DataOps in practice

2019-03-14

Lars Albertsson
Mimeria

Friction to production

Which NoSQL database would you recommend for this use case?

It won’t matter. Use something that you are used to. MySQL, Oracle?

Well, I’d prefer something else.

?

If we use an RDBMS, Ops have rules and opinions that slow us down.

Distance to production

What are your data scientists up to?

They have received a data dump and built a model.

Great, that is the first 1%.
What's next?

We will hand it off to a developer team, who hands it off to operations when the model is translated to Java.

Risky operations

How to I test the pipeline?

You temporarily change the output path and run manually.

What if I forget to change path?

Don’t do that.

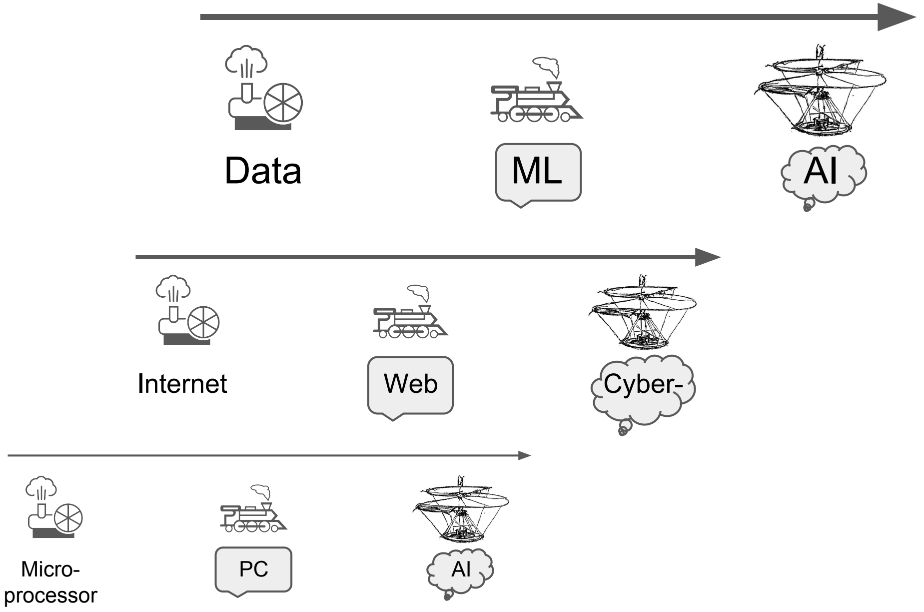
Disrupted or disruptor

Friction

Silos

Operational risk

Digital revolution steam engines



Properties of disruptors

Sustained ROI from machine learning

Short time from idea to production

Homogeneous data platform

Data processing pipelines

Major cloud + open source

Organisation aligned by use case

Data internally democratised

Diverse teams: data science, dev, ops

DataOps

Building data processing software

MIND THE GAP

Running data processing software

Purpose

Sustained ROI from machine learning

Short time from idea to production

Context

Homogeneous data platform

Data processing pipelines

Major cloud + open source

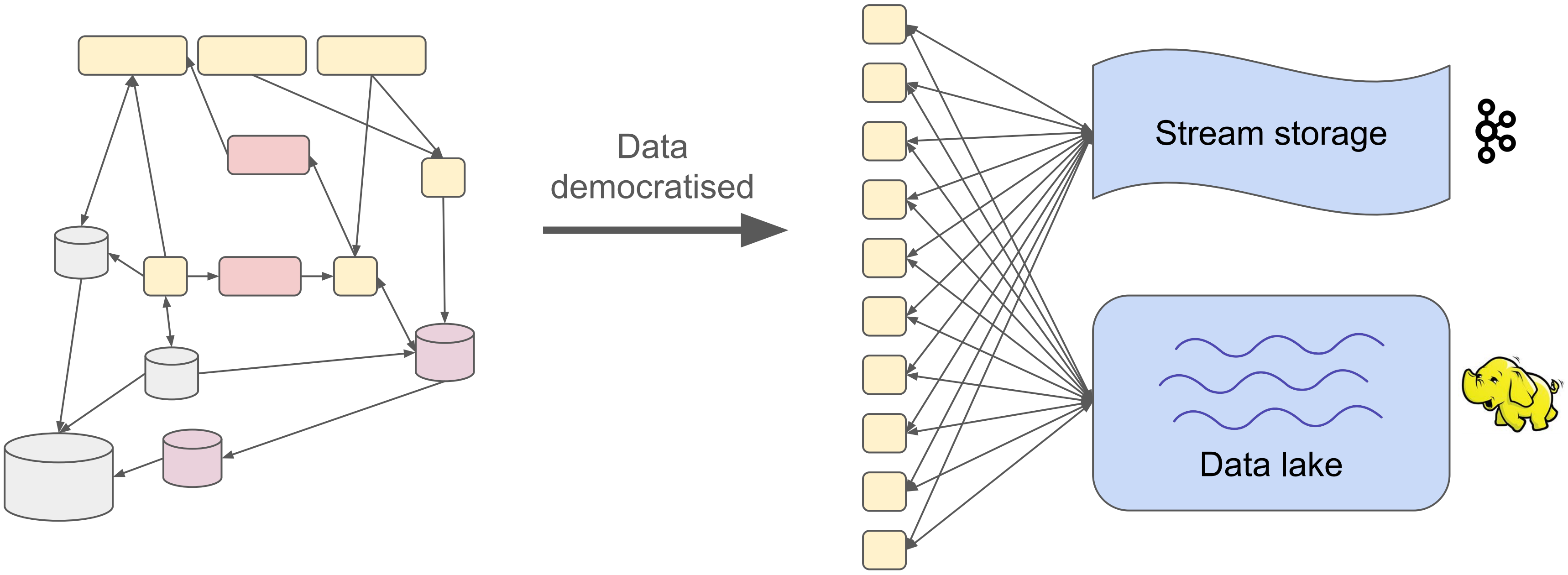
Means

Organisation aligned by use case

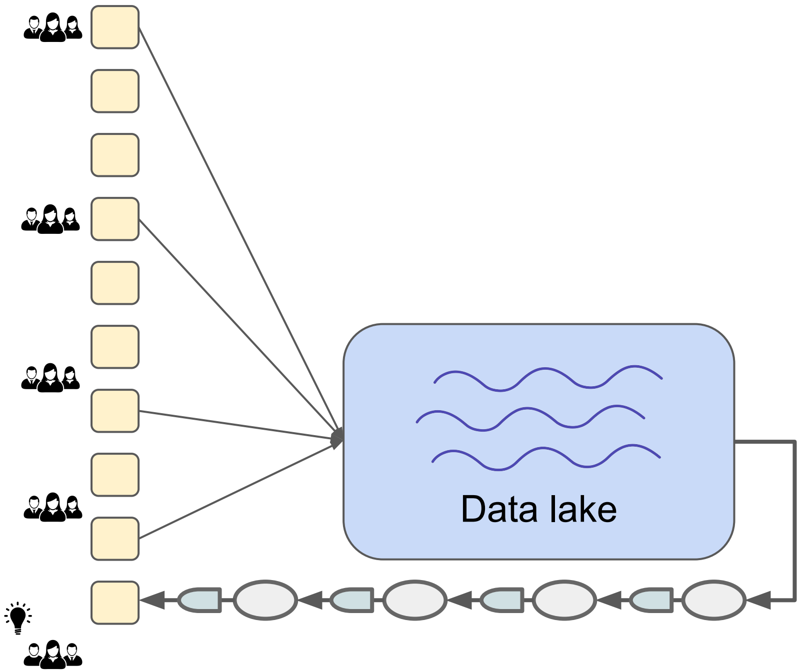
Data internally democratised

Diverse teams: data science, dev, ops

Big data - a collaboration paradigm

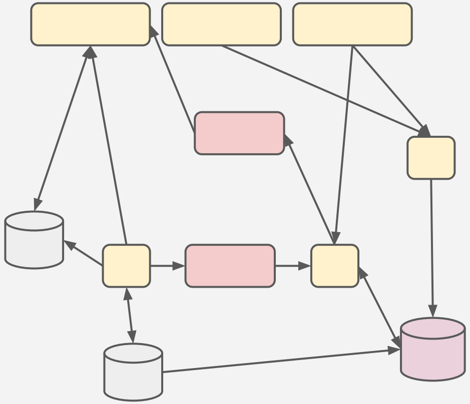


Data pipelines



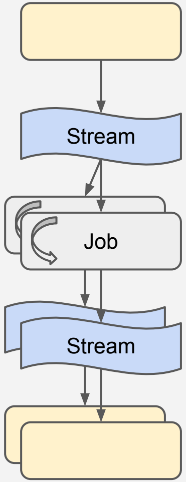
Data integration timescales

Online



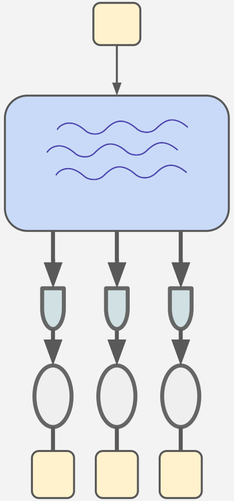
- SOA / microservices
- Synchronous RPC
- 1-100 ms

Nearline



- Stream storage (Kafka)
- Asynchronous event processing
- 10 ms - 1 hour

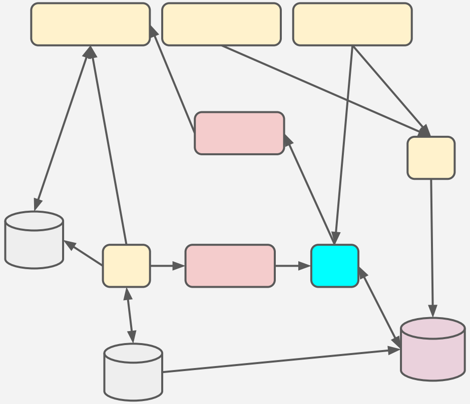
Offline



- File storage (Hadoop)
- Asynchronous batch processing
- 10 minutes -

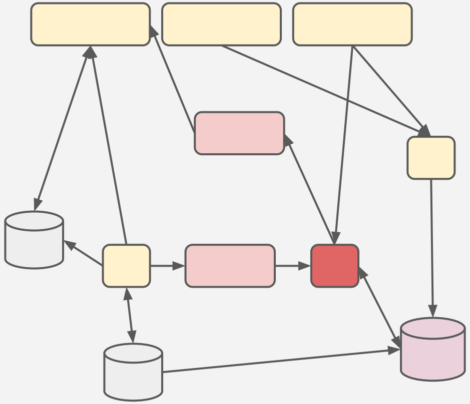
Operational manoeuvres - online

Upgrade



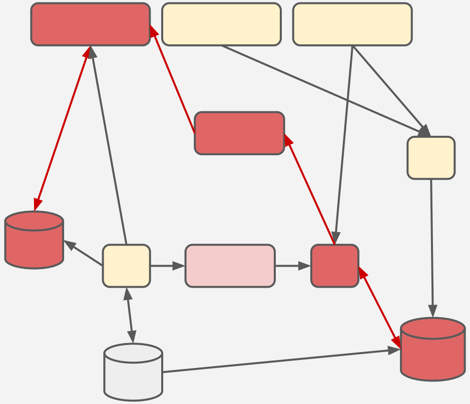
- Careful rollout
- Risk of user impact
- Proactive QA

Service failure



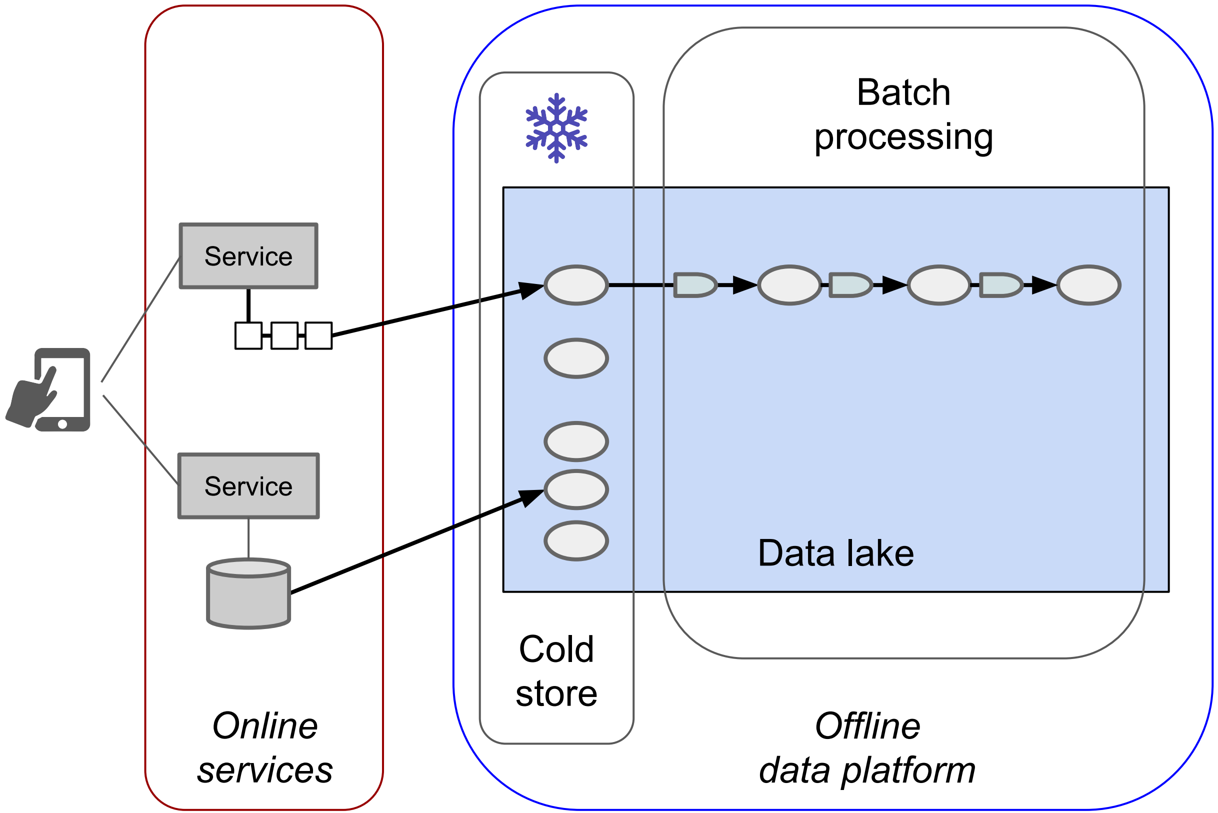
- User impact
- Data loss
- Cascading outage

Bug

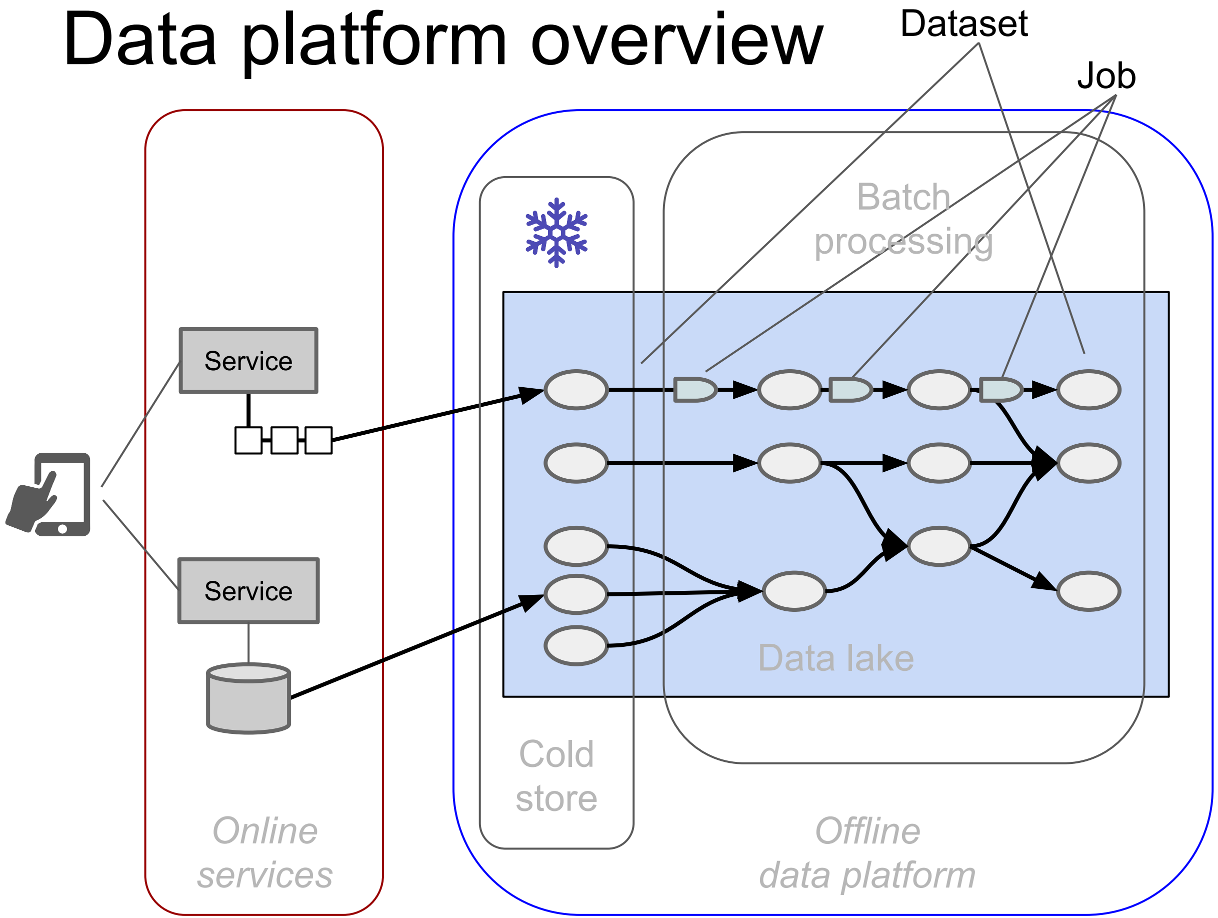


- User impact
- Data corruption
- Cascading corruption

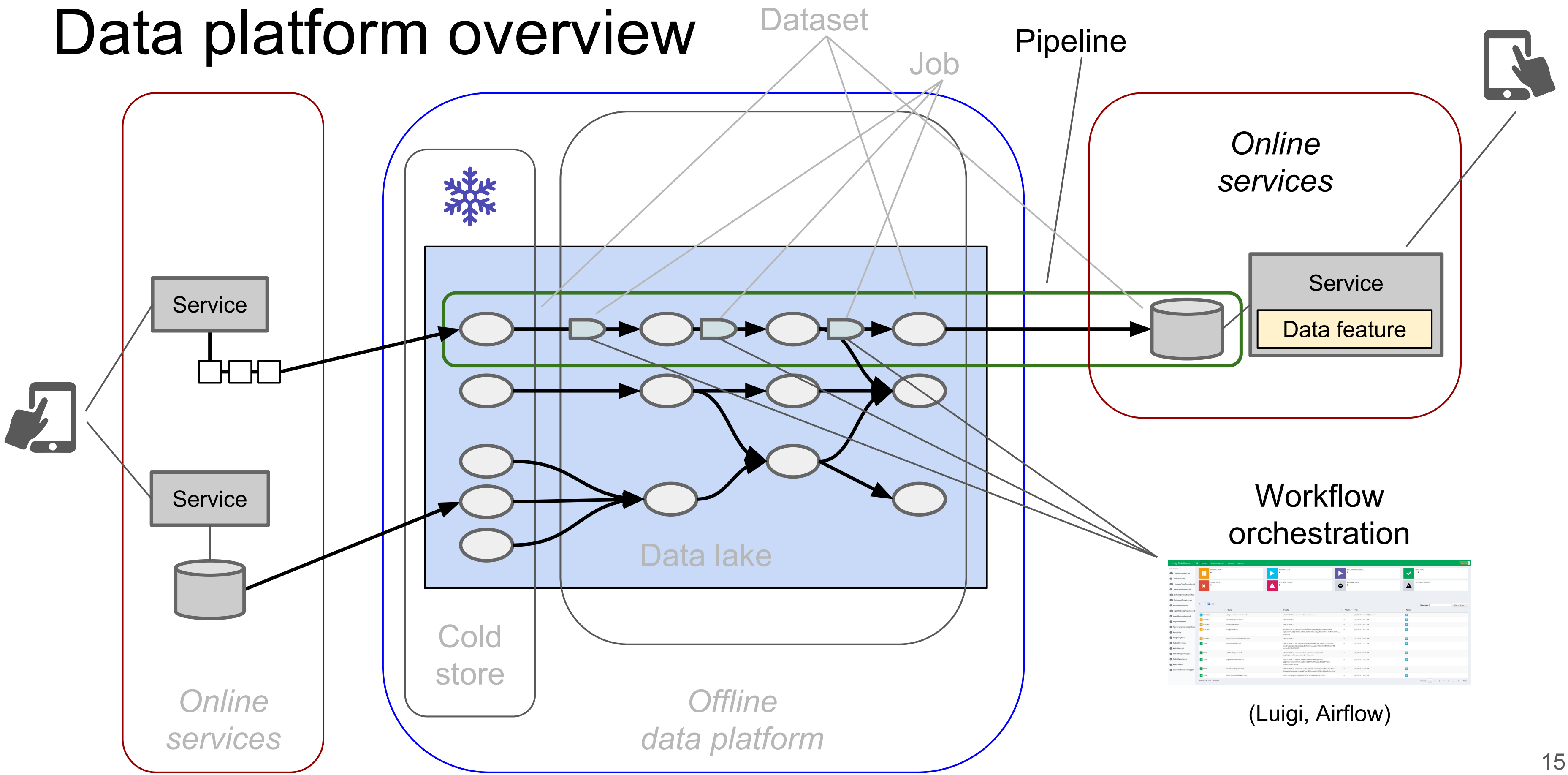
Data platform overview



Data platform overview

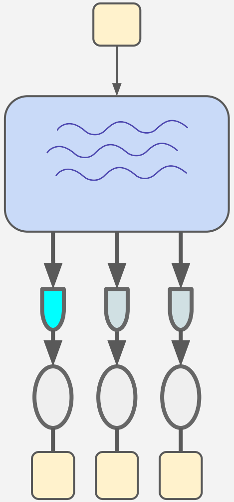


Data platform overview



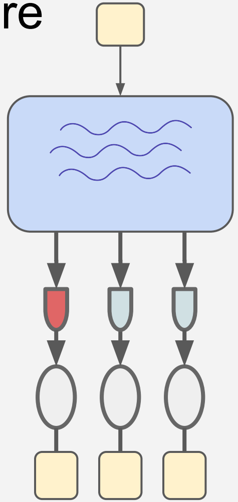
Operational manoeuvres - offline

Upgrade



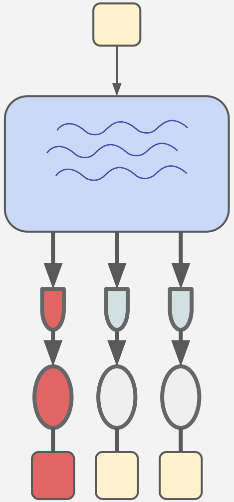
- Instant rollout
- No user impact
- Reactive QA

Service failure



- Pipeline delay
- No data loss
- No downstream impact

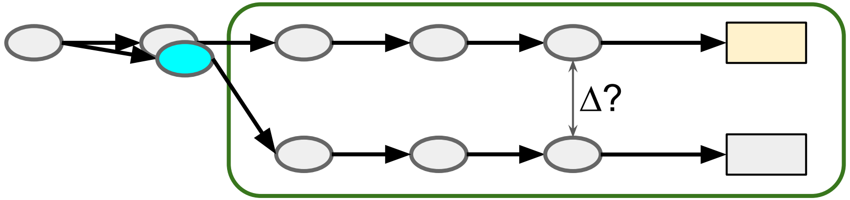
Bug



- Temporary data corruption
- Downstream impact

Production critical upgrade

- Dual datasets during transition
- Run downstream parallel pipelines
    - Cheap
    - Low risk
    - Easy rollback
- Easy to test end-to-end
    - Upstream team can do the change



No dev & staging environment needed!

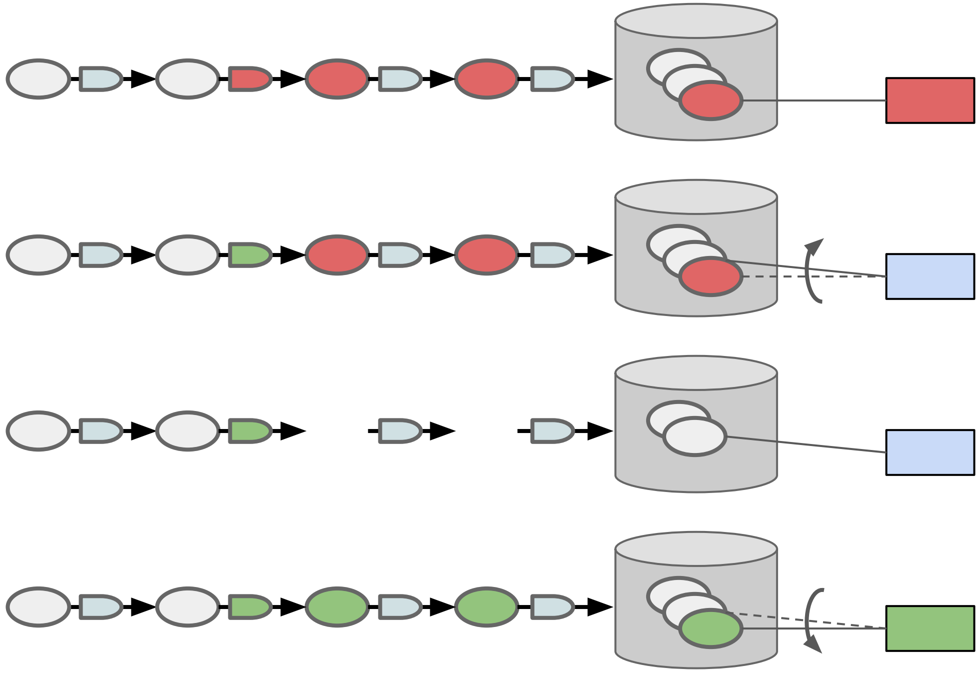
Life of an error, batch pipelines

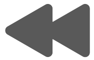
- Faulty job, emits bad data

  1. Revert serving datasets to old
  2. Fix bug
  3. Remove faulty datasets
  4. Backfill is automatic (Luigi)

  Done!

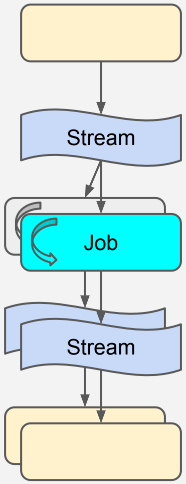
- Low cost of error
  - Reactive QA
  - Production environment sufficient





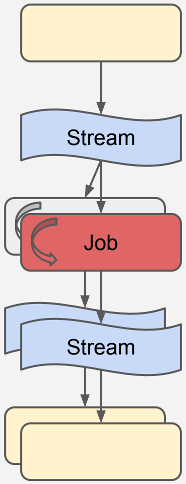
Operational manoeuvres - nearline

Upgrade



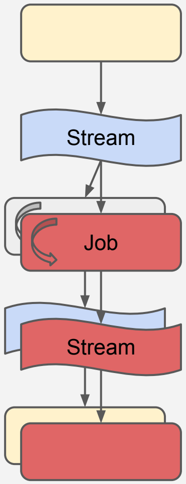
- Swift rollout
- Parallel pipelines
- User impact, QA?

Service failure



- Pipeline delay
- No data loss
- Downstream impact?

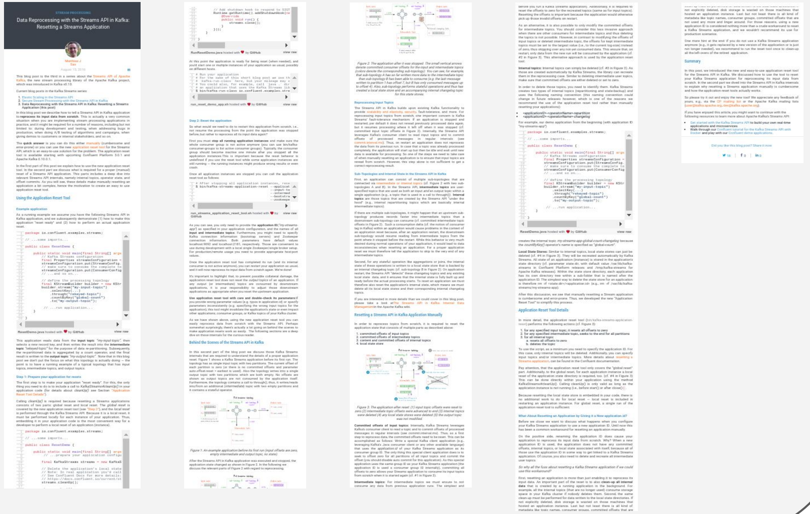
Bug



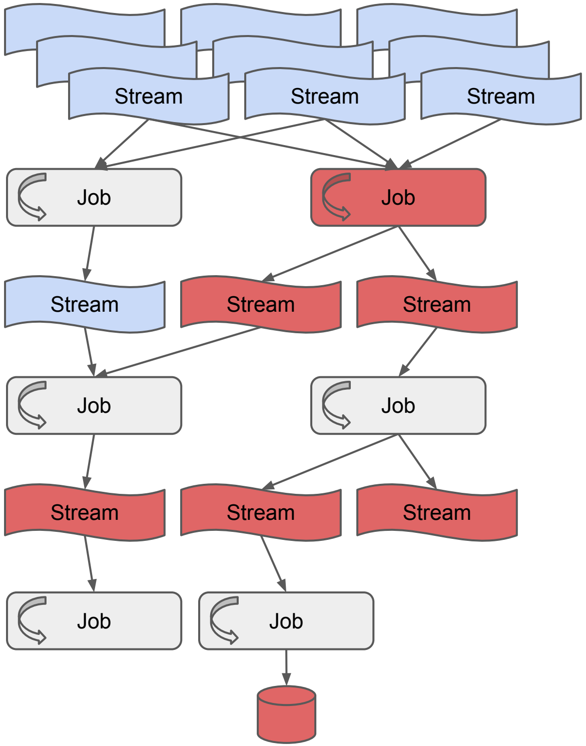
- Data corruption
- Downstream impact

Life of an error, streaming

Reprocessing in Kafka Streams

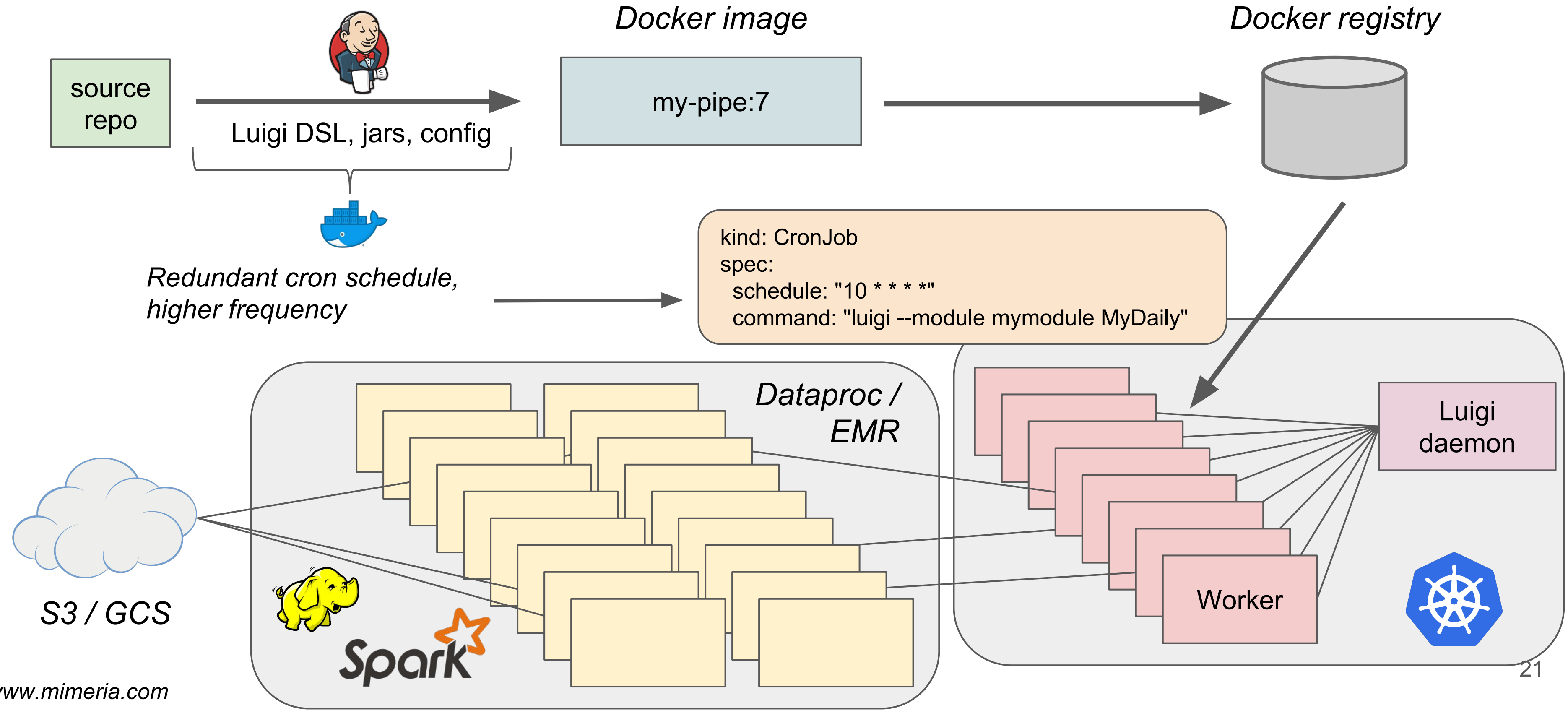

STREAM PROCESSING

Data Reprocessing with the Streams API in Kafka: Resetting a Streams Application



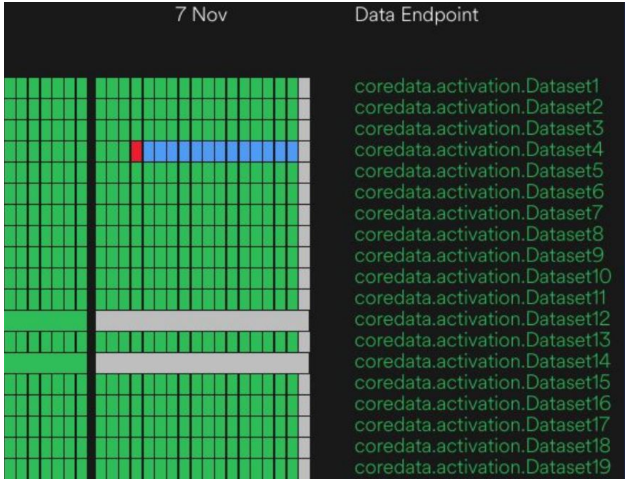
- Works for a single job, not pipeline. :-(

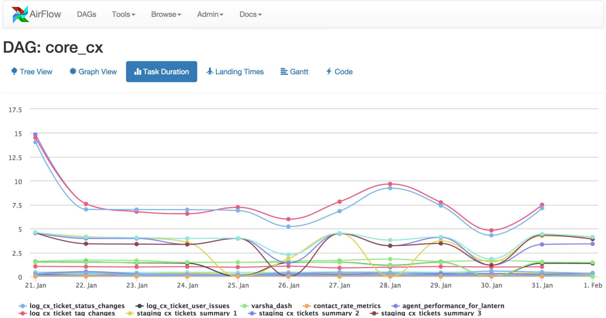
Deployment example, cloud native

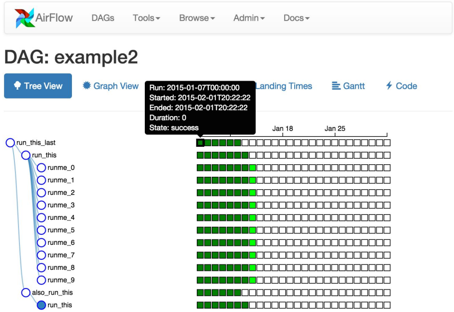


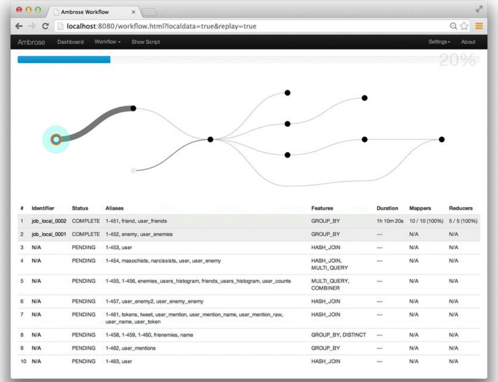
Monitoring timeliness, examples

- Datamon - Spotify internal
- Twitter Ambrose (dead?)
- Airflow



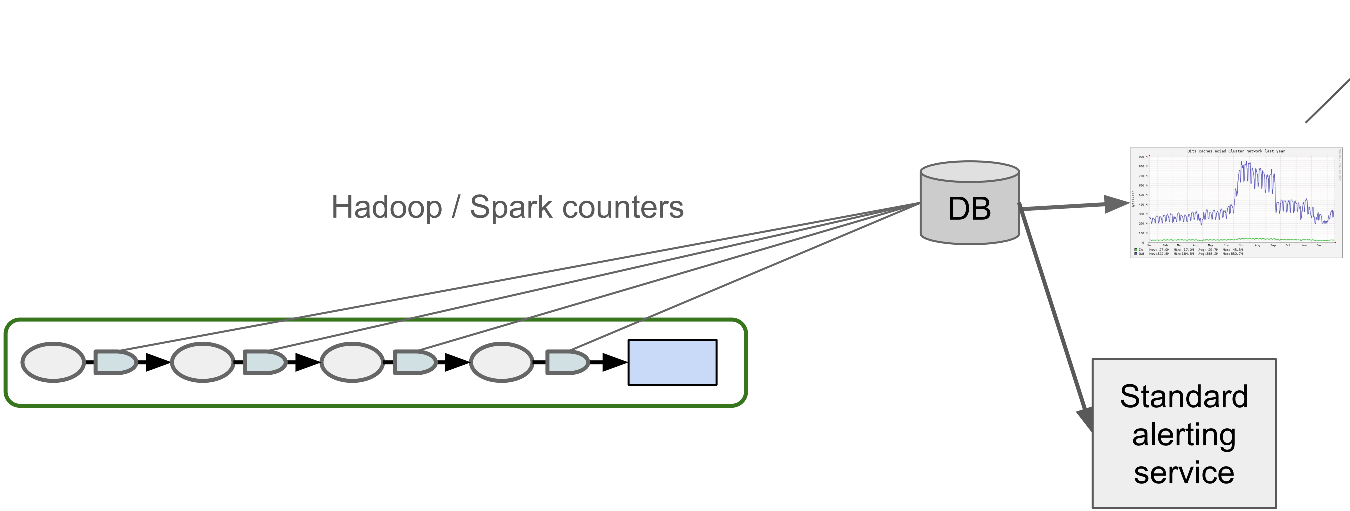






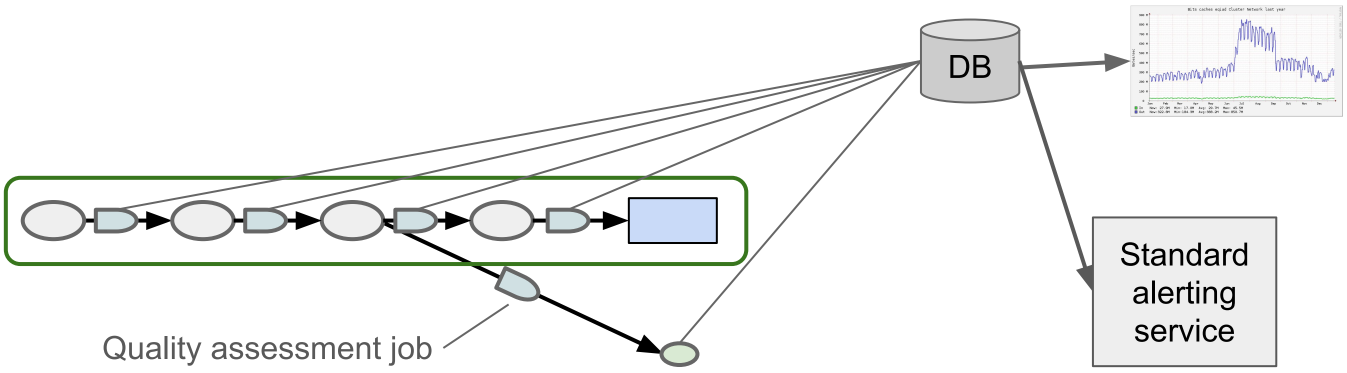
Measuring correctness: counters

- Processing tool (Spark/Hadoop) counters
  - Odd code path => bump counter
  - System metrics



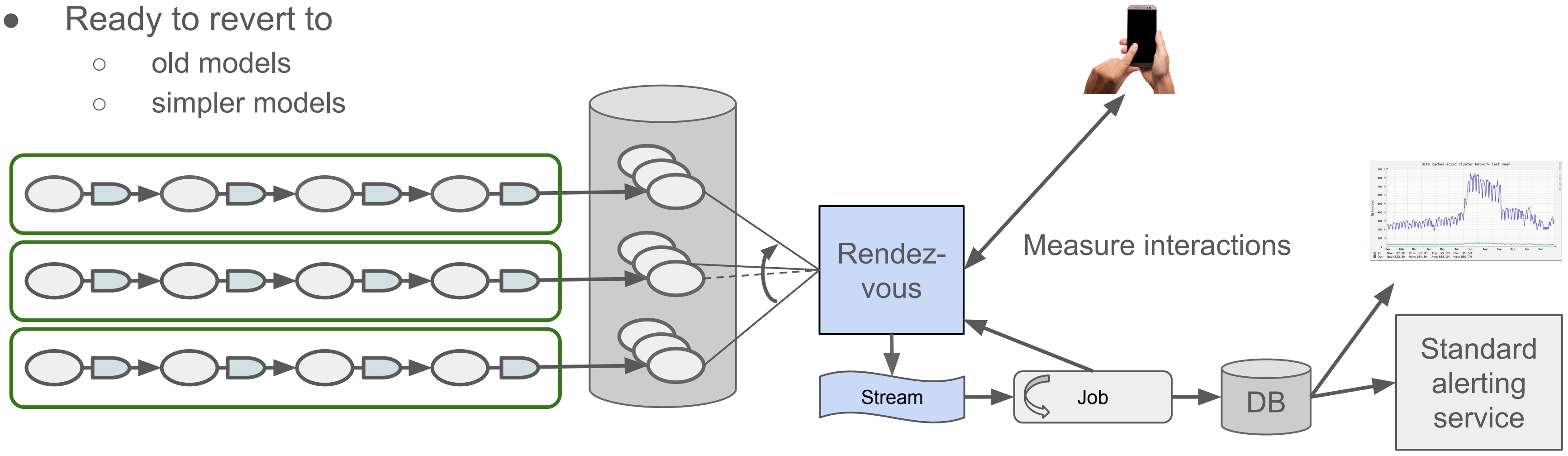
Measuring correctness: pipelines

- Processing tool (Spark/Hadoop) counters
  - Odd code path => bump counter
  - System metrics
- Dedicated quality assessment pipelines



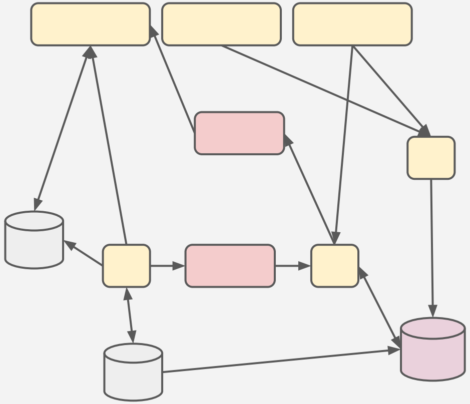
Machine learning operations

- Multiple trained models
  - Select at run time
- Measure user behaviour
  - E.g. session length, engagement, funnel
- Ready to revert to
  - old models
  - simpler models

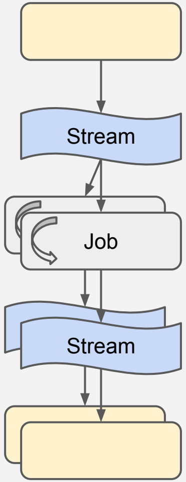


Data processing tradeoff

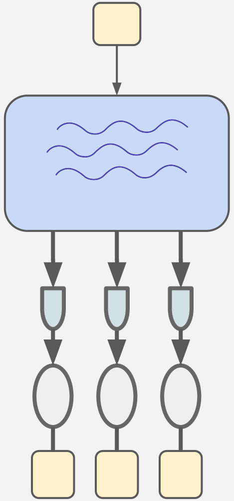
Online



Nearline



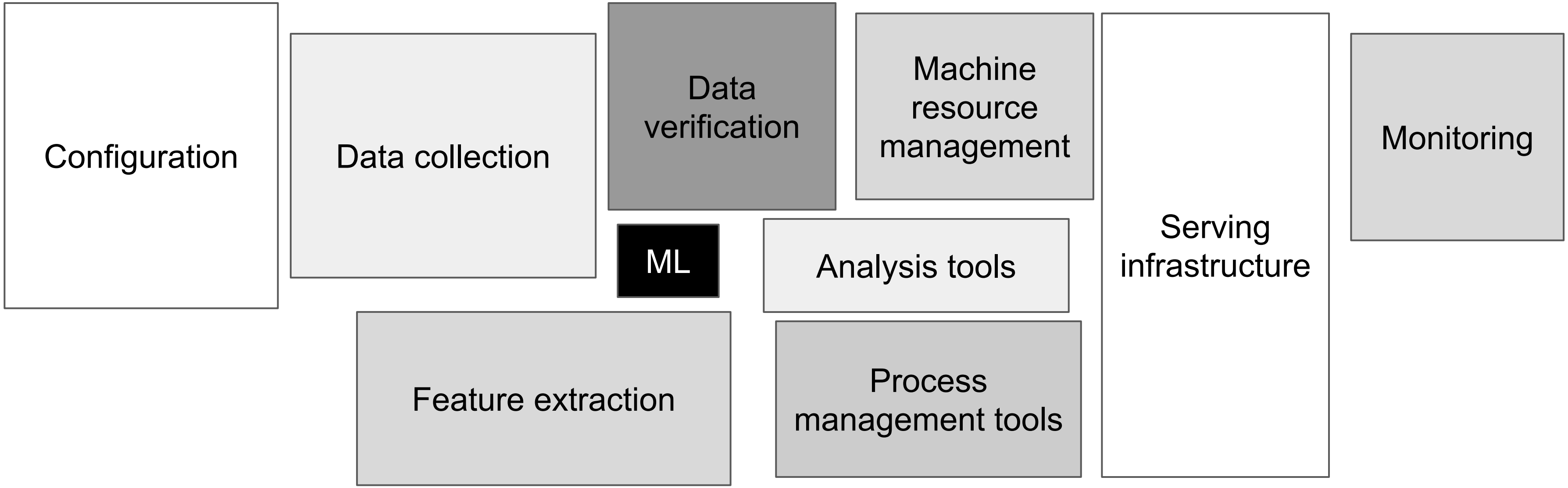
Offline



Data speed ⟷ Innovation speed

Bonus slides

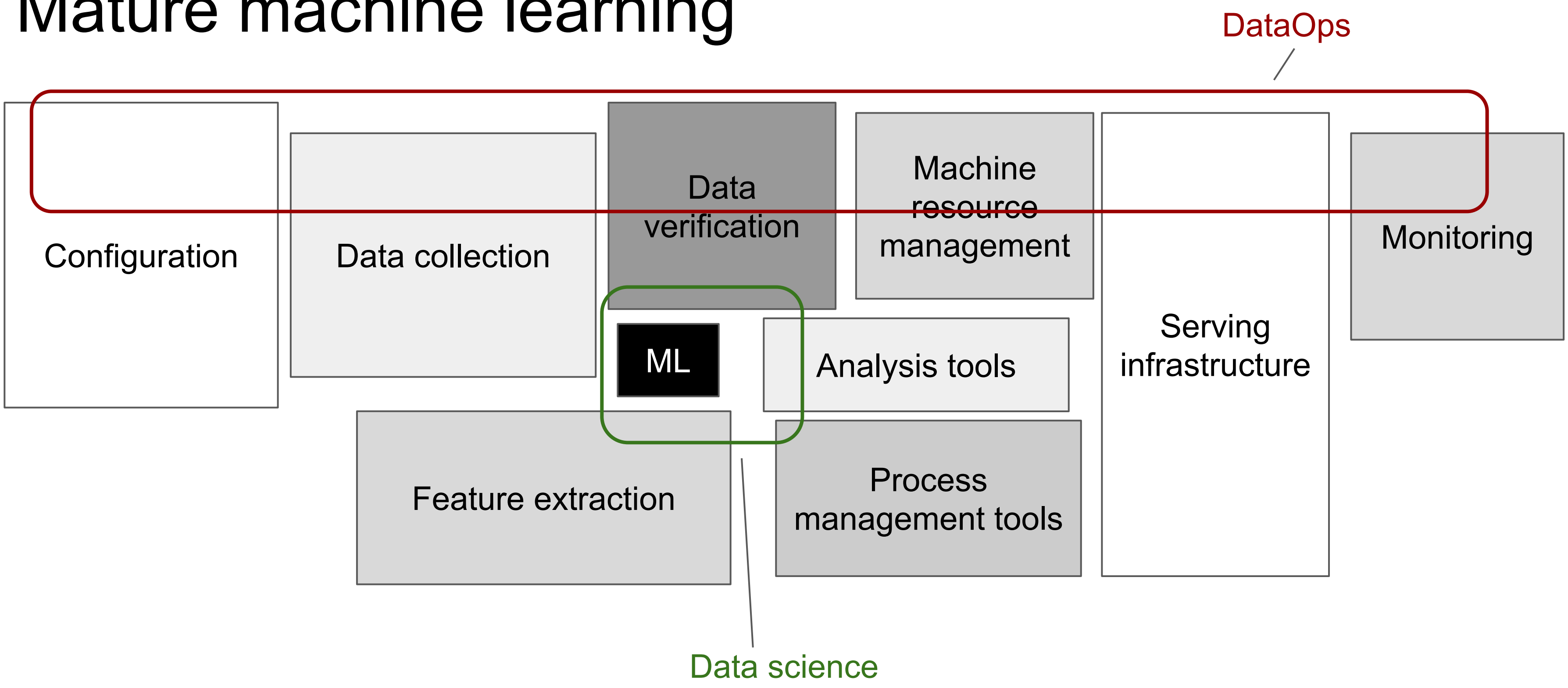
Machine learning products



Size = effort
Colour = code complexity

Credits: “Hidden Technical Debt in Machine Learning Systems”, Google, NIPS 2015

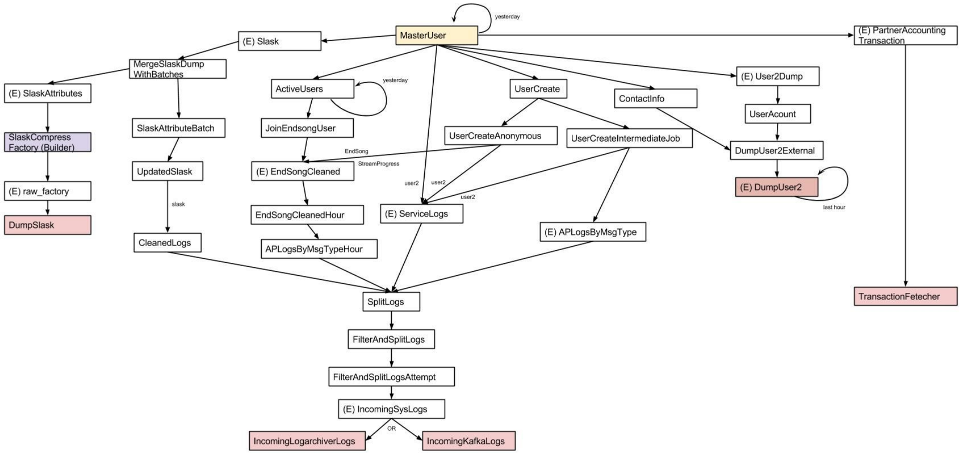
Mature machine learning



Complex business logic - MDM @ Spotify

- 10 pipelines like this
- Pipeline dev environment
- Pipeline continuous deployment infrastructure

One team of five engineers



Complex business logic - all Hadoop @ Spotify

- 2K unique jobs, 20K daily
- ~100 teams
- Almost all features involve lake
- Multiple processing tools
- Homogeneous infrastructure
  - Storage
  - Workflow management
- 2500 nodes, 50K cores, 100+TB mem, 100+PB store

- Migrating to Google cloud
- 750 BigQuery users
- 3M queries = 500 PB / month