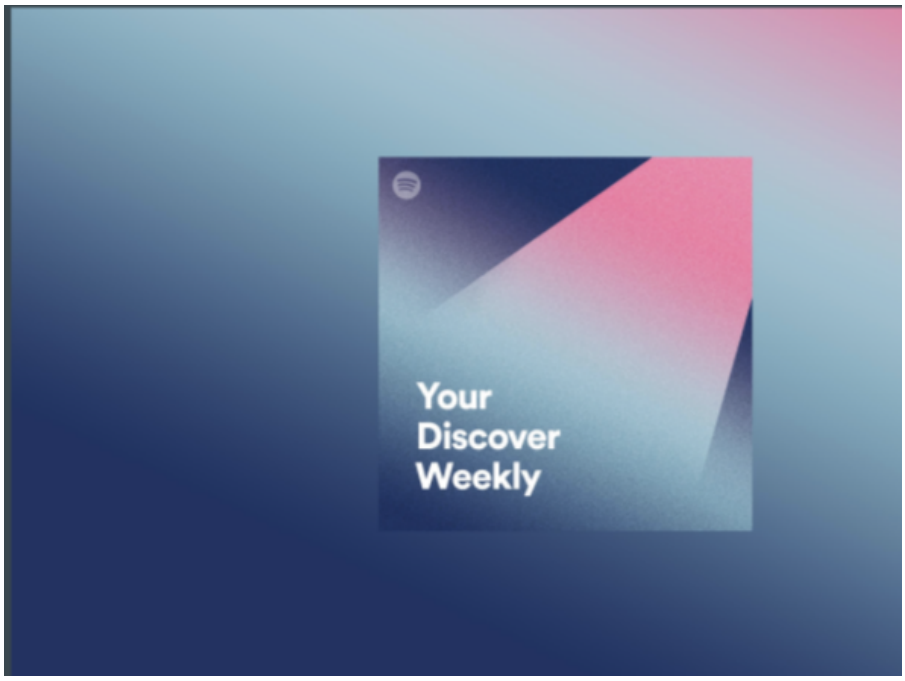
README.md

Spotify Recommendations

Overview

The Problem @ ✋

- I like Spotify and I like discovering music
- Music generation is very popularity based nowadays
- Discovery/interaction with music happens via playlists mostly nowadays
- Music curation tools for playlists are scarce
- Spotify has a lot of tools for interacting with their data/features



Million Playlist Dataset 

- Spotify hosted the 2018 RecSys challenge of 1 Million Playlists made by users
- Publicly released the dataset

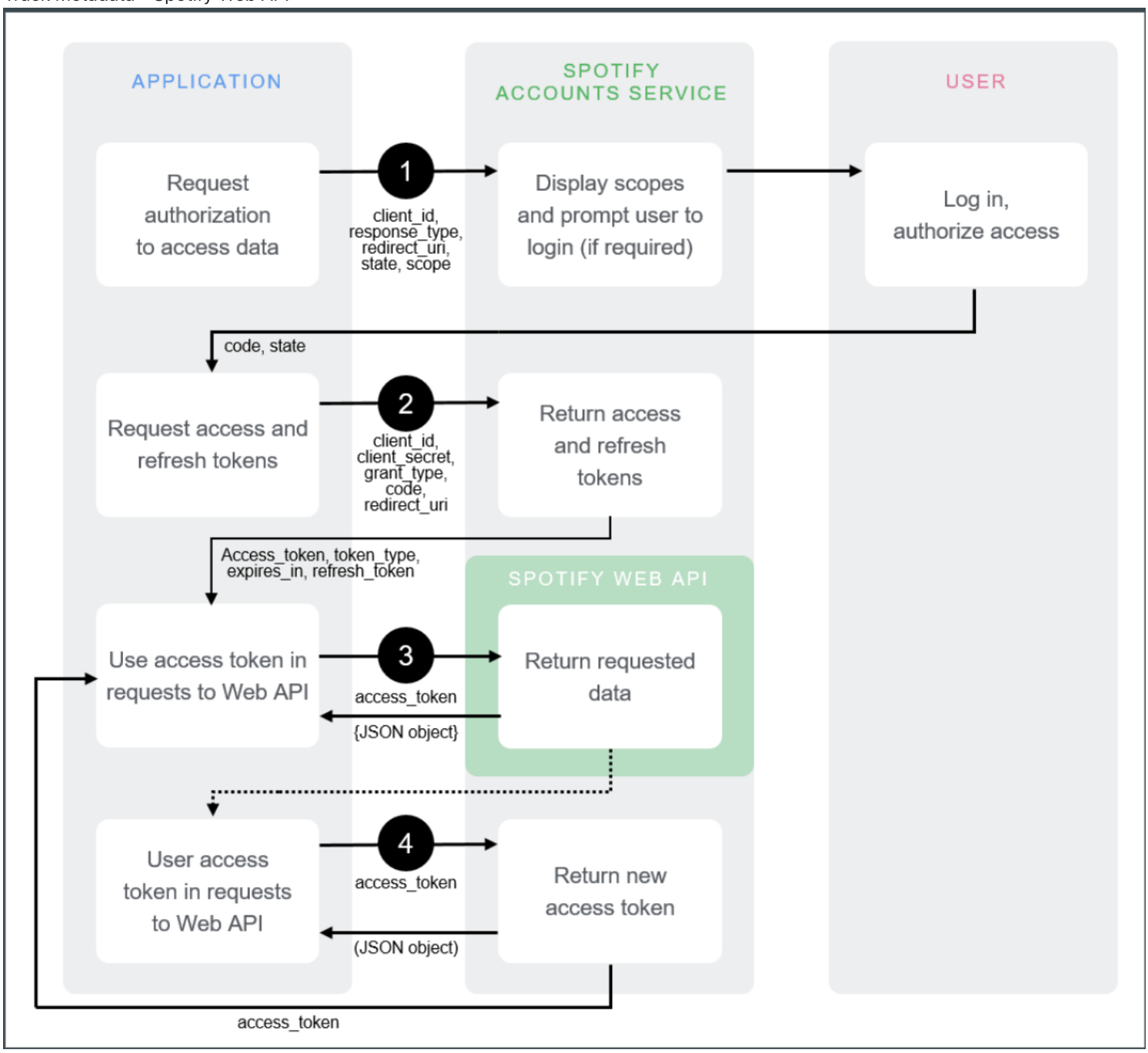
- No associated metadata of tracks

Property	Value
Number of playlists	1,000,000
Number of tracks	66,346,428
Number of unique tracks	2,262,292
Number of unique albums	734,684
Number of unique artists	295,860
Number of unique playlist titles	92,944
Number of unique normalized playlist titles	17,381
Average playlist length (tracks)	66.35

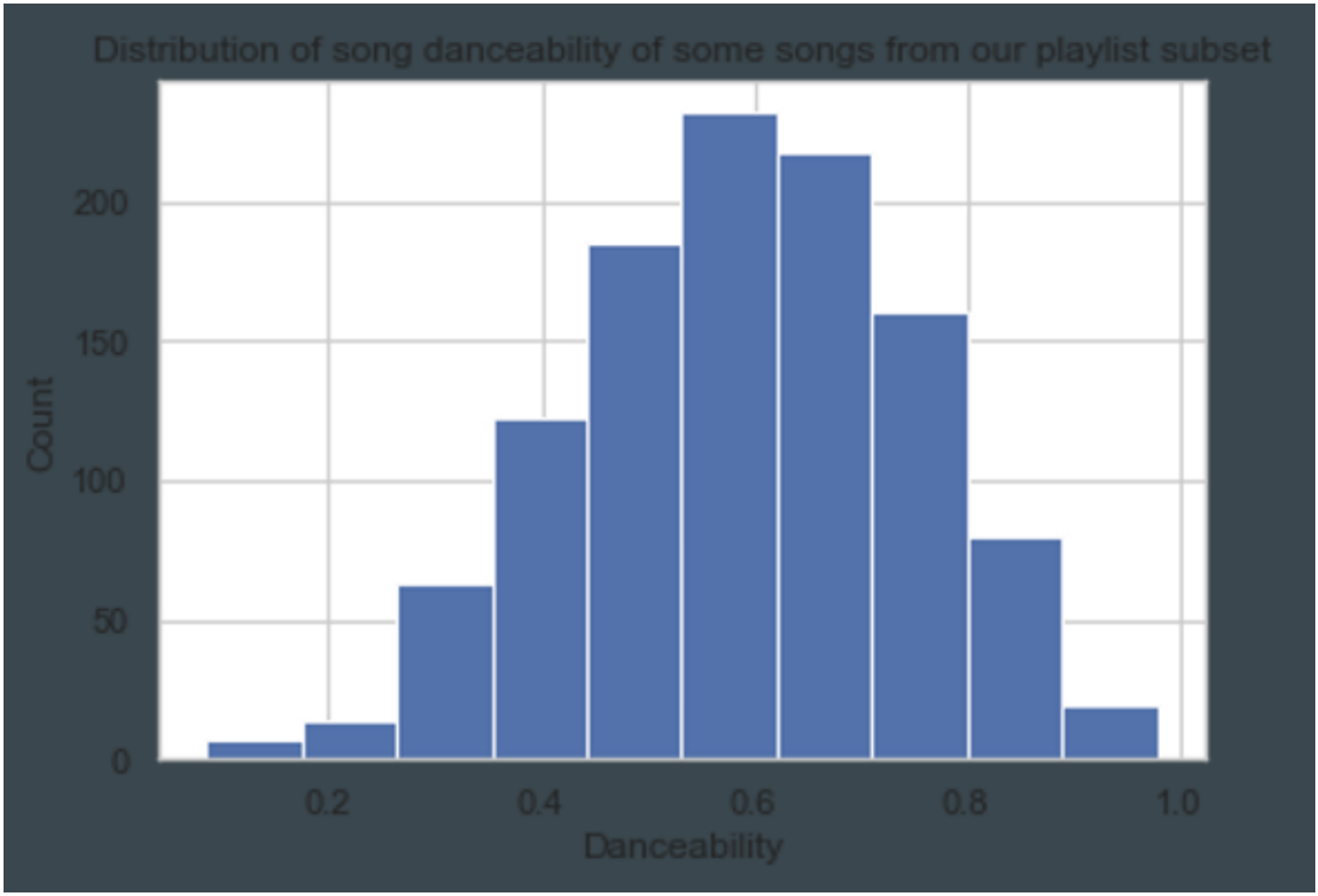

```
{
        "name": "musical",
        "collaborative": "false",
        "pid": 5,
        "modified_at": 1493424000,
        "num_albums": 7,
        "num_tracks": 12,
        "num_followers": 1,
        "num_edits": 2,
        "duration_ms": 2657366,
        "num_artists": 6,
        "tracks": [
            {
                "pos": 0,
                "artist_name": "Degiheugi",
                "track_uri": "spotify:track:7vqa3sDmtEaVJ2gcvxtRID",
                "artist_uri": "spotify:artist:3V2paBXEoZIAhfZRJmo2jL",
                "track_name": "Finalement",
                "album_uri": "spotify:album:2KrRMJ9z7Xjoz1Az4O6UML",
                "duration_ms": 166264,
                "album_name": "Dancing Chords and Fireflies"
            },
            {
                "pos": 1,
                "artist_name": "Degiheugi",
                "track_uri": "spotify:track:23EOmJivOZ88WJPUbIPjh6",
                "artist_uri": "spotify:artist:3V2paBXEoZIAhfZRJmo2jL",
                "track_name": "Betty",
                "album_uri": "spotify:album:3lUSlvjUoHNA8IkNTqURqd",
                "duration_ms": 235534,
                "album_name": "Endless Smile"
            },
            {
                "pos": 2,
                "artist_name": "Degiheugi"
```

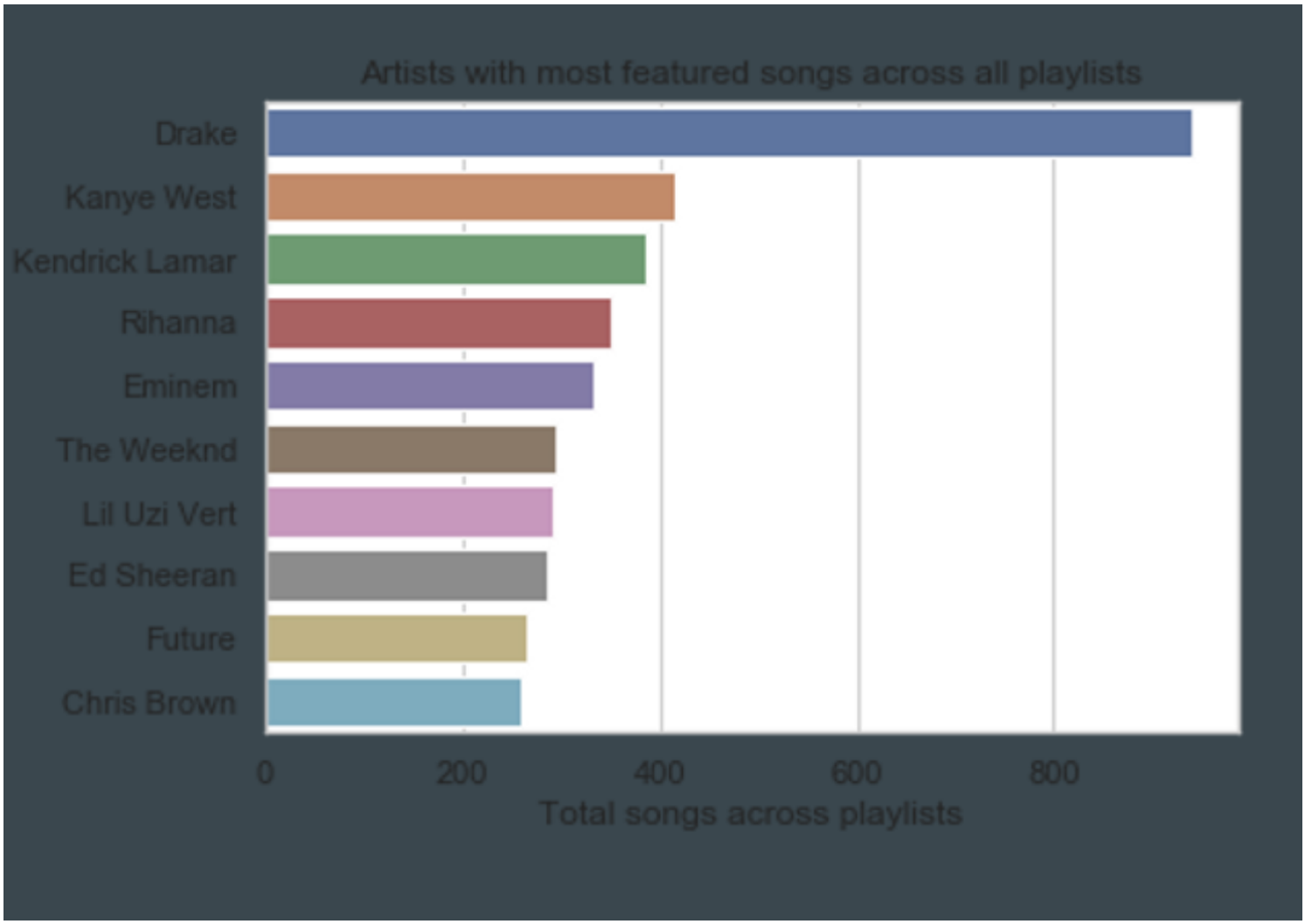

"artist_name": "Beginegi",
"track_uri": "spotify:track:1vaffTCJxkyqeJY7zF9a55",
"artist_uri": "spotify:artist:3V2paBXEoZIAhfZRJmo2jL",
"track_name": "Some Beat in My Head",
"album_uri": "spotify:album:2KrRMJ9z7Xjoz1Az4O6UML",
"duration_ms": 268050,
"album_name": "Dancing Chords and Fireflies"
},
// 8 tracks omitted
{
"pos": 11,
"artist_name": "Mo' Horizons",
"track_uri": "spotify:track:7iwx00eBzeSSSy6xfESyWN",
"artist_uri": "spotify:artist:3tuX54dqgS8LsGUvNzgrpP",
"track_name": "Fever 99\u00b0",
"album_uri": "spotify:album:2Fg1t2tyOSGWkVYHlFfXVf",
"duration_ms": 364320,
"album_name": "Come Touch The Sun"
}
],
}

Track Metadata - Spotify Web API



EDA - MPD + Track Metadata





The Approach 🤠

- Build infrastructure/tooling to easily build and deploy Spotify-based applications and models
- Do so in a streamlined reproducible fashion
- Automate as much as possible
- Infrastructure-as-code as much as possible
- Make it easy for a SWE/DE/DS/MLE/DevOps person to extend upon
- Use a MLFlow wrapped custom cosine similarity model as a test of E2E tooling from the ML end

Infrastructure considerations

- arbitrary custom model can be made easily
- models can connect to growing data sources
- deployment of said model is straightforward
- can be managed by ci/cd easily
- adaptable to data, models, and code changes

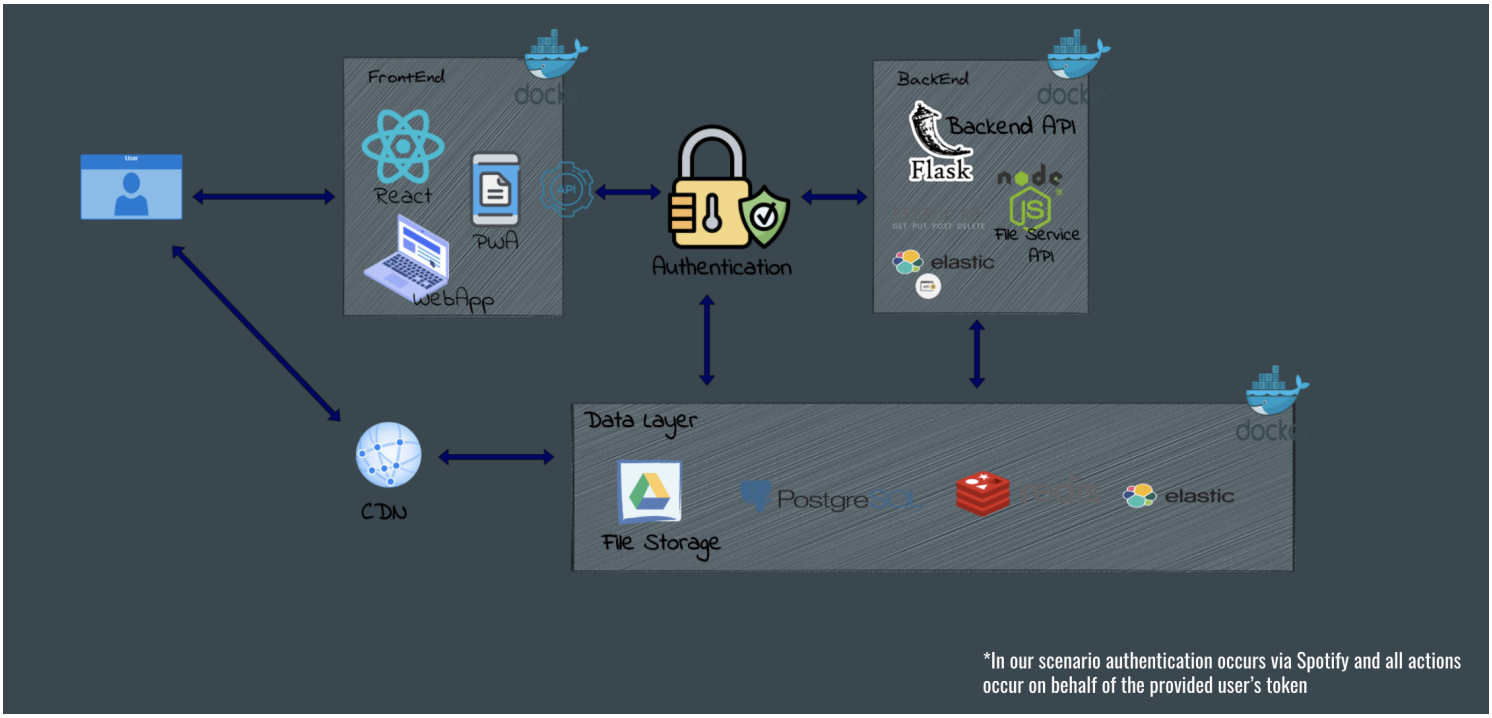
The Experiment - Let's Keep it Simple

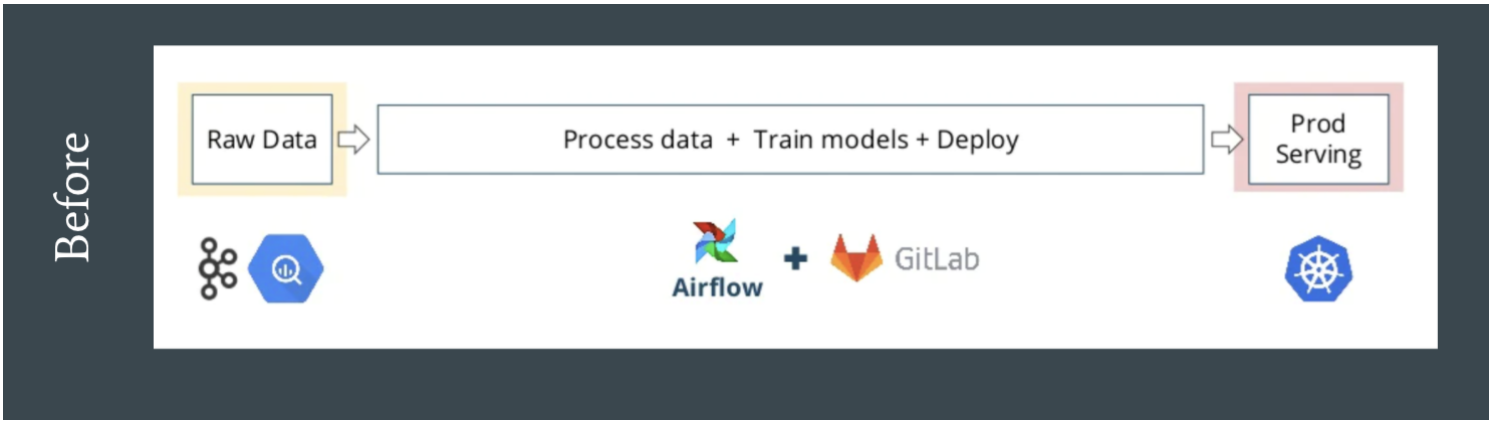
- Data: Create training and test data by way of obfuscation on MPD
- Take all the playlists and randomly remove a certain amount of songs and use as training data
- Use those removed songs and use as testing data

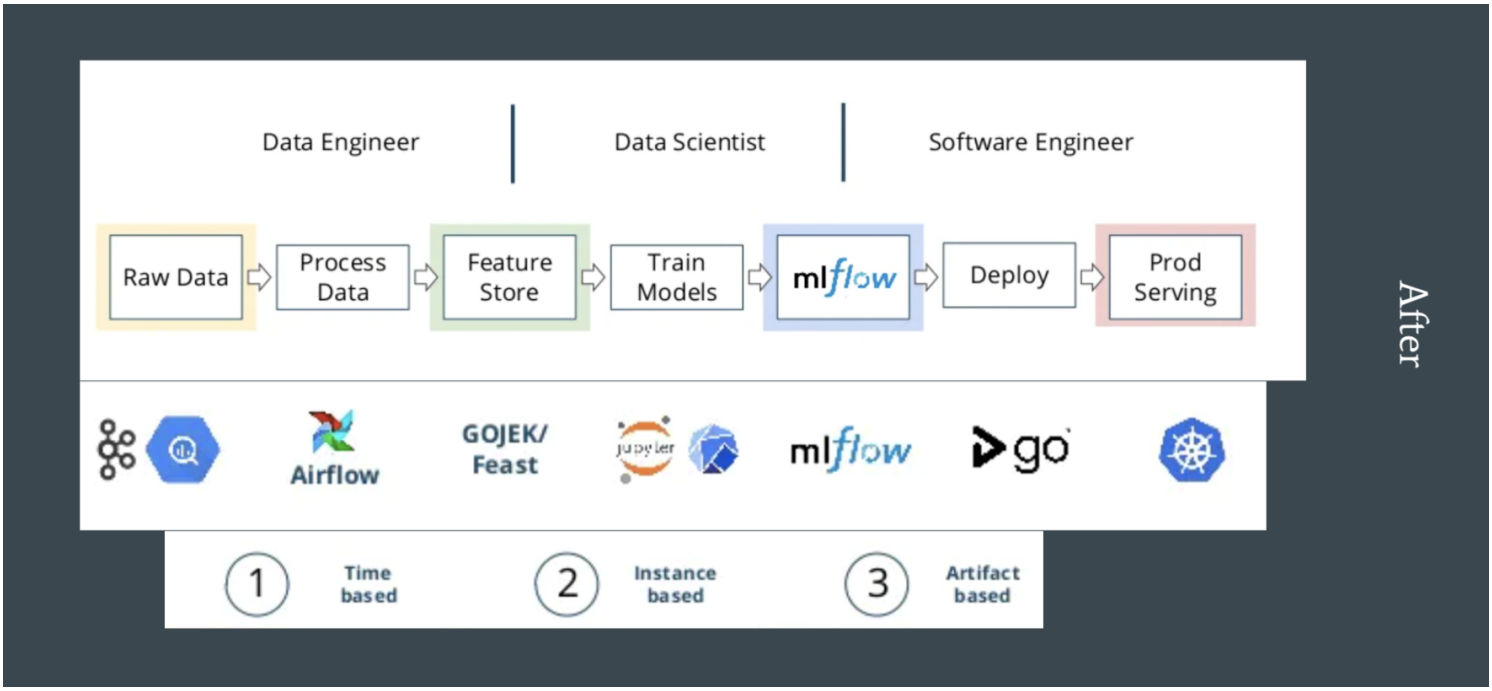
- Task: Use training songs in the playlist to predict the songs that were removed Method: Aggregate features of the songs to create a representative feature representation for the playlist w.r.t songs and use a similarity search
- Features used: Spotify given track features
- Search: Cosine Similarity

Introducing SpotifyPlaylists! 🎶

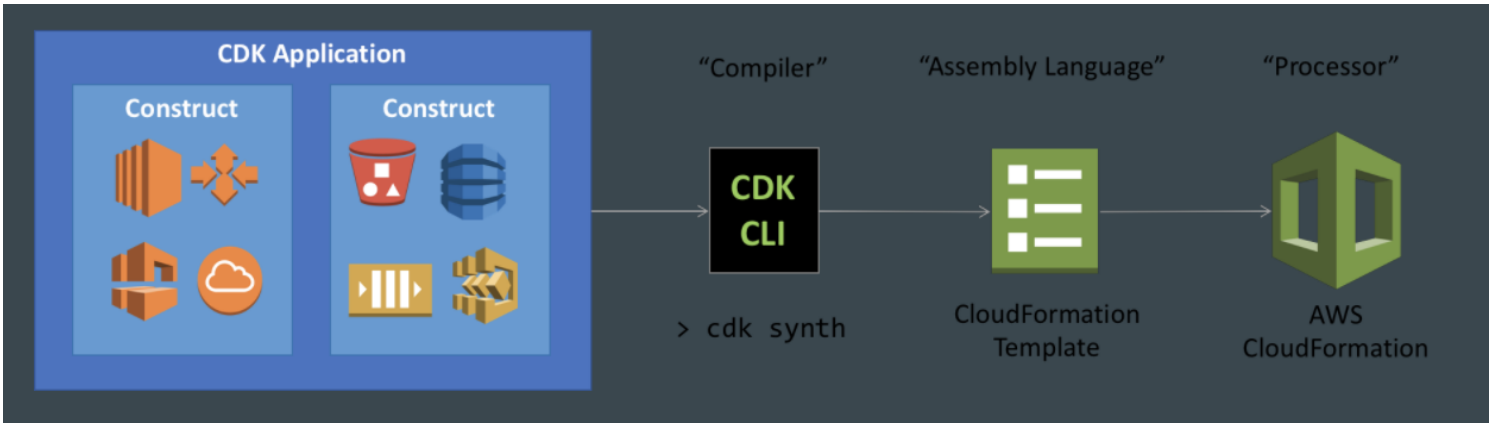
High Level Software Architecture

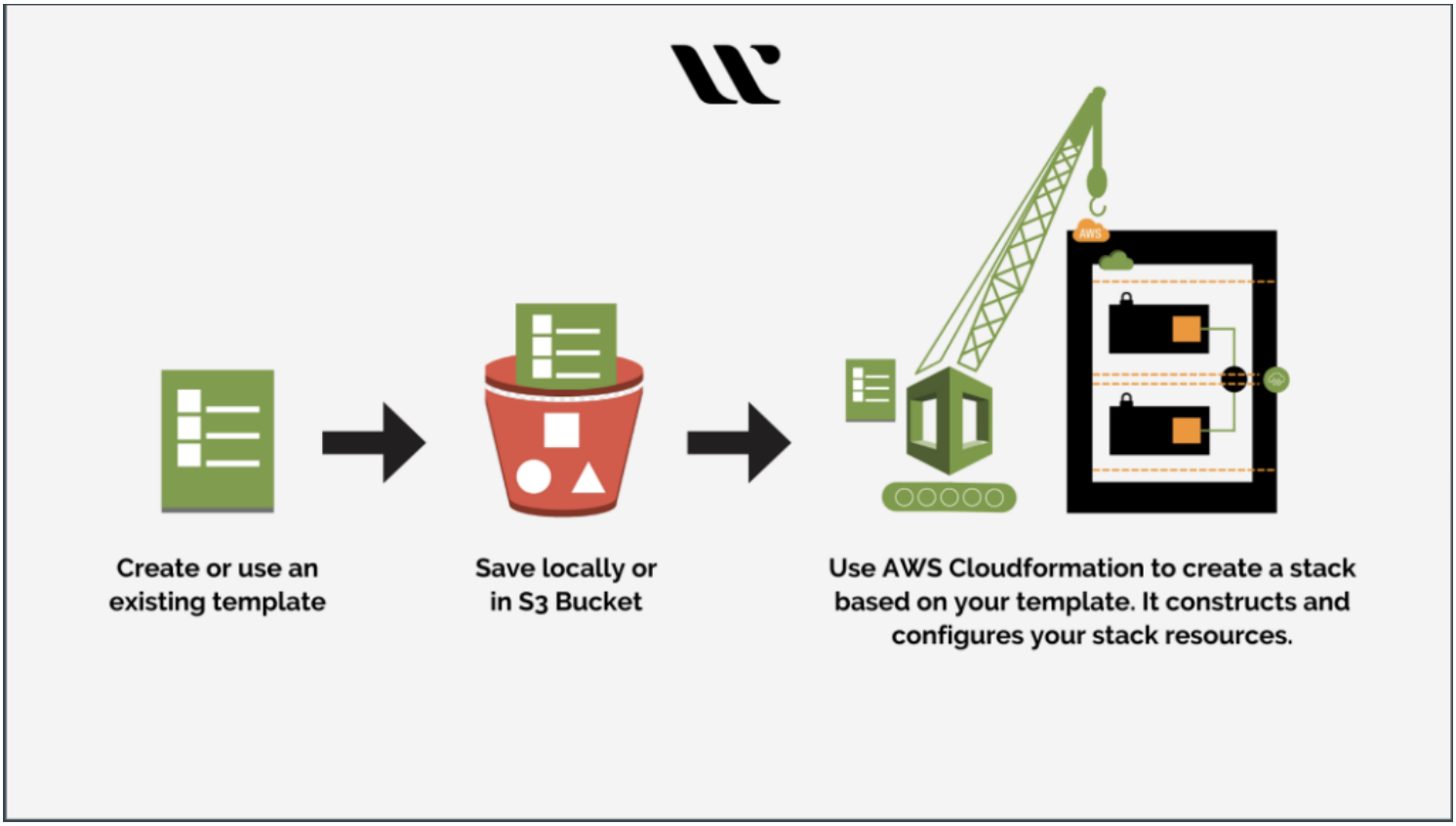


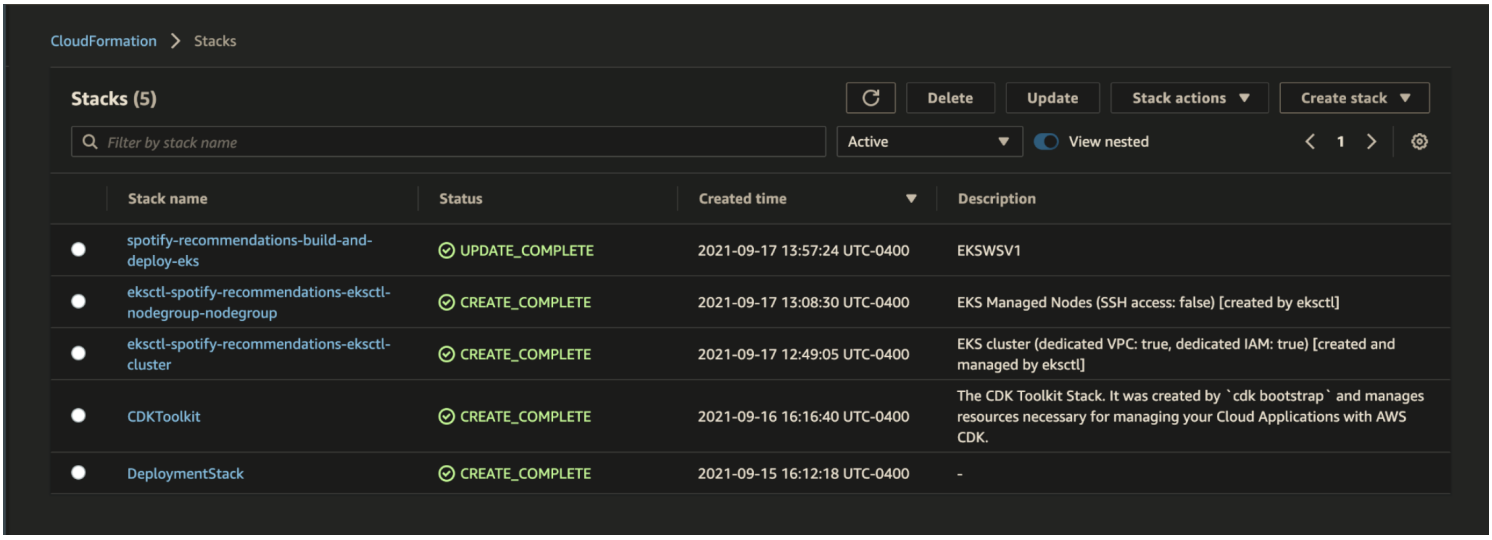




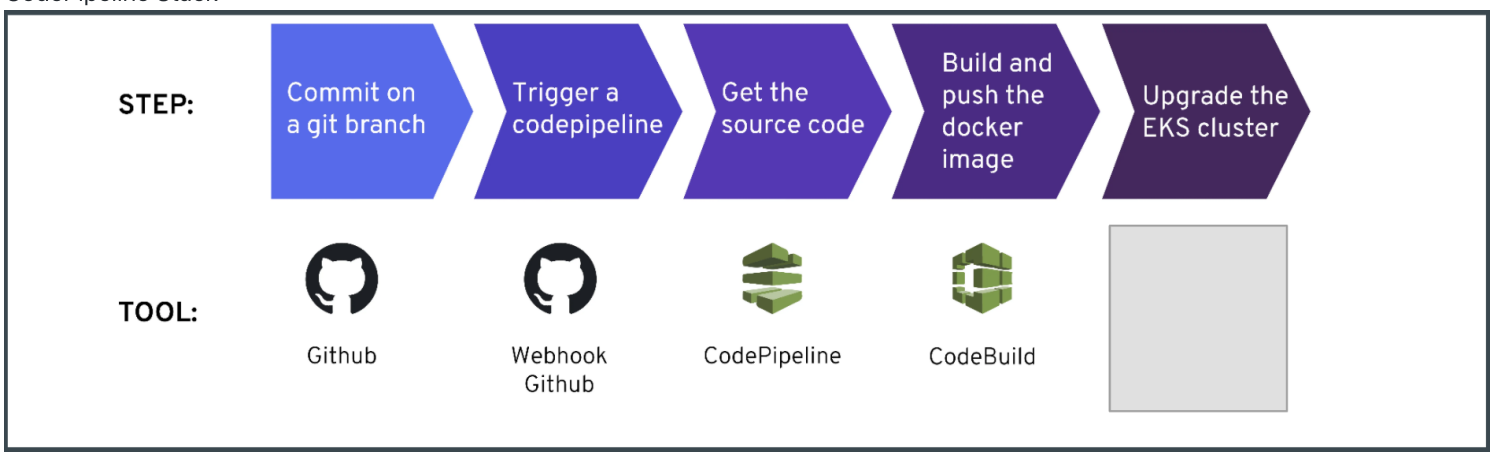
Deployment Stack







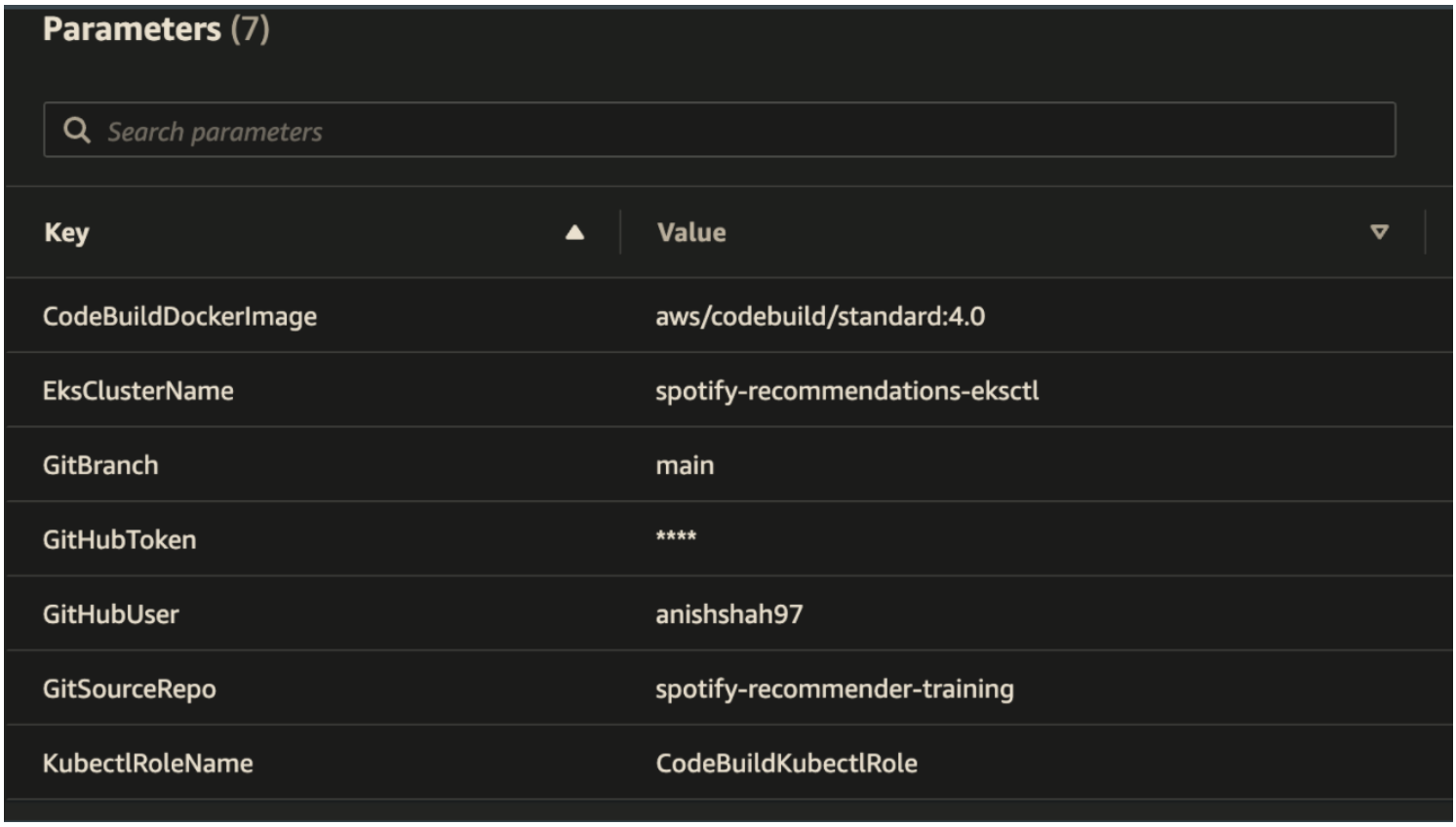
CodePipeline Stack

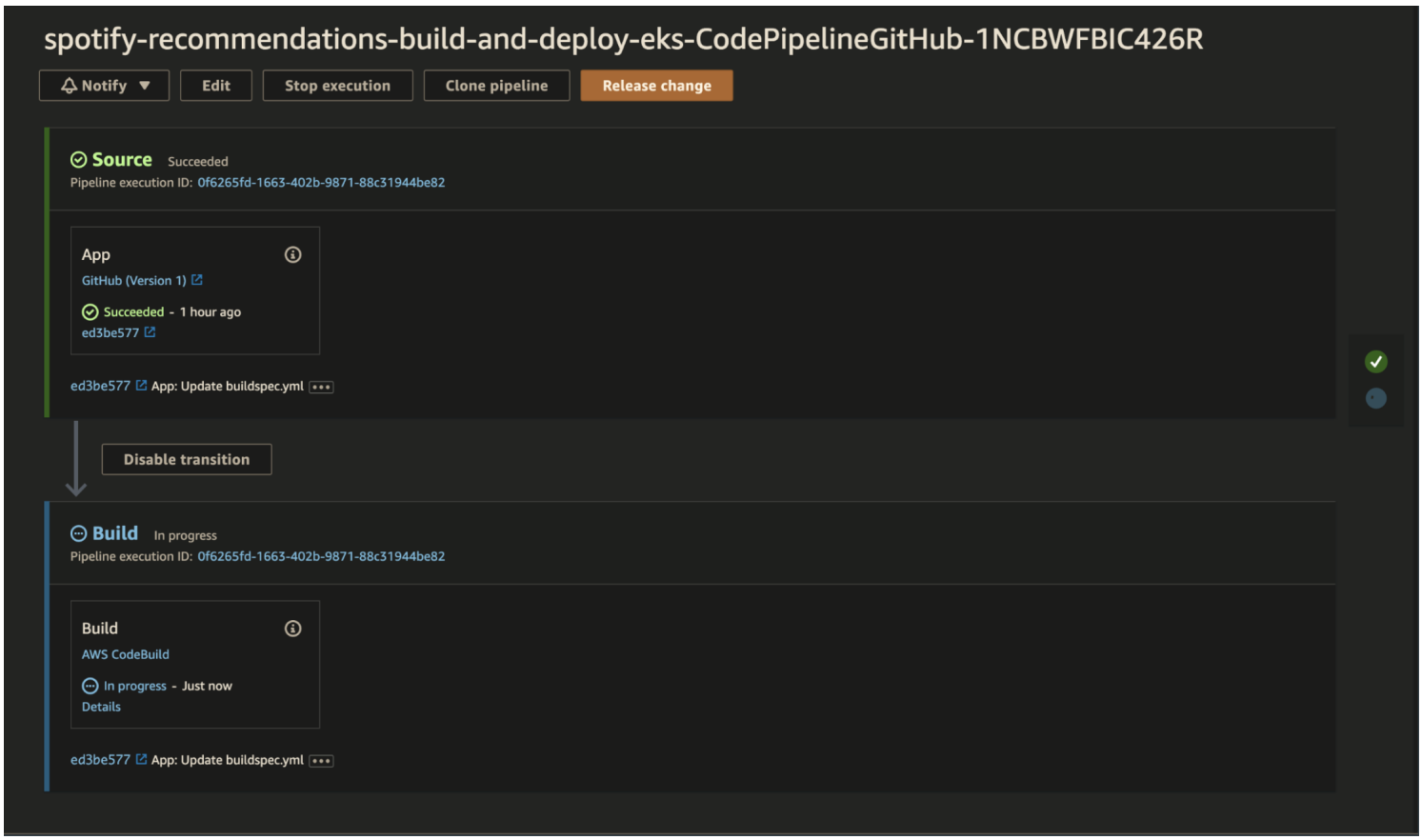


Resources (6)

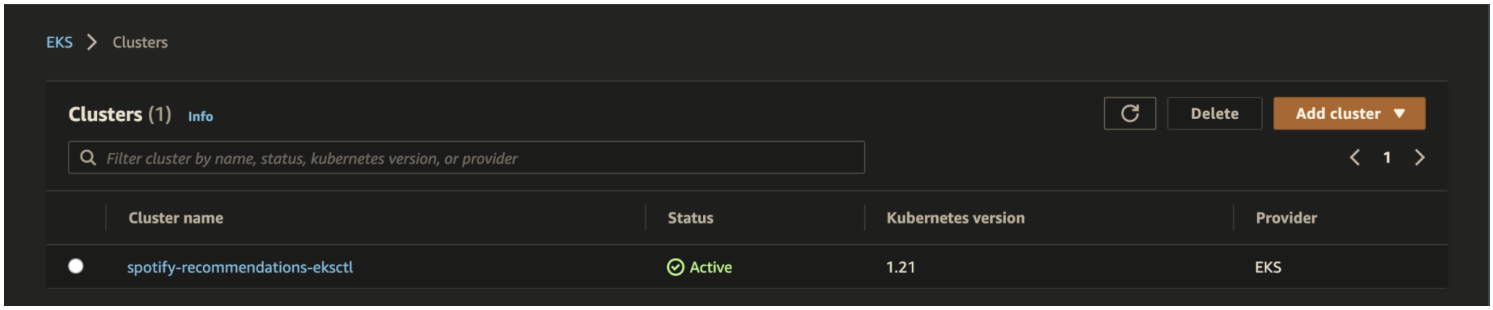
Search resources

Logical ID	Physical ID	Type	Status
CodeBuildProject	spotify-recommendations-build-and-deploy-eks	AWS::CodeBuild::Project	UPDATE_COMPLETE
CodeBuildServiceRole	spotify-recommendations-build-CodeBuildServiceRole-1374IJ0LEX8ST	AWS::IAM::Role	CREATE_COMPLETE
CodePipelineArtifactBucket	spotify-recommendations-codepipelineartifactbuck-nlm20ugvjdwi	AWS::S3::Bucket	CREATE_COMPLETE
CodePipelineGitHub	spotify-recommendations-build-and-deploy-eks-CodePipelineGitHub-1NCBWFBIC426R	AWS::CodePipeline::Pipeline	UPDATE_COMPLETE
CodePipelineServiceRole	spotify-recommendations-bu-CodePipelineServiceRole-17GZSS7Z34GTC	AWS::IAM::Role	CREATE_COMPLETE
EcrDockerRepository	spotify-recommendations-build-and-deploy-eks-ecrdockerrepository-lf4r2vvxh0fp	AWS::ECR::Repository	CREATE_COMPLETE

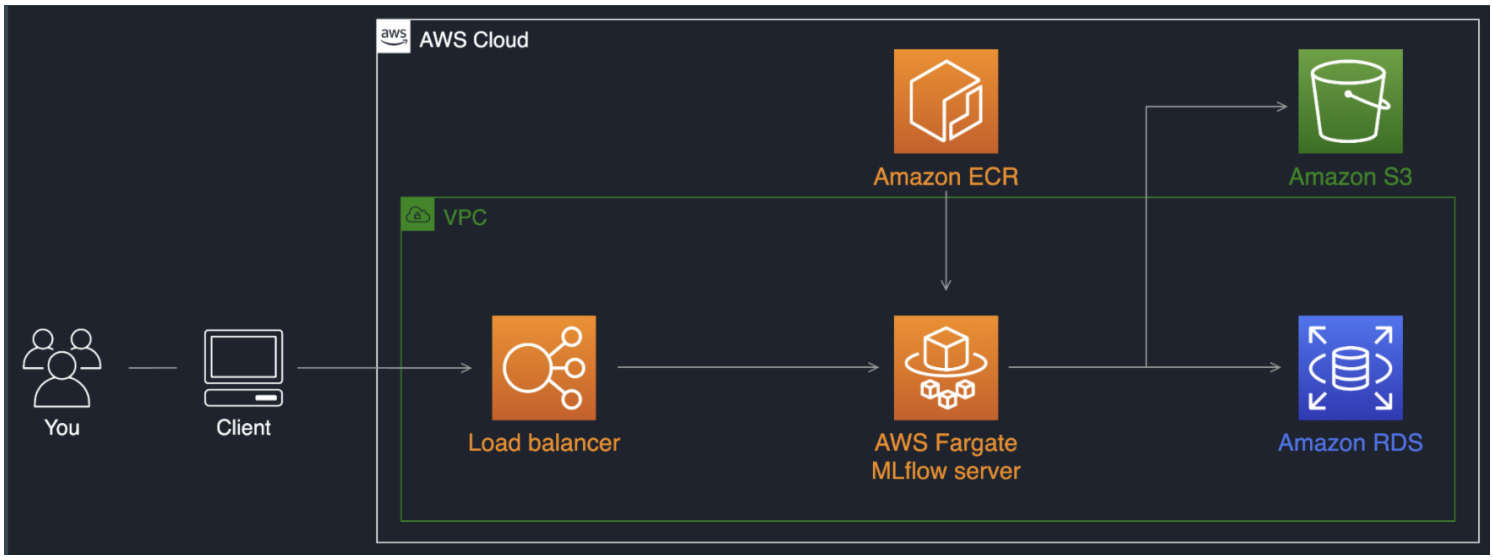

Parameters (7)
Search parameters
Key
Value
CodeBuildDockerImage
aws/codebuild/standard:4.0
EksClusterName
spotify-recommendations-eksctl
GitBranch
main
GitHubToken
****
GitHubUser
anishshah97
GitSourceRepo
spotify-recommender-training
KubectlRoleName
CodeBuildKubectlRole


spotify-recommendations-build-and-deploy-eks-CodePipelineGitHub-1NCBWFBIC426R
Notify
Edit
Stop execution
Clone pipeline
Release change
Source
Succeeded
Pipeline execution ID: 0f6265fd-1663-402b-9871-88c31944be82
App
GitHub (Version 1)
Succeeded - 1 hour ago
ed3be577
ed3be577 App: Update buildspec.yml
Disable transition
Build
In progress
Pipeline execution ID: 0f6265fd-1663-402b-9871-88c31944be82
Build
AWS CodeBuild
In progress - Just now
Details
ed3be577 App: Update buildspec.yml

Pre-requirements: EKS Cluster (using eksctl)

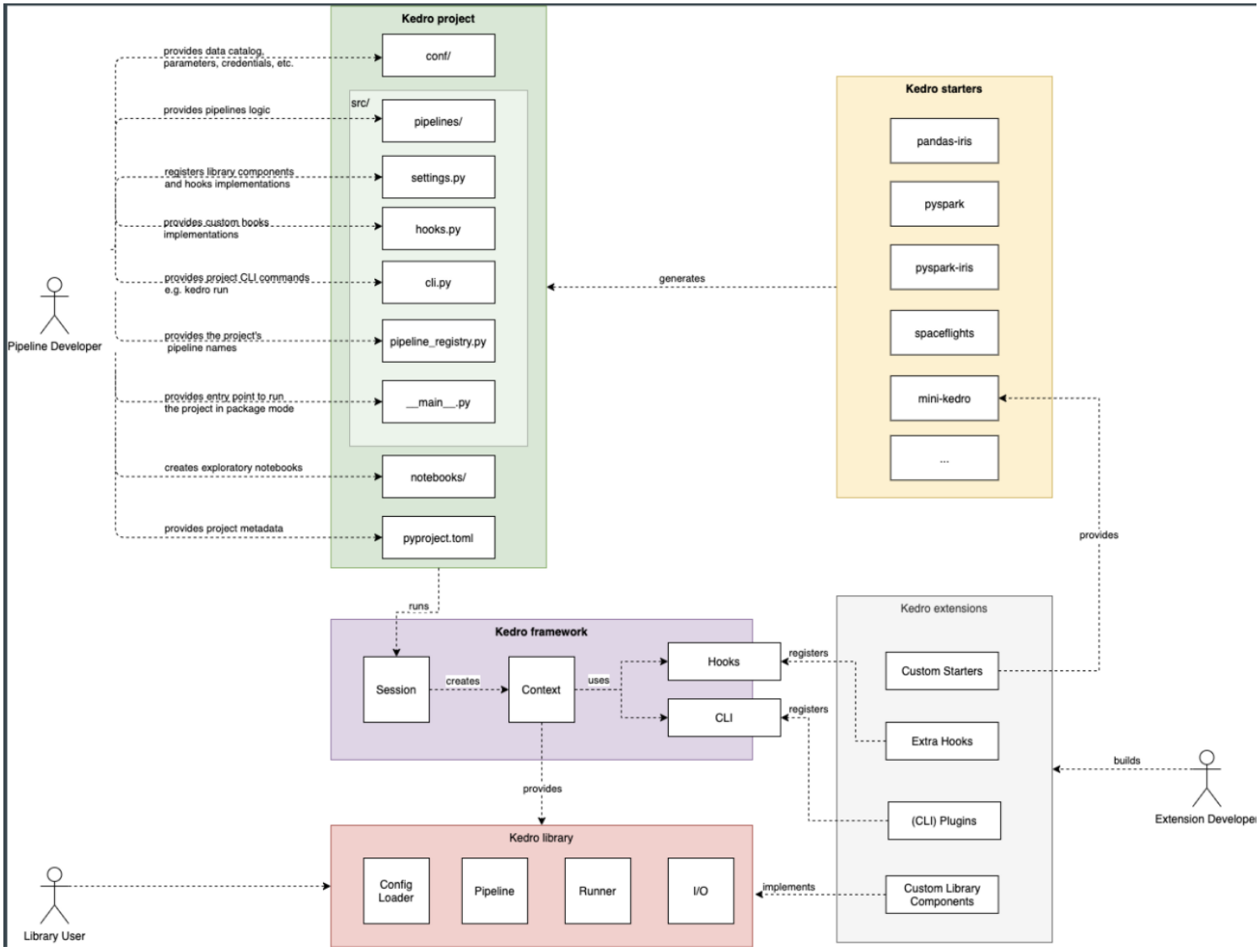


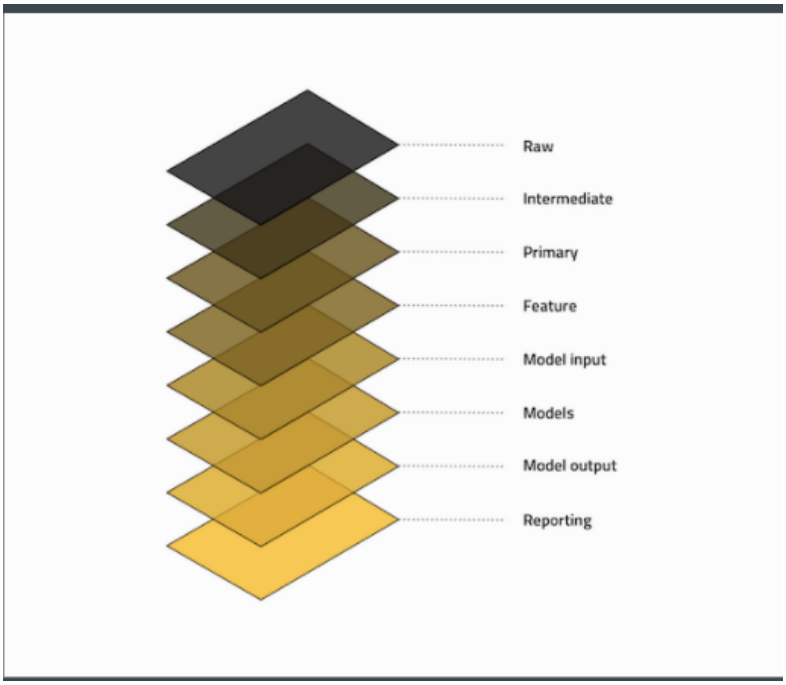
MLFlow Stack

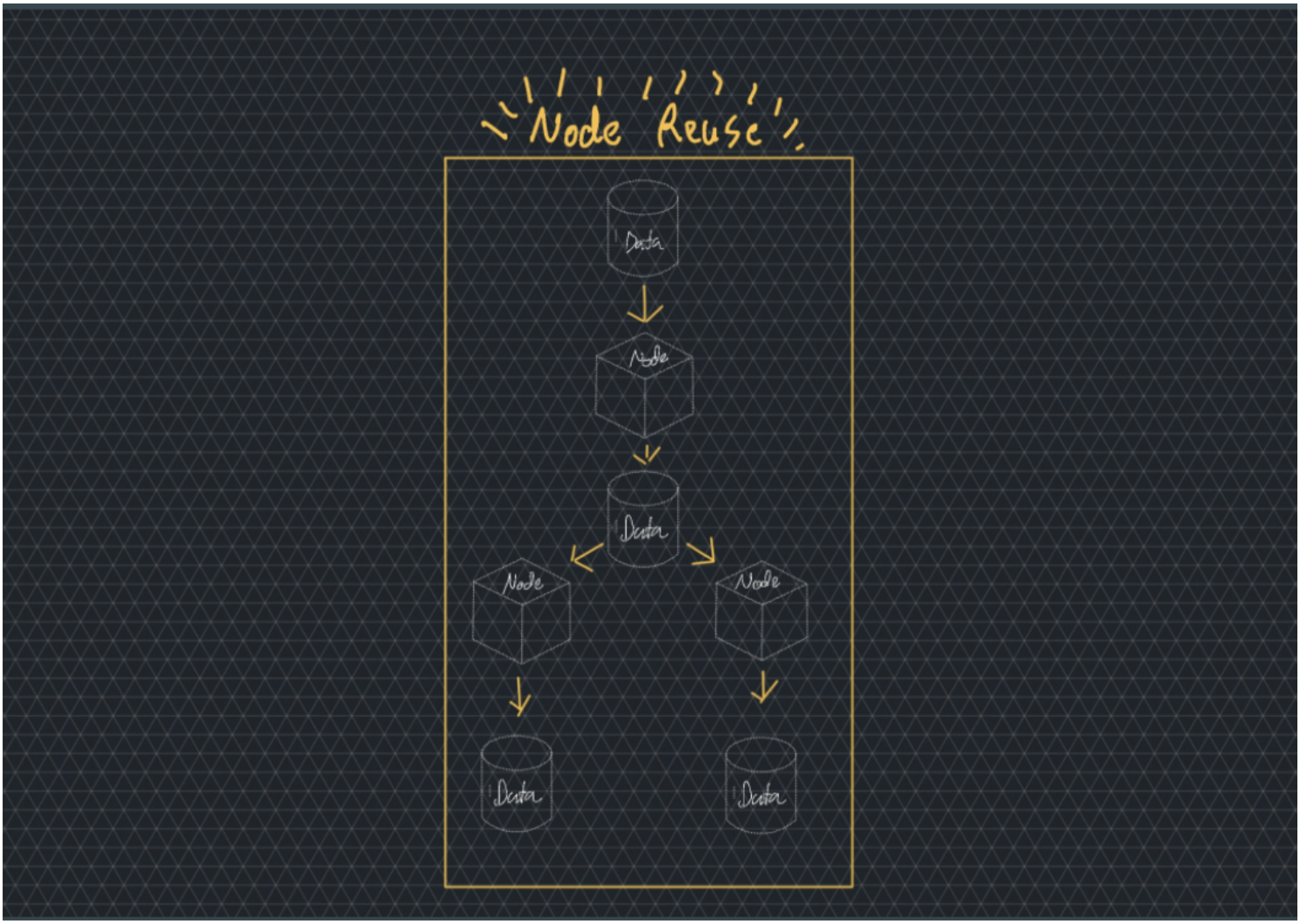


Timestamp	Logical ID	Status	Status reason
2021-09-15 16:26:15 UTC-0400	DeploymentStack	CREATE_COMPLETE	-
2021-09-15 16:26:12 UTC-0400	MLFLOWServiceTaskCountTargetAUTOSCALINGE4A5776E	CREATE_COMPLETE	-
2021-09-15 16:26:12 UTC-0400	MLFLOWServiceTaskCountTargetAUTOSCALINGE4A5776E	CREATE_IN_PROGRESS	Resource creation Initiated
2021-09-15 16:26:11 UTC-0400	MLFLOWServiceTaskCountTargetAUTOSCALINGE4A5776E	CREATE_IN_PROGRESS	-
2021-09-15 16:26:09 UTC-0400	MLFLOWServiceTaskCountTarget5798B92F	CREATE_COMPLETE	-

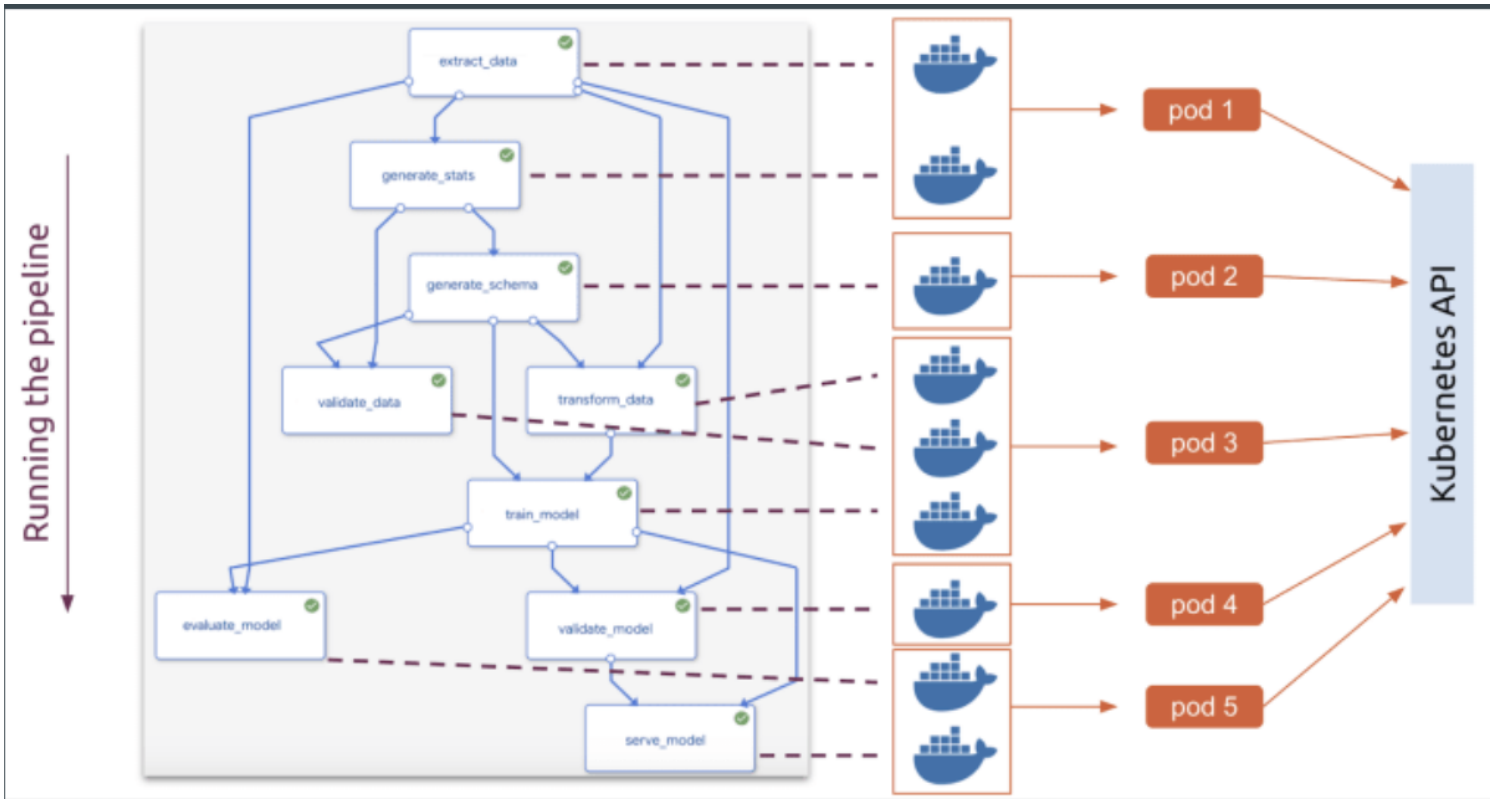
Codebase - Kedro

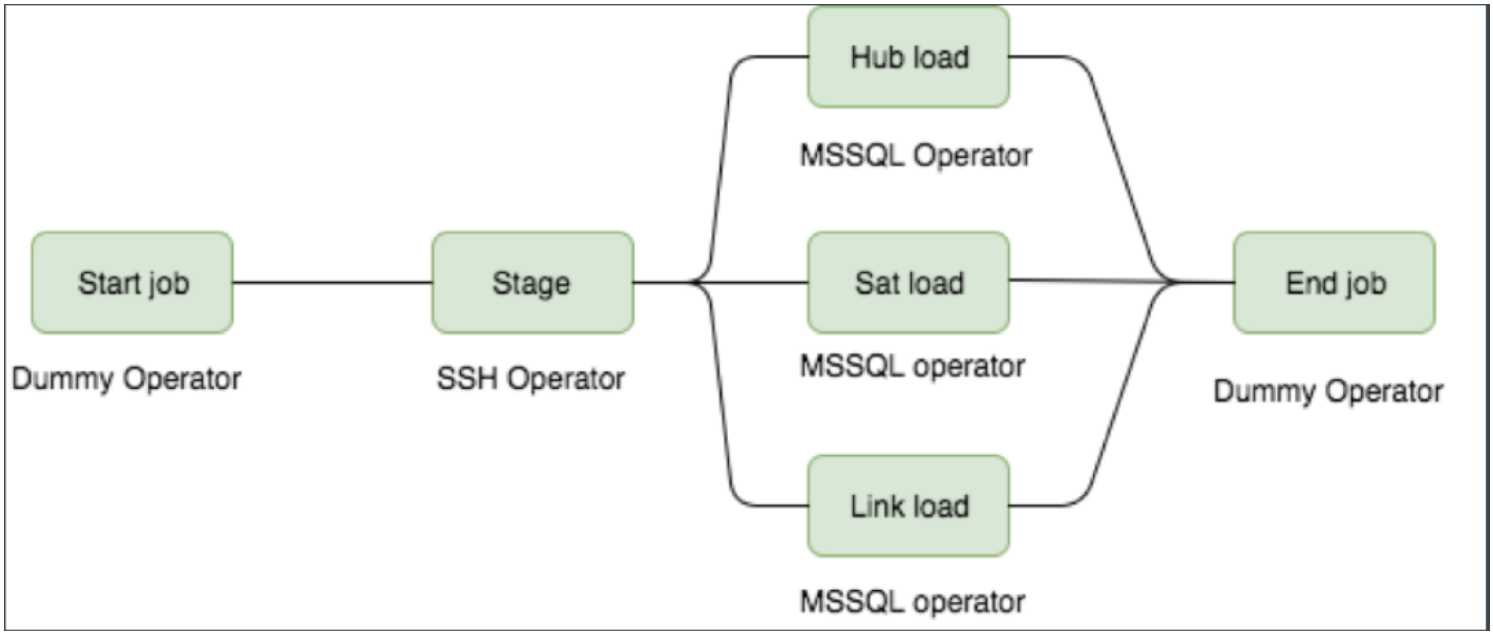



Raw
Intermediate
Primary
Feature
Model input
Models
Model output
Reporting

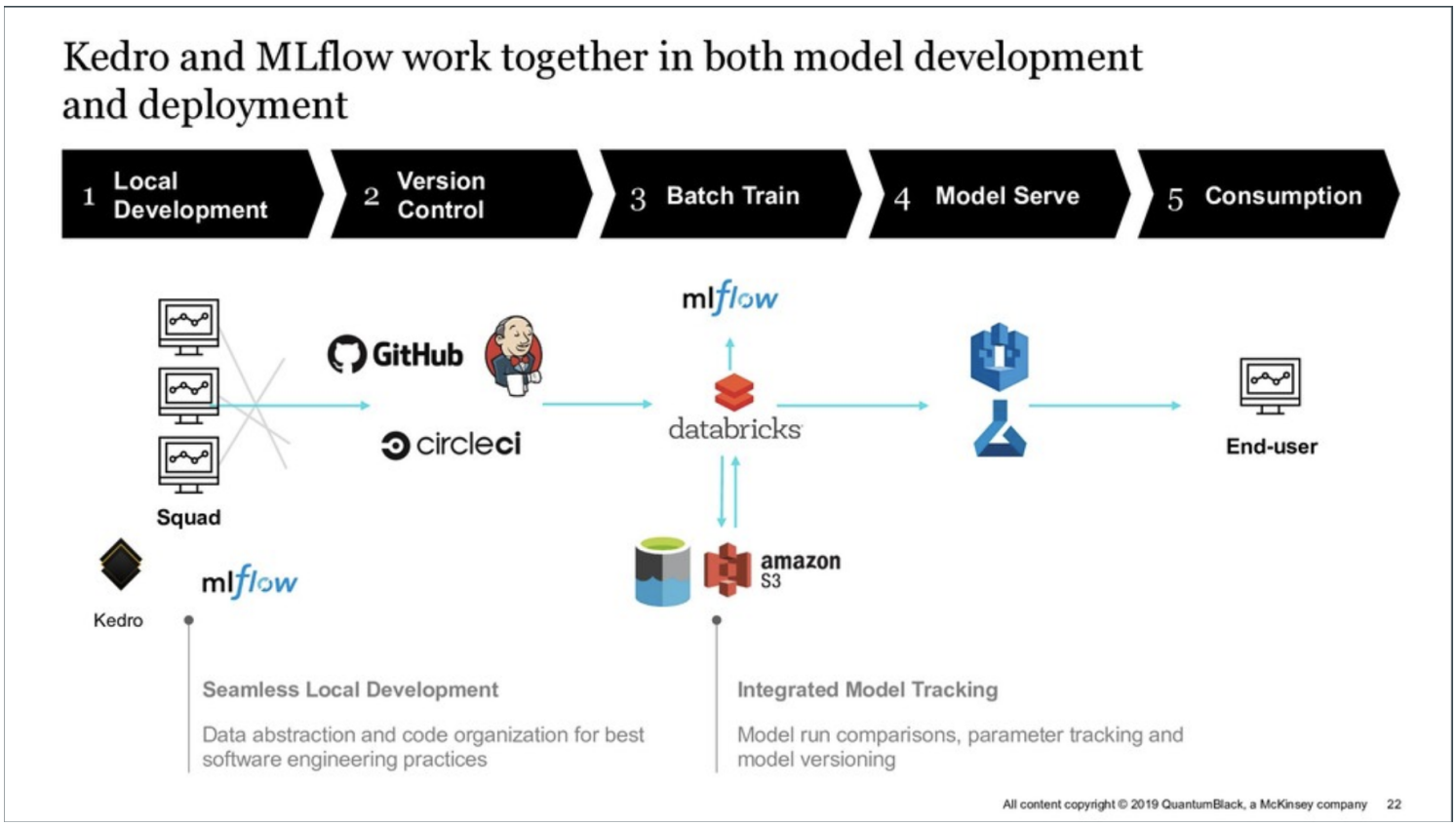

Node Reuse
Data
Node
Data
Node
Node
Data
Data

An Aside - DAGS

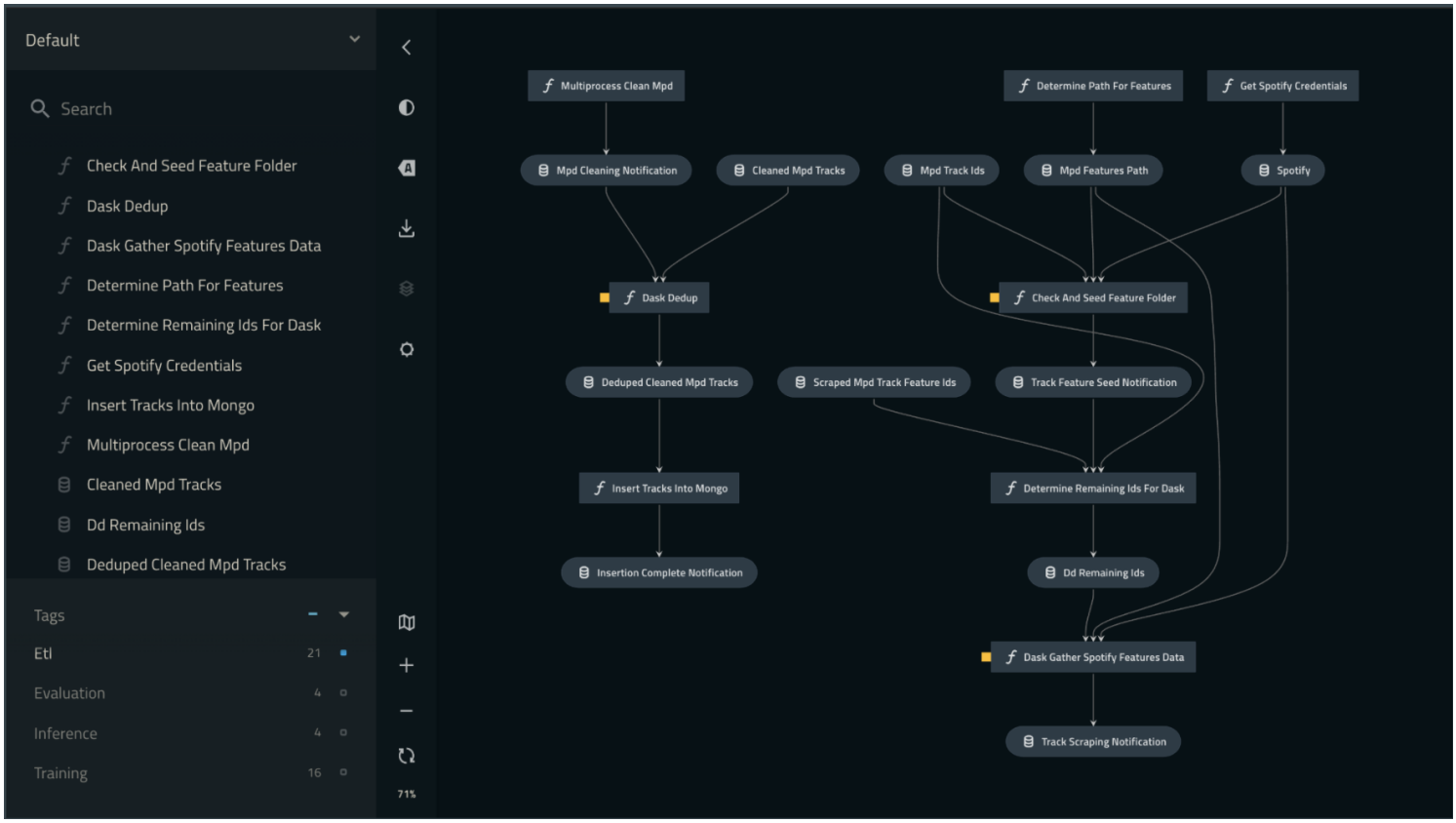


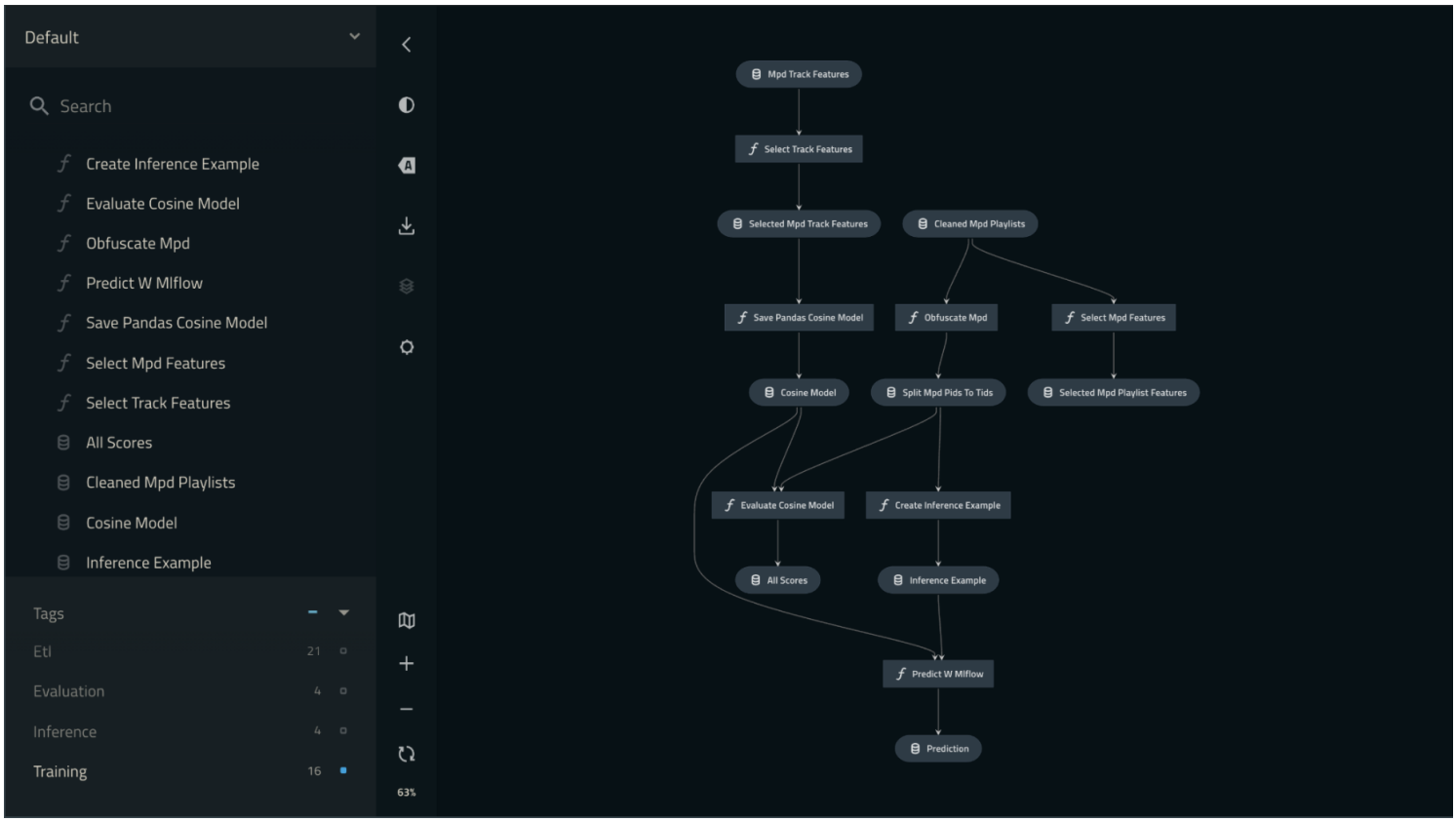


Kedro - MLFlow

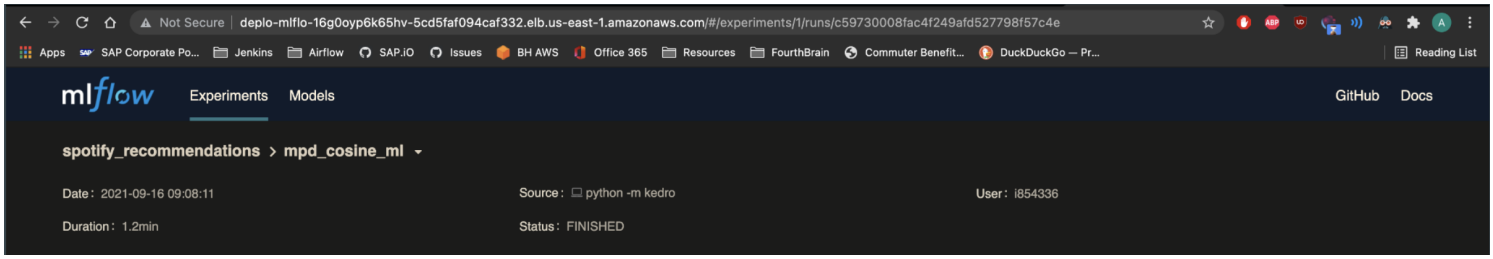


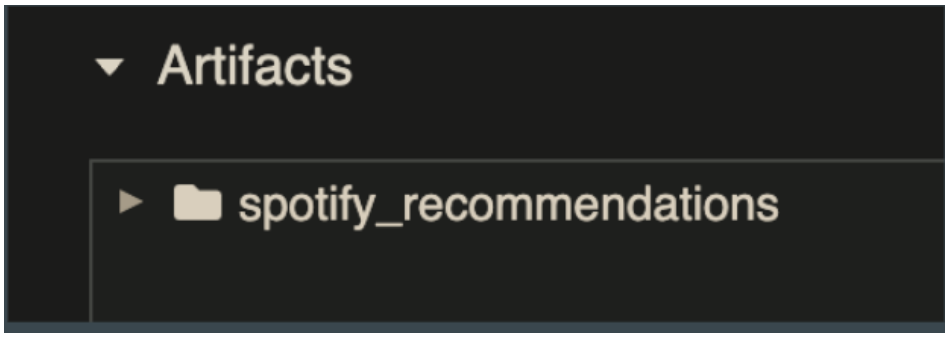
Training

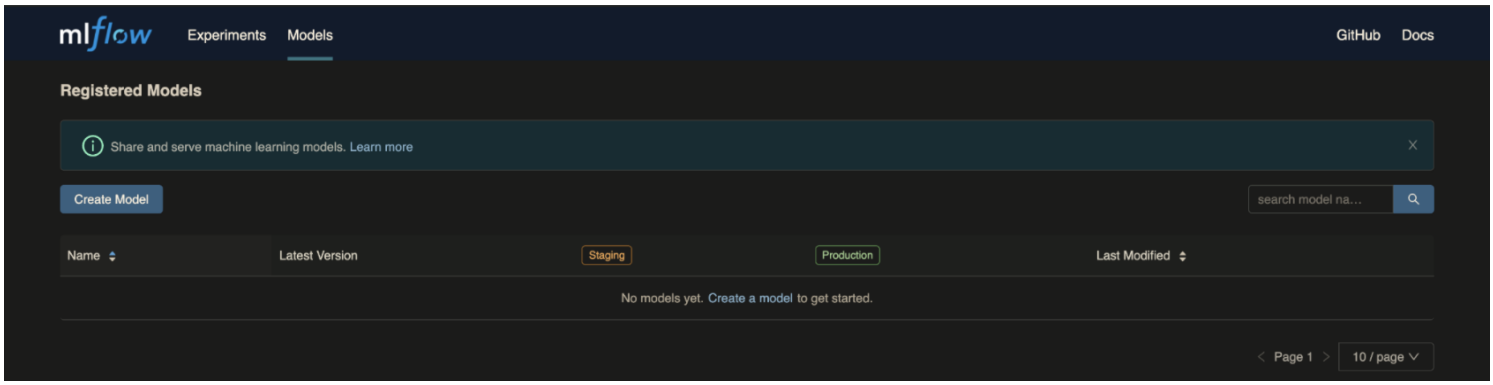

Default
Search
Check And Seed Feature Folder
Dask Dedup
Dask Gather Spotify Features Data
Determine Path For Features
Determine Remaining Ids For Dask
Get Spotify Credentials
Insert Tracks Into Mongo
Multiprocess Clean Mpd
Cleaned Mpd Tracks
Dd Remaining Ids
Deduped Cleaned Mpd Tracks
Tags
Etl 21
Evaluation 4
Inference 4
Training 16
71%
Multiprocess Clean Mpd
Determine Path For Features
Get Spotify Credentials
Mpd Cleaning Notification
Cleaned Mpd Tracks
Mpd Track Ids
Mpd Features Path
Spotify
Dask Dedup
Check And Seed Feature Folder
Deduped Cleaned Mpd Tracks
Scraped Mpd Track Feature Ids
Track Feature Seed Notification
Insert Tracks Into Mongo
Determine Remaining Ids For Dask
Insertion Complete Notification
Dd Remaining Ids
Dask Gather Spotify Features Data
Track Scraping Notification


Default
Search
Create Inference Example
Evaluate Cosine Model
Obfuscate Mpd
Predict W Mlflow
Save Pandas Cosine Model
Select Mpd Features
Select Track Features
All Scores
Cleaned Mpd Playlists
Cosine Model
Inference Example
Tags
Etl 21
Evaluation 4
Inference 4
Training 16
63%
Mpd Track Features
Select Track Features
Selected Mpd Track Features
Cleaned Mpd Playlists
Save Pandas Cosine Model
Obfuscate Mpd
Select Mpd Features
Cosine Model
Split Mpd Pids To Tids
Selected Mpd Playlist Features
Evaluate Cosine Model
Create Inference Example
All Scores
Inference Example
Predict W Mlflow
Prediction

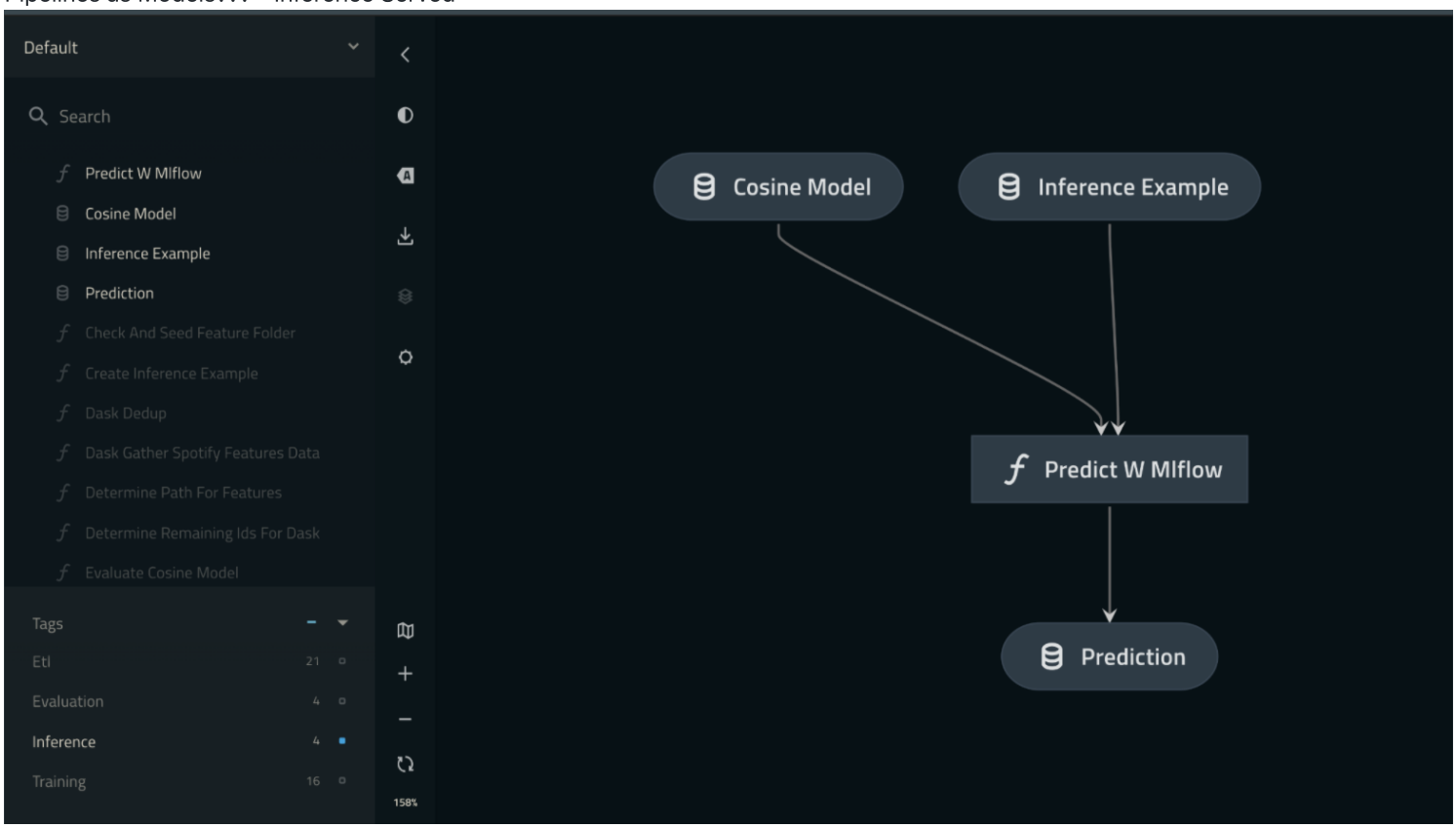
The Dynamic Duo in Action



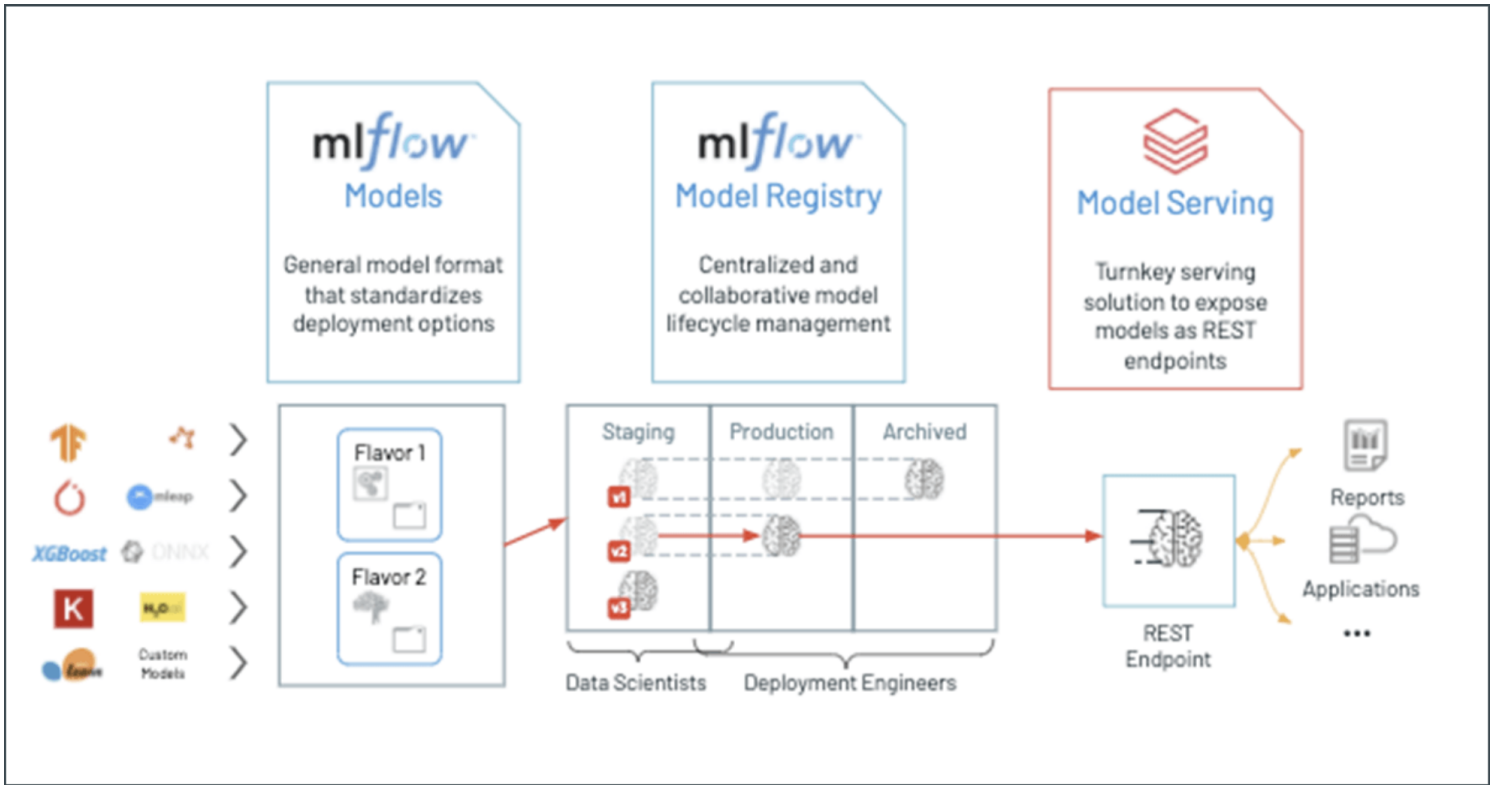




Pipelines as Models??? - Inference Served



You're sure that's the model?

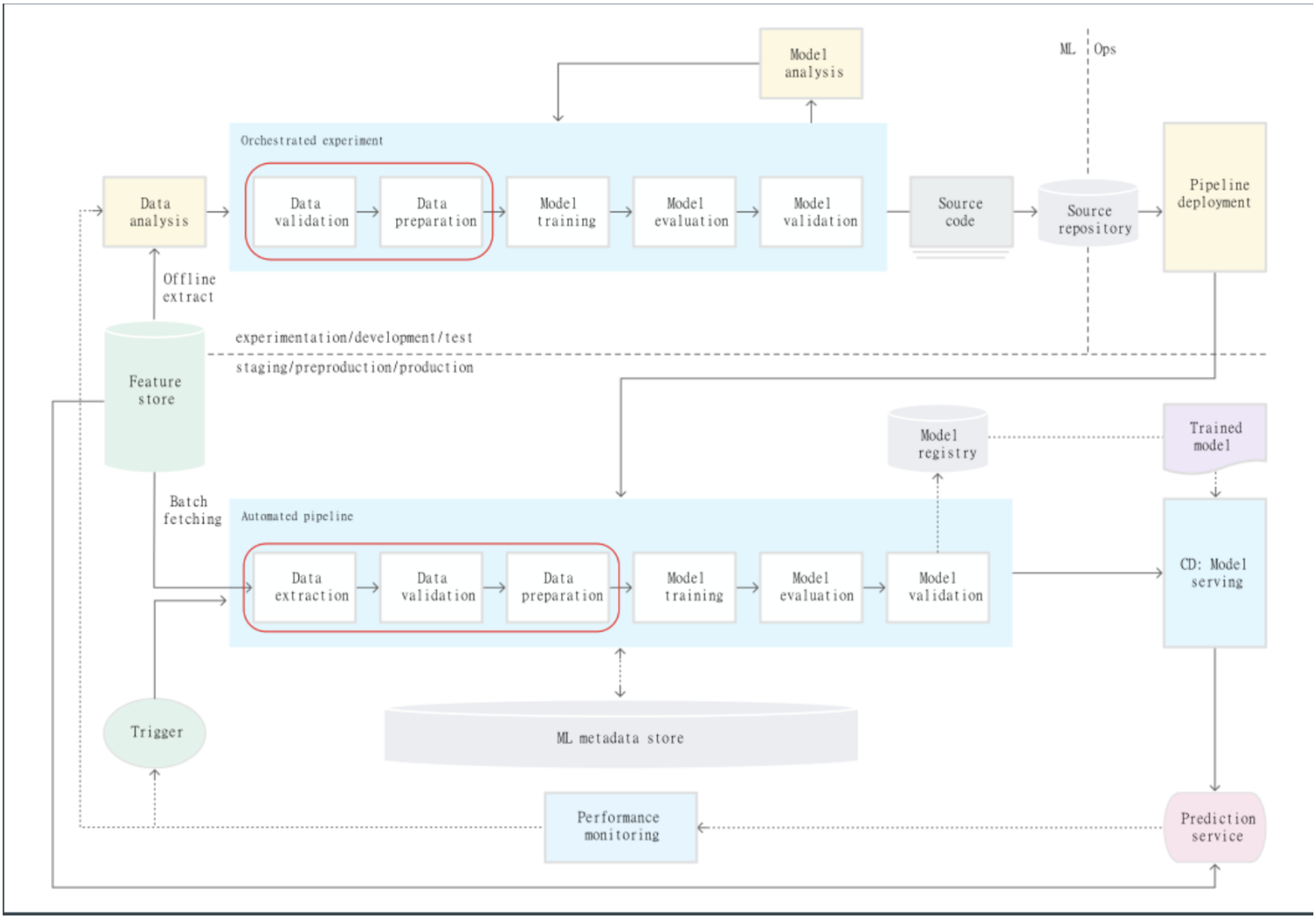


Demo App

- MLFlow: http://deplo-mlflo-16g0oyp6k65hv-5cd5faf094caf332.elb.us-east-1.amazonaws.com/#/
- Model Endpoint: https://spotify-recommendations-crun-ctreuw63uq-uw.a.run.app/invocations

Demo Video of Endpoint

Recap



#TODO:

- Add Prometheus and Grafana metrics
- Replace with an actually good model
- Connect all components of my application (Front-end and Back-end)
- Use MLFlow Model Registry to add streamlined Blue/Green Deployments and Canary Deployments
- Use CDK to spin up the CloudFormation template to spin up the aforementioned CodePipeline Stack
- Fix codebase to package minimal requirements for a better served model container
- Port from custom containers to something like a Seldon deployment if possible

Development

Kedro Overview

This is your new Kedro project, which was generated using `Kedro 0.17.4`.

Take a look at the Kedro documentation to get started.

Rules and guidelines

In order to get the best out of the template:

- Don't remove any lines from the `.gitignore` file we provide

- Make sure your results can be reproduced by following a data engineering convention
- Don't commit data to your repository
- Don't commit any credentials or your local configuration to your repository. Keep all your credentials and local configuration in `conf/local/`

How to install dependencies

Declare any dependencies in `pyproject.toml` as per usage with `poetry`. You can also use the `src/requirements.txt` is you use the `kedro install` and `build-reqs` flow.

How to run your Kedro pipeline

You can run your Kedro project with:

```
kedro run
```

How to test your Kedro project

Have a look at the file `src/tests/test_run.py` for instructions on how to write your tests. You can run your tests as follows:

```
kedro test
```

To configure the coverage threshold, go to the `.coveragerc` file.

Project dependencies

For the `poetry` workflow, to generate/update dependencies, use the `poetry add` functionality alongside adjusting the `pyproject.toml` to take advantages of `poetry`'s automatic dependancy resolution

If you are using the `kedro install` workflow, to generate or update the dependency requirements for your project: `kedro build-reqs`

This will copy the contents of `src/requirements.txt` into a new file `src/requirements.in` which will be used as the source for `pip-compile`. You can see the output of the resolution by opening `src/requirements.txt`.

After this, if you'd like to update your project requirements, please update `src/requirements.in` and re-run `kedro build-reqs`.

Further information about project dependencies

How to work with Kedro and notebooks

> Note: Using `kedro jupyter` or `kedro ipython` to run your notebook provides these variables in scope: `context`, `catalog`, and `startup_error`.
>
> Jupyter, JupyterLab, and IPython are already included in the project requirements by default, so once you have run `kedro install` you will not need to take any extra steps before you use them.

Jupyter

To use Jupyter notebooks in your Kedro project, you need to install Jupyter:

```
pip install jupyter
```

After installing Jupyter, you can start a local notebook server:

```
kedro jupyter notebook
```

JupyterLab

To use JupyterLab, you need to install it:

```
pip install jupyterlab
```

You can also start JupyterLab:

```
kedro jupyter lab
```

IPython

And if you want to run an IPython session:

```
kedro ipython
```

How to convert notebook cells to nodes in a Kedro project

You can move notebook code over into a Kedro project structure using a mixture of cell tagging and Kedro CLI commands.

By adding the `node` tag to a cell and running the command below, the cell's source code will be copied over to a Python file within `src/<package_name>/nodes/`:

```
kedro jupyter convert <filepath_to_my_notebook>
```

> *Note:* The name of the Python file matches the name of the original notebook.

Alternatively, you may want to transform all your notebooks in one go. Run the following command to convert all notebook files found in the project root directory and under any of its sub-folders:

```
kedro jupyter convert --all
```

How to ignore notebook output cells in `git`

To automatically strip out all output cell contents before committing to `git`, you can run `kedro activate-nbstripout`. This will add a hook in `.git/config` which will run `nbstripout` before anything is committed to `git`.

> *Note:* Your output cells will be retained locally.

Package your Kedro project

Further information about building project documentation and packaging your project