Data Versioning & Quality, Feature Stores and Labeling

Robert Clements

MSDS Program

University of San Francisco

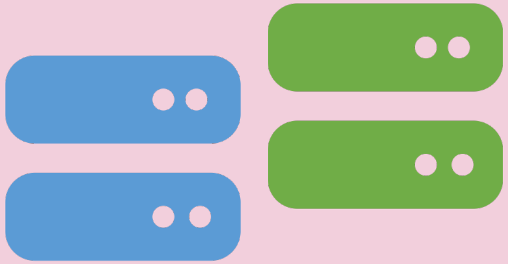
Announcements

- Grading on labs
- Project teams and ideas
  - 19 teams
  - Some duplicative ideas, but overall good
- Be in class on Thursday to complete first half of HW 1. Bring an actual pen or pencil to class.
- Do Quiz 1 before tomorrow night

What to Expect

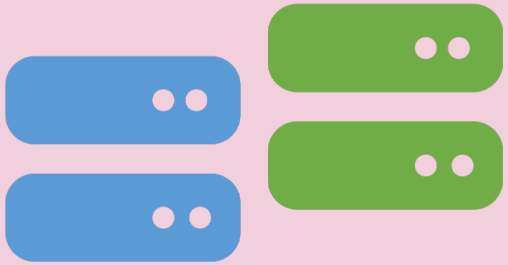
- Goal: to learn about the importance of data versioning in the model development process.
- How: in the lab we will use the very popular DVC (data version control) tool.
- Note: we are not going to build data pipelines (data engineering) but instead use version control to keep track of our data used for our models.

NAS, Network Drives, File systems



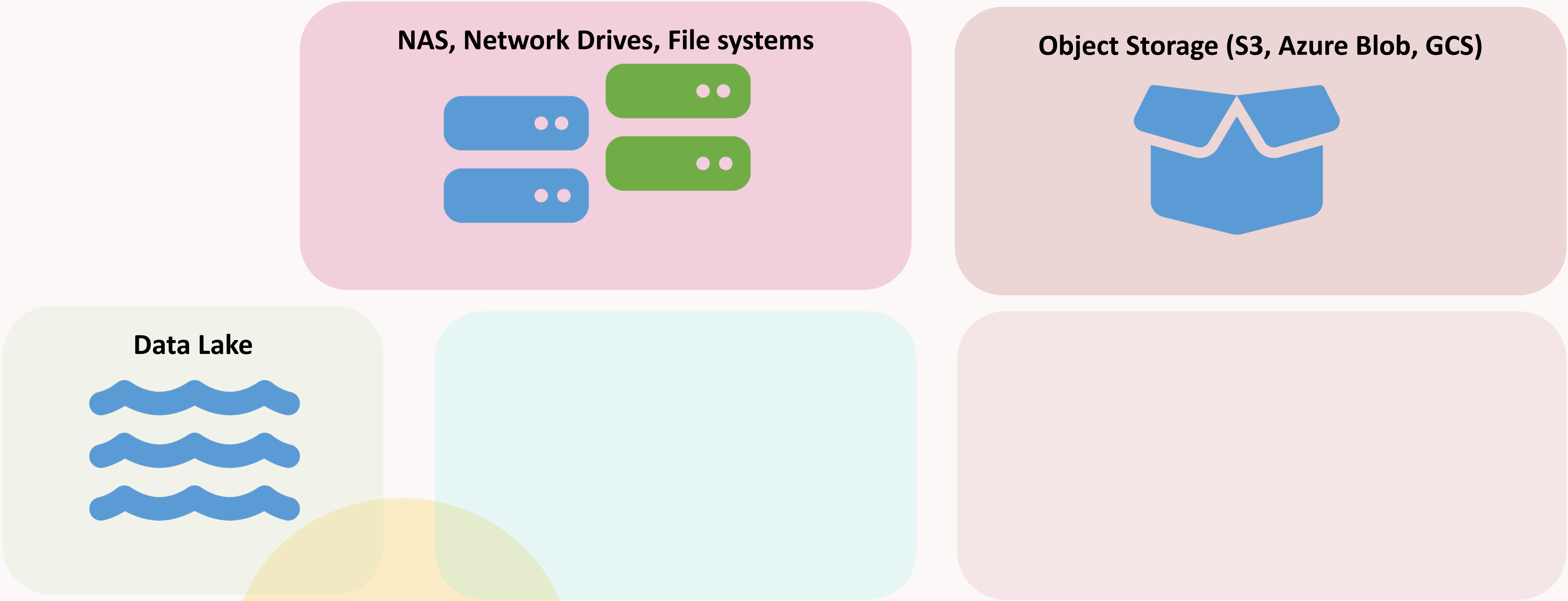
All types of files, just like on your laptop or cloud drive

NAS, Network Drives, File systems

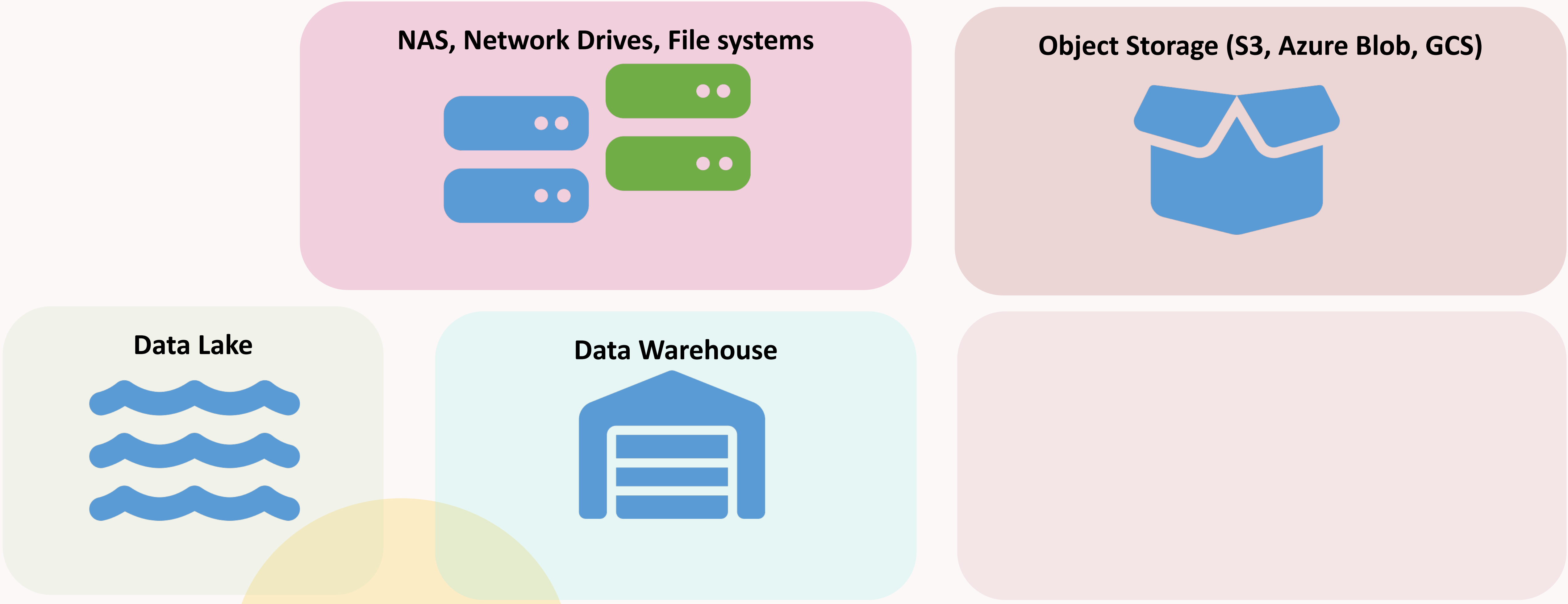


Object Storage (S3, Azure Blob, GCS)

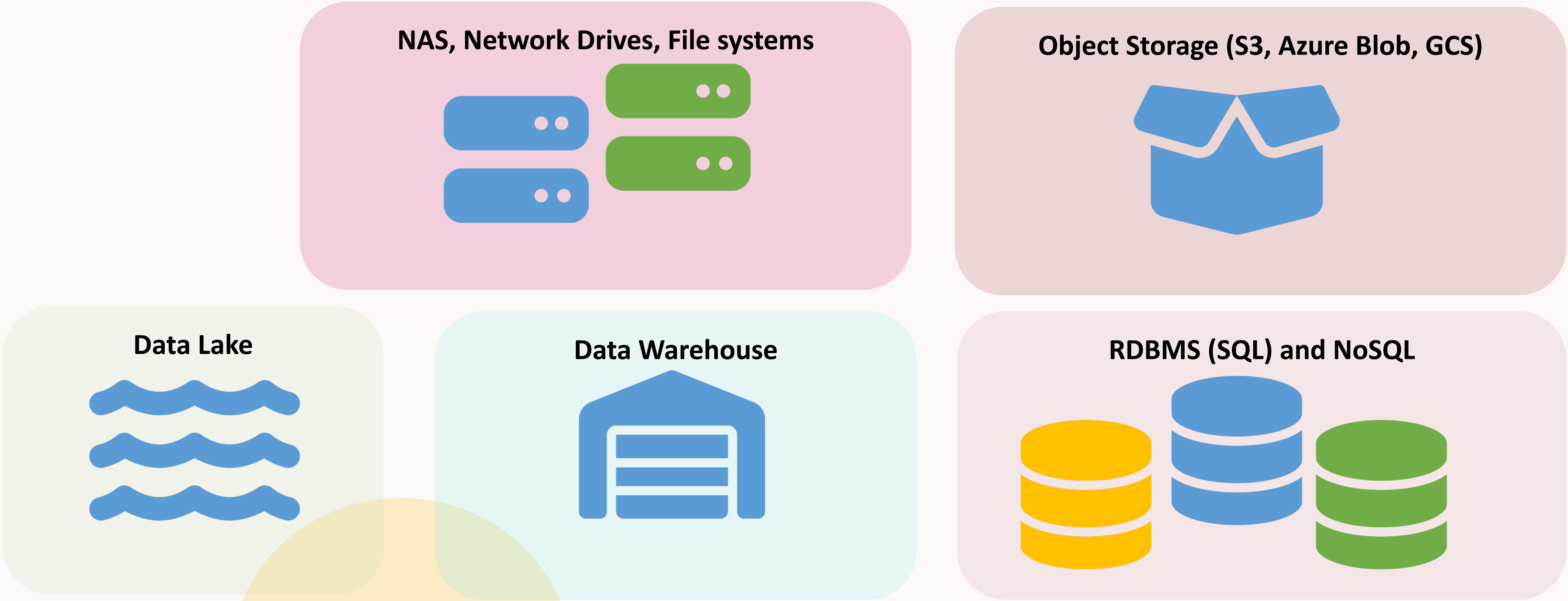
Similar to file system, store binaries, with redundancy and security.



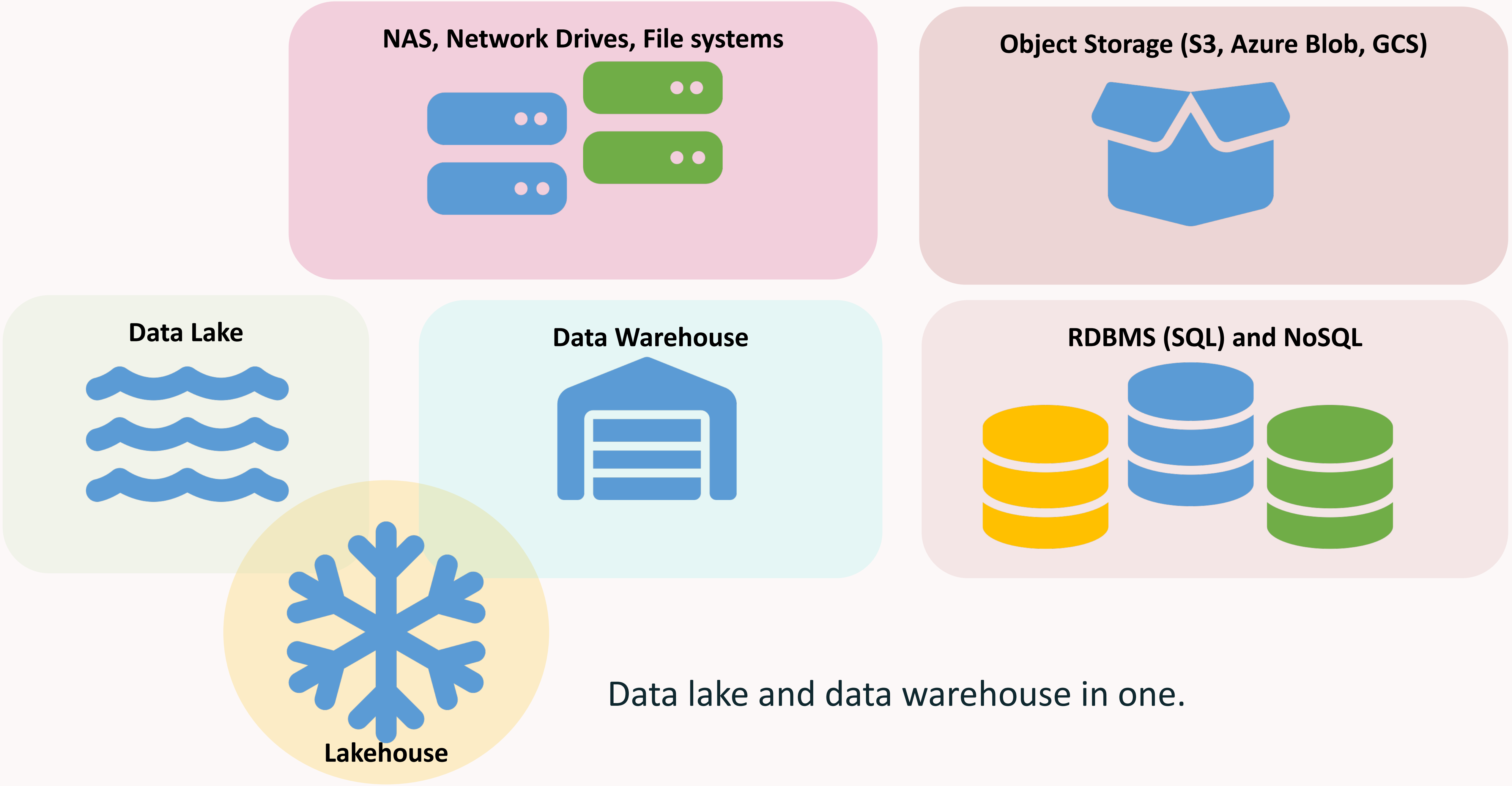
Dumping ground for raw data.



Nice, clean data using the extract-transform-load process.



Structured, semi-structured, unstructured and persistent data for analytics.



Data lake and data warehouse in one.

Data Pipelines

Though we won't be building pipelines, it's useful to know the main tools involved here tend to be Airflow, Prefect, Luigi, Dagster

Data Version Control

- Likely to iterate through many versions of data during development process
- Ideally can tie data to model/experiment
- data_v1.csv, data_v2.csv or dev_data.temp1, dev_data.temp2, etc. is bad practice and error-prone
- Recreating intermediate and final datasets from scratch is an option
  - True reproducibility
  - Sometimes not possible if org has bad data practices
- A good tool should make it easy to log and find a dataset used for a particular experiment

DVC

- Two main options: Git Large File Storage (LFS) and Data Version Control (DVC)
- DVC is similar to git
- CLI and VS Code extension
- Works on more than just data (e.g. models and experiments), but we’ll only use it for versioning data

Pipelines

Reproducible Pipelines

- All data should be reproducible, nothing adhoc

DVC Demo

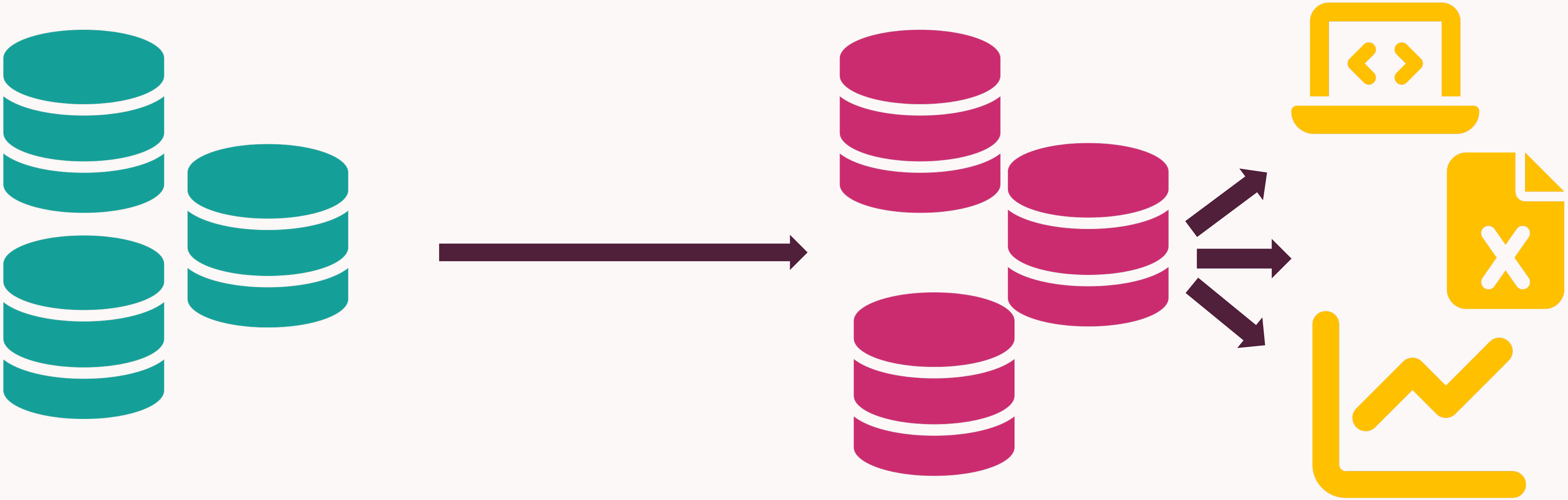
Data Quality

What to Expect

- Goal: to learn about the importance of data quality checking in the model development process.
- How: ~~in the lab we will use the very popular Great Expectations for data quality.~~
- Note: we are not going to build data pipelines (data engineering) but instead introduce how we might integrate quality control as part of a pipeline.

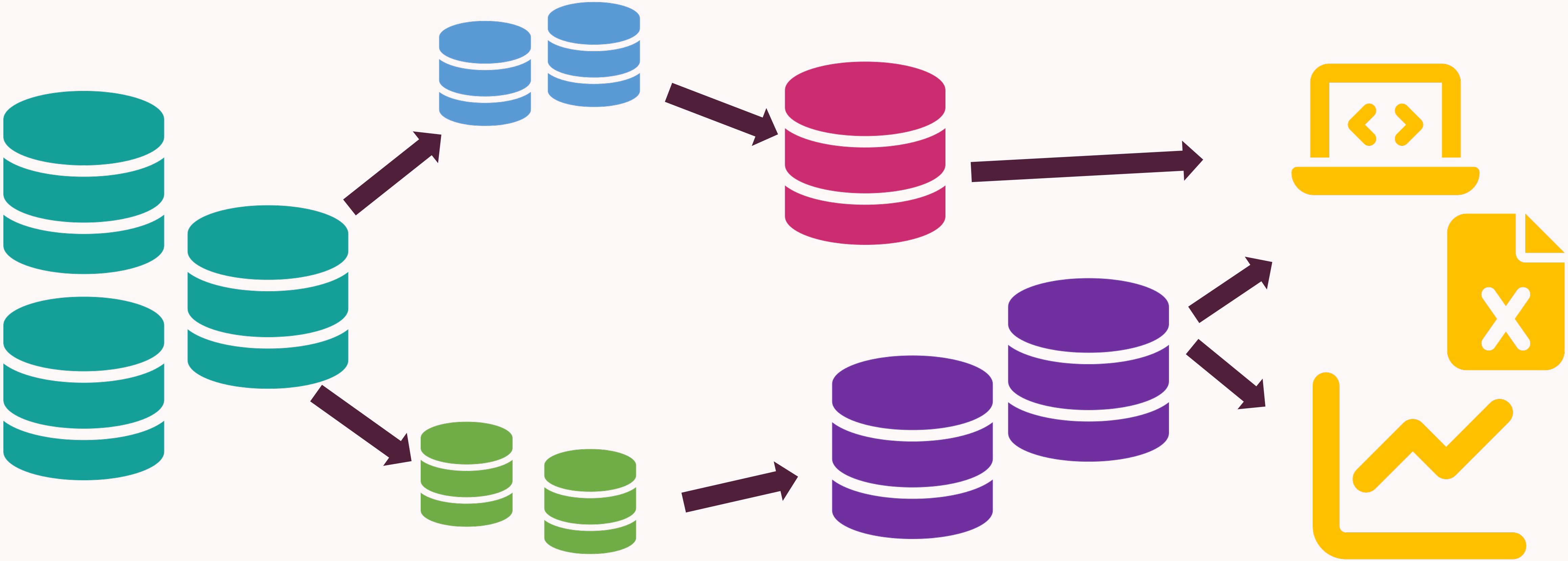
Data Quality

- Checking quality of upstream and downstream data sources is critical
  - Upstream and downstream data is used for many purposes, including model development/deployment, reporting, ad-hoc analyses, etc.



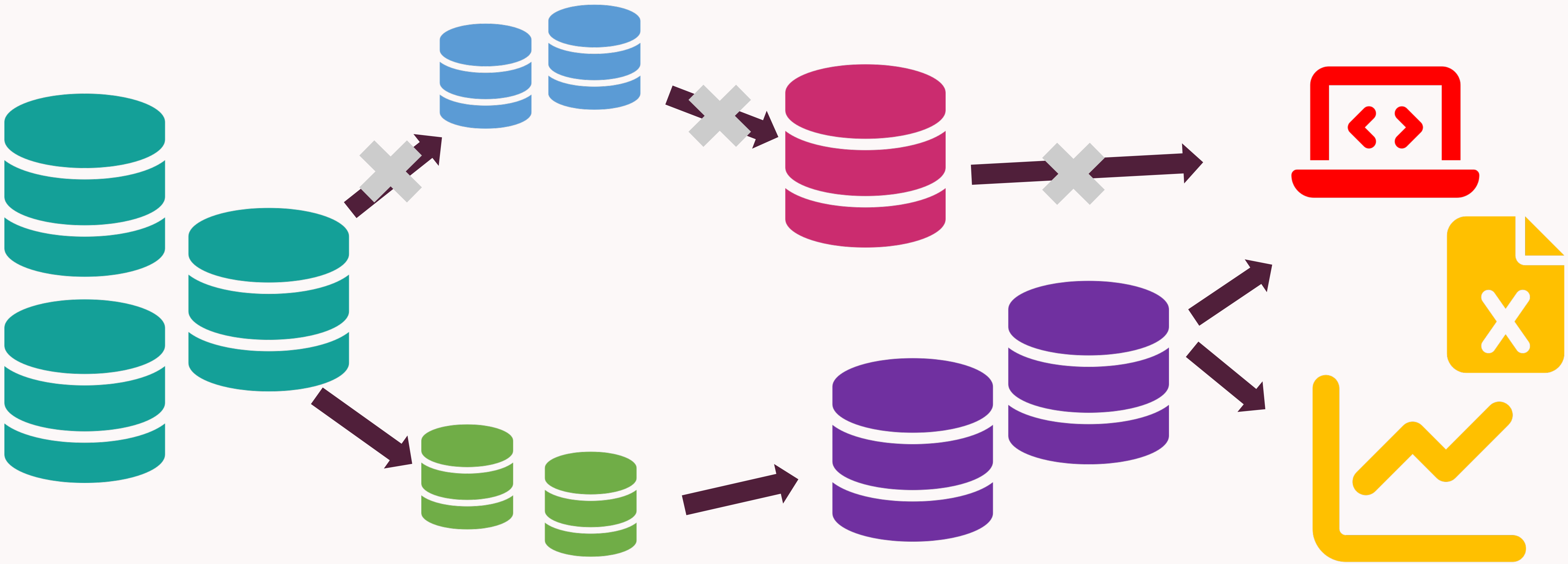
Data Quality

- Checking quality of upstream and downstream data sources is critical
  - Data easily gets fragmented, and can be owned by different teams



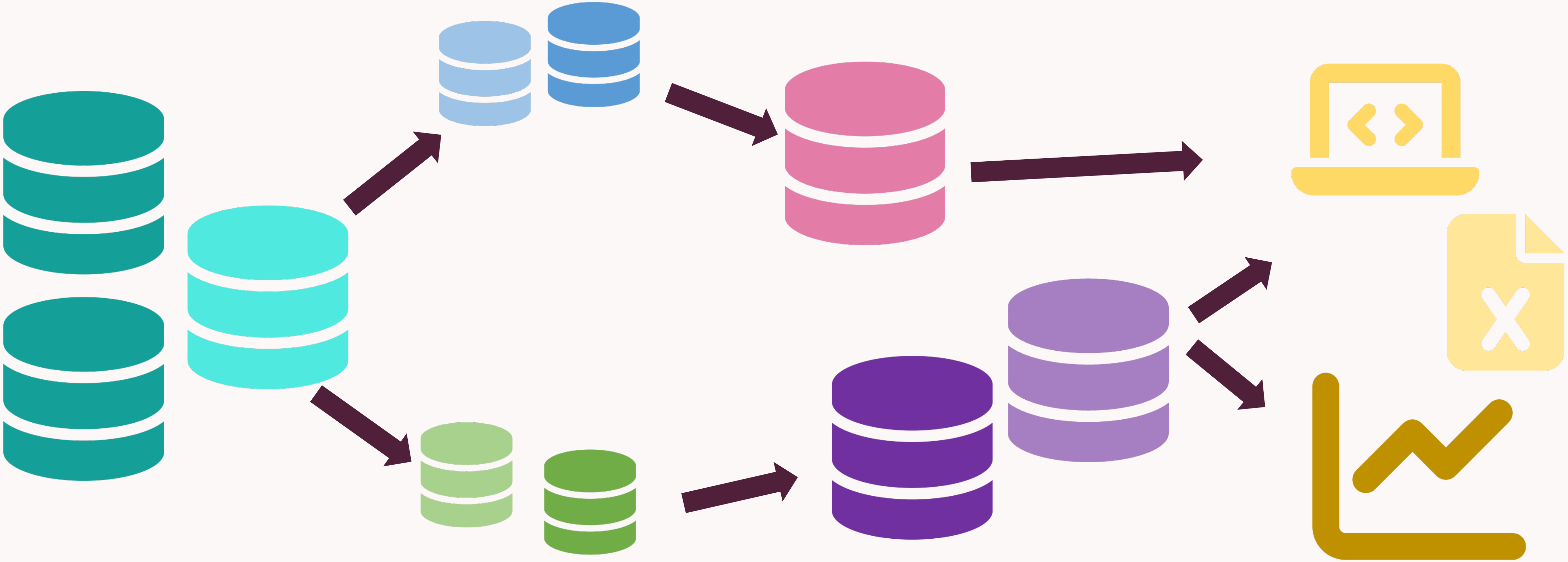
Data Quality

- Checking quality of upstream and downstream data sources is critical
  - Data pipelines break without warning



Data Quality

- Checking quality of upstream and downstream data sources is critical
  - Data/schema changes, sometimes without sufficient warning



Data Checks are Problem-Specific

Great Expectations

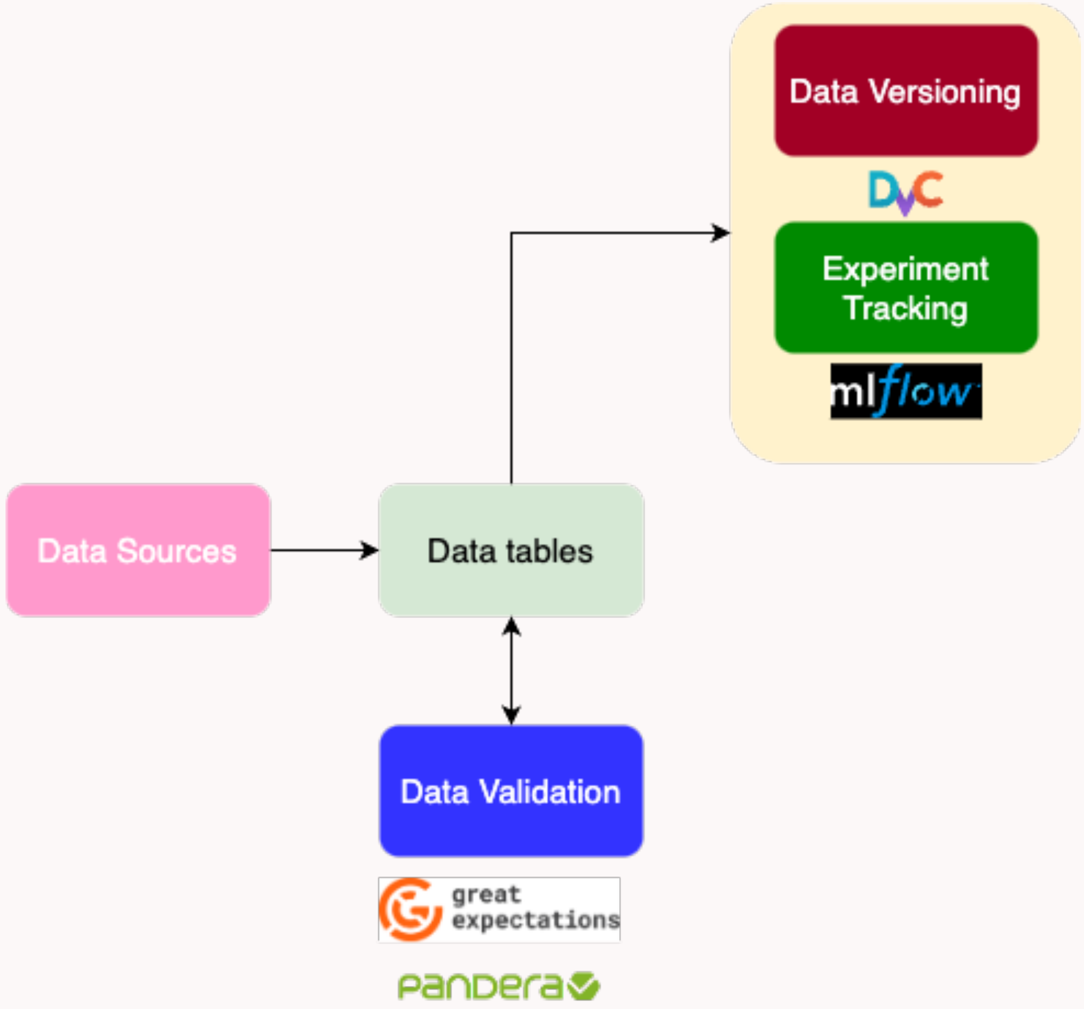
- Python-based declarative language for validating, documenting, and profiling data.
- Is NOT a pipeline execution or data versioning tool.
- Read the docs.

Great Expectations

- Great Expectations can be a part of the ETL pipeline execution

Alternatives

- Deepchecks (https://deepchecks.com/)
- Soda (https://www.soda.io/)
- Pandera (https://pandera.readthedocs.io/en/stable/)
- Deequ (https://github.com/awslabs/deequ): spark-based
- Data Validation Tool (https://github.com/GoogleCloudPlatform/professional-services-data-validator)



Data Versioning
DVC
Experiment Tracking
mlflow
Data Sources
Data tables
Data Validation
great expectations
pandera

Feature Stores and Platforms

What to Expect

- Goal: to learn about how the use of feature stores and platforms might help accelerate model development and ease model deployment.

- How: we will not be doing a feature store lab. Feel free to explore on your own.

Feature Store History

- In 2017, Uber wrote a blog post detailing [Michelangelo]

Feature Store

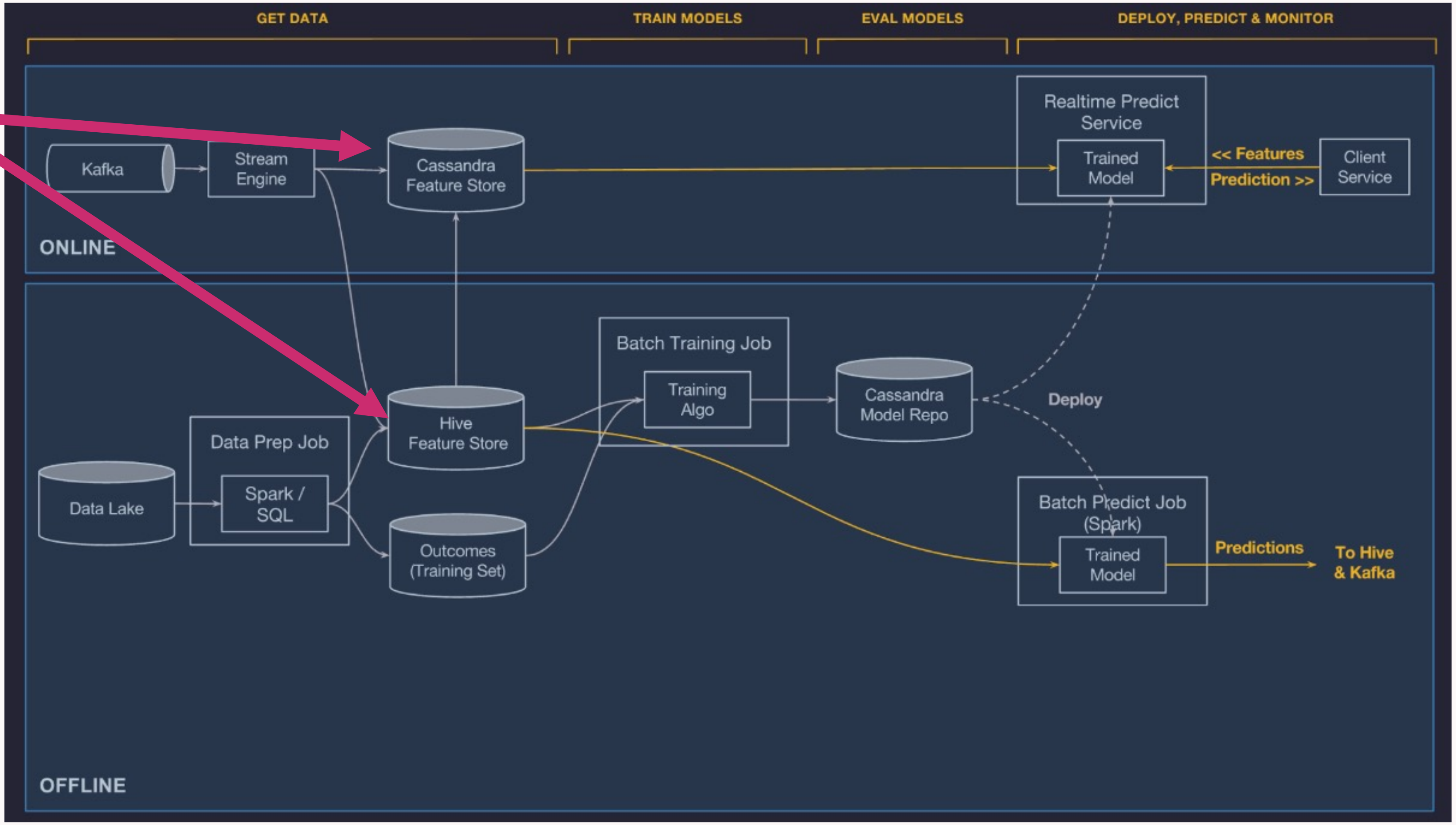
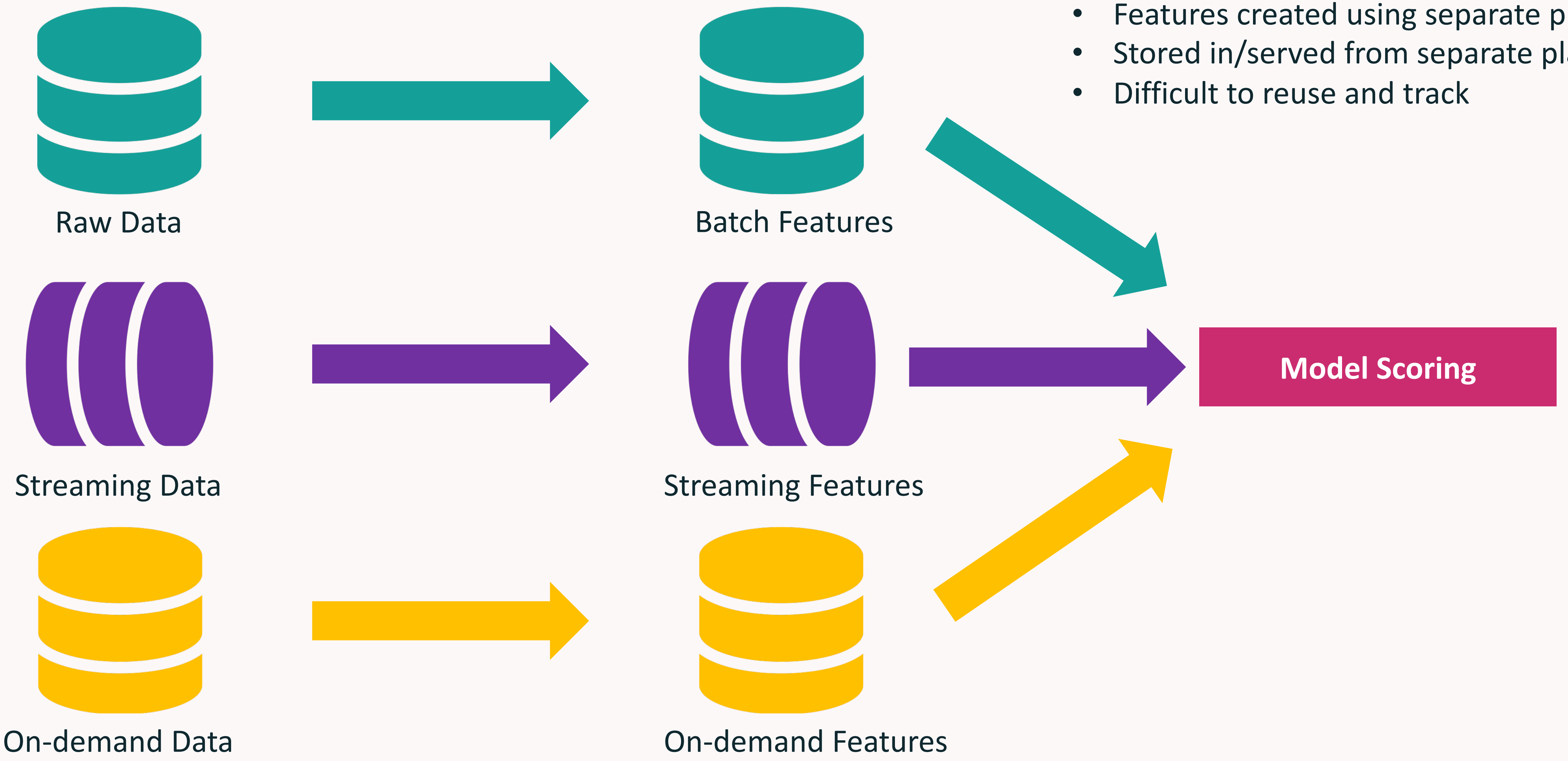


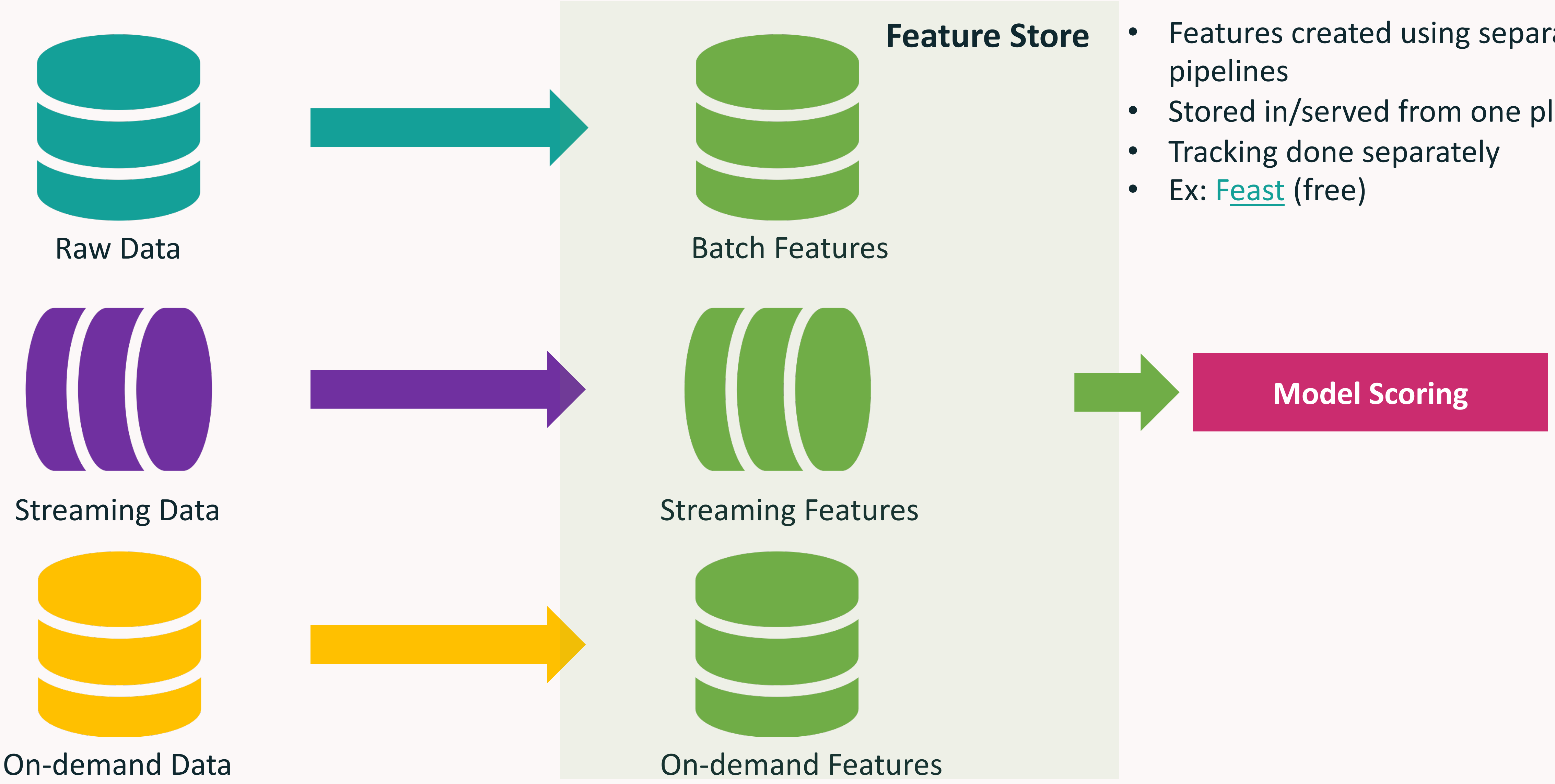
Image taken from their blog post

Feature Stores vs Platforms

- Features created using separate pipelines
- Stored in/served from separate places
- Difficult to reuse and track

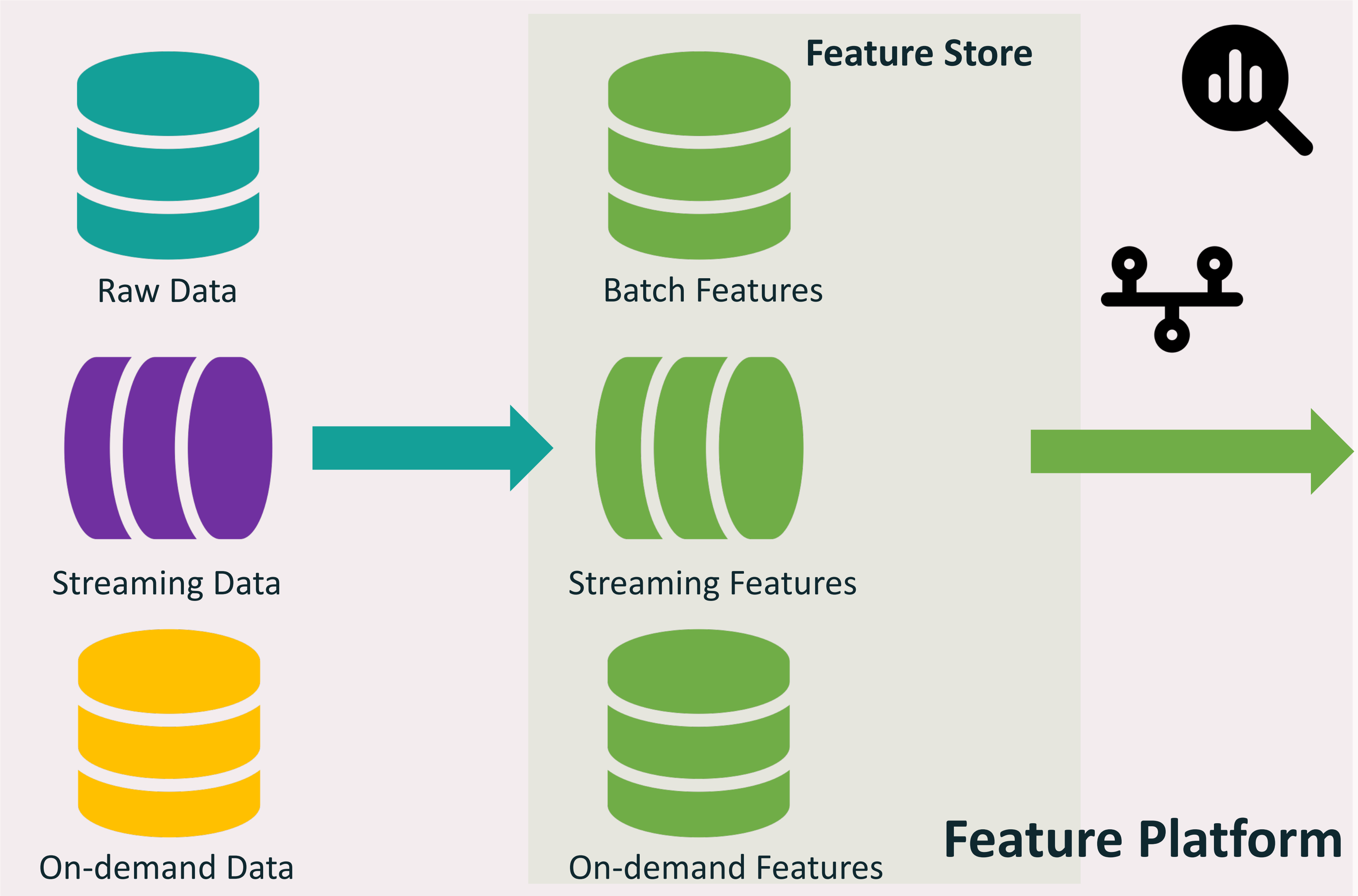


Feature Stores vs Platforms



- Features created using separate pipelines
- Stored in/served from one place
- Tracking done separately
- Ex: Feast (free)

Feature Stores vs Platforms



- Features orchestrated in one place
    - Dev/prod sync
- Stored in/served from one place
- Tracking, versioning included
- Ex: Tecton ($$) and Featureform

Feast

Feast is an open source feature store (not platform):

- Manages storage in other databases
- Integrates with many data sources (GCP, AWS, Azure, Snowflake) and storage (Postgres, Dynamo, Redis, and others)

Labeling

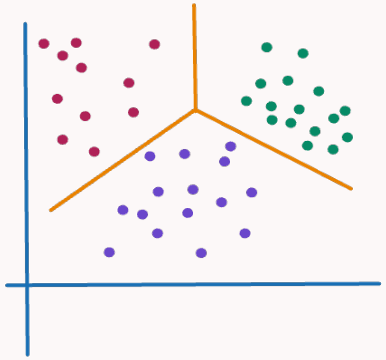
What to Expect

- Goal: we’ve learned about data quality and feature stores/platforms, so we should complete the picture and wrap everything up by learning about labeling solutions.

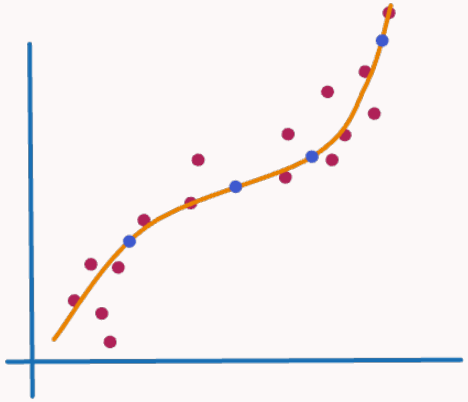
- How: we will not be doing a labeling lab. Feel free to explore on your own.

In some cases, we may not need to label

Unsupervised learning

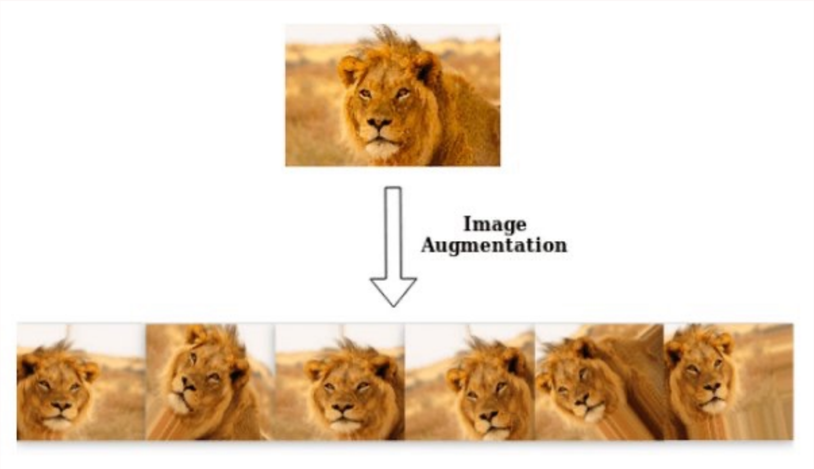


Semi-supervised learning



Augmentation

Image from https://www.quora.com/What-is-data-augmentation-in-CNN



Self-supervised learning

Synthetic data

Labeling Options

Read the Michelangelo blog post