ml-platform
    email: team-ml-platform@doordash.com
    slack_user: C043KKYL09H # eng-ml-mdp-alerts
    warehouse: ML_FEATURE_SERVICE
    vertical: Data Platform
    members: ...
  - name: ads-econ ...
  - name: ads-ml
    slack_user: C030M52NTA7 # ads-data-monitoring
    warehouse: ETL_BATCH_ANALYTICS_ADS_HIGH
    vertical: Ads
    members:
      - name: tony.x
      - name: andy.f
      - name: stanley.t
```

LEARNINGS

Declaratively feature pipelines

- Very easy to author jobs
- Warehouse cost and compute contention
    - team warehouses & queue
- Spark job performance & cost
    - spark tuning guidelines
    - auto optimizations
    - attribution & reporting
- Abandoned jobs
    - auto disable failing jobs
    - lineage with active model



LEARNINGS

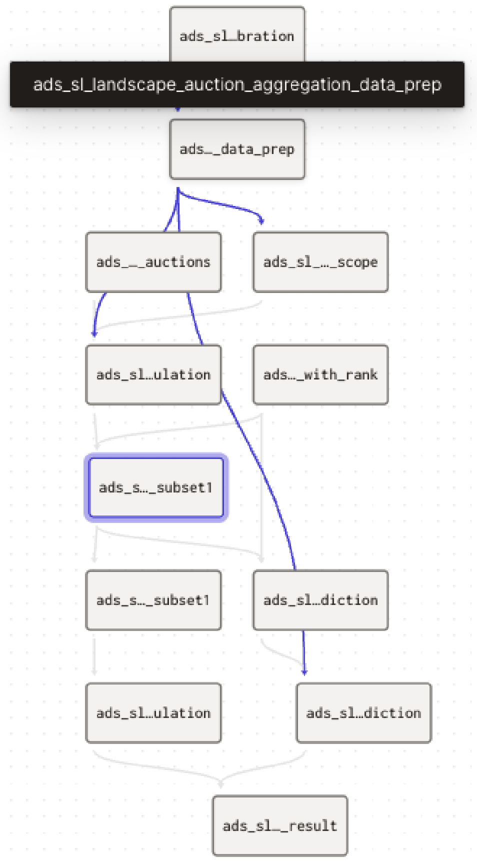
Auto backfill downstream jobs to create historical data for model training

- Improved dev velocity

LEARNINGS

Auto backfill downstream jobs to create historical data for model training

- Improved dev velocity
- High complexity and cost
    - explicit flag to trigger downstreams



LEARNINGS

Three core components

- Feature Registry
- Unified Execution Environment (Library Suite)
- Infrastructure Automation (Orchestration & Integrations)

LEARNINGS

Three core components

- Feature Registry
- Unified Execution Environment (Library Suite)
- Infrastructure Automation (Orchestration & Integrations)

- Single entry point, end-to-end experience
- Complexity & error detection
  - Categorize errors
  - Analytical metrics in addition to system observability

THANK YOU!

Q&A