Building Data Pipelines with Monitoring and Observability

Jiaqi Liu
@jiaqicodes
GOTO Chicago 2020

Jiaqi Liu
@jiaqicodes
GOTO Chicago 2020

Agenda

- Data Pipelines
- Challenges with Data Pipelines
- Designing Features:
  - Immutable Data
  - Dry Run Mode
  - Data Lineage
- Testing, Monitoring & Alerting

Data Pipelines

ETL Pipeline

- **Extract** data from a source, this could be scraping from a site, a large file, a realtime stream of data feeds.
- **Transform** the data - this could be joining the data with additional information for an enhanced data set, running through a machine learning model, or aggregating the data in some way.
- **Load** the data into a data warehouse or a user facing dashboard - wherever the end storage and display for data might be.

Batch

Periodic Process that reads data in bulk (typically from a filesystem or a database)

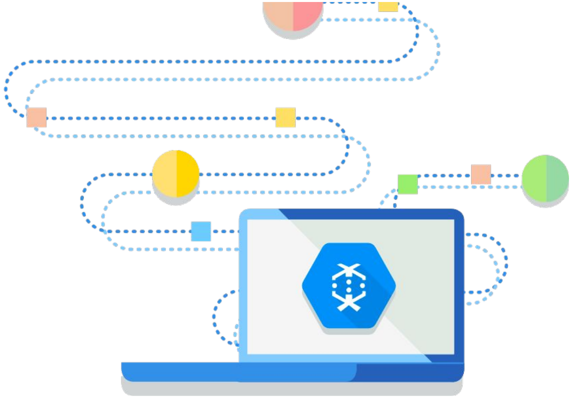
Stream

High throughput, low latency system that reads data from a stream or a queue

Luigi

AWS Data Pipeline

Apache Airflow



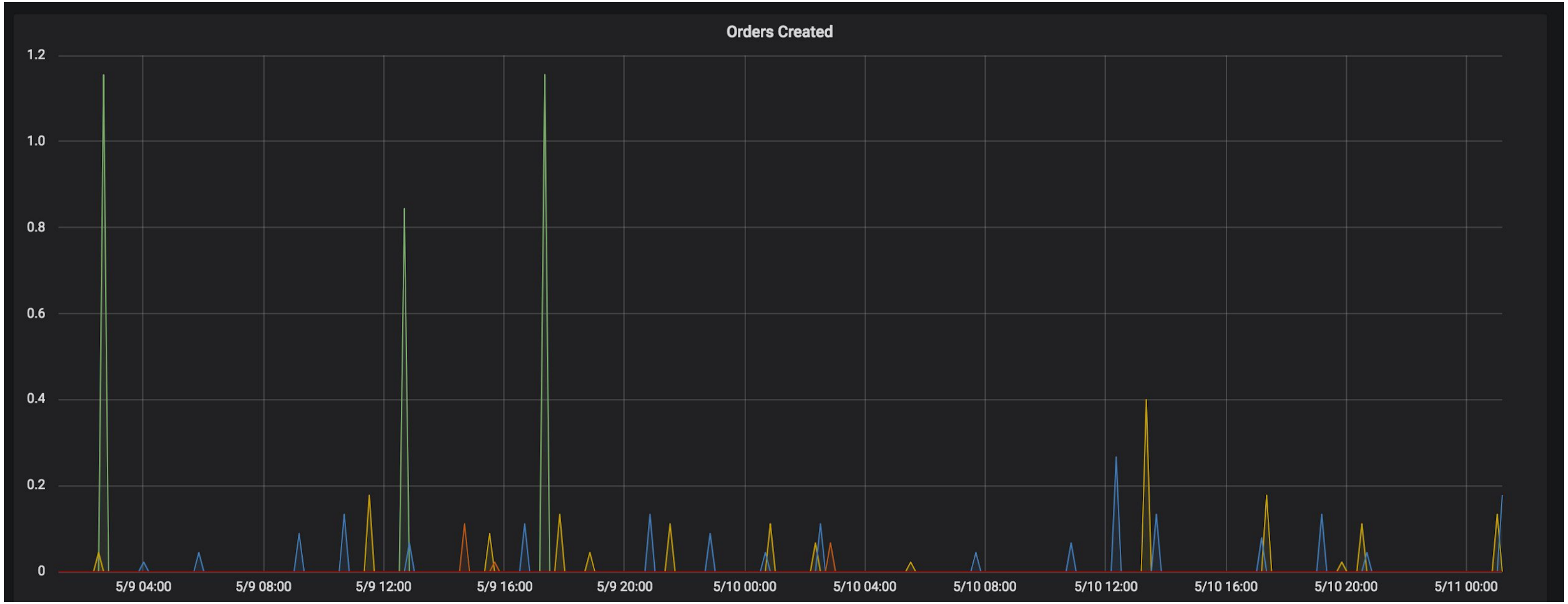
Google Dataflow

What Could Go Wrong?

Problems

- Batch Job is never scheduled
- Batch Job takes too long to run
- Data is malformed or corrupt
- Data is lost
- Stream is backed-up, Stream data is lost
- Non-deterministic models

Batch Jobs


Orders Created
1.2
1.0
0.8
0.6
0.4
0.2
0
5/9 04:00
5/9 08:00
5/9 12:00
5/9 16:00
5/9 20:00
5/10 00:00
5/10 04:00
5/10 08:00
5/10 12:00
5/10 16:00
5/10 20:00
5/11 00:00

Stream Jobs

Processing Throughput (Incoming Messages)
1.3 K
1.0 K
750
500
250
0
16:00
20:00
00:00
95% Processing Latency
12.5
10.0
7.5
5.0
2.5
0
16:00
20:00
00:00

Data Pipeline Concerns

Delayed Processing

Speed in Data Processing could be Core to Business

Data Integrity

Data is exposed or lost or malformed. A statistical model is producing highly inaccurate results

It's not enough to know that the pipeline is healthy, you also have to know that the data being processed is accurate.

Build data pipelines that support interpretability and observability

Interpretability

Not just understand what a model predicted but also **why.**

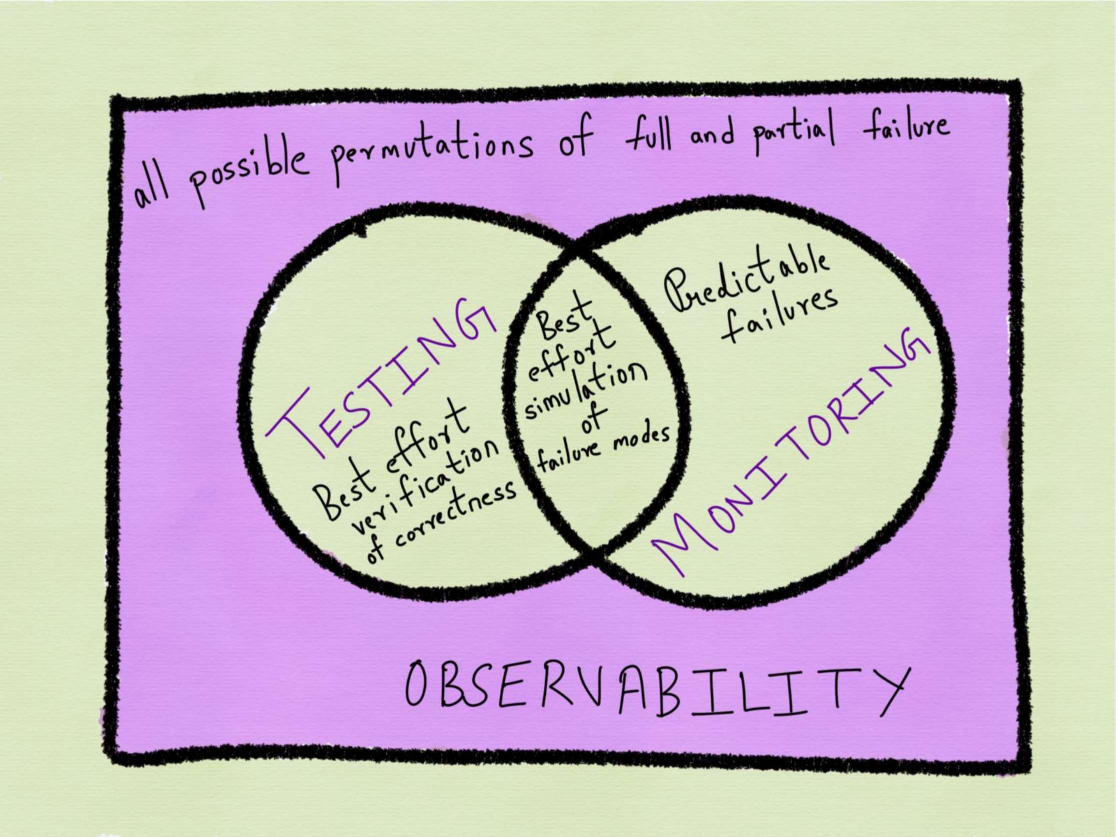
Allows for **debugging** and auditing machine learning models.

Interpretability

- Fairness
- Privacy
- Reliability
- Causality
- Trust

Observability

- SRE term
- Can’t catch for things you don’t know
- Focus on debugging



Cindy Sridharan - https://medium.com/@copyconstruct/monitoring-and-observability-8417d1952e1c

Pipeline Features

Build feature to support Interpretability and observability

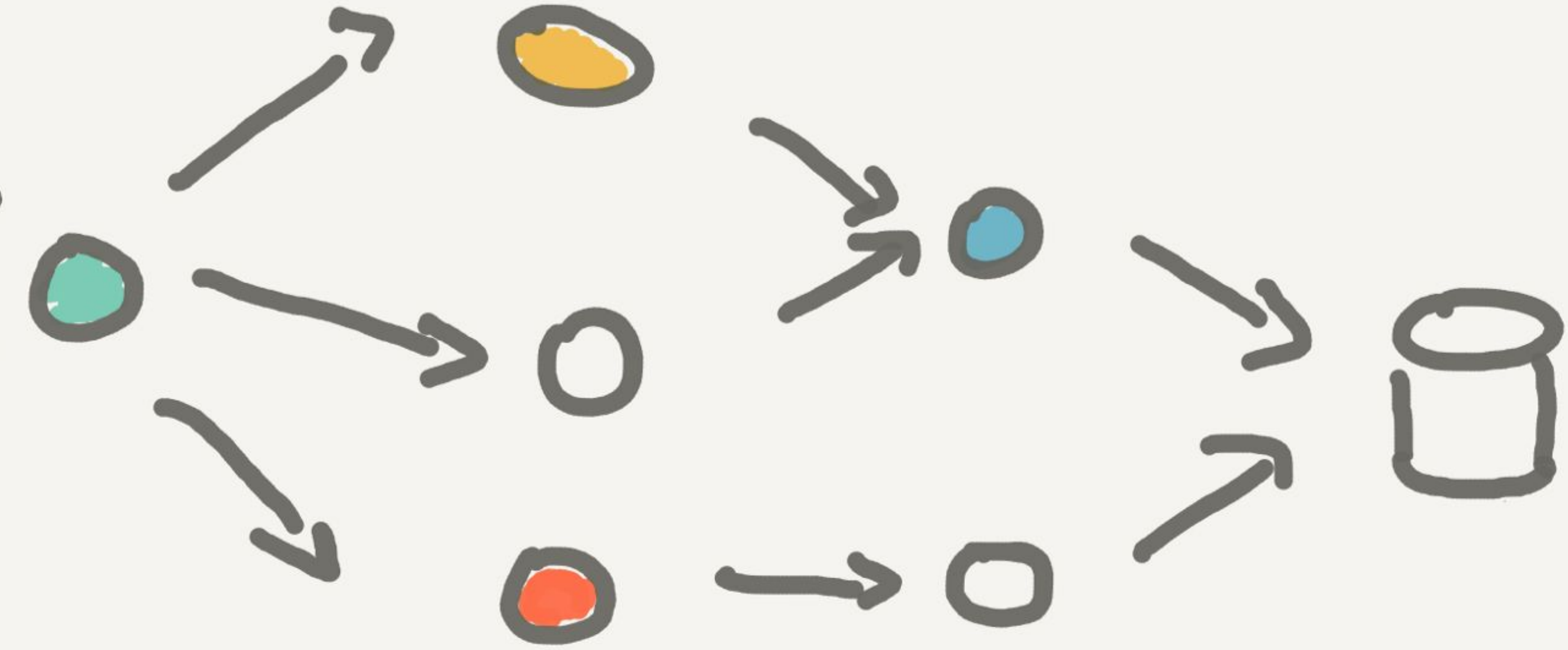
Features to Include

- Immutable Data
- Data Lineage
- Having a Test Run Feature

Immutable Data → Reproducible Outcomes

transaction_id	created_dt	account_id	transaction_type	amount
1	2019-01-15 01:00:24.032473	12345	credit	100
2	2019-01-30 10:01:07.683552	12345	debit	-5
3	2019-02-01 11:01:28.153952	12345	debit	-10

Data Lineage → **Diagnostics**



Tag Records With Metadata (version of code, source of data)

Log to a distributed tracer (use consistent unique identifiers to track)

Test Run → Validate Assumptions

What assumptions did you make about your data?

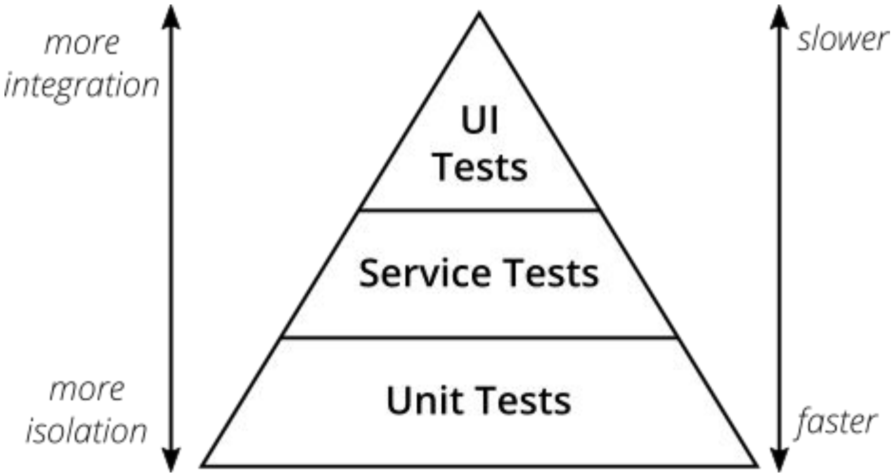
Schema

```
schema = {
    'bio': string
    'name': string
    'talk': {
        'description': string
        'link': string
        'session_type': string, // enum
        'title': string
        },
    'twitter': string}
}
```

Ability to test output of data transformation before committing to database

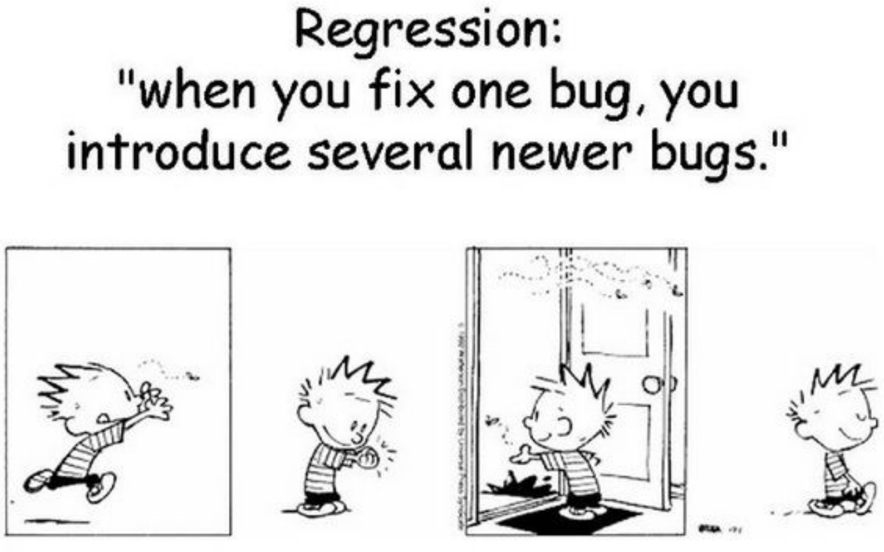
Testing, Monitoring, Alerting

Test Pyramid



https://martinfowler.com/articles/practical-test-pyramid.html

Regression Tests



https://www.ibeta.com/regression-testing-nutshell/

Champion/Challenger Model

93%

Model A Precision

95%

Model B Precision

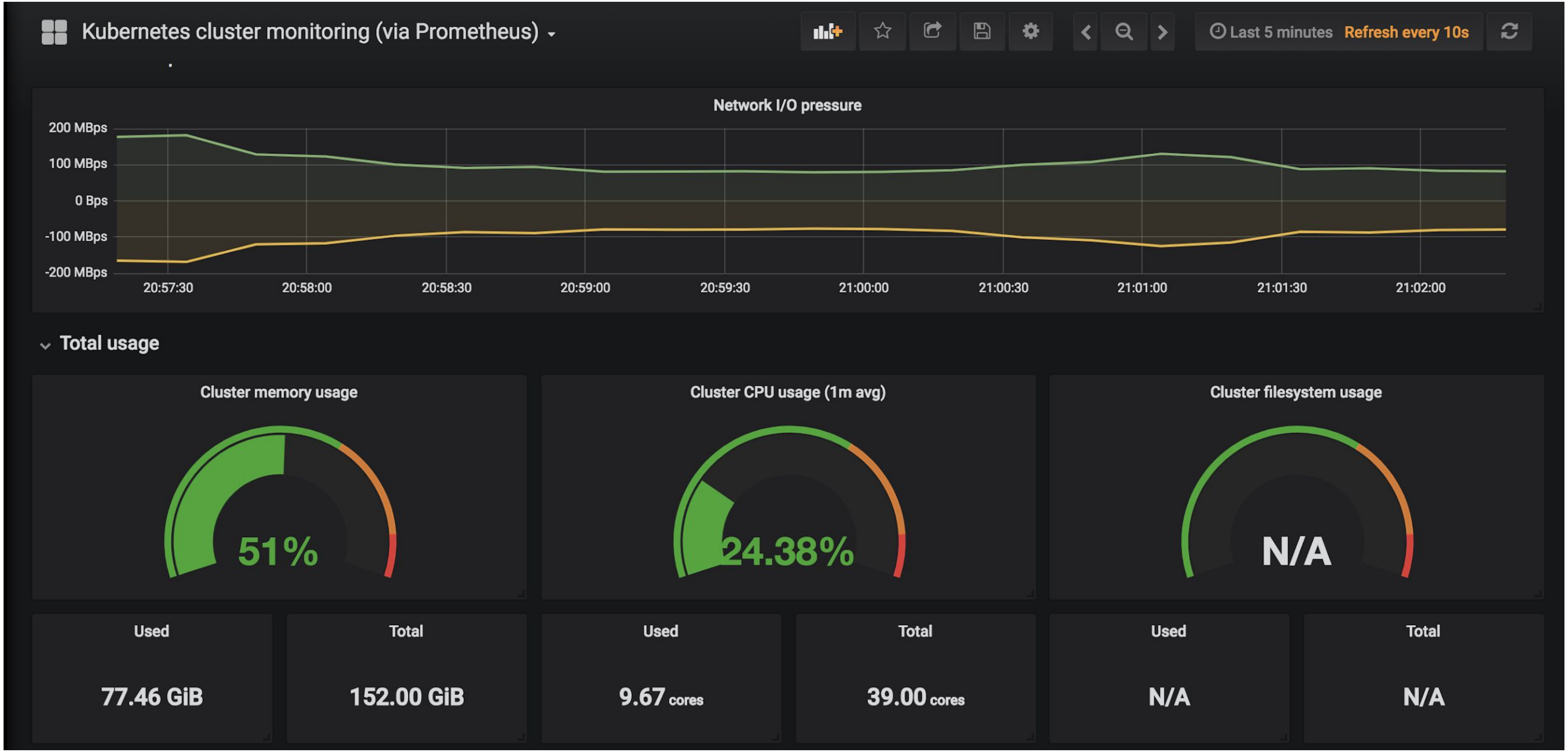
Monitoring & Testing

	Web Service	Data Pipeline
Health Check	Have some kind of health check endpoint and check that when you ping `/healthcheck` you get a 200 status code	Check that a job has succeeded
Integration Test	`POST` to one endpoint and expect to get the correct data from a corresponding `GET` endpoint	Verify some fake data made its way through the data transformation *This can be hard to replicate if there's no easy way to feed fake data into the data pipeline
Latency	Measure the average response time of an API	Measure time it takes for data pipeline to complete

Monitoring Tools

Prometheus

Grafana


Kubernetes cluster monitoring (via Prometheus)
Last 5 minutes
Refresh every 10s
Network I/O pressure
200 MBps
100 MBps
0 Bps
-100 MBps
-200 MBps
20:57:30
20:58:00
20:58:30
20:59:00
20:59:30
21:00:00
21:00:30
21:01:00
21:01:30
21:02:00
Total usage
Cluster memory usage
51%
Cluster CPU usage (1m avg)
24.38%
Cluster filesystem usage
N/A
Used
77.46 GiB
Total
152.00 GiB
Used
9.67 cores
Total
39.00 cores
Used
N/A
Total
N/A

Time Series Metrics

Metrics to be scraped by prometheus:

```
job_last_success_unixtime = prometheus_client.Gauge('job_last_success_unixtime',
    'Time of last successful batch job')
job_duration_seconds = prometheus_client.Summary('job_duration_seconds',
    'Duration of batch job run in seconds')
```

Metrics are calculated at the end of the pipeline as such:

```
with job_duration_seconds.time():
  run_pipeline()
  time_now = int(time.time())
  job_last_success_unixtime.set(time_now)
```

Alerting

```
ALERT BatchJobFailed
    IF time() - job_last_success_unixtime > (3 * 60 * 60)
    FOR 15m
    LABELS { severity="high", owner="data-platform" }
    ANNOTATIONS {
      summary = "The batch job has failed.",
      description = "The {{ $labels.job }} has not succeeded in over 3 hours",
```

Set a threshold that works for you.
Establish a baseline and go from there.

Page on symptoms not root causes.
Create trail of causes for diagnostics

Data Lineage, Immutable Data, Test Run -> Ease of development when working with evolving data

Monitoring & Alerting -> Overall Pipeline Observable

Questions?