aws startups &

Datahack

Datahack Meetup

Dataset Versioning and Feature Stores for Machine Learning

Julian Sprung

Julian Sprung

ML Solutions Architect, Startups

Amazon Web Services, Berlin

Working with European, Middle Eastern and African startups building their machine learning solutions on AWS.

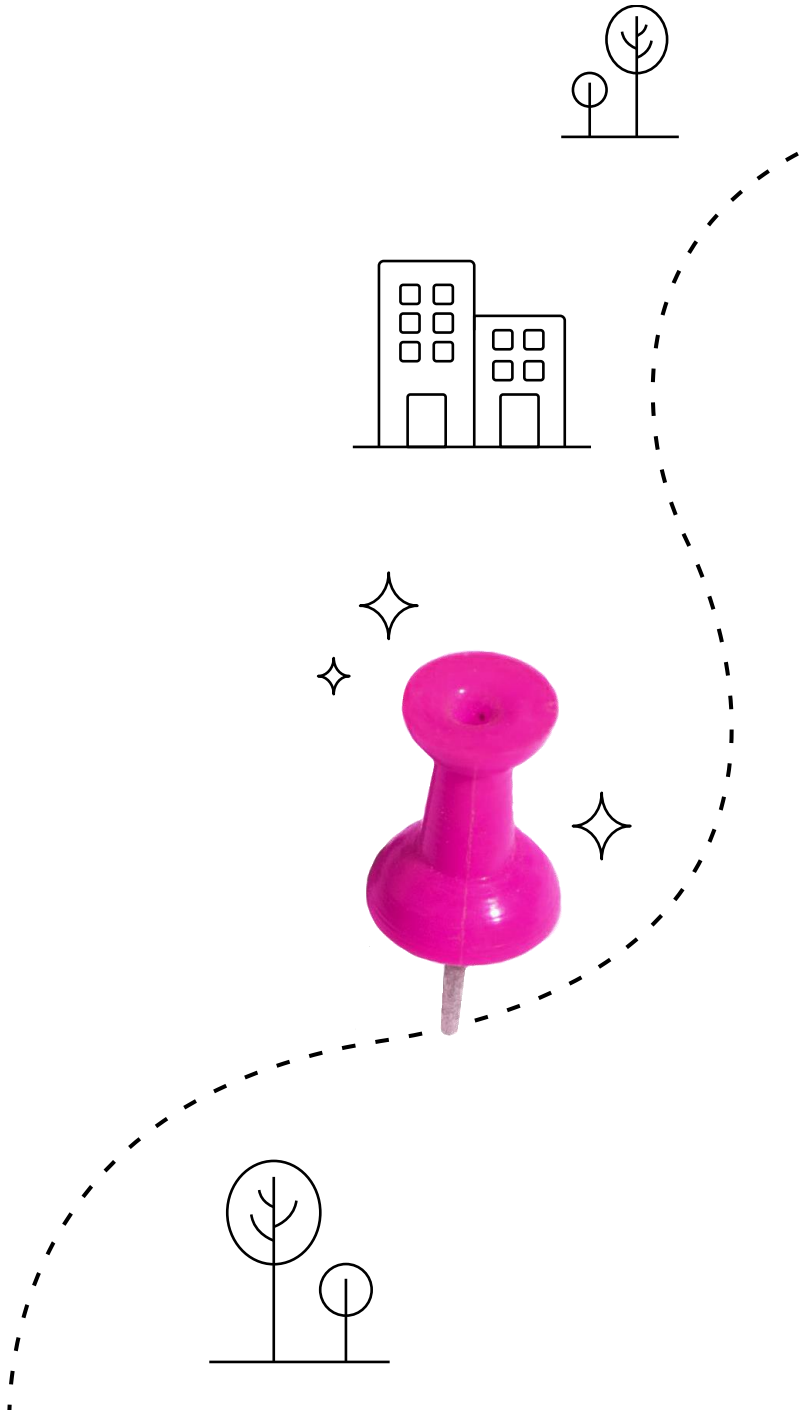
Former co-founder and CTO at Genesis Cloud – GPU cloud infrastructure for ML and rendering.

Find me on LinkedIn:
https://www.linkedin.com/in/julian-sprung-1b18a689/

aws startups

Agenda

1. Motivation
2. Data Versioning Approaches
    1. Git (LFS)
    2. Object Storage (Amazon S3)
    3. Data Version Control (DVC)
3. Feature Stores
4. SageMaker Feature Store Demo

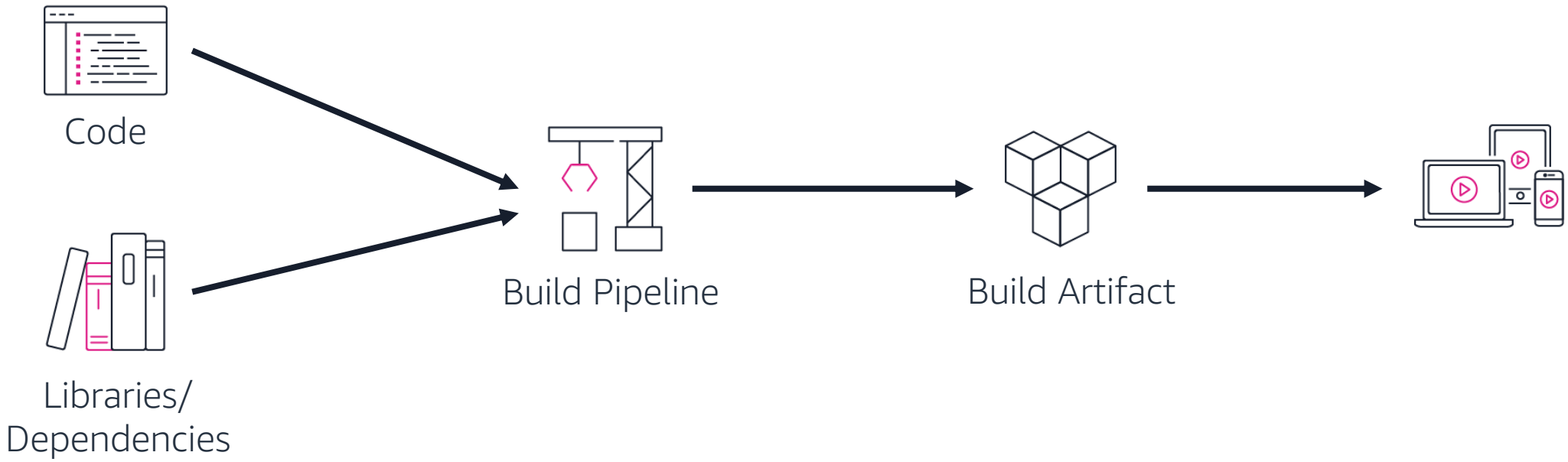


aws startups

Motivation

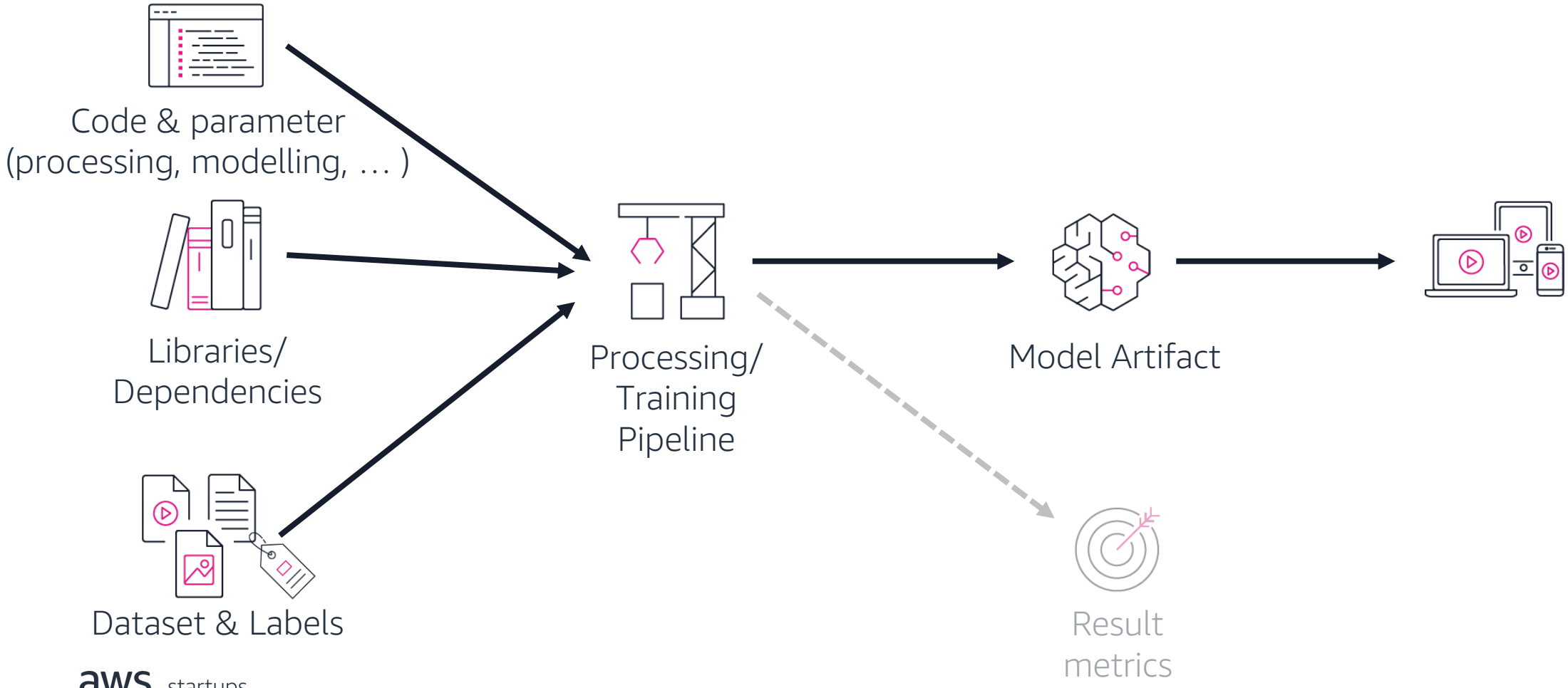
aws startups

Excursion: SW Development – Build Pipelines



aws startups

Reproducible Pipelines: What about Data Science?



aws startups

Why bother about it?

- Best practice data quality management
    - Garbage in, garbage out
- Reproducible training and retraining
- Reuse and share preprocessed datasets
- Impossible to debug data errors
- Testing: i.e. integration tests for your full ML pipeline
- Explainability (debugging predictions)
- Auditing, regulatory and ethical AI requirements
- …

85% AI projects deliver erroneous outcomes due to bias in Data, algorithms or teams managing it.

92% Of AI practitioners reported **Data cascading** effects are prevalent in their AI/ML projects.

Sources:: https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence and https://storage.googleapis.com/pub-tools-public-publication-data/pdf/0d556e45afc54afeb2eb6b51a9bc1827b9961ff4.pdf

What is the cost to your business of making wrong or suboptimal predictions?
(Caused by training on the wrong data)

aws startups

Data Versioning Approaches

aws startups


git
&
Git Large File Storage

Git LFS : Why not use Git?

- Git
  - Great for text files, i.e. code, that Git can handle as DAG
  - Bad for binary files, images, videos etc.
    - Especially for larger files or changing files
      (GitHub file size limit 100MB)
  - Huge repos, slow cloning, branch switching etc…

Git LFS : Why not use Git?

- Git
  - Great for text files, i.e. code, that Git can handle as DAG
  - Bad for binary files, images, videos etc.
    - Especially for larger files or changing files (GitHub file size limit 100MB)
  - Huge repos, slow cloning, branch switching etc…

- Git LFS (large file storage)
  - An open source git extension for versioning large files
  - Little change to git workflows
  - Small repos and faster cloning and fetching
  - Data storage on remote git LFS server
  - File size limitations (2GB GitHub free, 5GB GitHub Enterprise)

Git Large File Storage

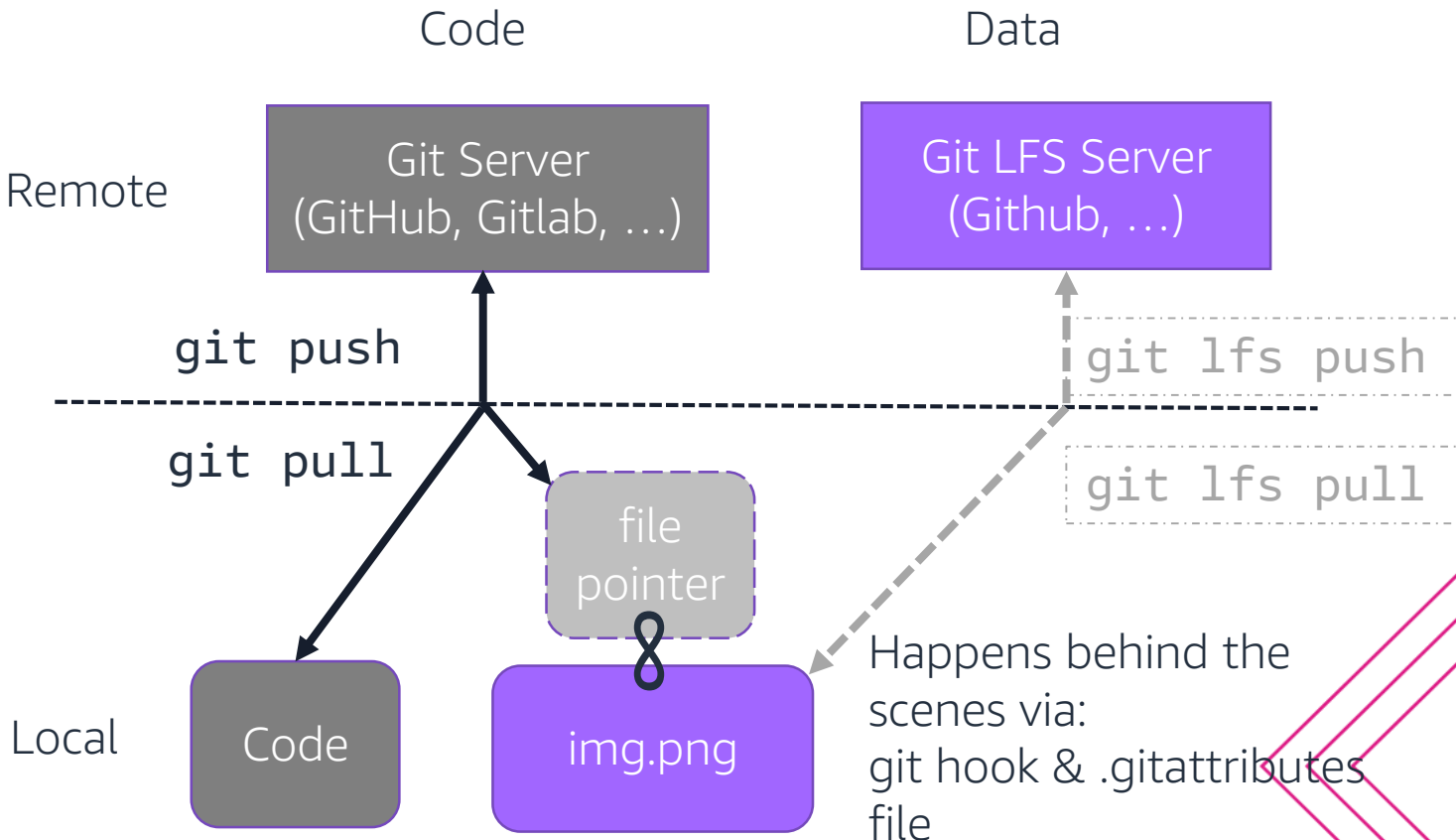
Git LFS: Quick Overview

```
$ git lfs install

$ git lfs track “*.png”

$ git add path-to-file/img.png

$ git commit -m “add image”
$ git push
…
$ git pull
```



aws startups

Git LFS: Interacting with the files

- Git (lfs) pull
- Just like local files, e.g. git updates pointer to actual files

Git LFS: A few Downsides

- Requires Git LFS Server (GitHub, Gitlab, Atlassian, self-hosted)
- Max file size implementations ~2GB
- No direct storage integrations (Amazon S3, Snowflake, Azure, GCP etc)
- Data transfer…
- Not great for sharing datasets across projects

Cloud Object Storage

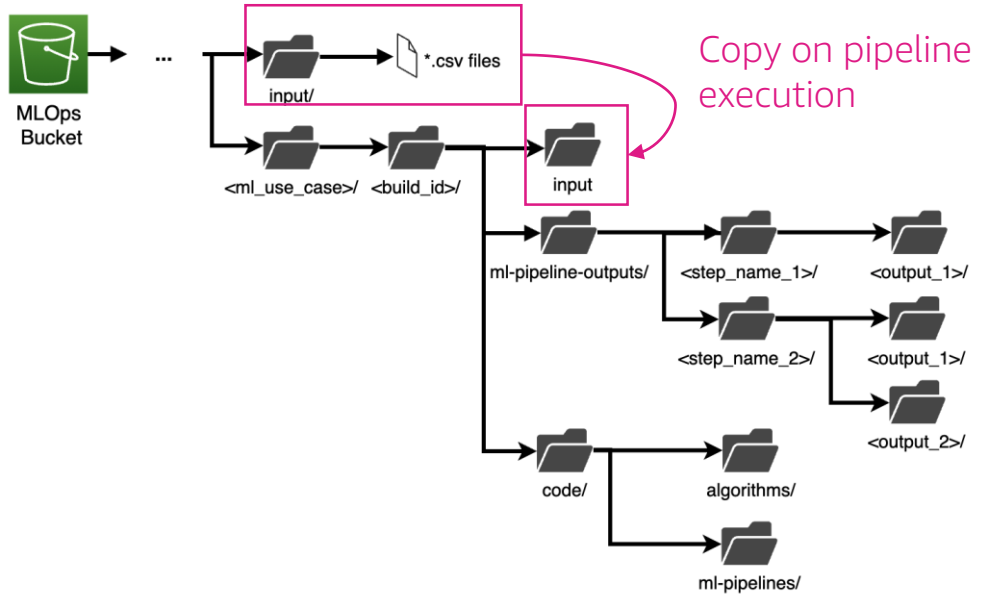
Amazon S3

Object storage (S3) with immutable files

Option 1) Manifest files

```
{"source-ref": "S3 bucket location 1"}
{"source-ref": "S3 bucket location 2"}
                  ...
{"source-ref": "S3 bucket location n"}
```

Option 2) Copy datasets



- Caveats:
  - Relies on data immutability
  - Manifest files require code or a system that will load the listed files
  - Copying creates data duplication, i.e. not suited for large/changing data sets

aws startups

Object storage (S3): Interacting with your files

- Download before training/processing
  - Fully replicated
  - Sharded/split
- Options to pipe on-the-fly
- High-performance/Caching options FSx for Lustre

- See also AWS blog: Choose the best data source for your Amazon SageMaker training job

aws startups

DVC: Data Version Control

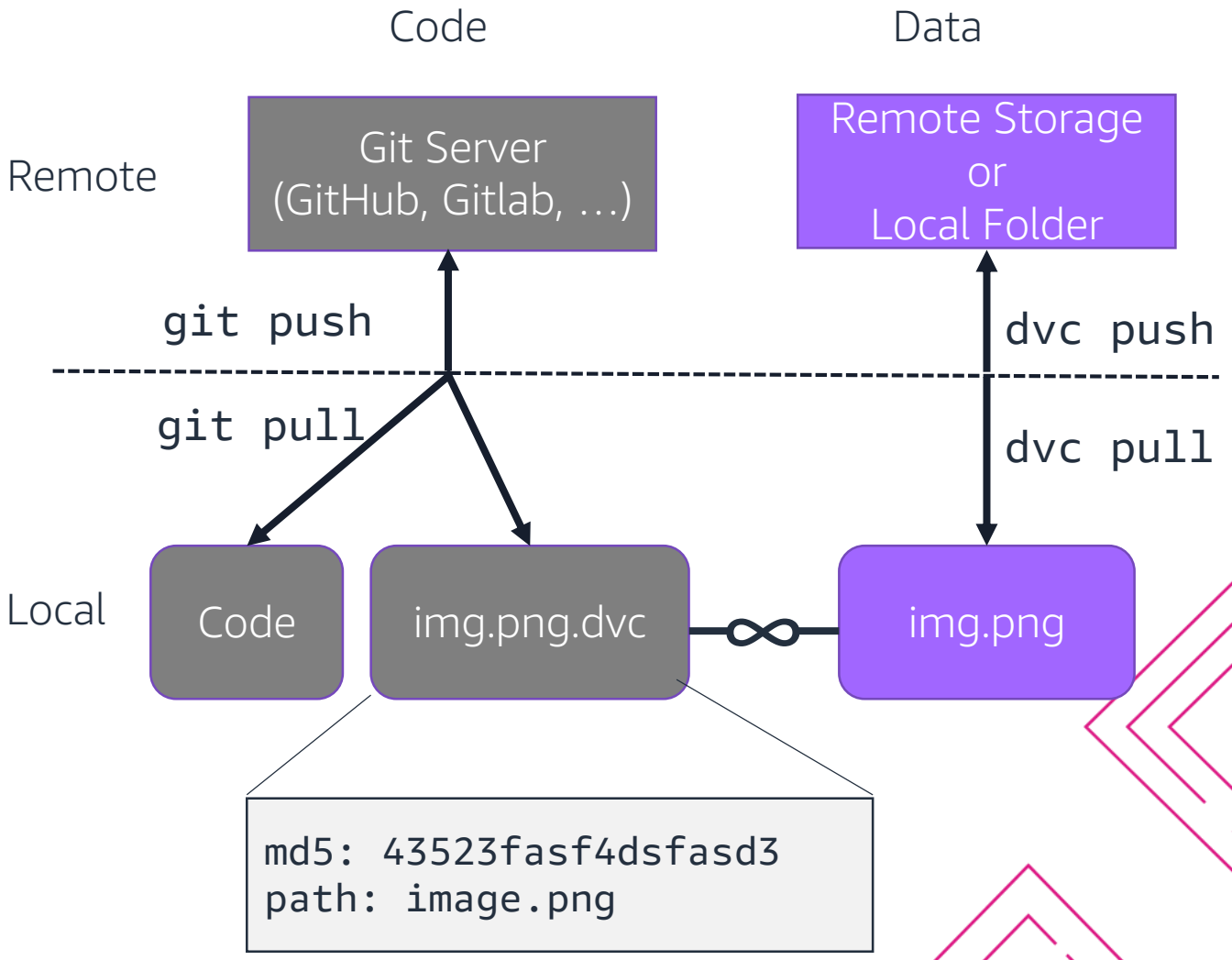
- “Open-source Version Control System for Machine Learning Projects”
- Closely adopting git style syntax
- Track Datasets (but also ML Workflows and Projects)
- Many storage backend options (Amazon S3, Azure, GCP, SSH, local)

- Versions control features beyond dataset tracking:
  - Model artifacts
  - Data pipelines
  - Metrics and Parameters

DVC Quick Overview

```
$ dvc init
$ dvc remote add -d myrepo s3://mybucket

$ dvc add image.png
$ git commit -m “data: add image”
$ dvc push
…
$ git checkout <>
$ dvc pull
```



aws startups

DVC: Interacting with your files

- Within your Git/DVC project:
  - `dvc pull` + local processing

- Outside the original project
  - `dvc get` (download only)
  - `dvc import` (download + tracking)
  - Python API: `dvc.api.open` or `dvc.api.read`

```python
import dvc.api

with dvc.api.open(
    "get-started/data.xml",
    repo="https://github.com/iterative/dataset-registry")
    as fd:
        # fd.read()
```

aws startups

DVC Demo

```
$ git init
$ dvc init
$ git status
$ git commit -m ‘initial’
### for local: $ dvc remote add -d dvc-remote /tmp/dvc-store
$ dvc remote -d s3-dataset s3://dvc-demo-datahack/demodata/
$ git commit ./dvc/config -m “configure dvc”
<Copy a file >
$ dvc add file/folder
$ git commit .gitignore data.dvc
$ git tag -a ‘v1’ -m “raw data set”
$ dvc push
…
```

aws startups

...and there are more (selected)

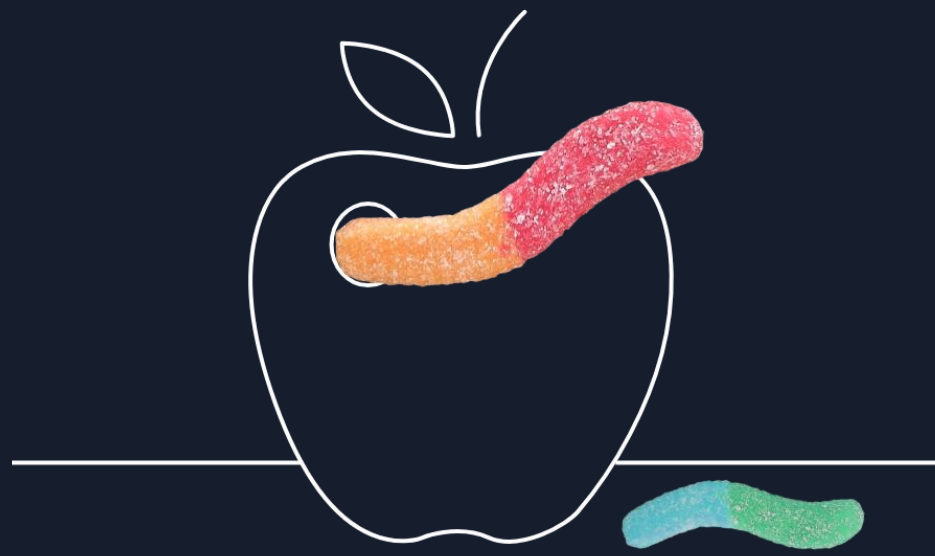
Pachyderm automates and scales the machine learning lifecycle while guaranteeing reproducibility including: Data Driven Automation, Petabyte Scalability, End-to-End Reproducibility

The lakeFS open source project for data lakes allows data versioning, rollback, debugging, testing in isolation, and more – all in one.

Delta Lake is an open-source storage framework that enables building a Lakehouse architecture with compute engines including Spark, PrestoDB, Flink, Trino, and Hive and APIs for Scala, Java, Rust, Ruby, and Python..

aws startups

Feature Stores



aws startups

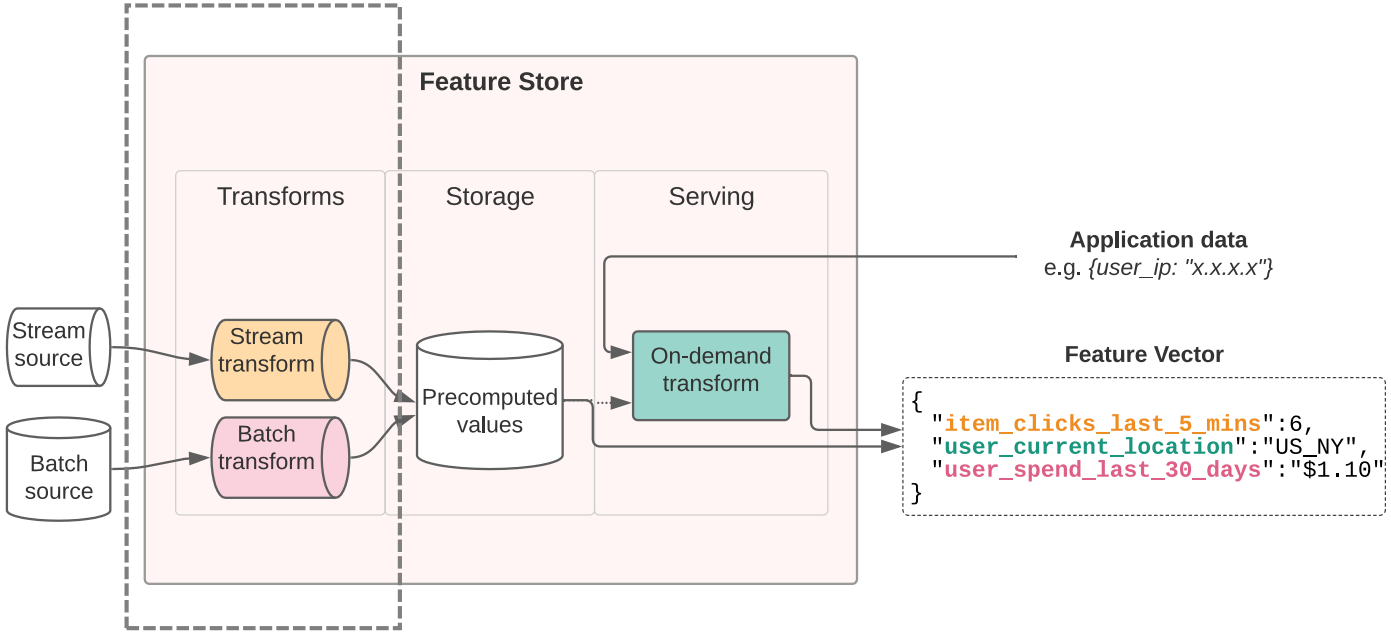
What is an Feature Store?

A central repository to ingest, store and serve features for machine learning

A "**feature**" refers to the entire column in the dataset

Transaction_id	in_foreign_country	size_compared_to_avg_transaction	fraud?
7485	False	0.8x	False
46854	True	21.2x	True
3521	True	1.1x	False

A "**feature value**" refers to a single value of a feature column



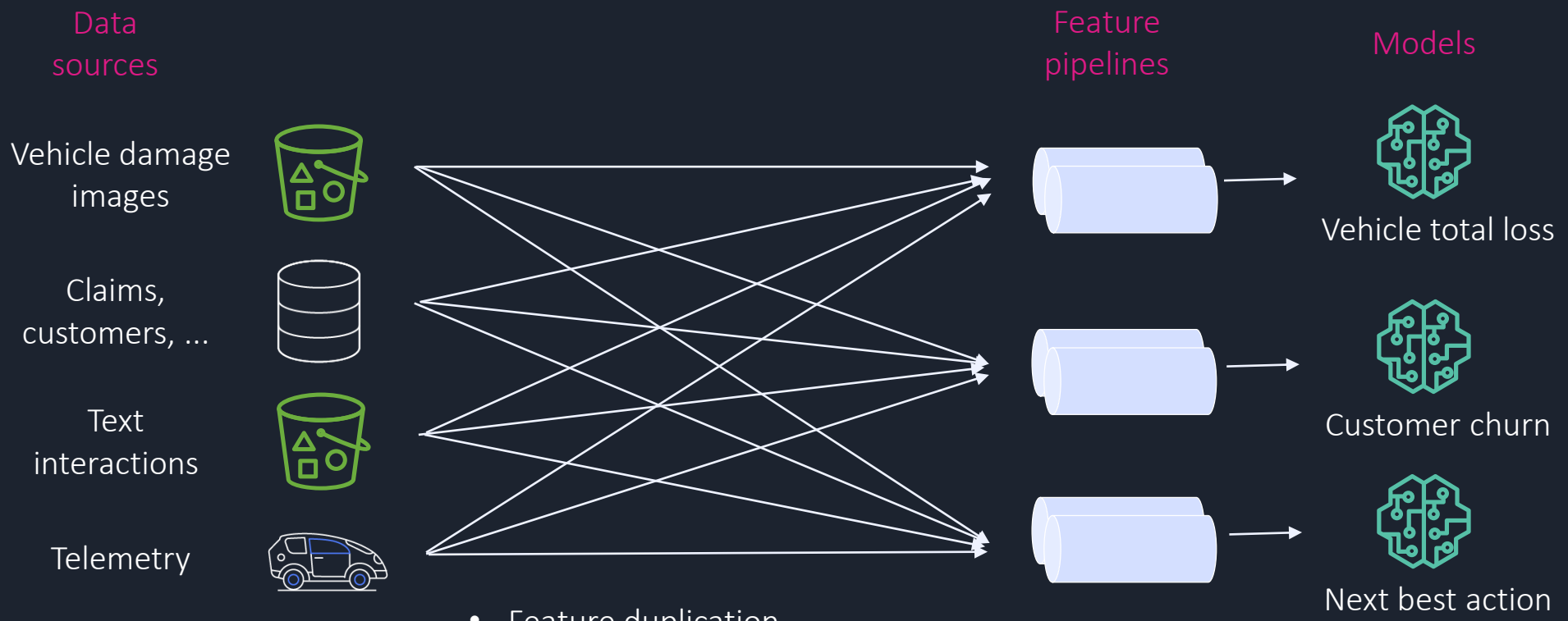
Transforms are not always part of the feature store itself

source: https://feast.dev/blog/what-is-a-feature-store/

aws startups

What is a feature store trying to solve?

Standalone feature engineering for each new model

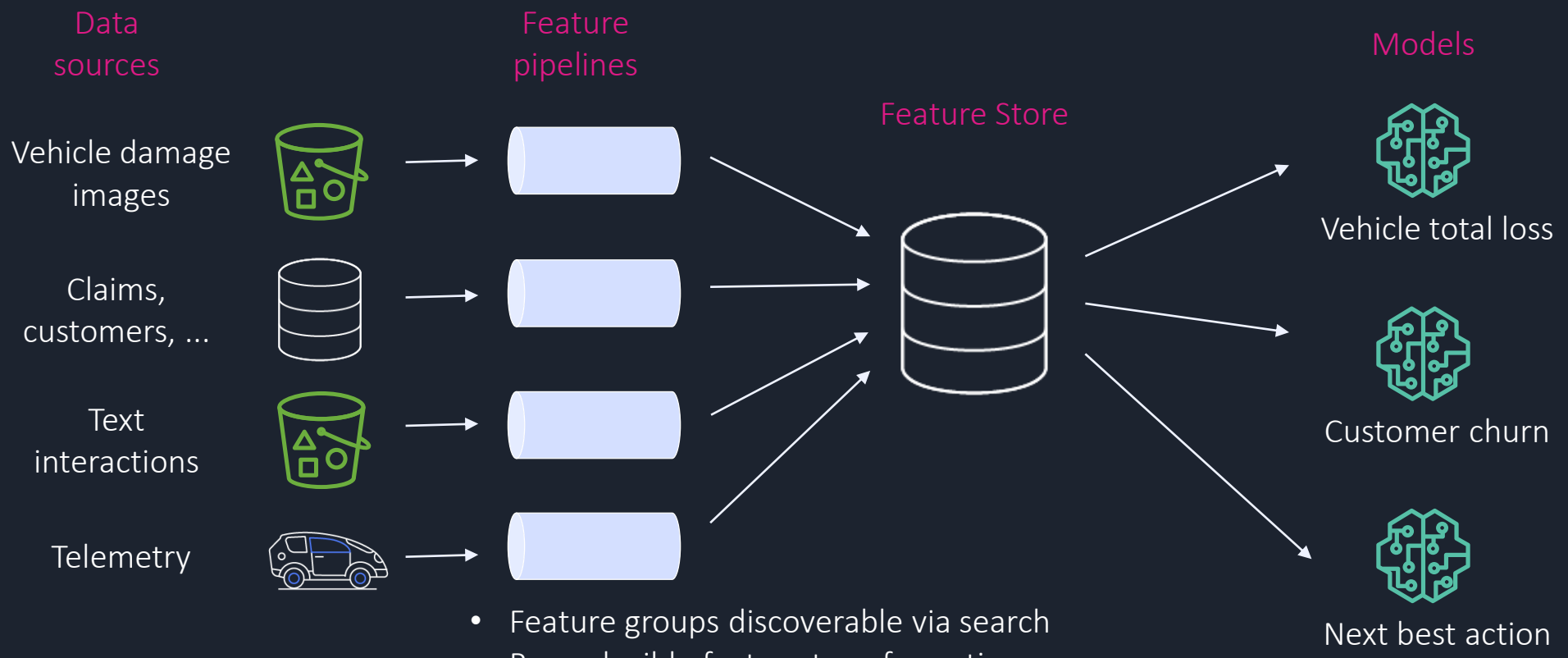


- Feature duplication
- Slow time to market
- Inaccurate predictions

aws startups

With A Feature Store...

Build features once, reuse them across teams and models



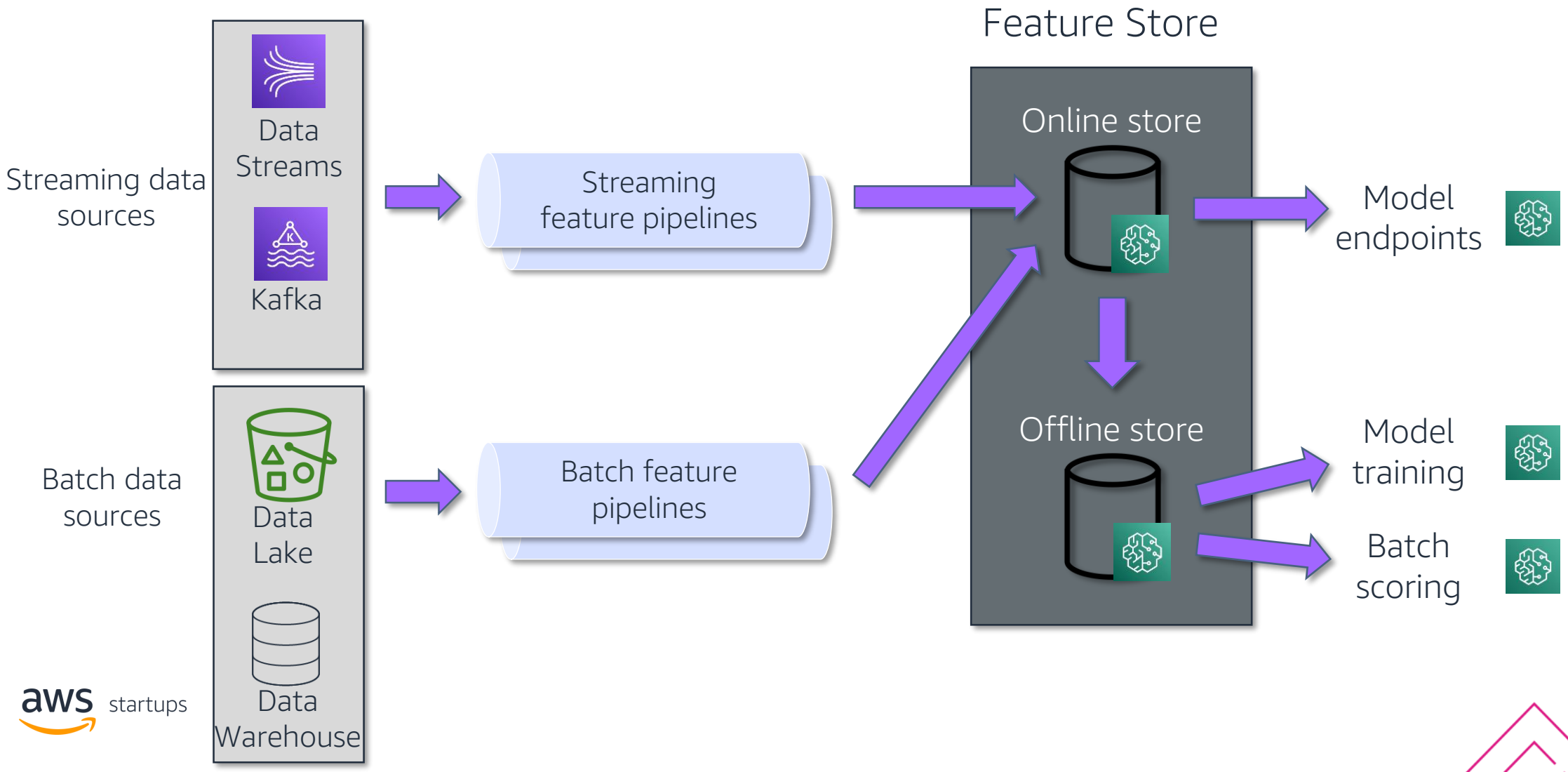
- Feature groups discoverable via search
- Reproducible feature transformations
- Extract accurate training datasets
- Low latency lookups for inference
- Consistent features for training and inference

aws startups

Why Feature Stores?

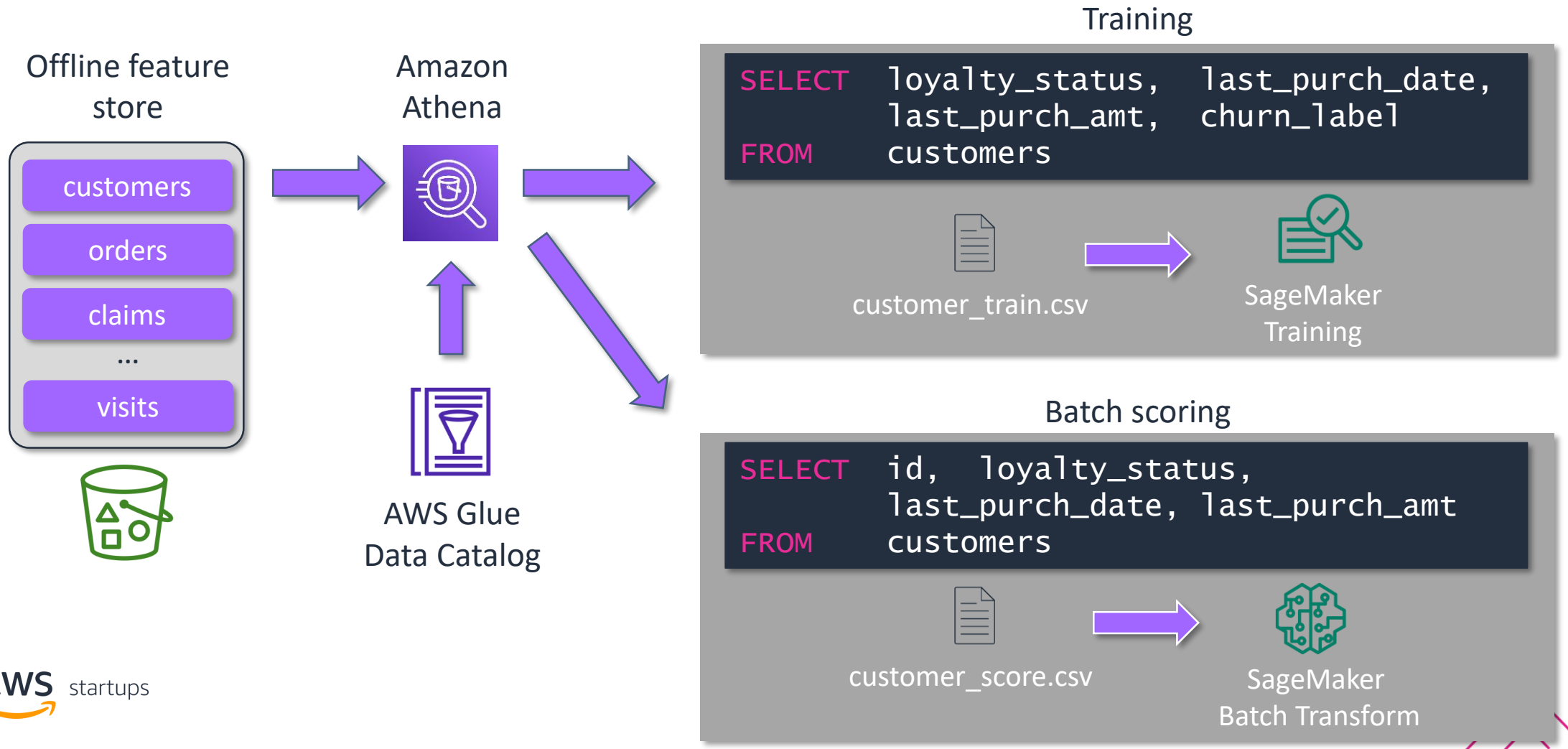
- Feature Engineering
    - Standardize your features across teams
    - Enable feature sharing, discovery, and reuse
    - Security and governance of features
- Training:
    - Dataset versioning: point-in-time dataset retrieval via query (no data leakage)
    - Reproducibility for training datasets
    - Combining multiple features into training data
- Serving:
    - System for low latency serving of latest features matching an ID
    - Data or feature consistency across training and serving

Feature Store in Context



aws startups

Reproducible Training Datasets: Query the Feature Store



Query features interactively, or with Python SDK

Amazon SageMaker

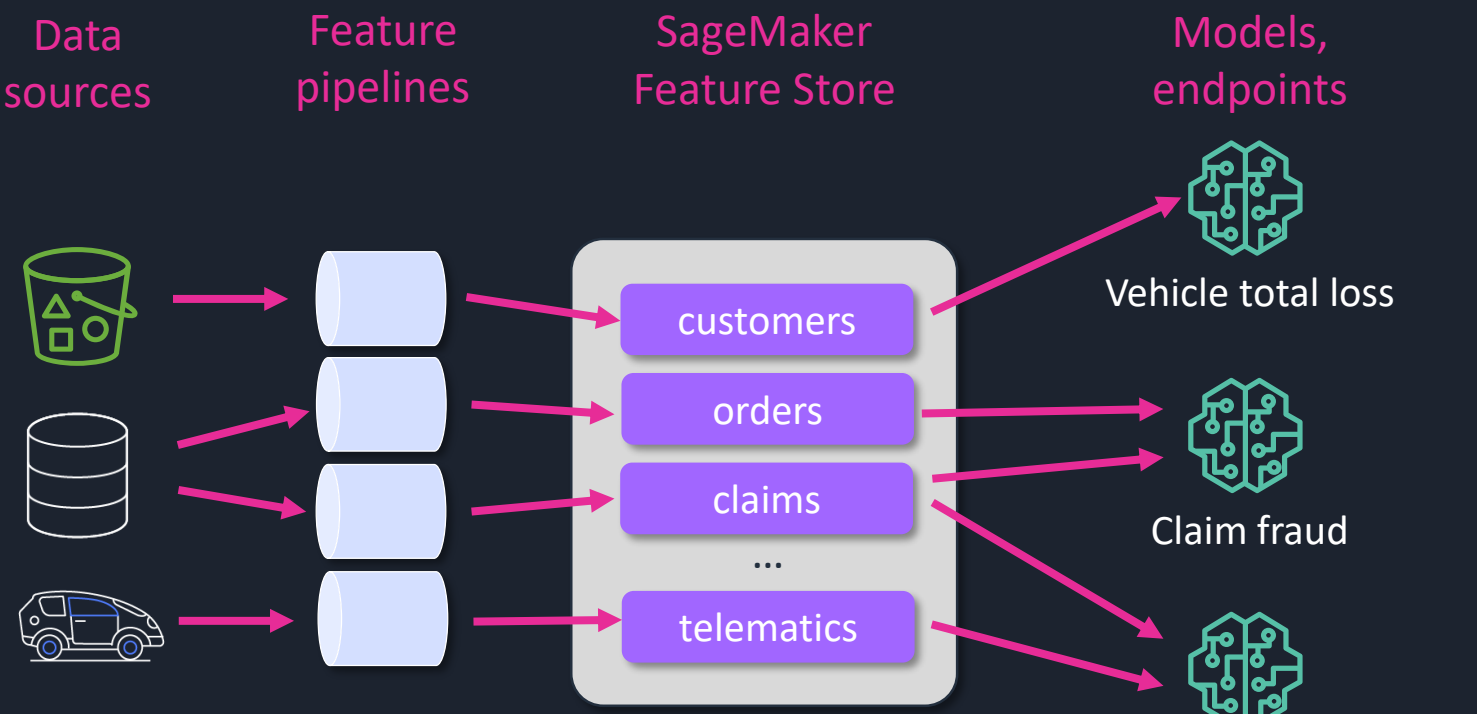
Athena console

Python SDK

```
s = f'SELECT COUNT(*) FROM "{fg.athena_query().table_name}" ' + \
        'WHERE fl_date = \'2020-03-31\''
q = feature_group.athena_query()
q.run(s, output_location=output_location)
q.wait()
df = q.as_dataframe()
```

aws startups

ML lineage from data to model



aws startups

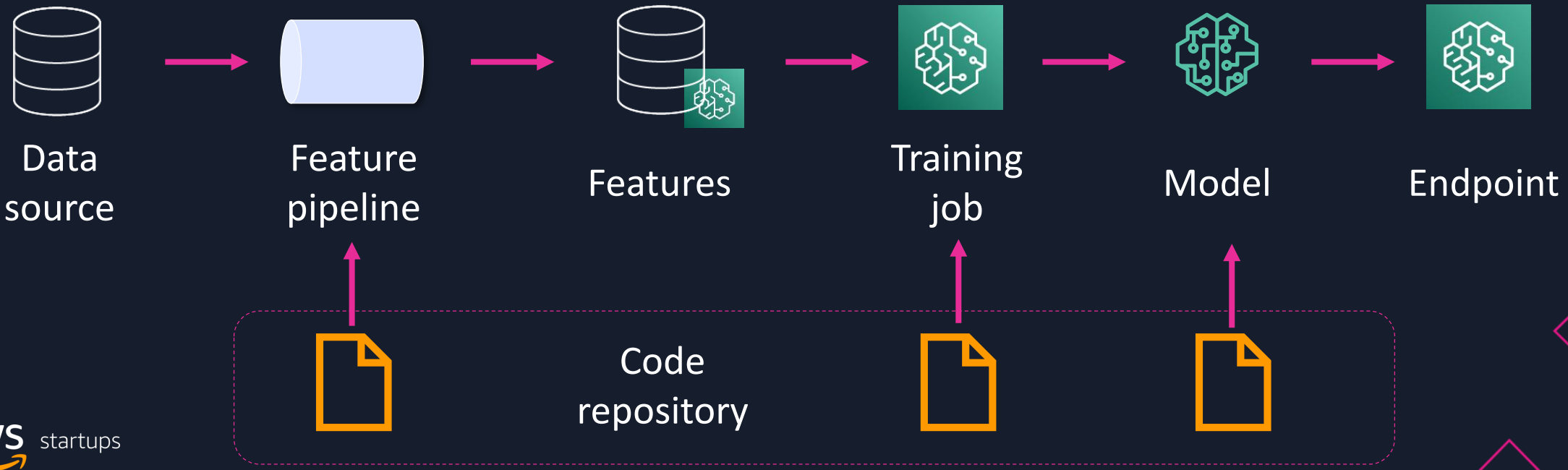
ML lineage

Feature management

- What's the impact of a feature change?
- Do existing features cover my use case?
- …

Auditing and troubleshooting

- What features do these predictions count on?
- How were the features built?
- …



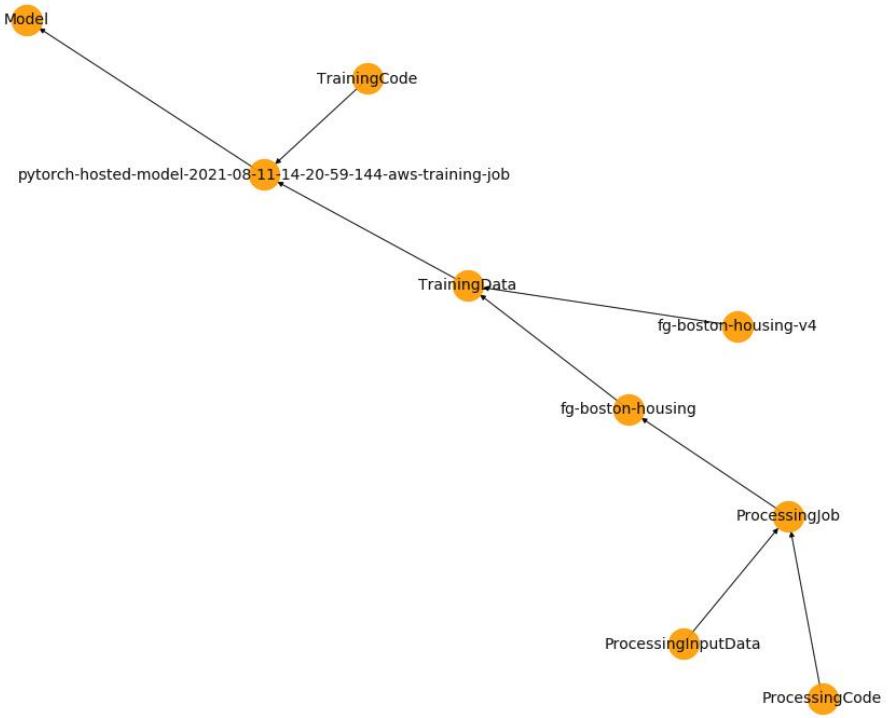
aws startups

Track Lineage from Query to model as DAG

```
ml_lineage = MLLineageTracking(sagemaker_session)

lineage = ml_lineage.create_ml_lineage(estimator,
        model_name=model_name,
        query=query,
        sagemaker_processing_job_description=
             preprocessing_job_description
        )

ml_lineage.graph()
```



Feature Store Landscape

A few selected popular feature stores…but there many more

Amazon SageMaker

Tecton

FEAST

Google Vertex AI

HOPSWORKS

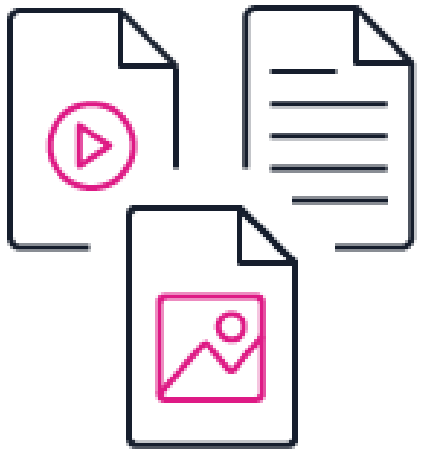
Databricks

aws startups

To get an overview check out https://www.featurestore.org/

What about other data, i.e. unstructured data?

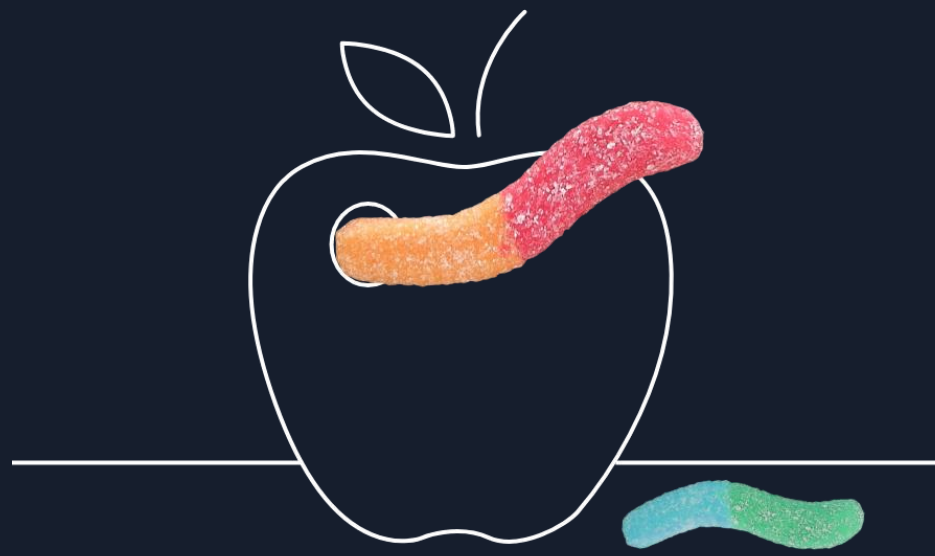
- Most feature stores focus on structured, e.g. tabular, data



- Options to include unstructured data (images, videos, …):
  - Limited support for blobs or text documents as strings
  - Convert arrays to strings, e.g. images
  - Feature with path to files stored on object storage
  - Convert unstructured data into structured, e.g. an image into a classification
  - Limited support for embeddings

SageMaker Feature Store

A quick demo



aws startups

SageMaker Feature Store – 10min Demo

https://www.youtube.com/watch?v=kgI0Rwn41S0&t=458s **(from 7:42 until 18:10)**

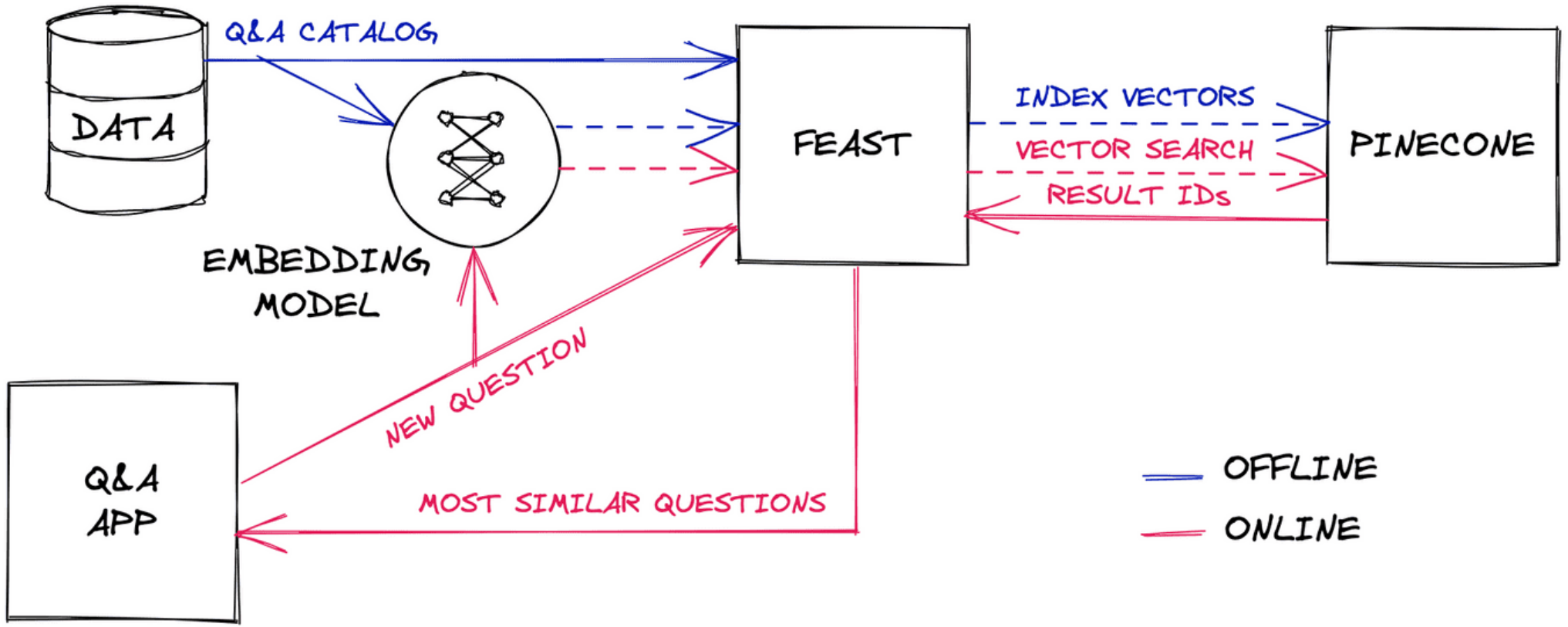
Q&A

aws startups

aws startups

Thank you

Feature Store + Vector DB for embeddings + KNN



https://www.pinecone.io/docs/examples/feast-feature-store/