

# 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**  
**Prediction Requests**  
**Teams on ML Platform**

**30** Thousand

**300** Thousand/Second

**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

## 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,  
  0.0,  
  9.321,  
  -2.20,  
  1.01,  
  0.0,  
  ...,  
]
```

Process of creating features from raw data is **feature engineering**.

# Features at Spotify

## Technical

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

**Features are a huge problem**

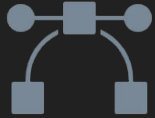
## Organization

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

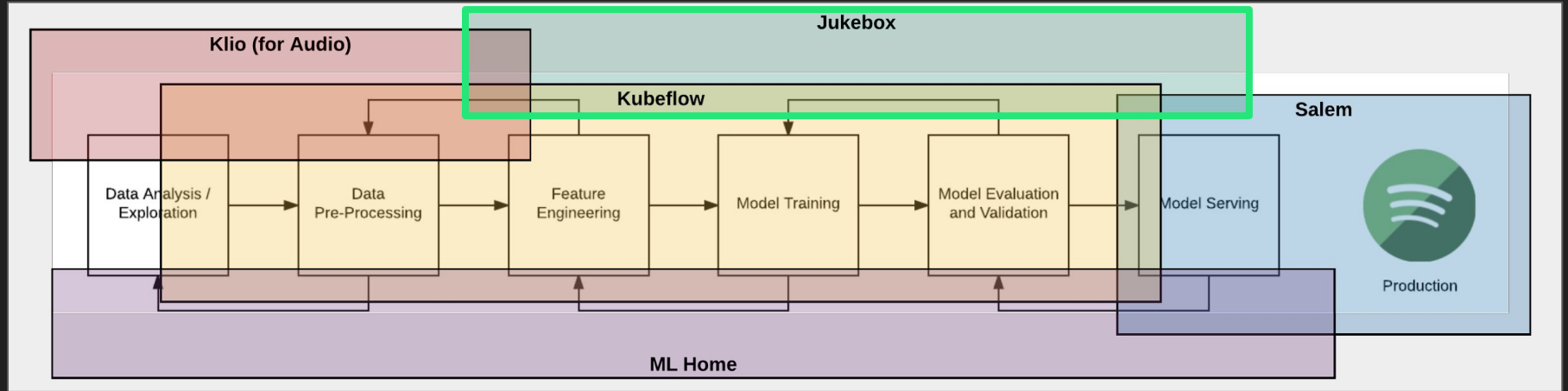


# A brief history of our platform

pre 2017

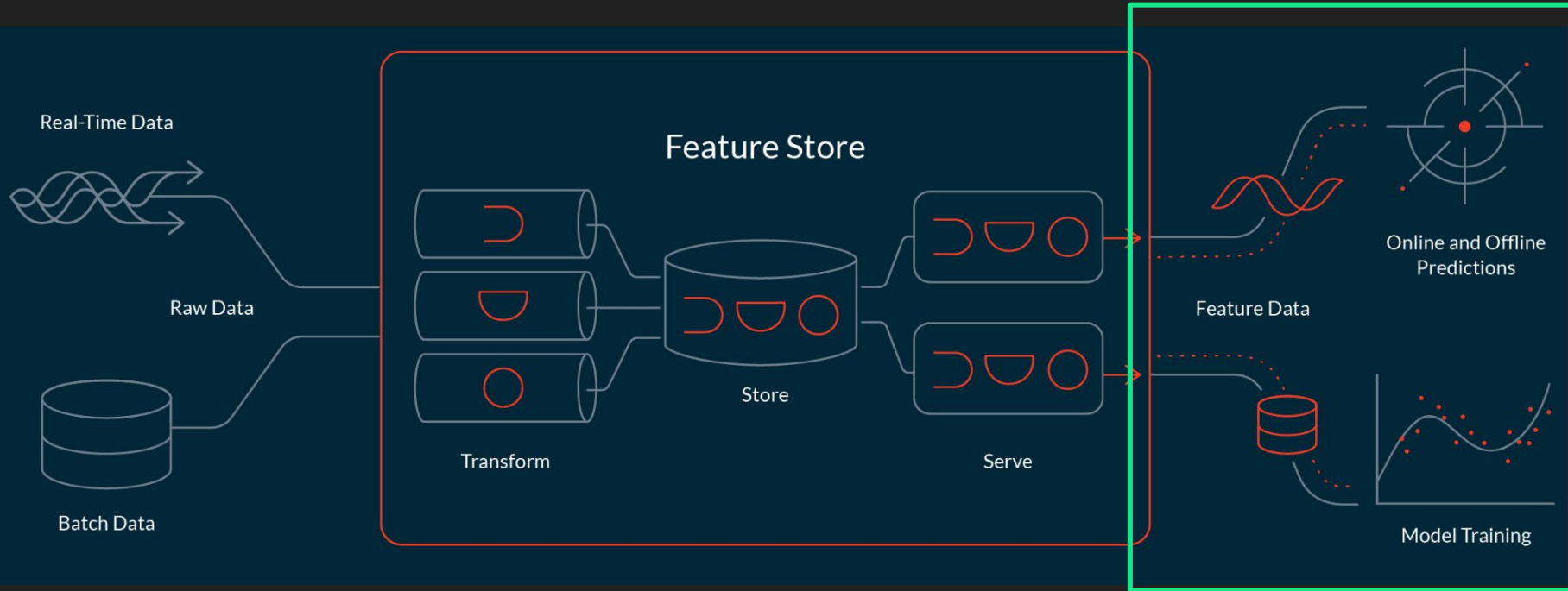


# Quick background on the ML Platform



5 products that serve various stages of the ML Lifecycle

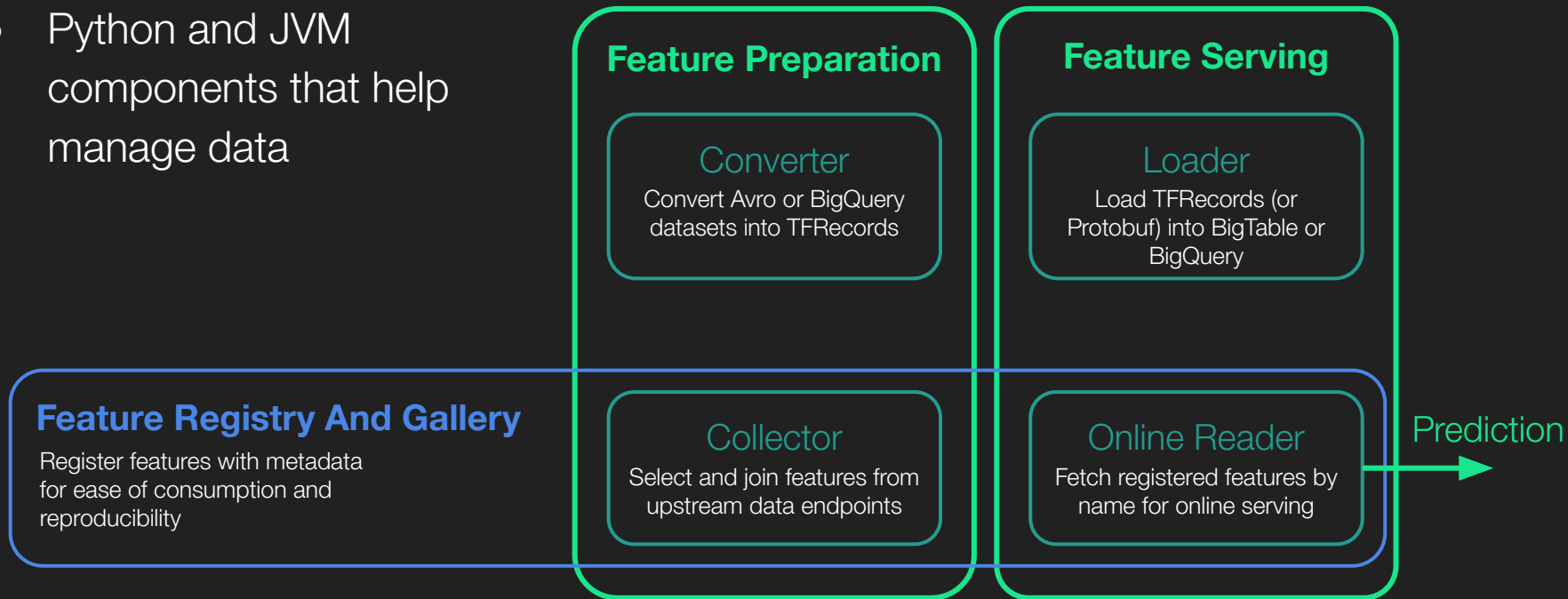
# Refresher



# Where we were

- Python and JVM components that help manage data

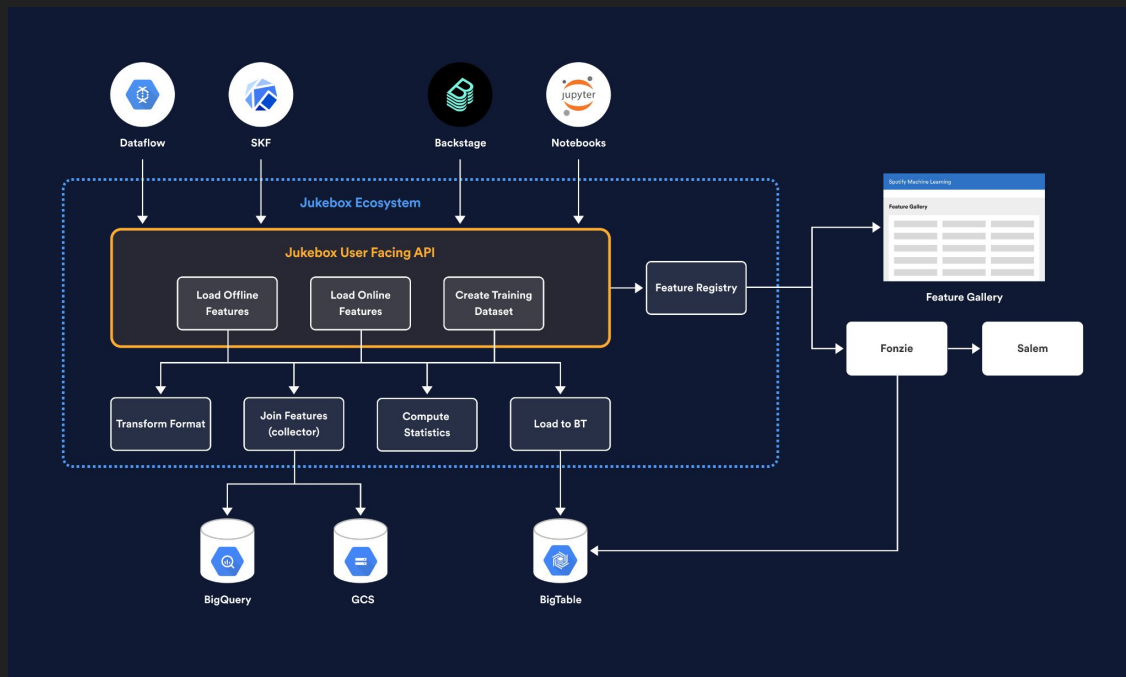
## JUKEBOX



**Decentralized management through libraries**

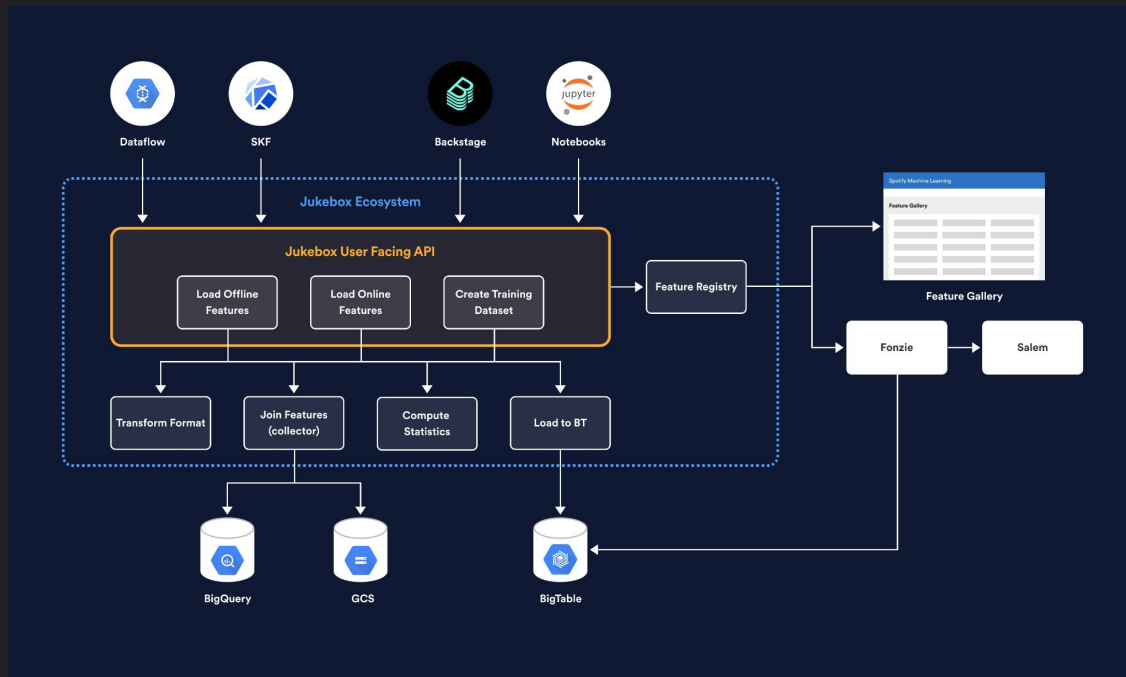
# Jukebox 2.0 Overview

- 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?

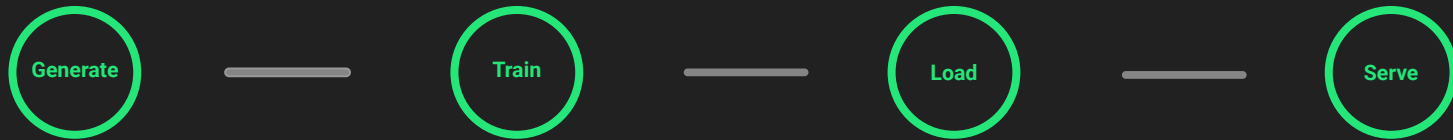
## Jukebox 1.0

- 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

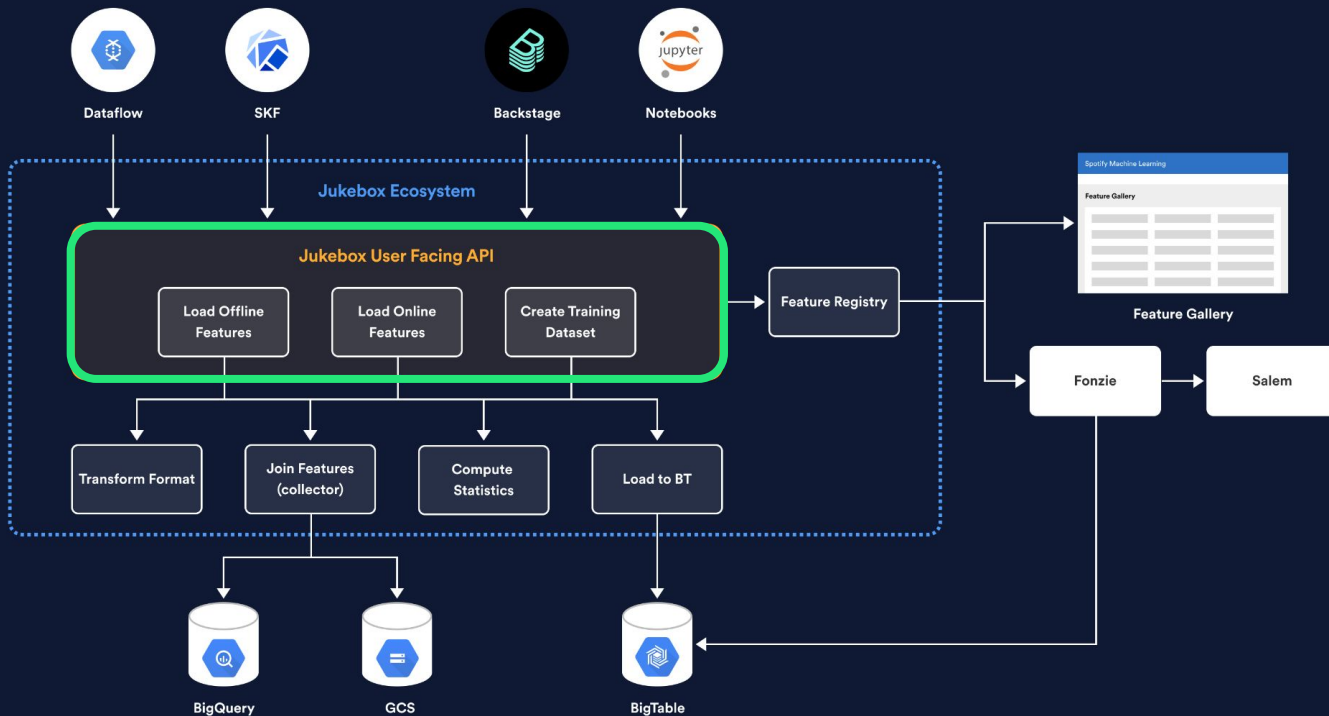
- **Register your features and reference them by name (simplifies code and reduces chance of incorrect definition)**
- **No more managing pesky library versions or dependencies to get the latest Jukebox updates**
- **<soon> Profiling information in the Feature Marketplace UI, and automatic feature fetching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

# Feature Workflows





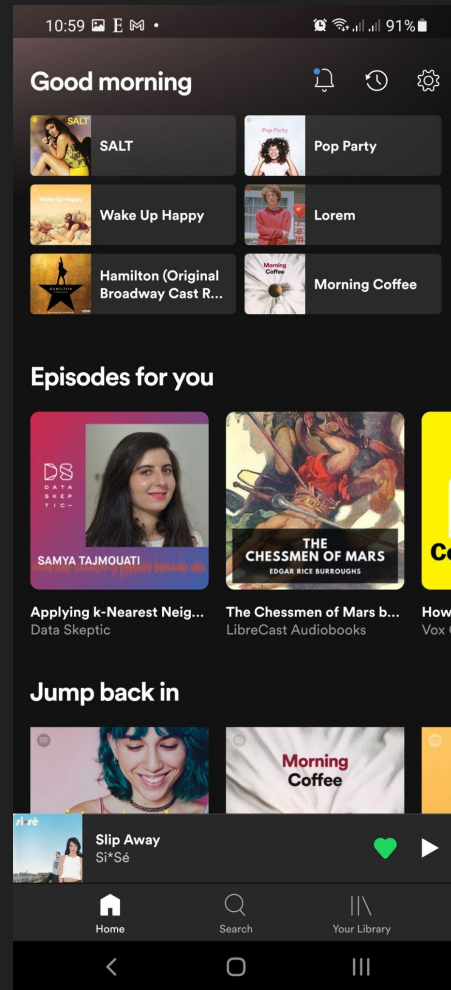
# Feature Workflows



# 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

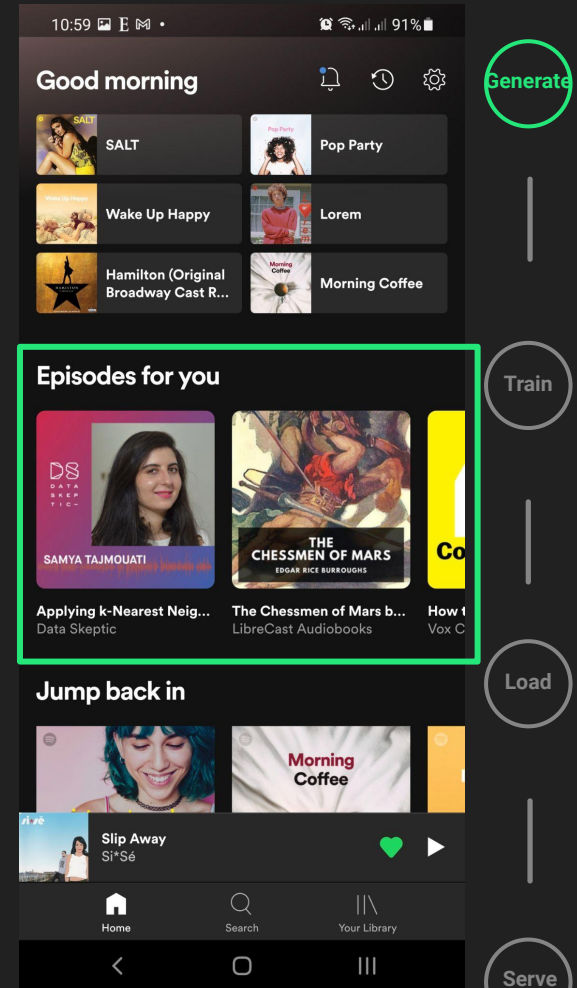
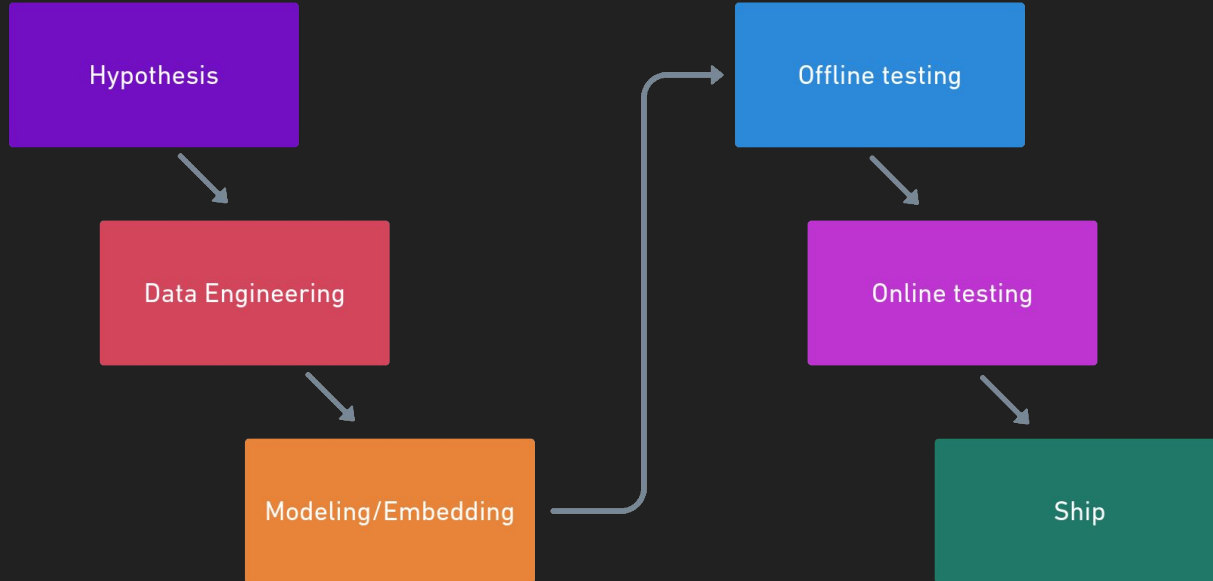
- Bag-of-words
- BERT

## Context

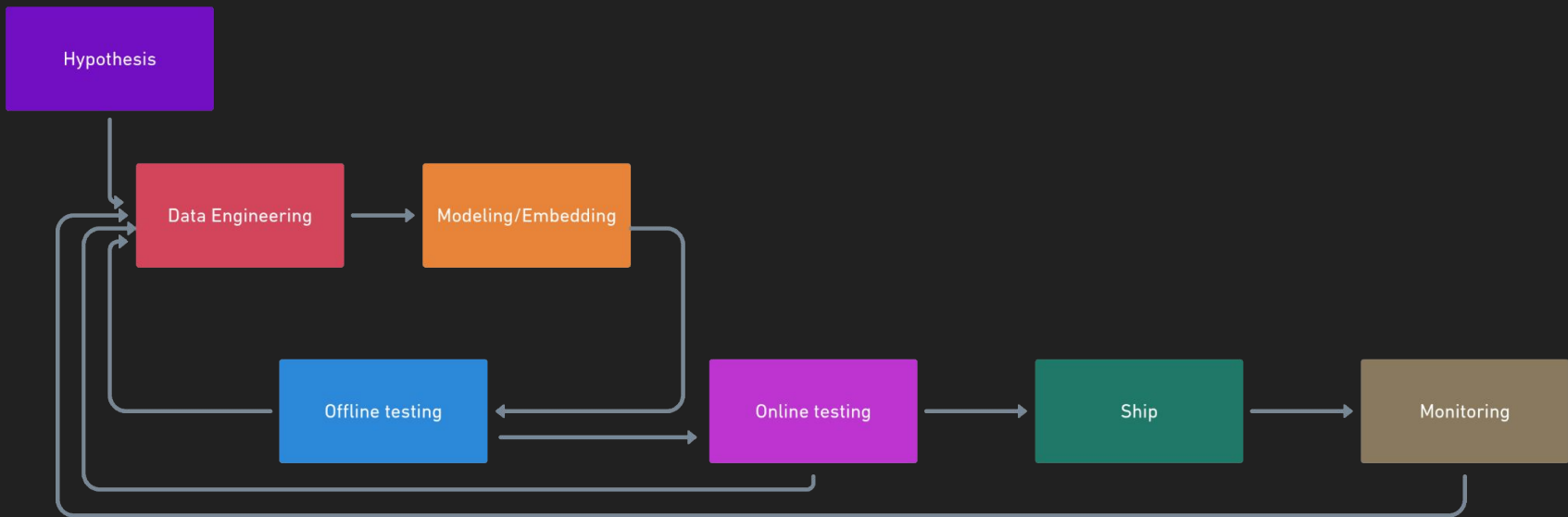
- Local time, device, etc.

A circular button with a red border and the word "Generate" in red text.A vertical line separator.A circular button with a red border and the word "Train" in red text.A vertical line separator.A circular button with a red border and the word "Load" in red text.A vertical line separator.A circular button with a red border and the word "Serve" in red text.

# Feature Engineering



# Feature Engineering -- real



Generate

Train

Load

Serve

# Feature Engineering -- trouble

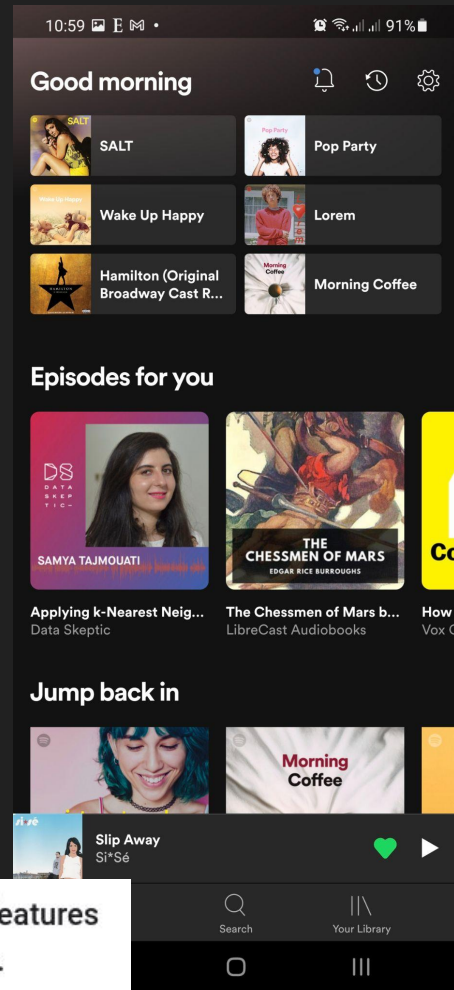
## Home x Salem

#home-ml-ops

July 2020



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 :(

A circular button with a green border and the word "Generate" in green text.A circular button with a white border and the word "Train" in white text.A circular button with a white border and the word "Load" in white text.A circular button with a white border and the word "Serve" in white text.

# Feature Engineering

Killer features for fast iteration

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

Generate

Train

Load

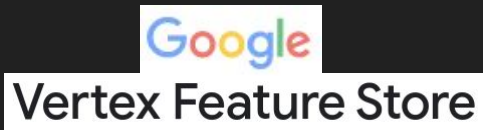
Serve



# Feature Engineering



Zipline



Hopsworks

# Jukebox 2.0



FEAST



TECTON



# Adding a new feature

## First steps

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

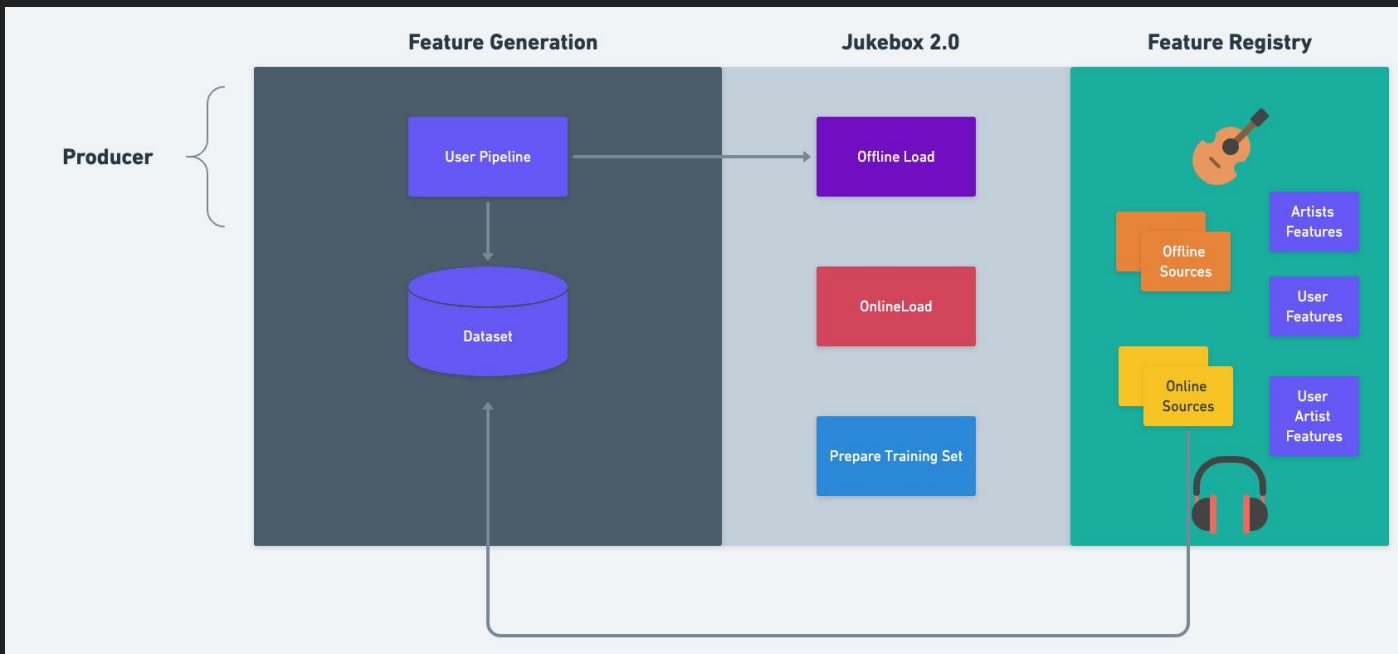
A circular button with a red border and the word "Generate" in red text.A circular button with a white border and the word "Train" in white text.A circular button with a white border and the word "Load" in white text.A circular button with a white border and the word "Serve" in white text.

# 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



Generate

Train

Load

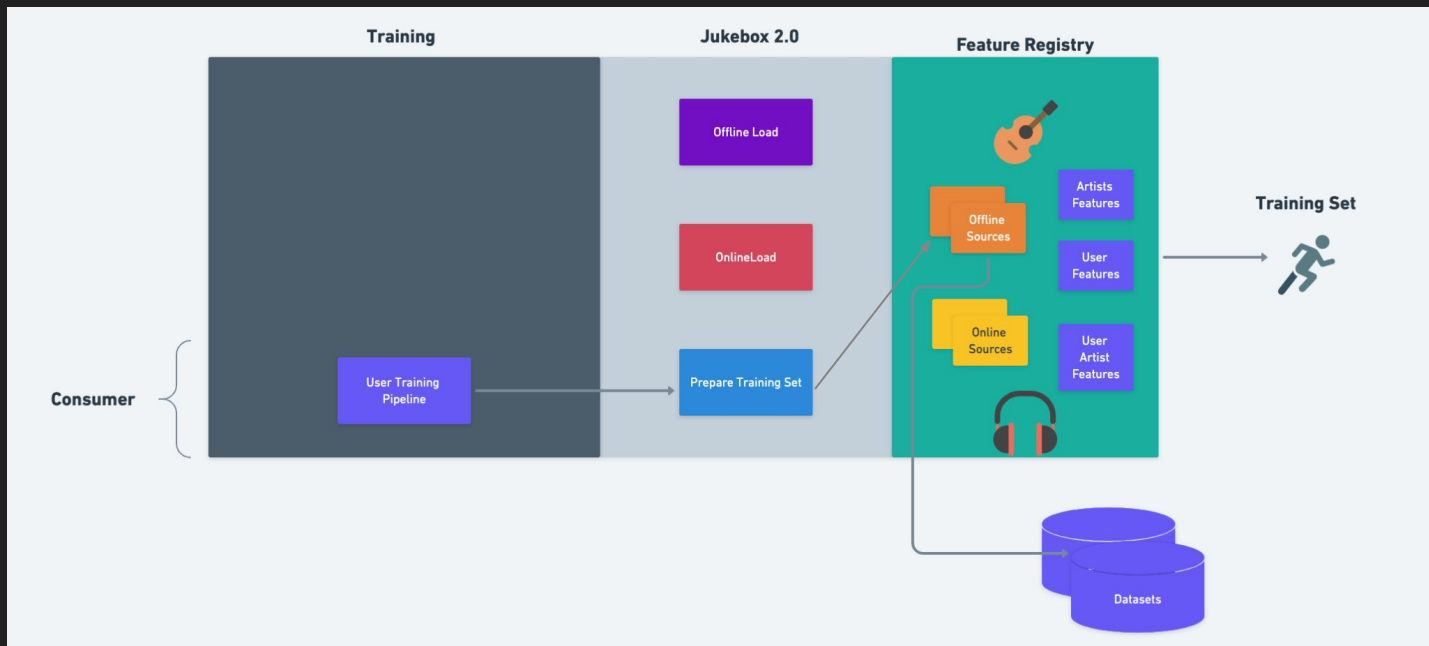
Serve

# 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 at the right point in time



Generate

Train

Load

Serve

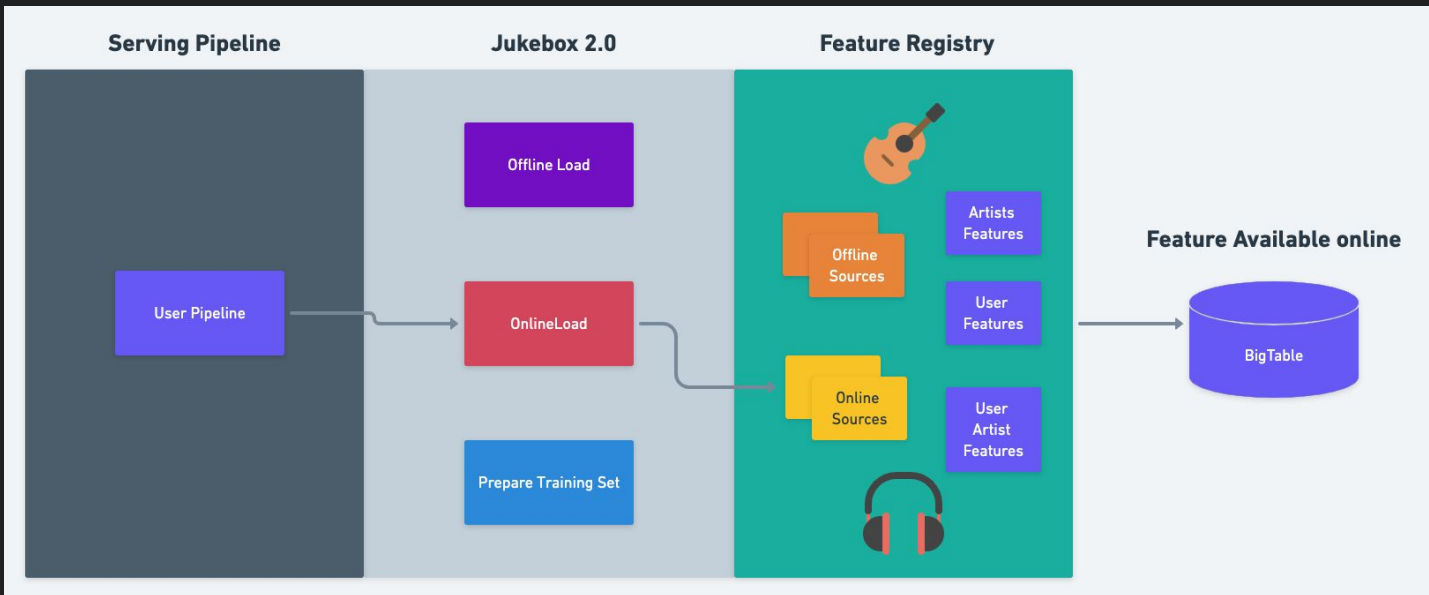
# Online Data Loader

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



Generate

Train

Load

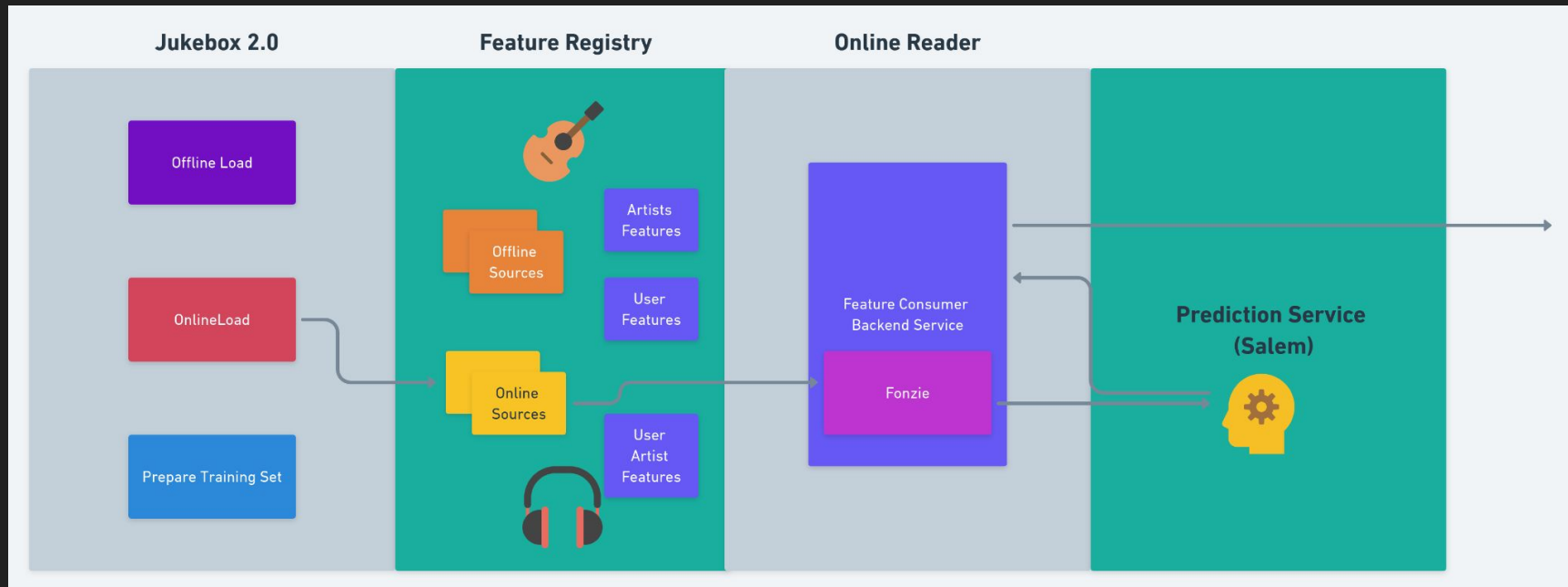
Serve

# 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

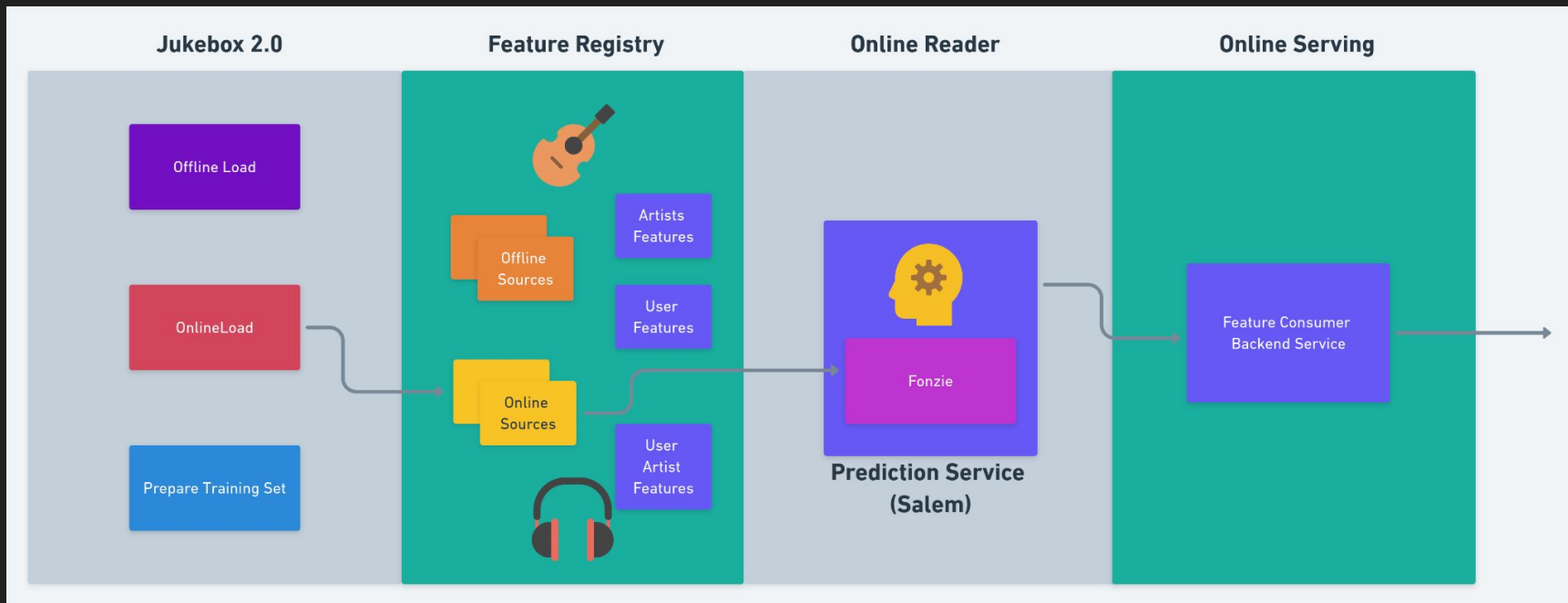
## Consuming a feature





# Online Serving

## Idealized feature consumer view



Generate

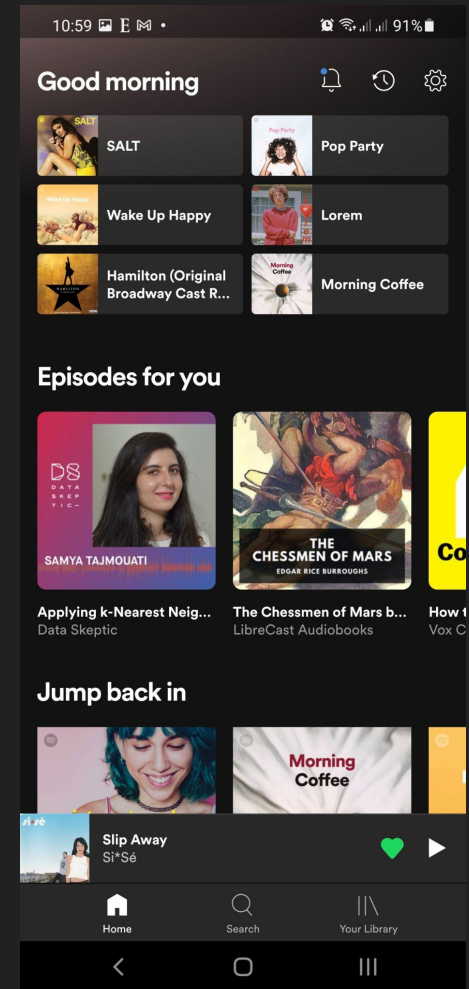
Train

Load

Serve

# Home Use Case

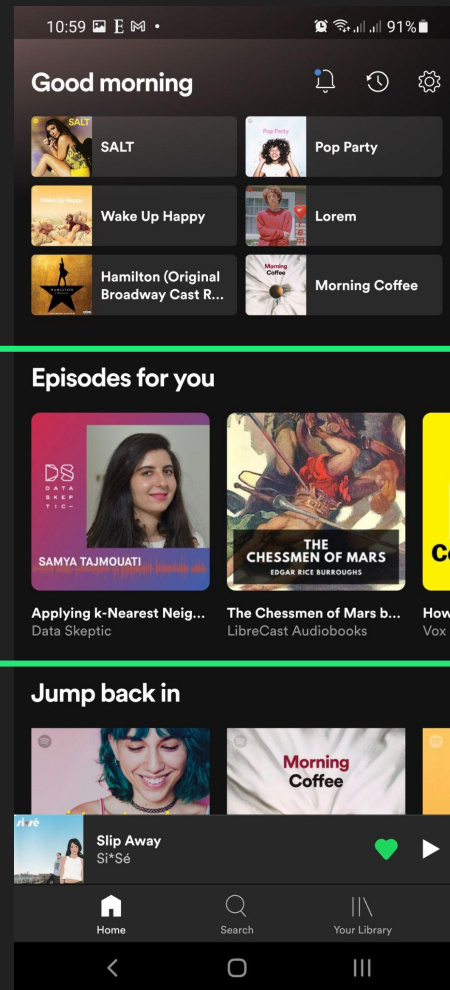
- Many models, multiple teams
- Daily, near-real-time and contextual features
- Five feature stores in all



# 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

## Spotify Machine Learning $\beta$

HOME PROJECTS **FEATURE GALLERY**

### Feature Gallery ☆ 🔗 SUPPORT

All Machine Learning features available through our Feature Marketplace.  
For more details on how to register a feature, check out the [Feature Documentation](#).

**Filters (0)** | **1003 Feature Families** ?

Feature Family	Entity	Feature Count	Features	
/show/showNormalizedMusicVector.batch	show	1	/show/PromotionsTargeting.JoinedShowNewFeatures.gcs/showNormalizedMusicVector.batch	<a href="#">DETAILS</a>
/entity/user_artist/component/user-artist-interactions-pipeline/days_28_summary.total_skips	user_artist	1	/entity/user_artist/component/user-artist-interactions-pipeline/name/days_28_summary.total_skips	<a href="#">DETAILS</a>
/user_artist/days_7_summary.top_track_counts_by_count	user_artist	1	/user_artist/user-artist-interactions-pipeline/days_7_summary.top_track_counts_by_count	<a href="#">DETAILS</a>
/user_artist/splitstreamsPerTrack_7dWindowMaxTracks_byPlatform	user_artist	1	/user_artist/listenerType/splitstreamsPerTrack_7dWindowMaxTracks_byPlatform	<a href="#">DETAILS</a>
/user_artist/splitstreamsPerTrack_maxStreamsPerTrack28DayWindow_byTopType	user_artist	1	/user_artist/ccd-streams/splitstreamsPerTrack_maxStreamsPerTrack28DayWindow_byTopType	<a href="#">DETAILS</a>
/user_adcreative/realtimegenre	user_adcreative	1	/user_adcreative/paradox-ae/realtimegenre	<a href="#">DETAILS</a>
/user/avg_danceability	user	2	/user/jukebox-apm-sample/avg_danceability, /user/ml-golden-path/avg_danceability	<a href="#">DETAILS</a>

# Feature Overview

[Feature Gallery](#) > [Feature Family](#)

## /track/artists

### OVERVIEW

#### Overview ☆

[? SUPPORT](#)

Feature Family is a group of features that share the same entity (i.e. user), and may express a similar idea or measurement (i.e. country). They can span across different sources.



Filters (0) | 1 Feature ②

🔍 Search



Feature	Entity	Owner	Online Source	Offline Source	Lineage	
/track/catapult-metadata-entities/artists	track	catapult	N/A	MetadataEntities.Track.gcs	0 Models	<a href="#">DETAILS</a>

1 of 1



1



# Feature Details

## Feature Details ✕

OVERVIEW
QUALITY
LINEAGE

---

Feature Family ⓘ	/track/artists
Feature ⓘ	/track/catapult-metadata-entities/artists
Entity	track
Owner	catapult
Online Source	N/A
Offline Source	<a href="#">MetadataEntities.Track.gcs</a>
Description	Lists the artists who contributed to a track specifying their role and order. The same artist can have different roles. The same role can have multiple artists that can be ordered using the artist_ordinal_number field.
Scope ⓘ	SCOPE_UNDEFINED
Life Cycle	LIFECYCLE_UNDEFINED
Data Type	DATA_TYPE_ANY
Lineage	0 Models

## Feature Details ✕

OVERVIEW
QUALITY
LINEAGE

---

### Offline Source

Data Endpoint: [MetadataEntities.Track.gcs](#)

### Scope

Scope from Data Endpoint.

Access	Life Cycle	TC4D
Broad	Production	

### Delivery

DAILY DELIVERY

SLO CONFIGURED

2021-09-27 is OK

\* Status Legend [TRY SLO CALCULATOR](#)

[Monitor this endpoint in Data Monitoring tool →](#)

Last 28 days compared to the previous 28 days.

10 hrs

DELIVERY TIME - P90

4h 49m

-2.0%

SLO COMPLIANCE

100.0%

0.0%

SLO BREACHES

0

BREACH DURATION

0 min

# 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

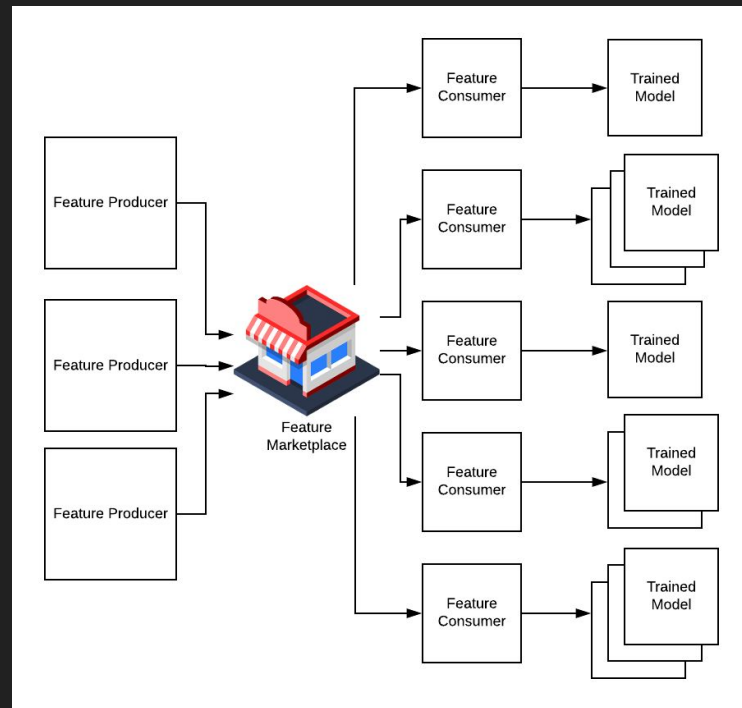
Build different datasets, serve different models – you won't need to build another data pipeline that joins features together or another backend service to query it when you need to fetch actual feature values for serving your ML model



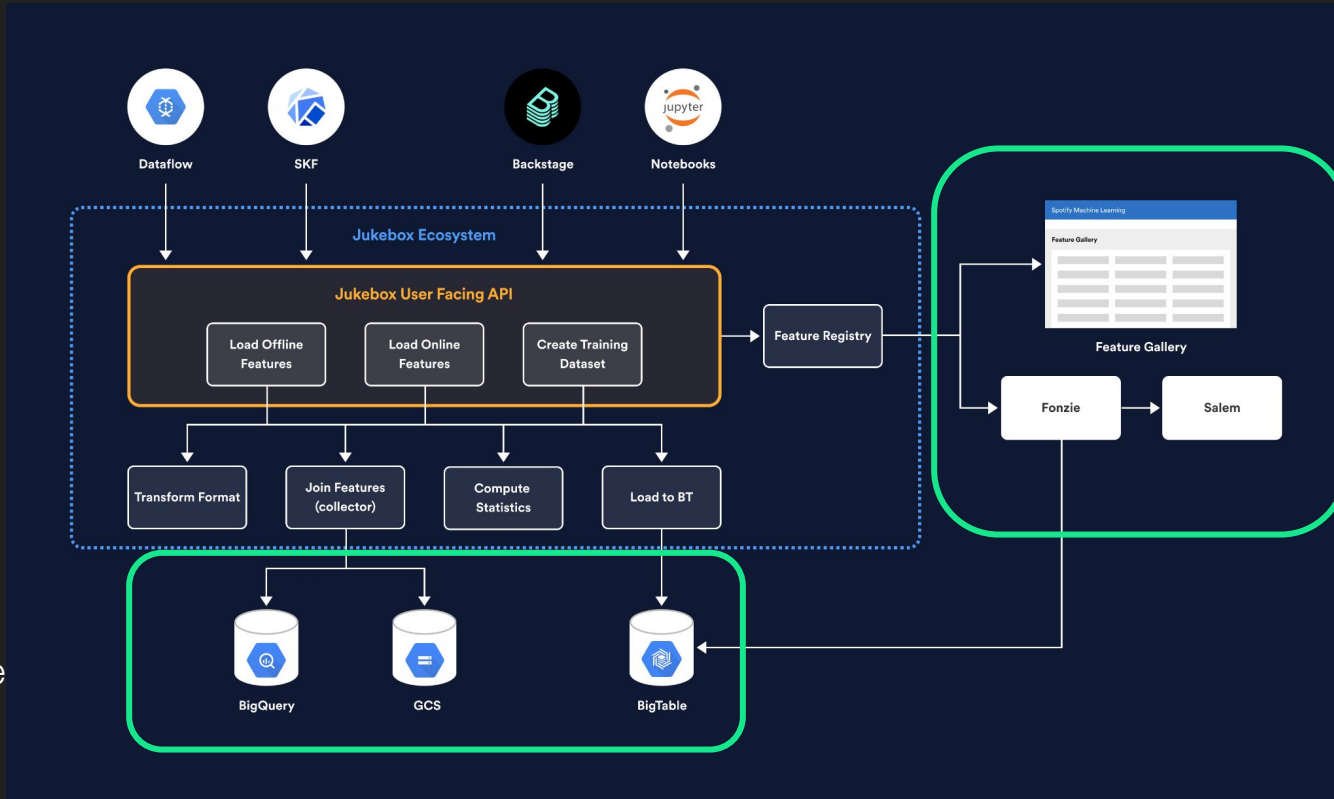
# Where we're going - Feature Marketplace

Saving Spotify's 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

**Daniel Kristjansson**  
danielk@spotify.com

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

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

@\_amankhan

