Jukebox 2.0

Aman Khan
Product Manager
Spotify

Daniel Kristjansson
Staff Engineer
Spotify

[Feature Store Summit] Jukebox

Aman Khan, Product Manager
Daniel Kristjansson, Staff Engineer

Overview

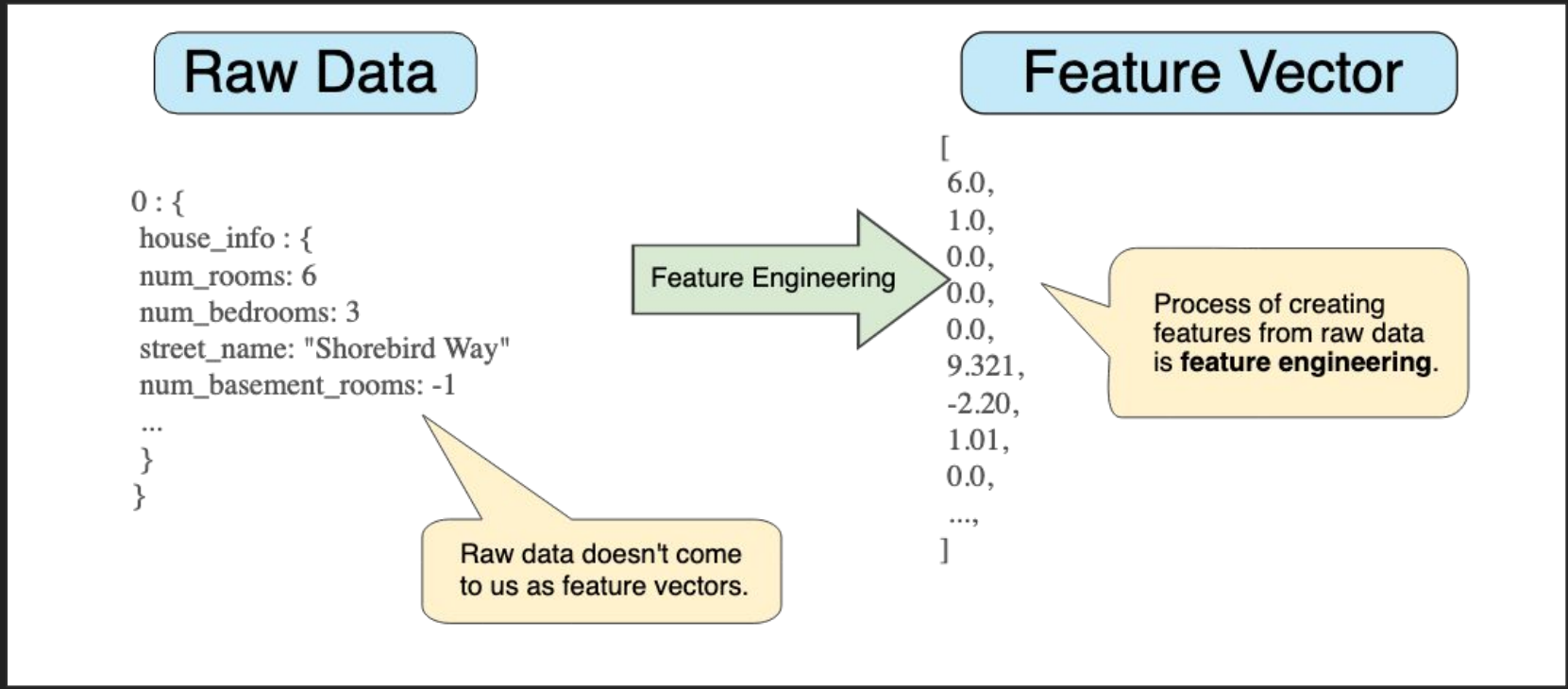
- ML Platform at Spotify
- What is Jukebox
- How it works
- Why it works like this
- Where we're going
  - Jukebox API
  - Feature Workflows
  - Feature Marketplace


Active Users 365 Million
Tracks 70 Million
Podcasts 2.9 Million
Playlists 4 Billion
Available in 178 Markets

Models Trained	30 Thousand
Prediction Requests	300 Thousand/Second
Teams on ML Platform	50

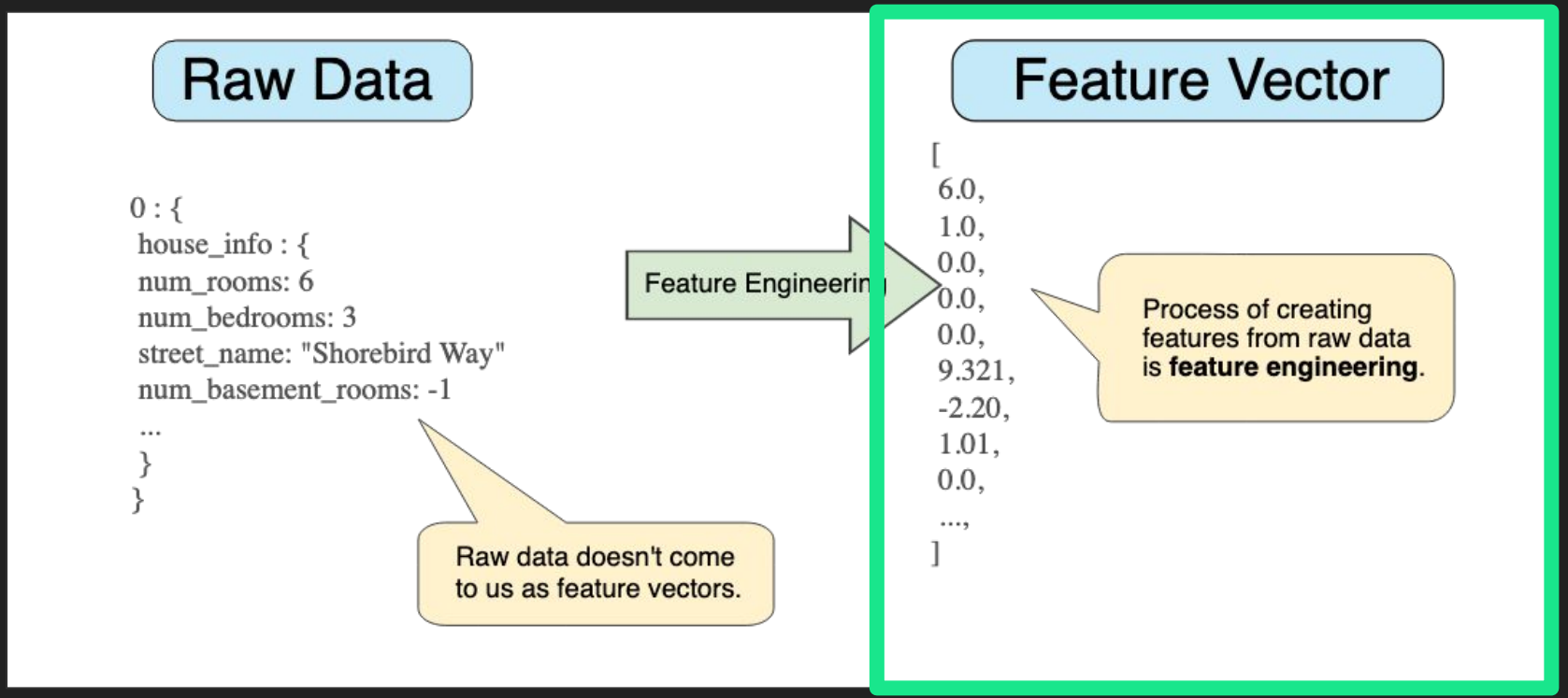
ML Platform 2020 Stats

Refresher


Raw Data
0 : {
house_info : {
num_rooms: 6
num_bedrooms: 3
street_name: "Shorebird Way"
num_basement_rooms: -1
...
}
}
Raw data doesn't come to us as feature vectors.
Feature Engineering
Feature Vector
[
6.0,
1.0,
0.0,
0.0,
0.0,
9.321,
-2.20,
1.01,
0.0,
...,
]
Process of creating features from raw data is **feature engineering**.

Refresher

Jukebox helps manage this for model use case



Features at Spotify

Technical

- Dynamic features
- Near "real time" features
- Complexity between trying to bridge offline and online

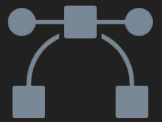
Organization

- Highly autonomous culture
- 50+ squads, across different use cases, recreating and serving the same features

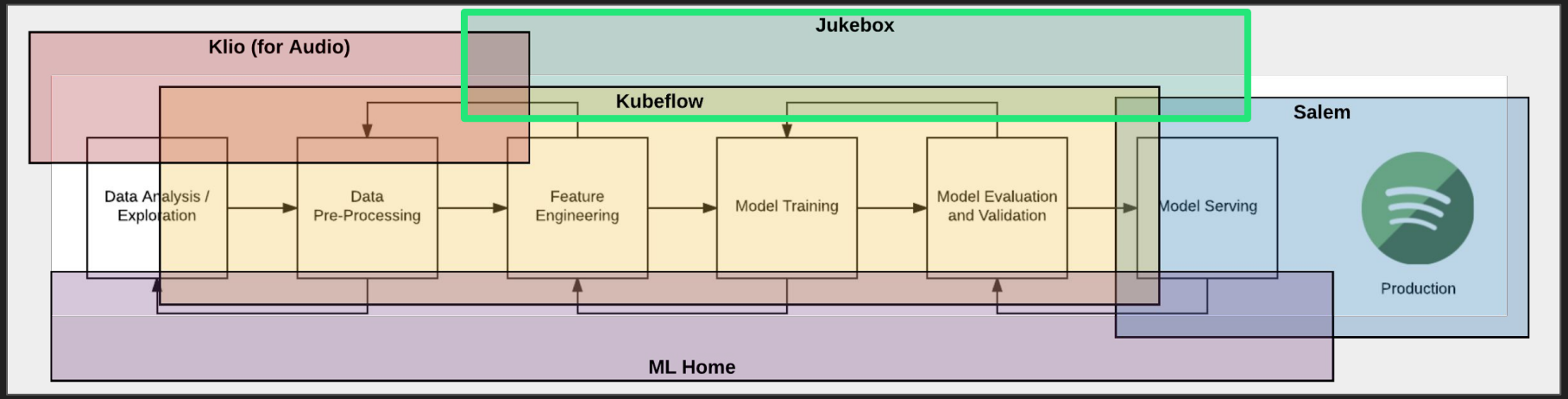
Features are a huge problem

A brief history of our platform

pre 2017

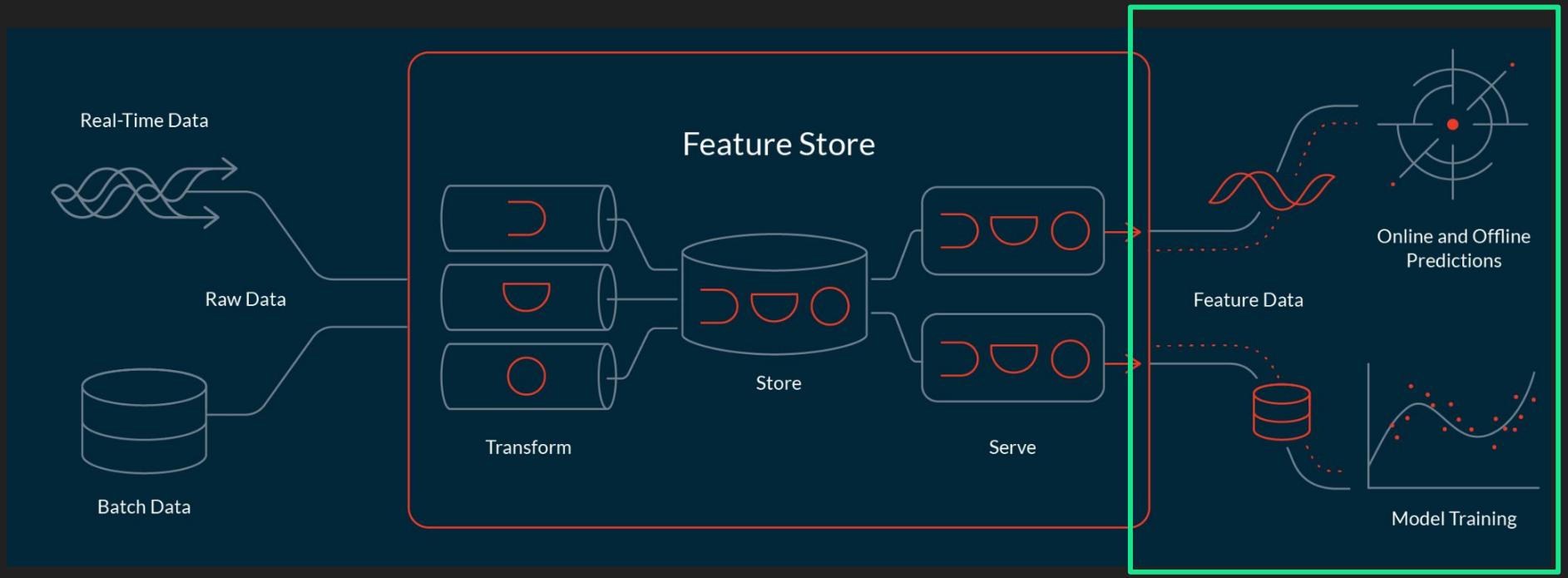


Quick background on the ML Platform



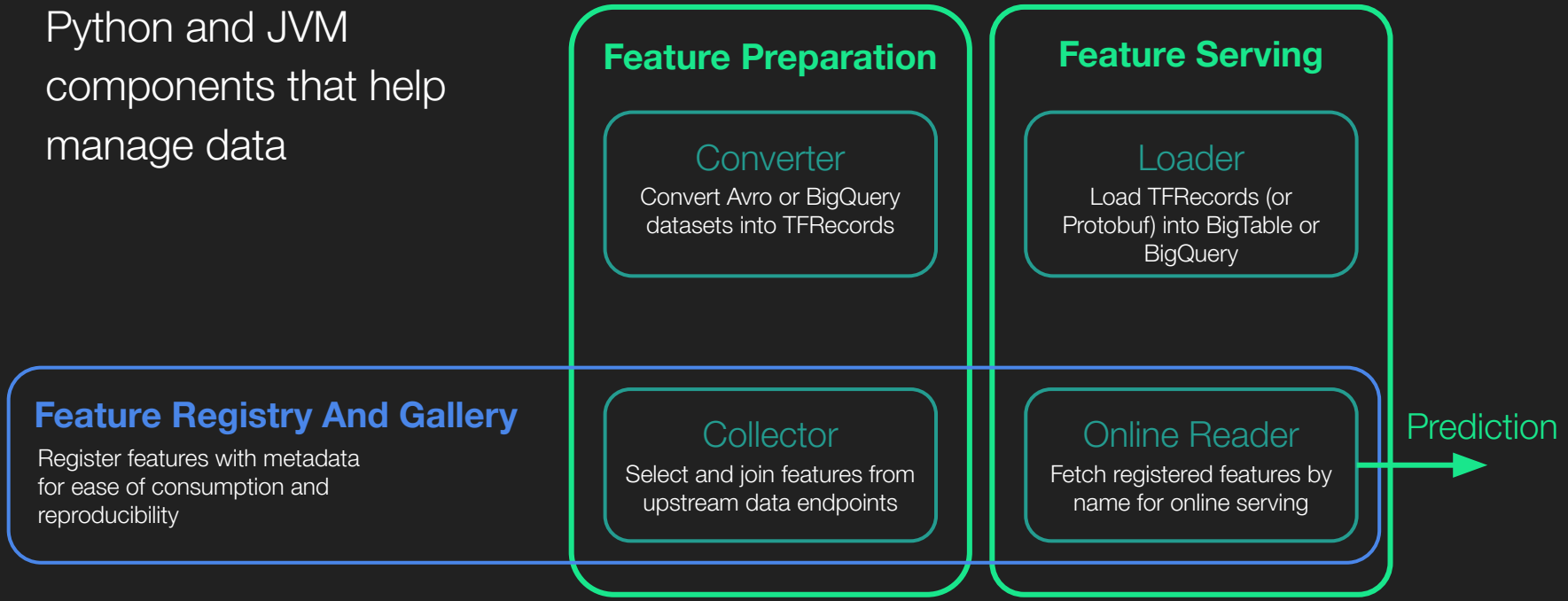
5 products that serve various stages of the ML Lifecycle

Refresher


Real-Time Data
Raw Data
Batch Data
Feature Store
Transform
Store
Serve
Feature Data
Online and Offline Predictions
Model Training

Where we were

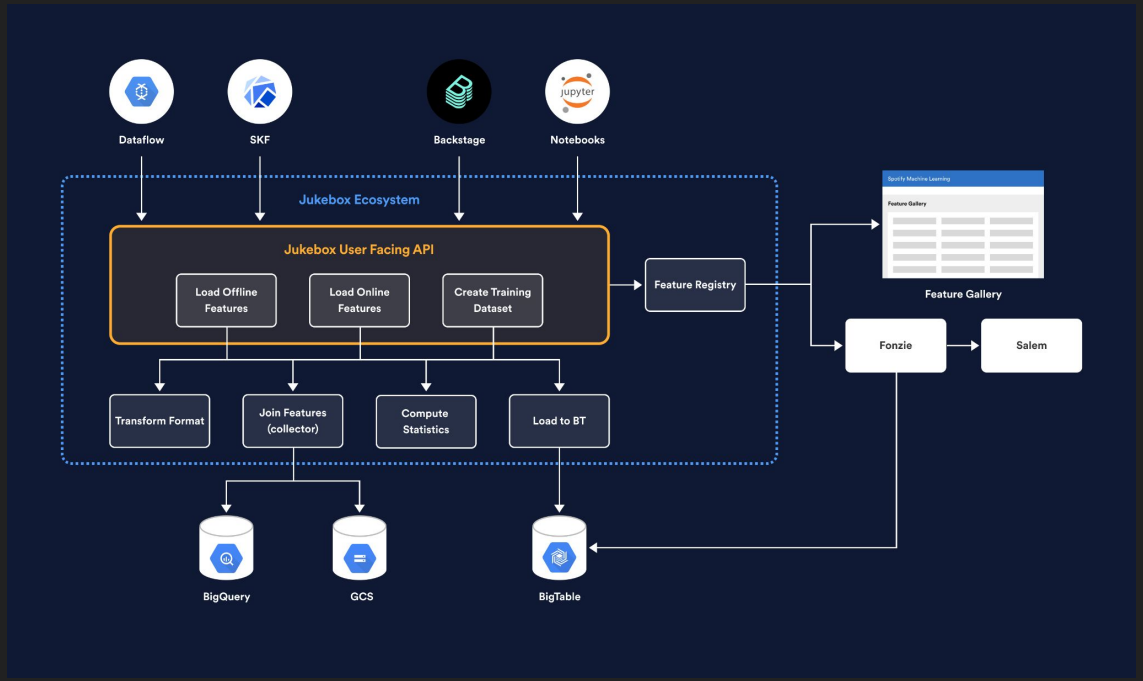
- Python and JVM components that help manage data



Decentralized management through libraries

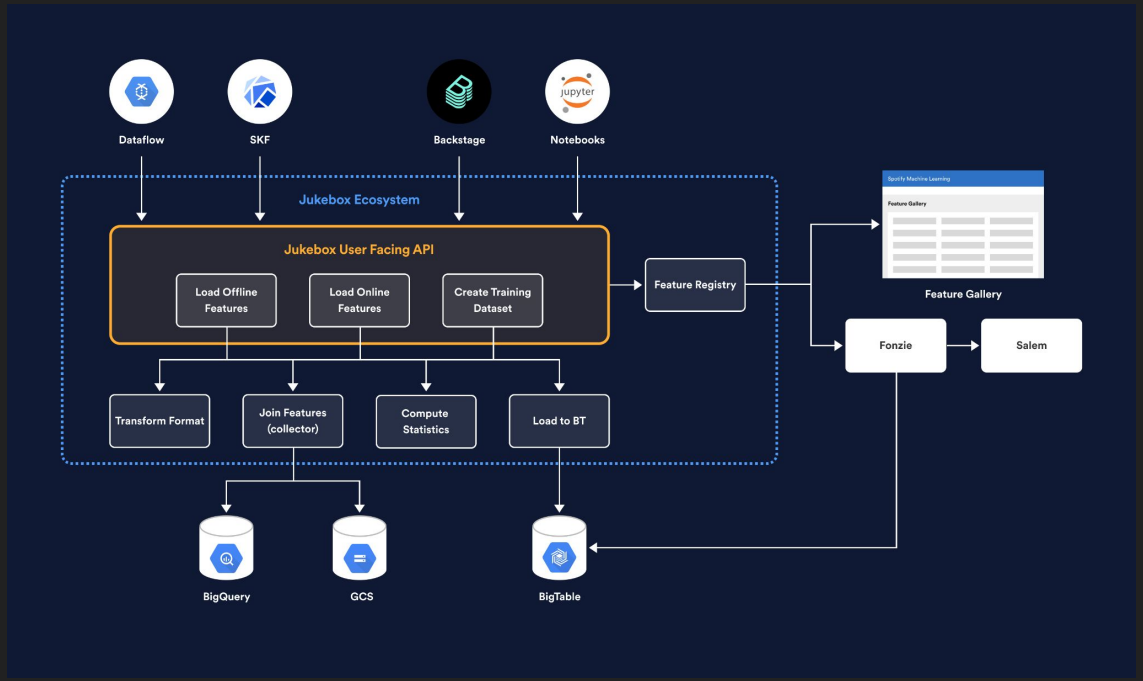
Jukebox 2.0 Overview

- Functionality exposed as a service API
  - Make features available for Jukebox ecosystem (load)
  - Prepare training dataset using point in time joins
  - Load feature set for online inference (Bigtable)



Jukebox 2.0 Overview

- Service will automatically:
  - Convert between formats
  - Register Bigtable + GCS locations in feature registry
  - Join datasets
  - Compute statistics
- Benefits
  - Centralized Management
  - User Experience
  - Development



User focused, centralized workflow

What's new?

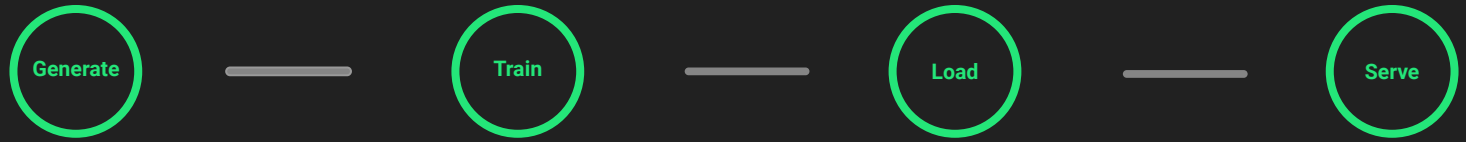
Jukebox 1.0

- Reference sources between training and serving workflows (error prone)
- Library dependencies and version maintenance for latest and greatest
- No trust in reusing other team's features
- No connection to rest of ML Platform

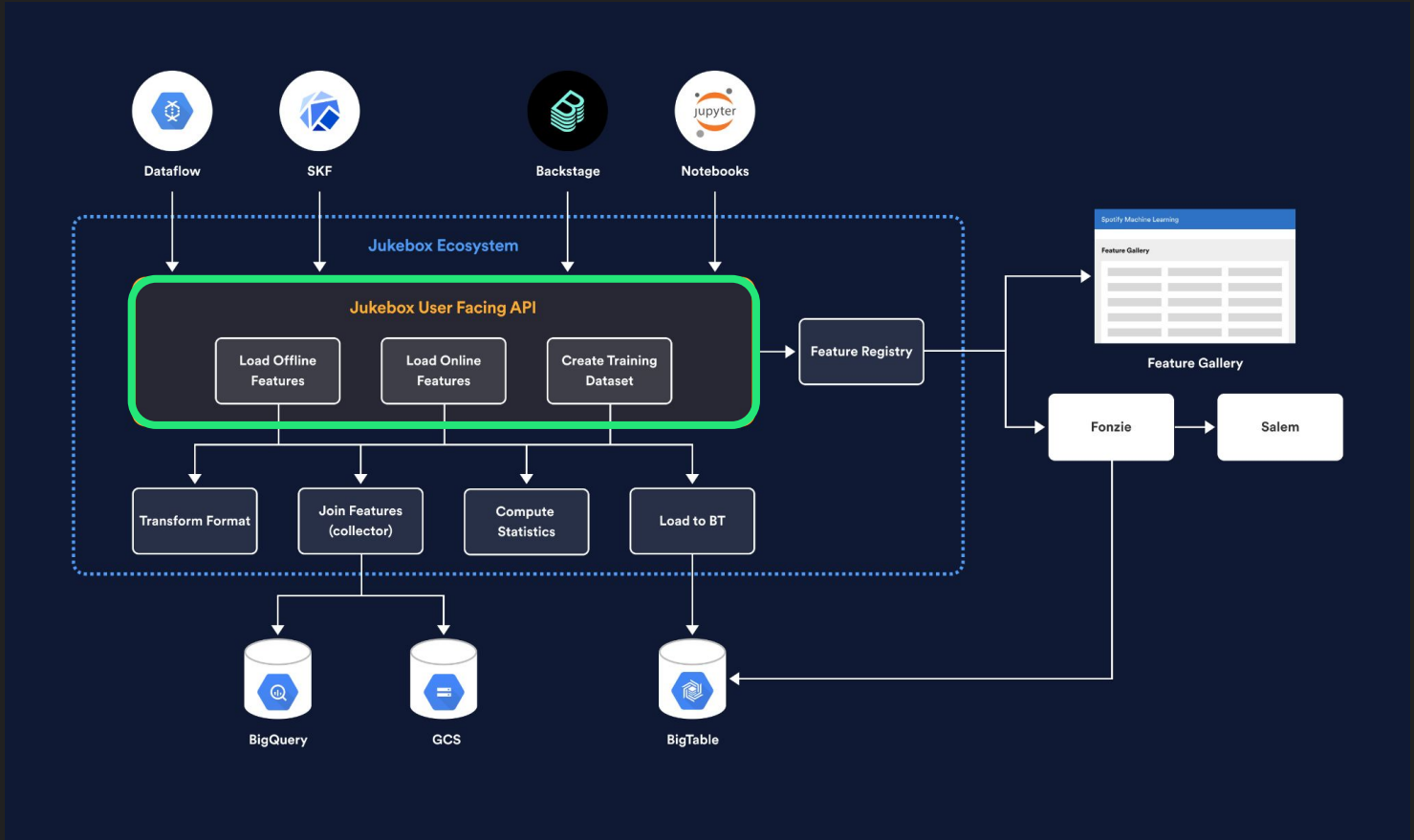
Jukebox 2.0

- **Register your features and reference them by name (simplifies code and reduces chance of incorrect definition)**
- **No more managing pesky library versions or dependencies to get the latest Jukebox updates**
- **<soon> Profiling information in the Feature Marketplace UI, and automatic feature fetching in Salem**
- ***Bonus* Point in time joins!**

Feature Workflows



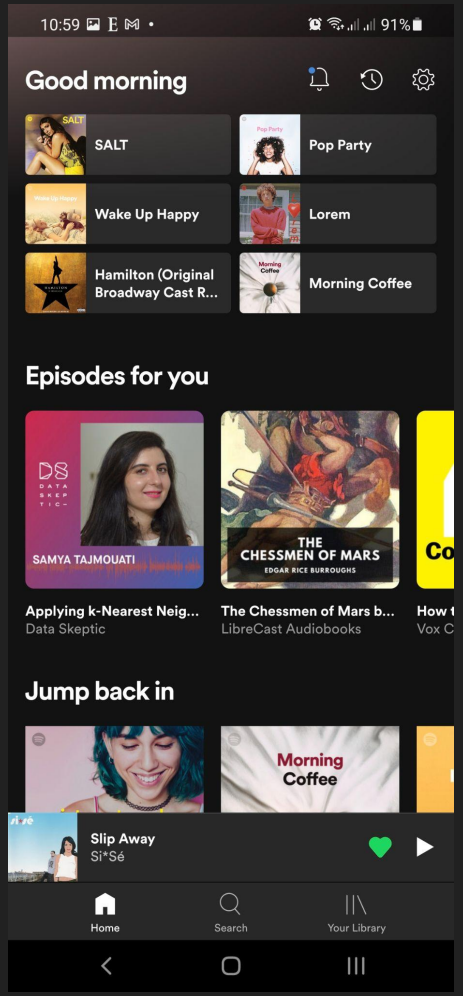
Feature Workflows



Home Use Case

Goal: Listener finds a session quickly that will sustain their satisfaction with Spotify

- Many models, multiple teams
- Daily, near-real-time and contextual features
- Five feature stores*



Feature Sources

Logs - Impression, Interaction, EndSong, etc

- Scio pipelines
- BigQuery

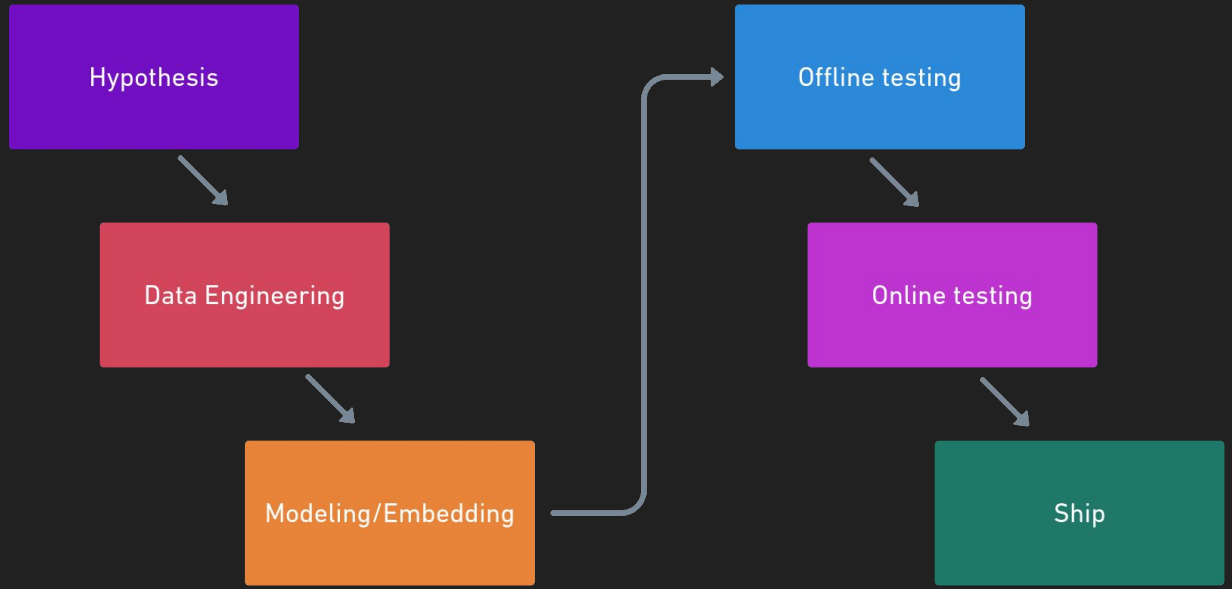
Embeddings - Music, Users, Episodes, etc

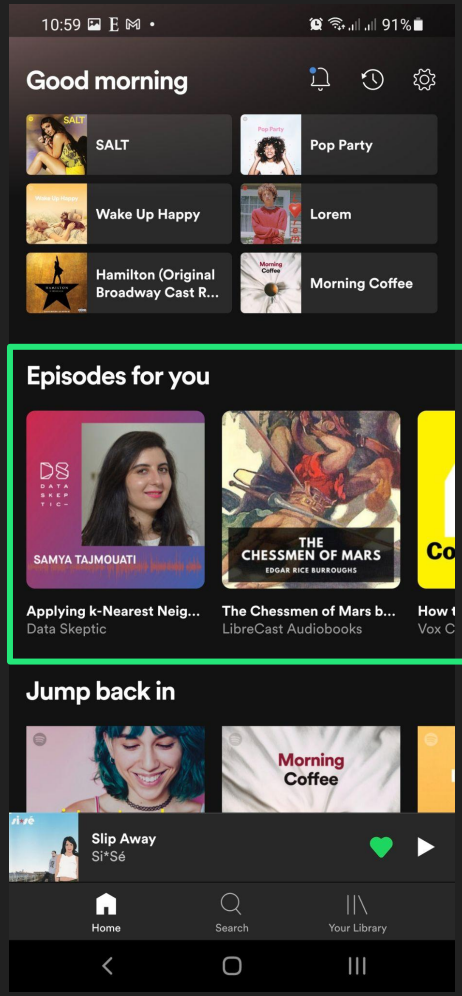
- Bag-of-words
- BERT

Context

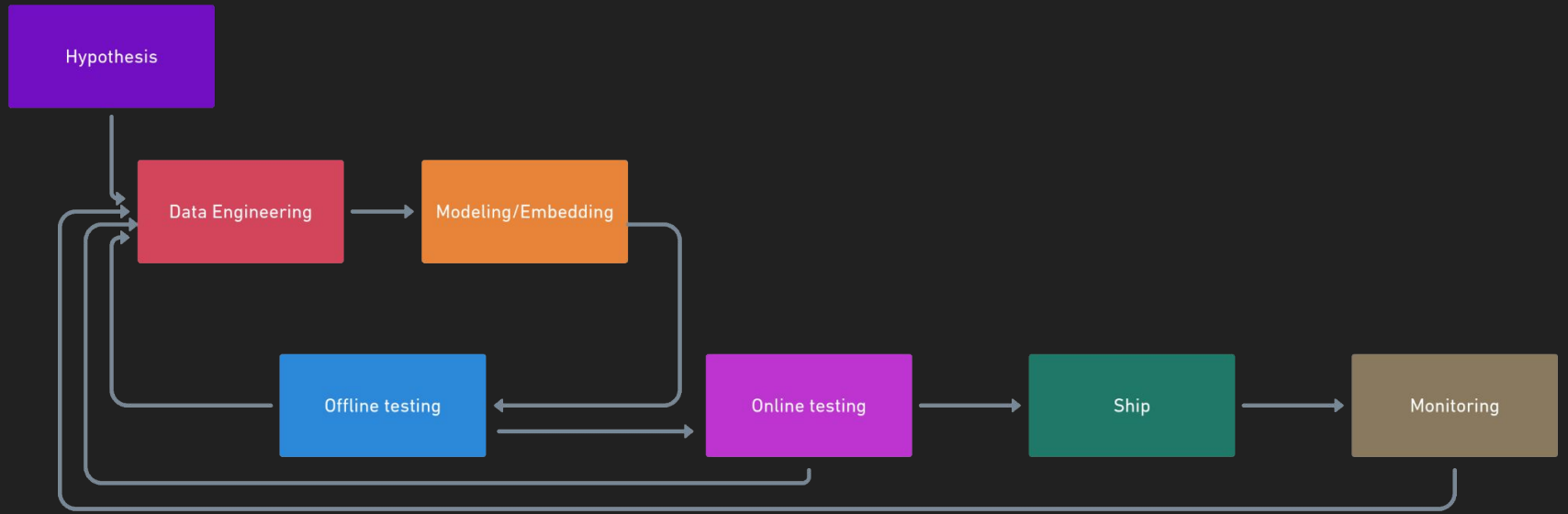
- Local time, device, etc.

Feature Engineering





Feature Engineering -- real



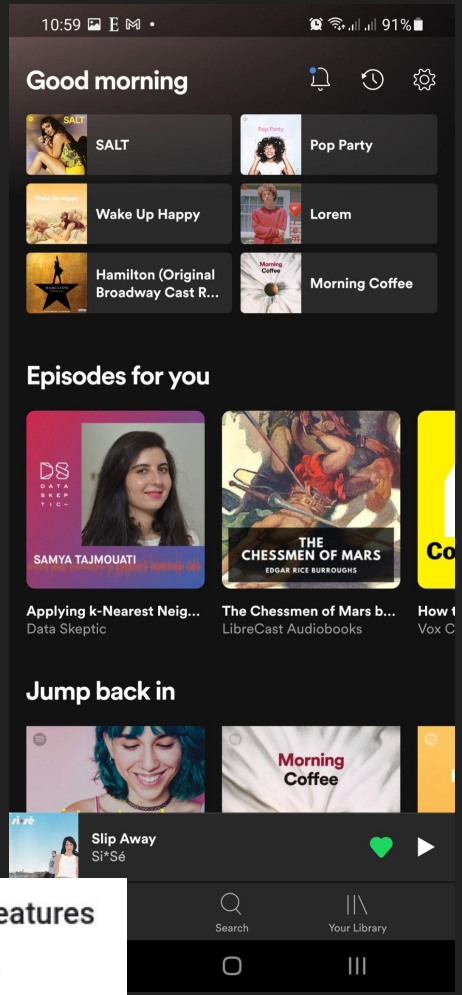
Generate
Train
Load
Serve

Feature Engineering -- trouble

Home x Salem

#home-ml-ops

July 2020



Generate

Train

Load

Serve

Rule #29: The best way to make sure that you train like you serve is to save the set of features used at serving time, and then pipe those features to a log to use them at training time.

Feature Engineering

Salem - new capabilities contribute to longer feature iteration cycles

- Feature logging for everyone :D
- Longer Feature Engineering workflow for everyone :(
- Longer running A/B tests :(

Feature Engineering

Killer features for fast iteration

- Point-in-time joins
- Automated backfills
- Streaming ingestion

Feature Engineering

Zipline

Google
Vertex Feature Store

Hopsworks

Jukebox 2.0

FEAST

tecton

Generate

Train

Load

Serve

Adding a new feature

First steps

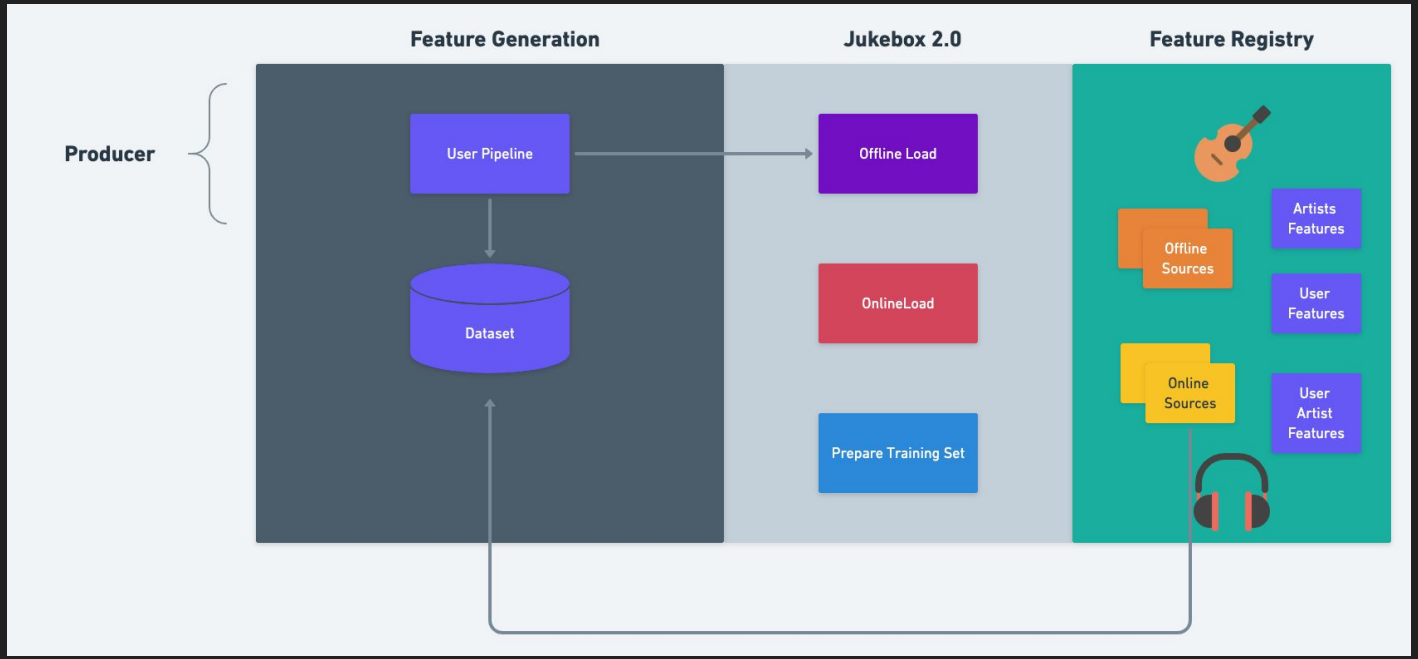
- **Register Feature**
- **Offline Load**
- **Prepare Training Set**

Example

```
    # Artist preference example - load all offline endpoints
    # Use parts of the first as a template for the rest of the jukebox.load-*.apm workflows
  - &jukebox-apm-defaults
    id: jukebox.load-offline.apm.artist_entities
    schedule: daily
    service_account: jukebox-examples@sp-ml-infra.iam.gserviceaccount.com
    docker_image: gcr.io/sp-ml-infra/jukebox-luigi:latest
    docker_args: [
        'load-offline',
        '--features', '{"/entity/artist/component/jukebox-apm-sample/name/genres":"genres","/entity/artist/component/jukebox-apm-sample/name/popularity_normal
        '--keys', '[{"primitive_entity": "PRIMITIVE_ENTITY_ARTIST", "identifier": "IDENTIFIER_BASE62", "column_name": "artist_gid"}]',
        '--data-endpoint', 'jukebox.examples.apm.artist_entities',
        '--storage-format', 'STORAGE_FORMAT_TF_EXAMPLE',
        '--project', 'sp-ml-infra',
        '--uri-prefix', 'gs://sp-ml-infra-temp-eu/apm/',
        '--date', '{}'
    ]
```

Create a Training Set

Feature Producer: Register Feature & Offline Load

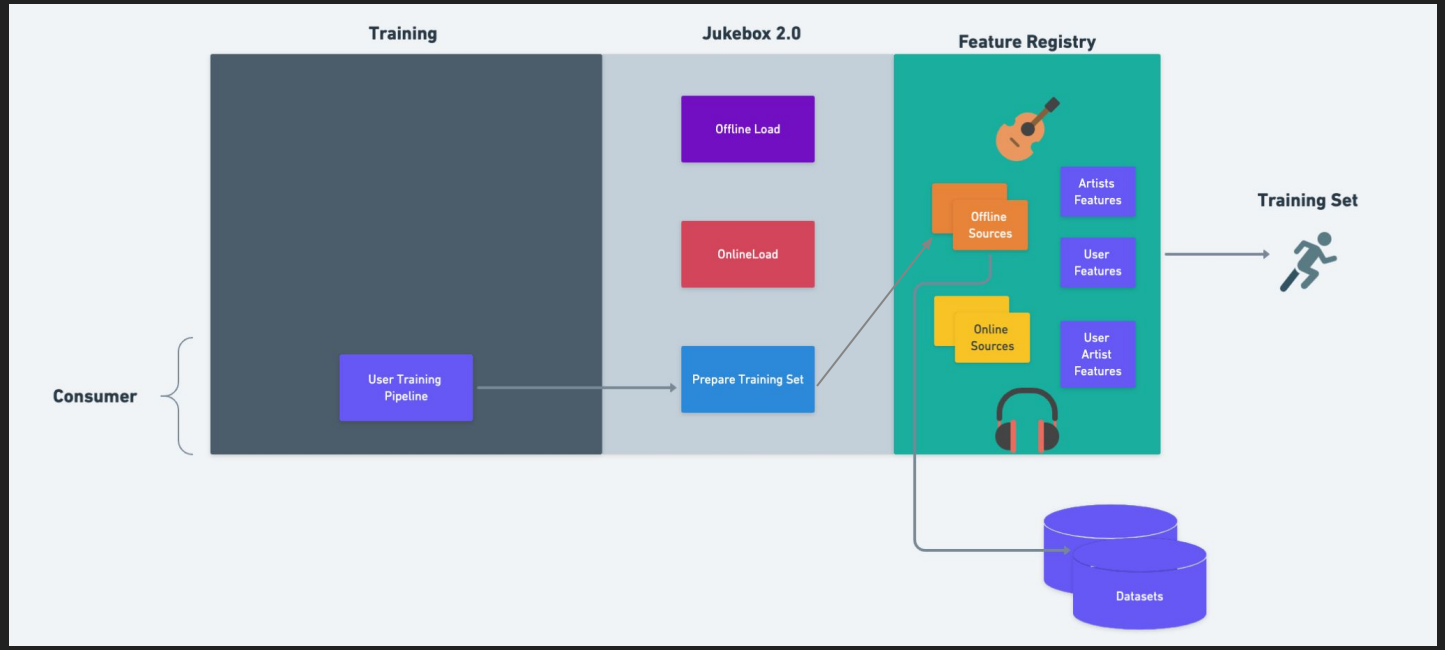


Training

1. Create label dataset in training pipeline

2. ***Prepare Training Dataset*** is called providing label dataset (logs) and features

3. Workflow job is run that will collect feature values <u>at the right point in time</u>



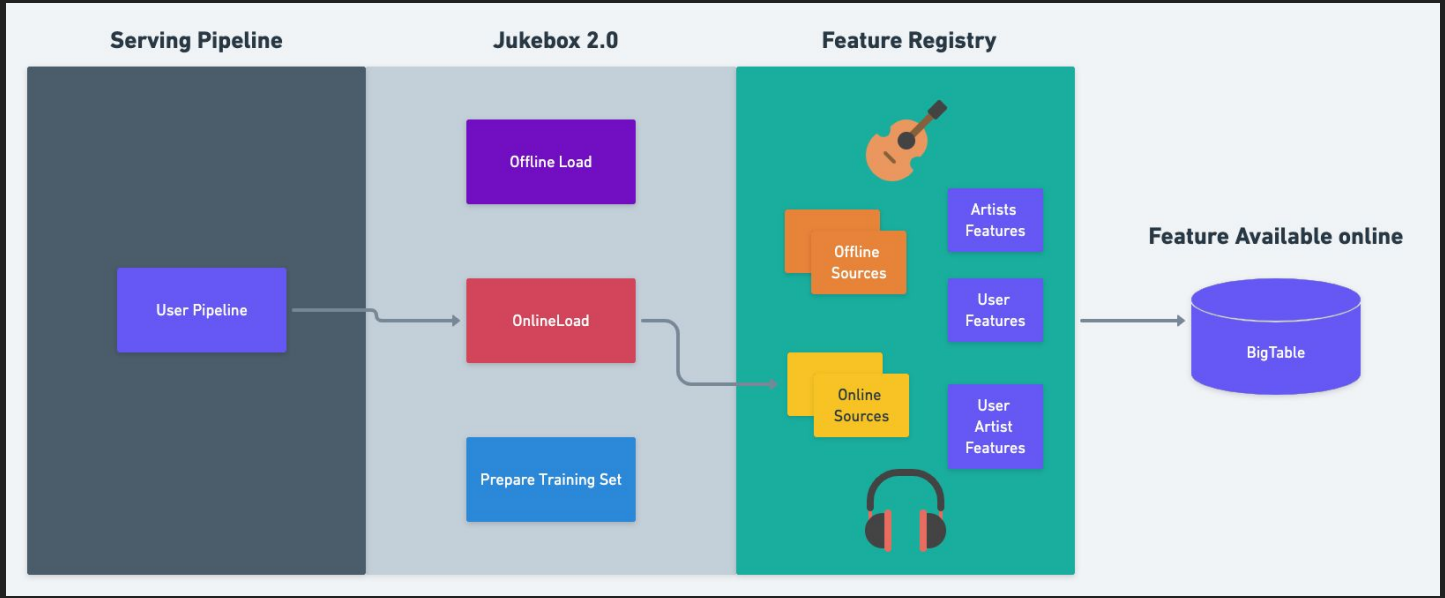
Online Data Loader

1. Features are specified to be loaded in an online store for serving, or for being refreshed

2. ***Online Load*** is called specifying features to loaded

3. Workflow is run, feature values are collected and loaded into BT

4. Online Source is created to reference newly loaded data in BT is created

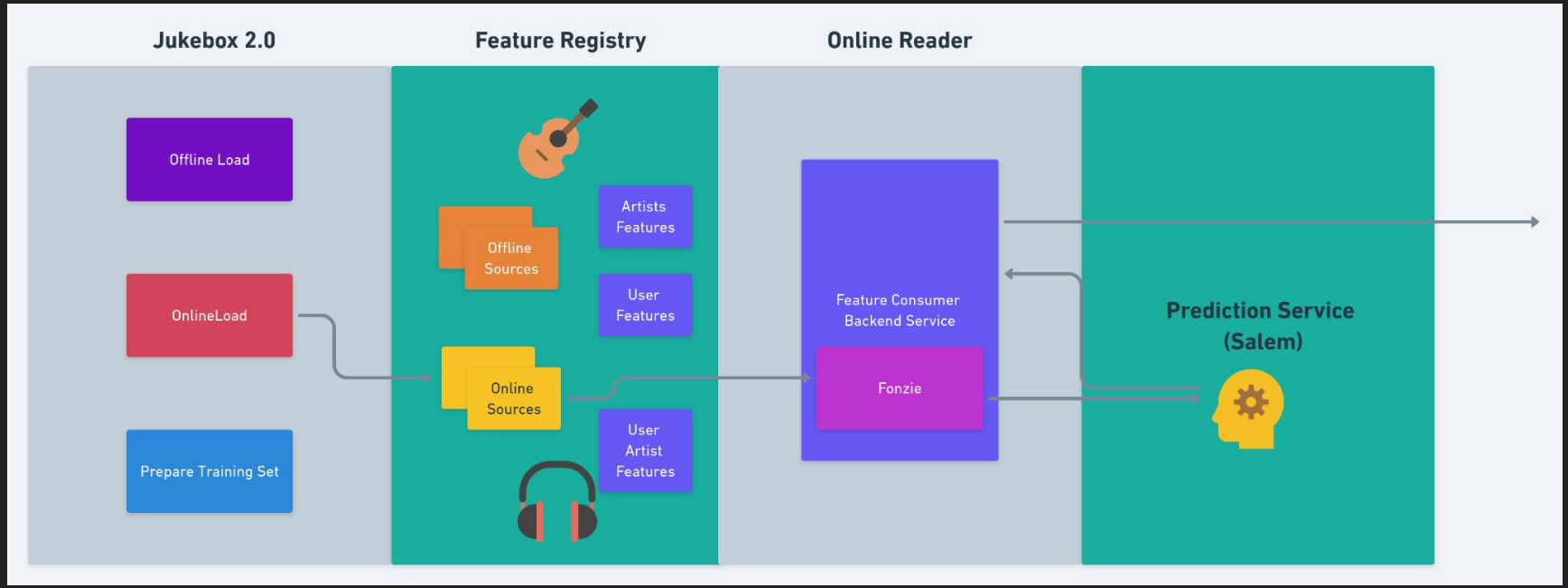


Online Load Example

```
  # Artist preference example - load into Bigtables by entity
- <<: *jukebox-apm-defaults
  id: jukebox.load-online.apm.artist
  docker_args: [
      'load-online',
      '--features', '{"/entity/artist/component/jukebox-apm-sample/name/genres":"genres","/entity/artist/component/jukebox-apm-sample/name/popularity_normal
      '--filter-keys', '[{"primitive_entity": "PRIMITIVE_ENTITY_ARTIST", "identifier": "IDENTIFIER_BASE62", "column_name": "artist_gid"}]',
      '--filter-data-endpoint', 'jukebox.examples.apm.UserArtistKeys',
      '--filter-storage-format', 'STORAGE_FORMAT_TF_EXAMPLE',
      '--uri-prefix', 'gs://jukebox-example',
      '--project', 'sp-ml-infra',
      '--bigtable-instance', 'jukebox-apm',
      '--bigtable-table', 'artist',
      '--date', '{}'
  ]
```

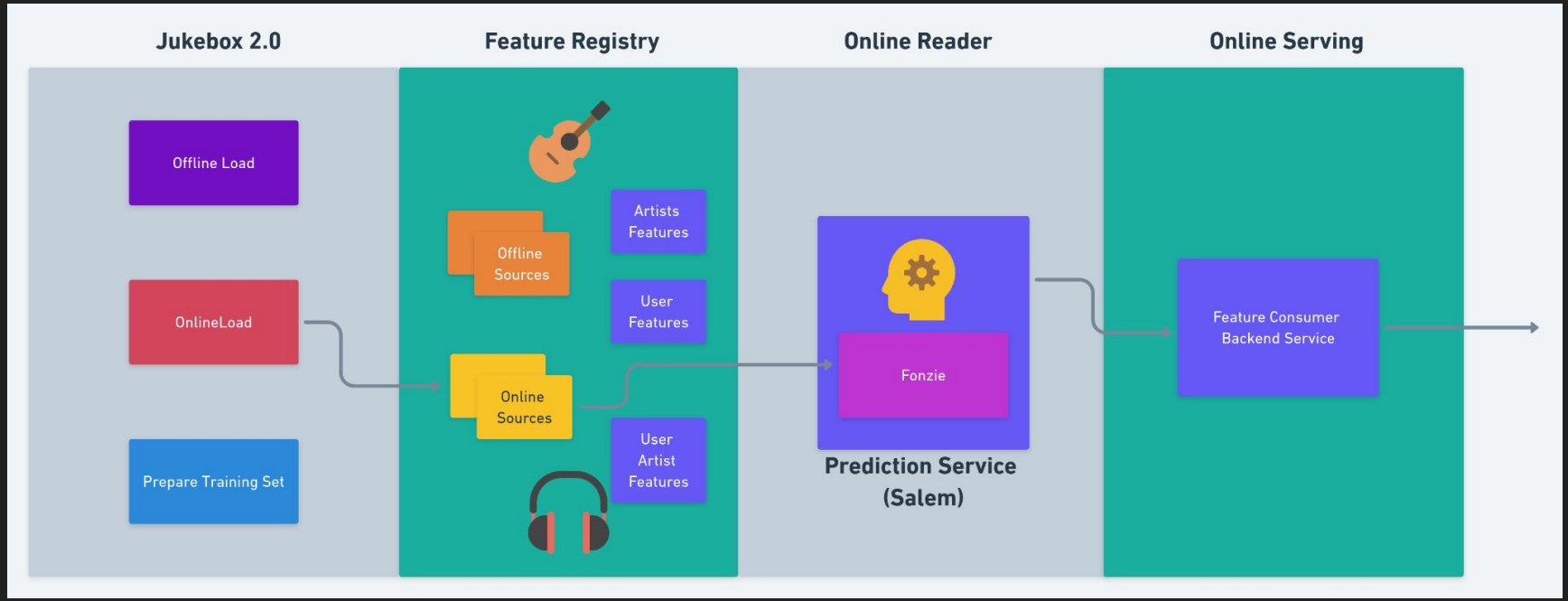
Online Serving

Consuming a feature



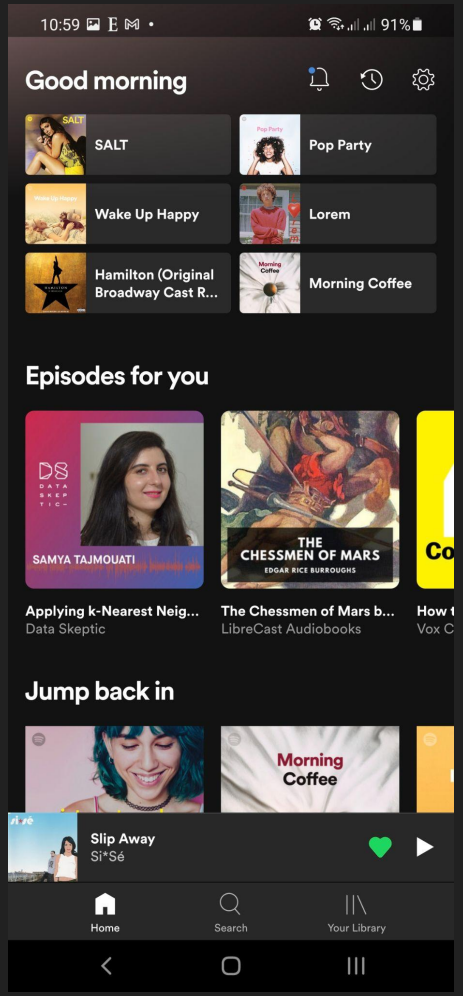
Online Serving

Idealized feature consumer view



Home Use Case

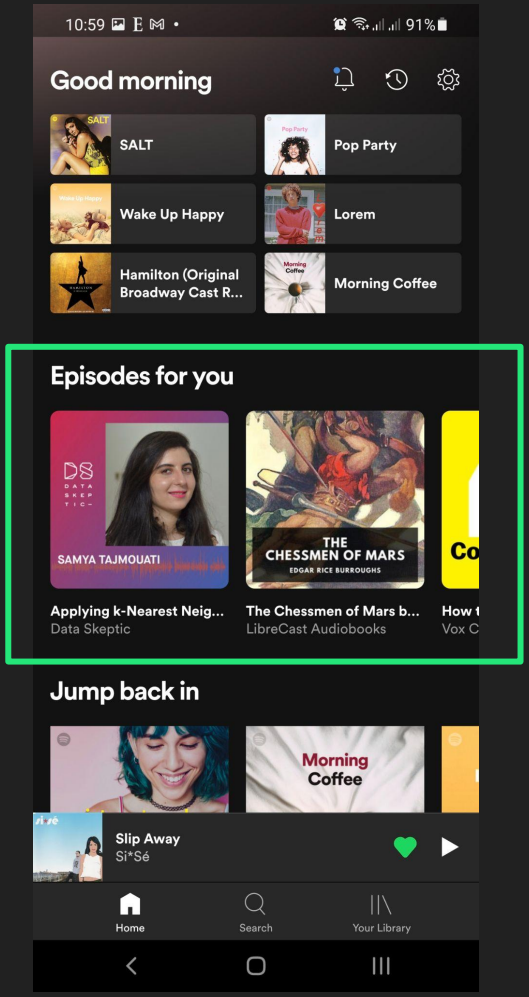
- Many models, multiple teams
- Daily, near-real-time and contextual features
- Five feature stores in all



Home Use Case

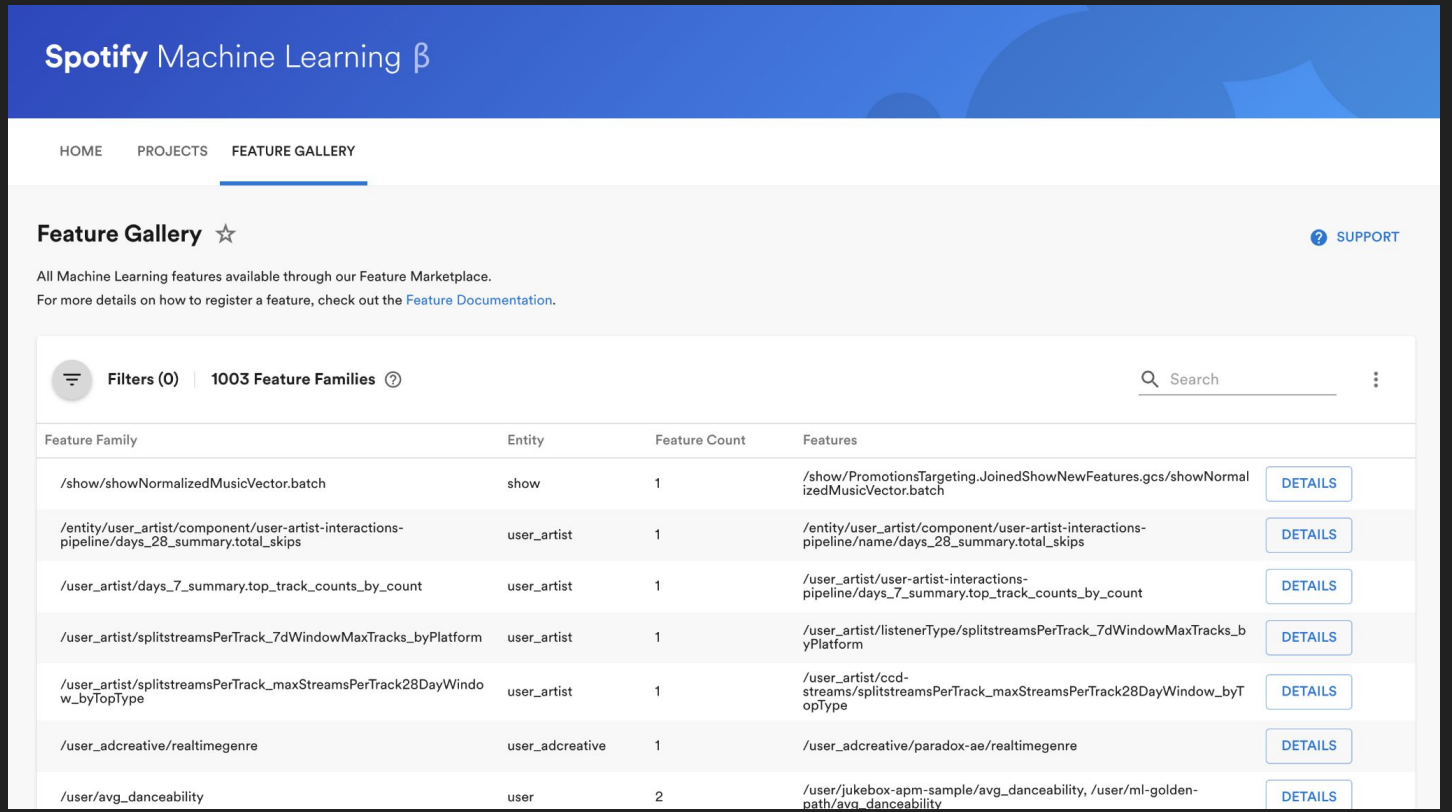
Many models, multiple teams

- Adoption can be piecemeal
- Feature registry useful for internal feature sharing
- API makes adding each feature easier

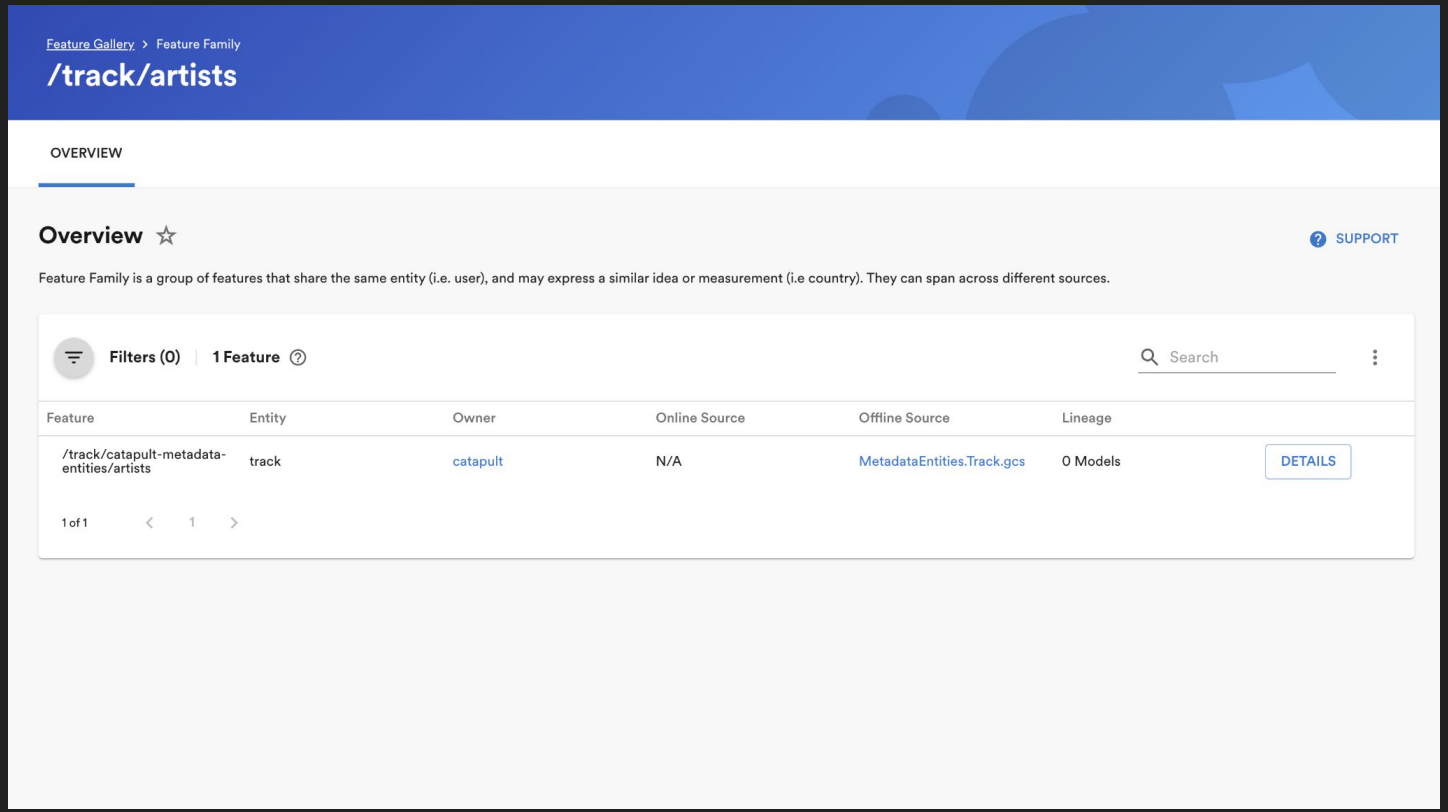


Feature Gallery

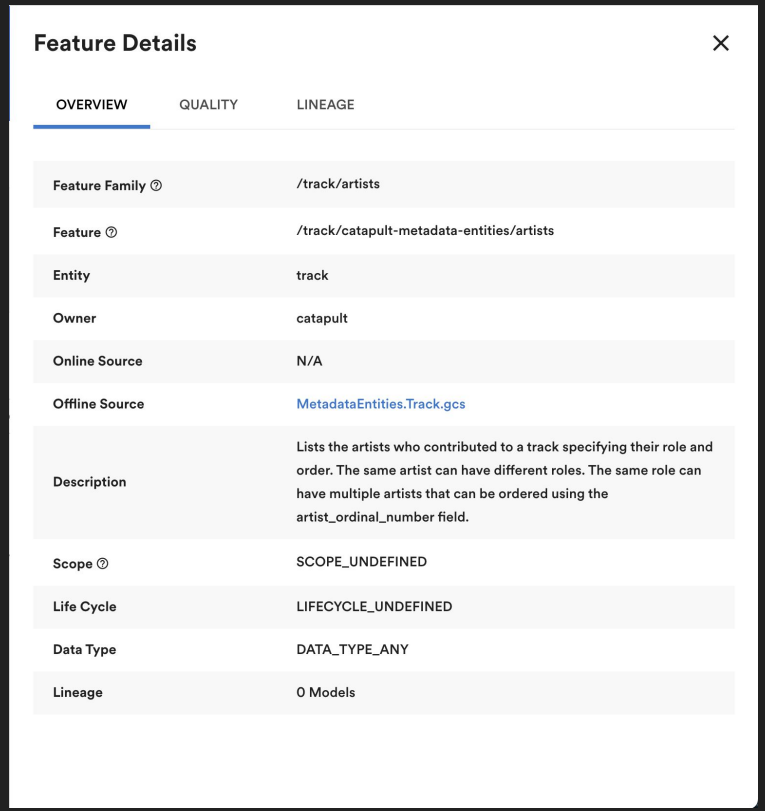
Gallery Homepage

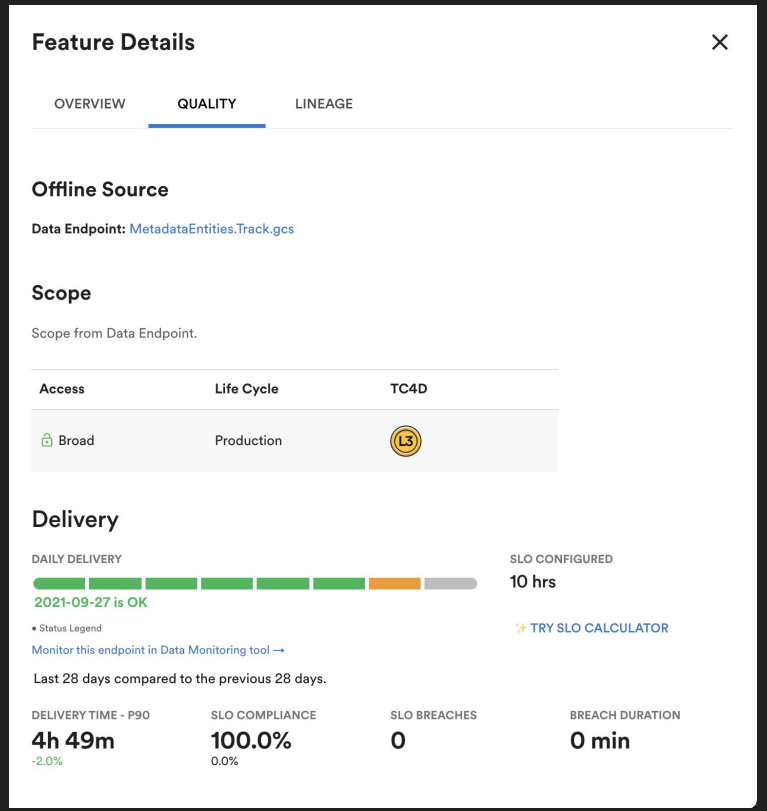


Feature Overview



Feature Details





In summary

Usability

Abstracts away complexity of feature workflow, and makes it easy to implement features

Trust

Jukebox enables a rich metadata layer containing information about ML model association, feature quality, and more

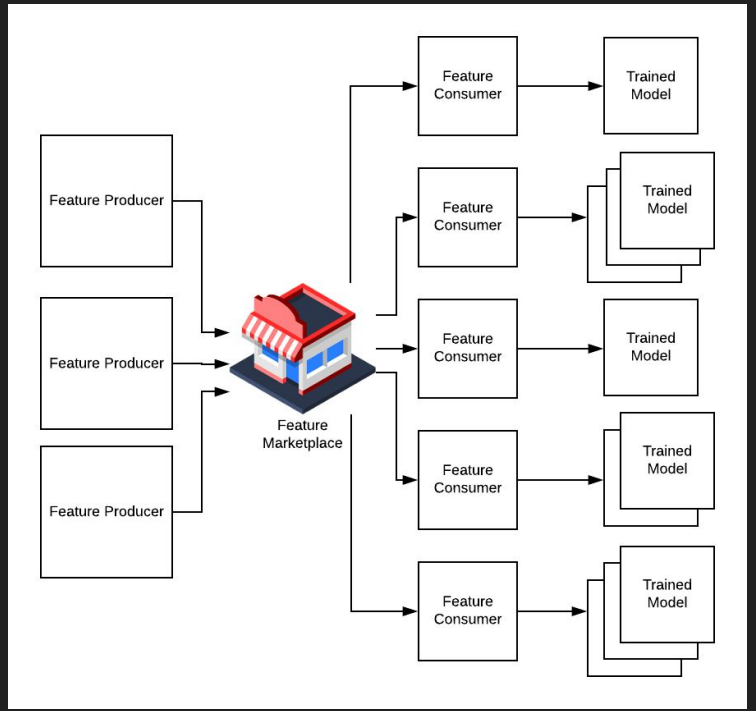
Built for reuse

Build different datasets, serve different models – you won't need to build another data pipeline that joins features together or another backend service to query it when you need to fetch actual feature values for serving your ML model

Where we're going - Feature Marketplace

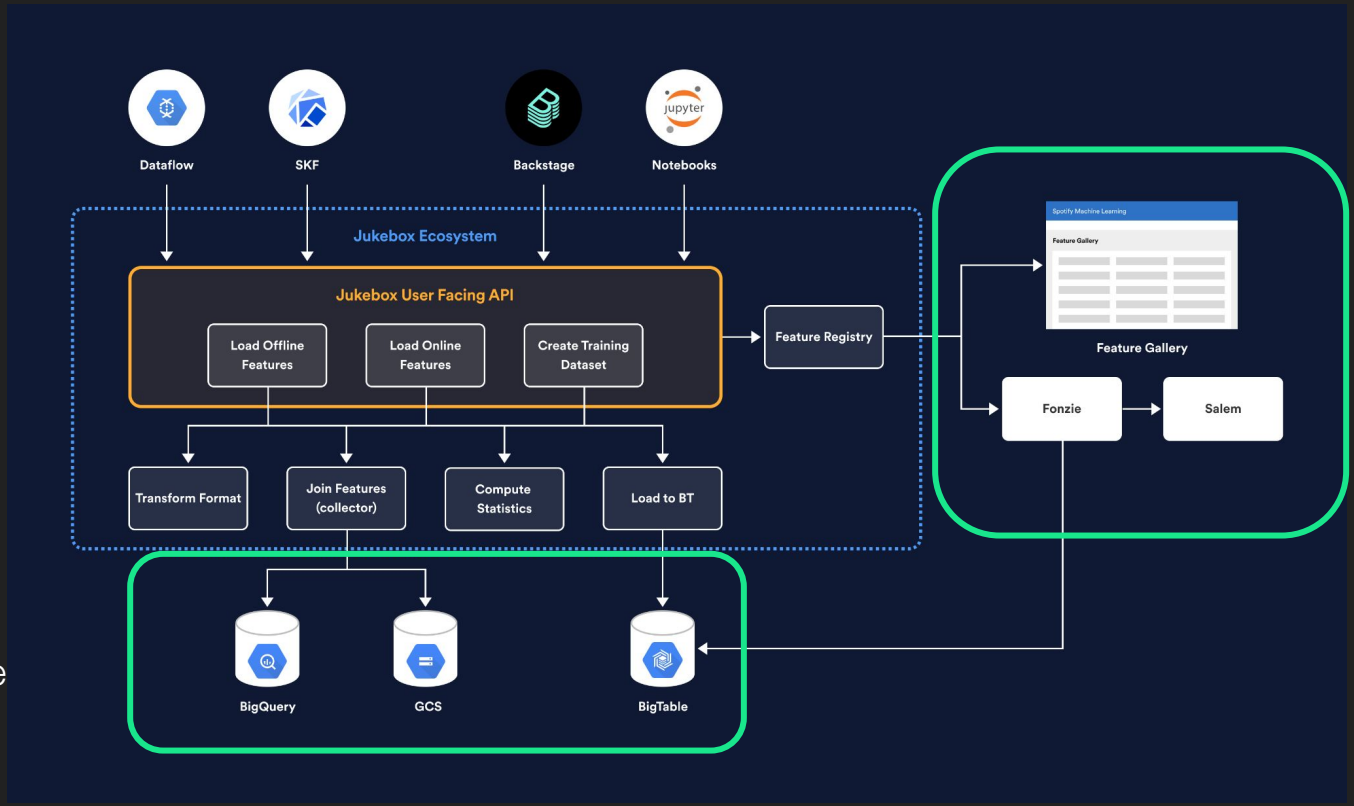
Saving Spotifiers time, money and toil through feature reuse and enhanced feature management.

- Builds on top of our API strategy and infrastructure
- Platformize the interface between producers and consumers
- Enhance trust with information: statistics, lineage, ownership



Future Improvements

Storage tradeoff for cost and performance



Automation

Model interpretability

Lessons Learned

- Listen to your customers (and if you can, rope them in to work with you)
- Feature stores are an evolving space
- Think about scale

Thank you!

Do you have any questions?

Aman Khan
amank@spotify.com

https://www.linkedin.com/in/amanberkeley

@_amankhan

Daniel Kristjansson
danielk@spotify.com

https://www.linkedin.com/in/danielkristjansson