

# Himalayan Peaks of Testing Data Pipelines

Ksenia Tomak, Dodo Engineering  
Pasha Finkelshteyn, JetBrains

# Who we are

# What is Big Data

# Who are DEs?

# What is a pipeline?

# Who needs pipelines

# QA of pipeline

QA ?= QC

# QA of pipeline

QA ≠ QC

QA is about processes and not only about software quality.

# Pyramid of testing. Unit



# Typical pipeline



# Unit testing of pipeline

What may we test here?

A pipeline should transform data correctly!

*Correctness is a business term*

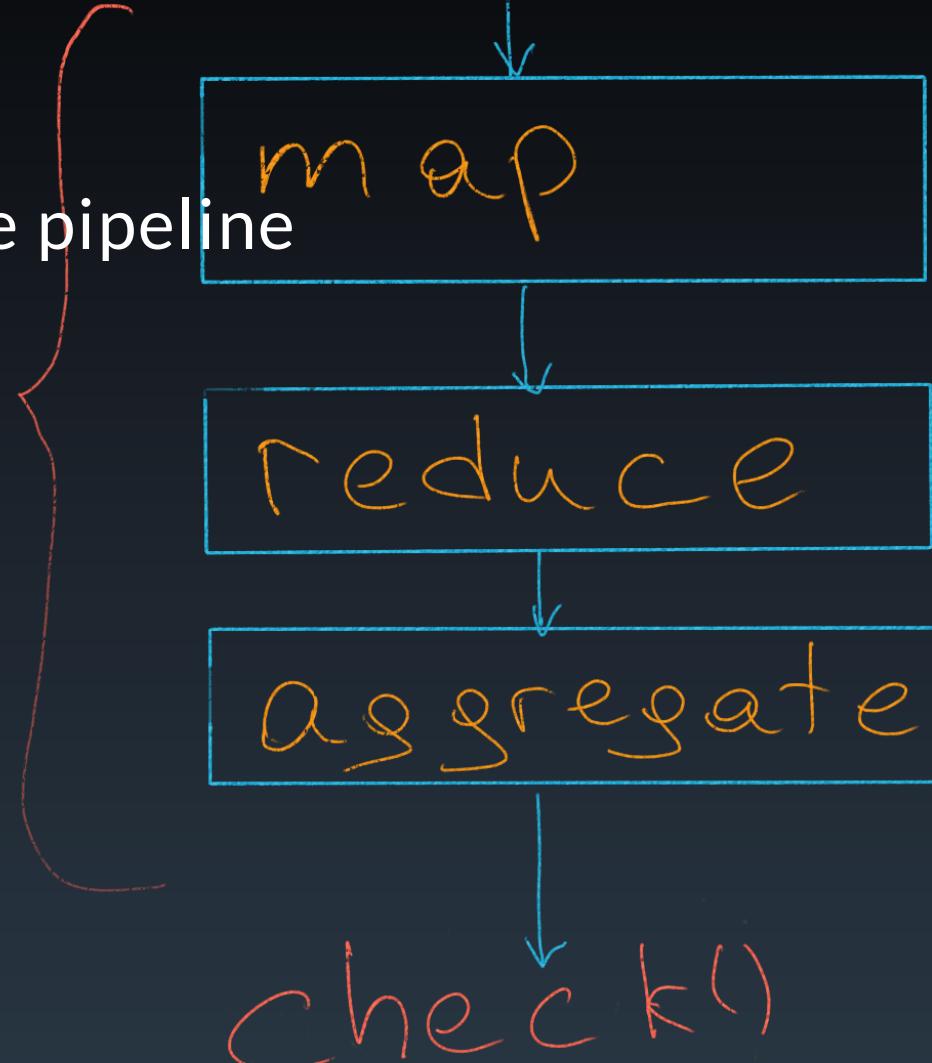
fake data

# Let's paste fakes!

Fake/mock input data

Reference data at the end of the pipeline

Separate  
Function

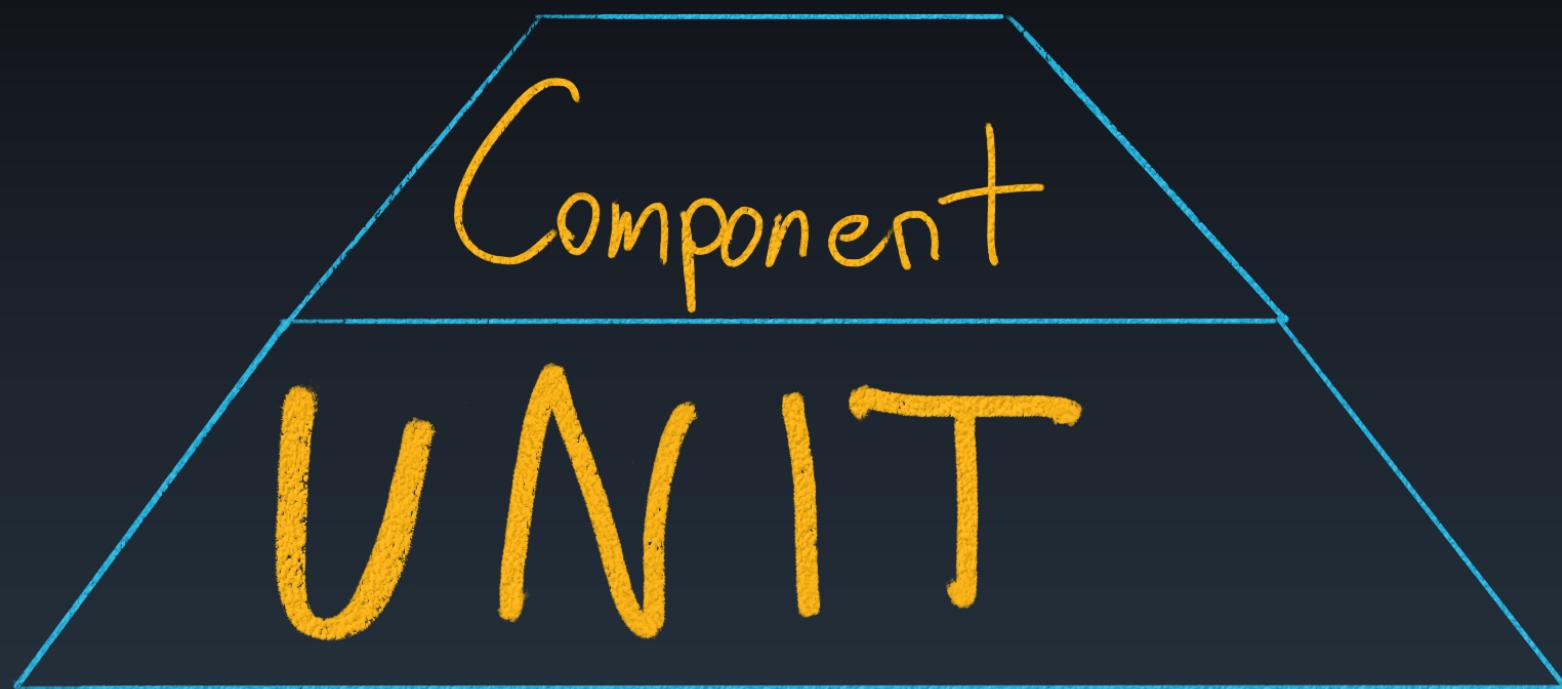


# Tools

[holdenk/spark-testing-base](#) ← Tools to run tests

[MrPowers/spark-daria](#) ← tools to easily create test data

# Component testing

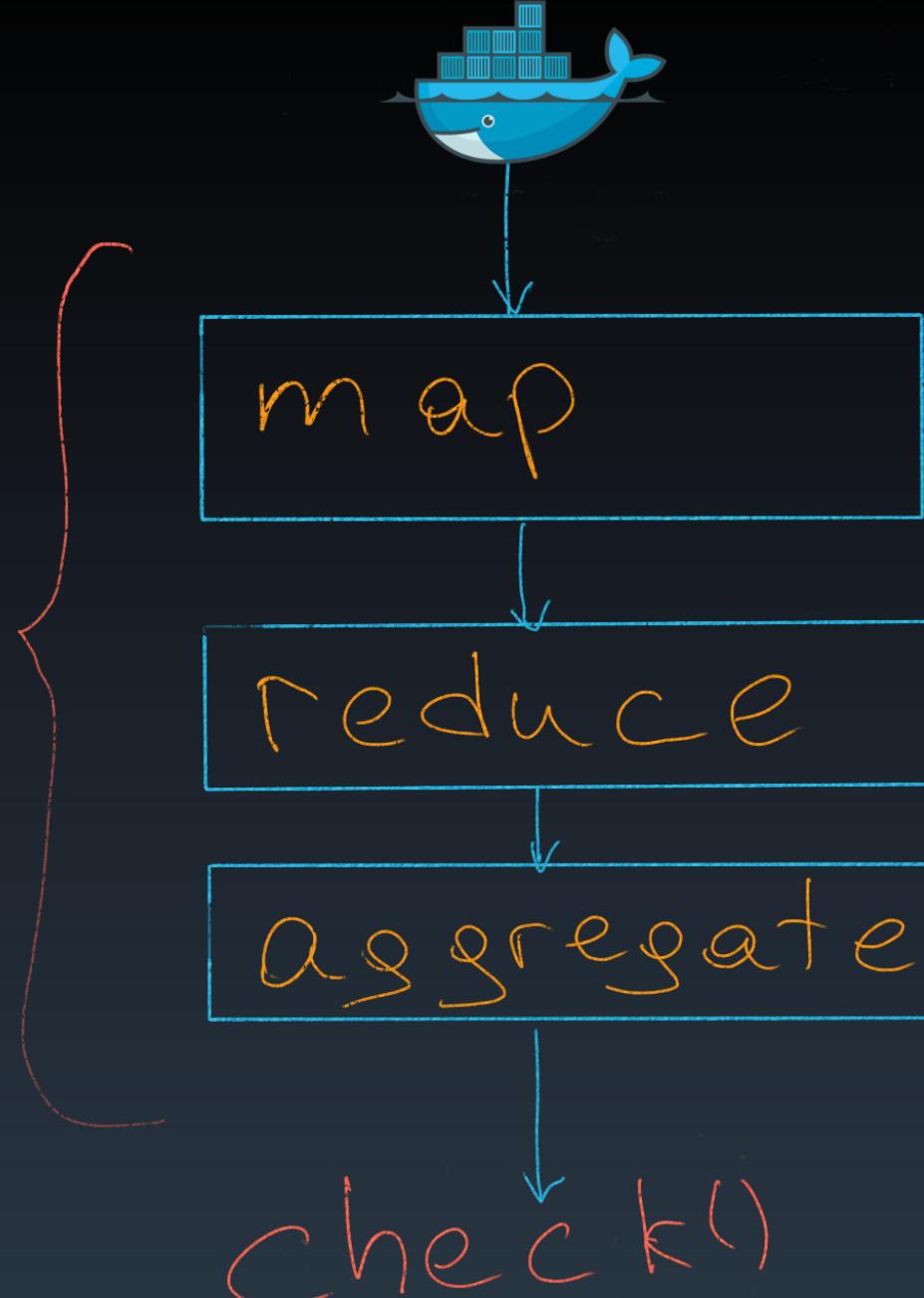




TEST CONTAINERS

# TestContainers

Separate  
Function



# TestContainers

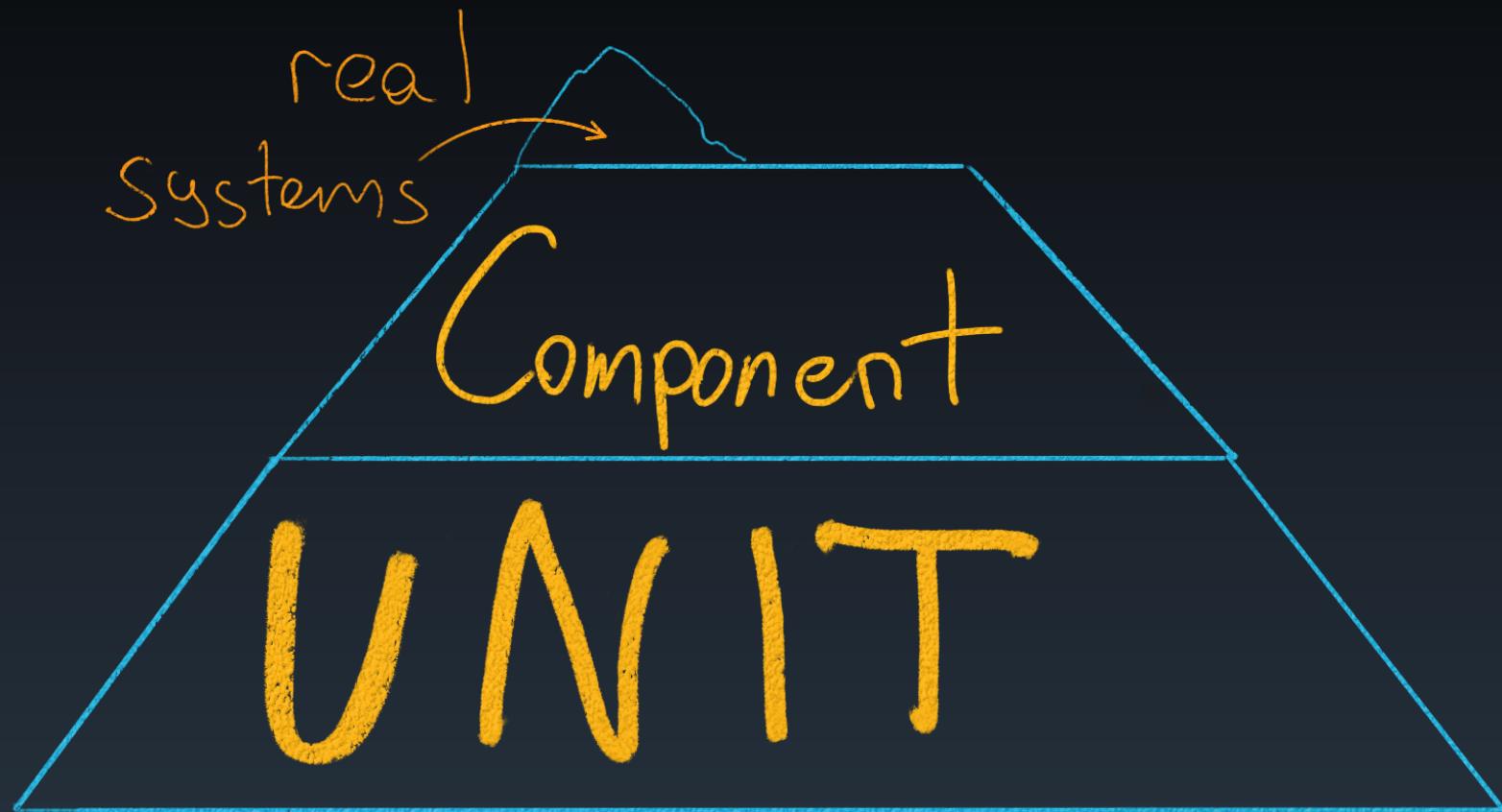
Supported languages:

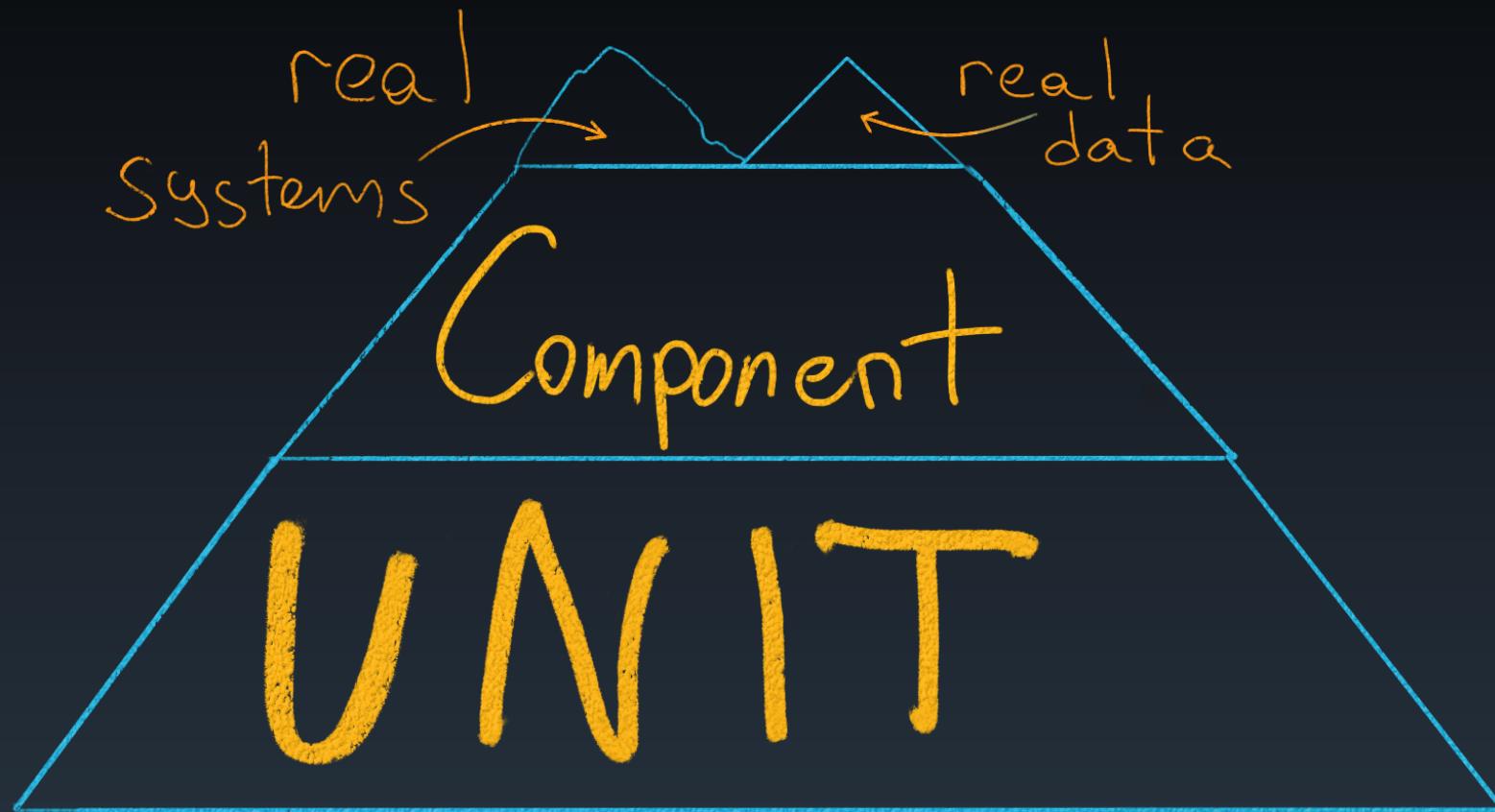
- Java (and compatibles: Scala, Kotlin, etc.)
- Python
- Go
- Node.js
- Rust
- .NET

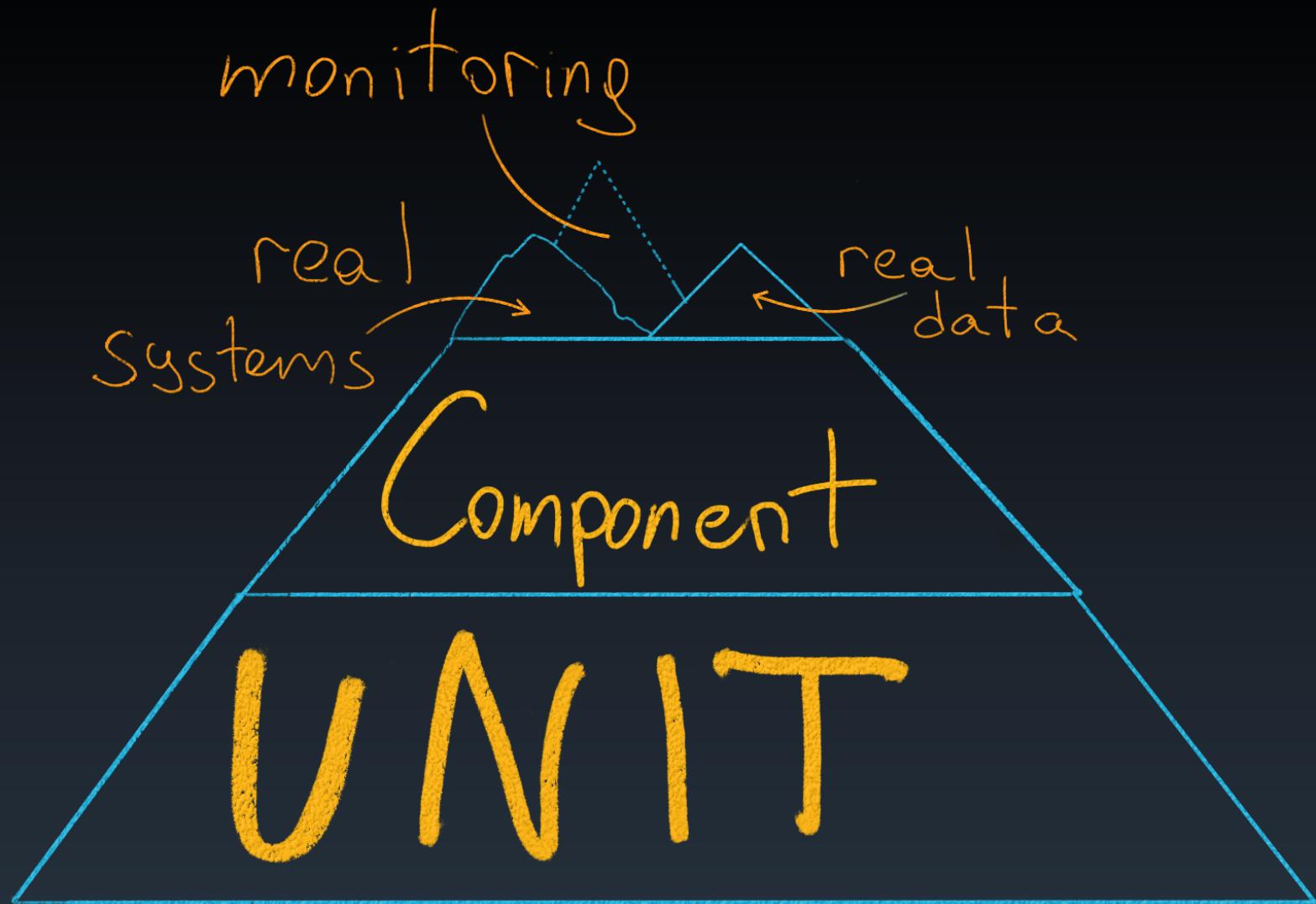
# Test Containers

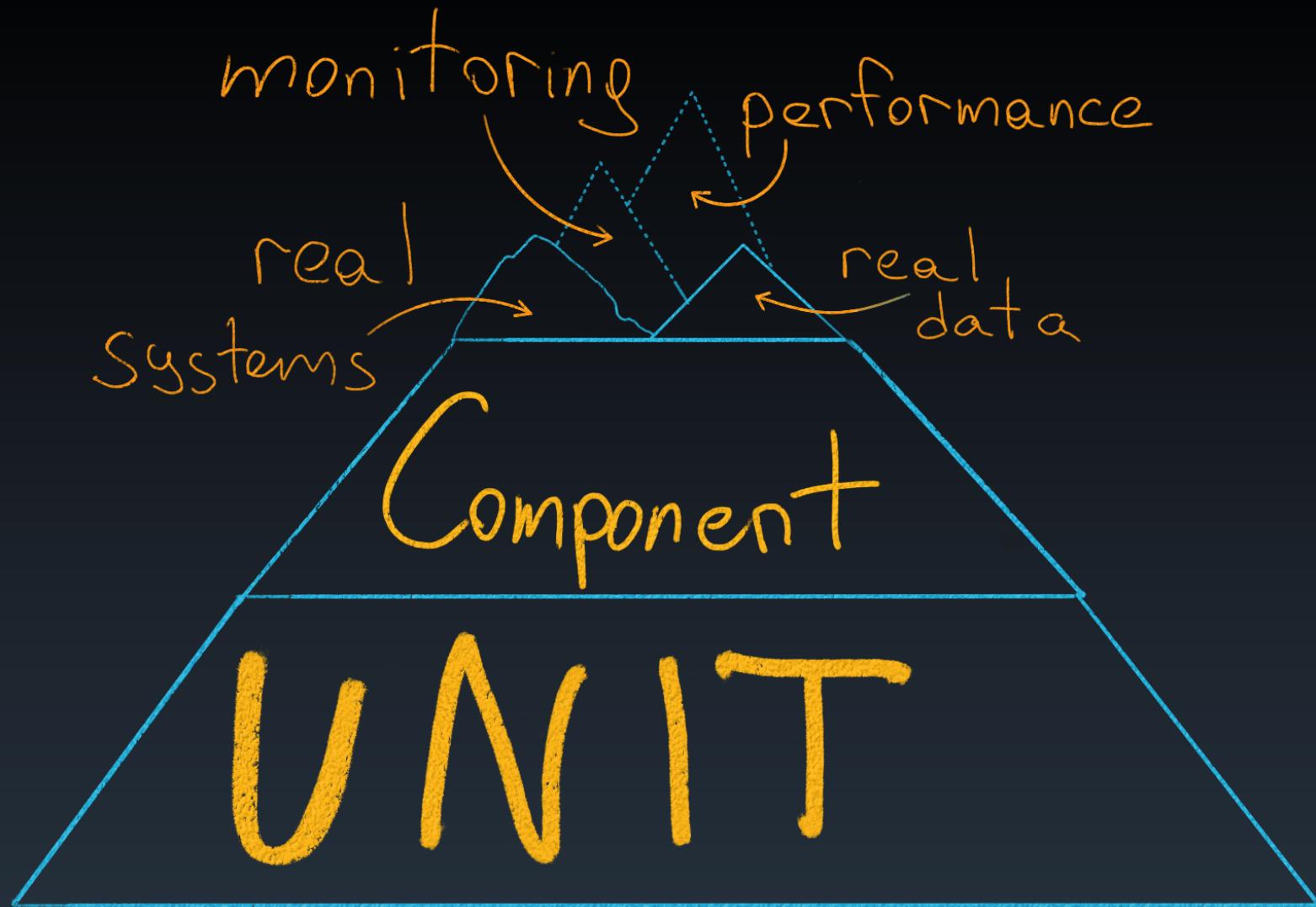
```
import sqlalchemy
from testcontainers.mysql import MySqlContainer

with MySqlContainer('mysql:5.7.17') as mysql:
    engine = sqlalchemy.create_engine(mysql.get_connection_url())
    version, = engine.execute("select version()").fetchone()
    print(version) # 5.7.17
```









# Real systems

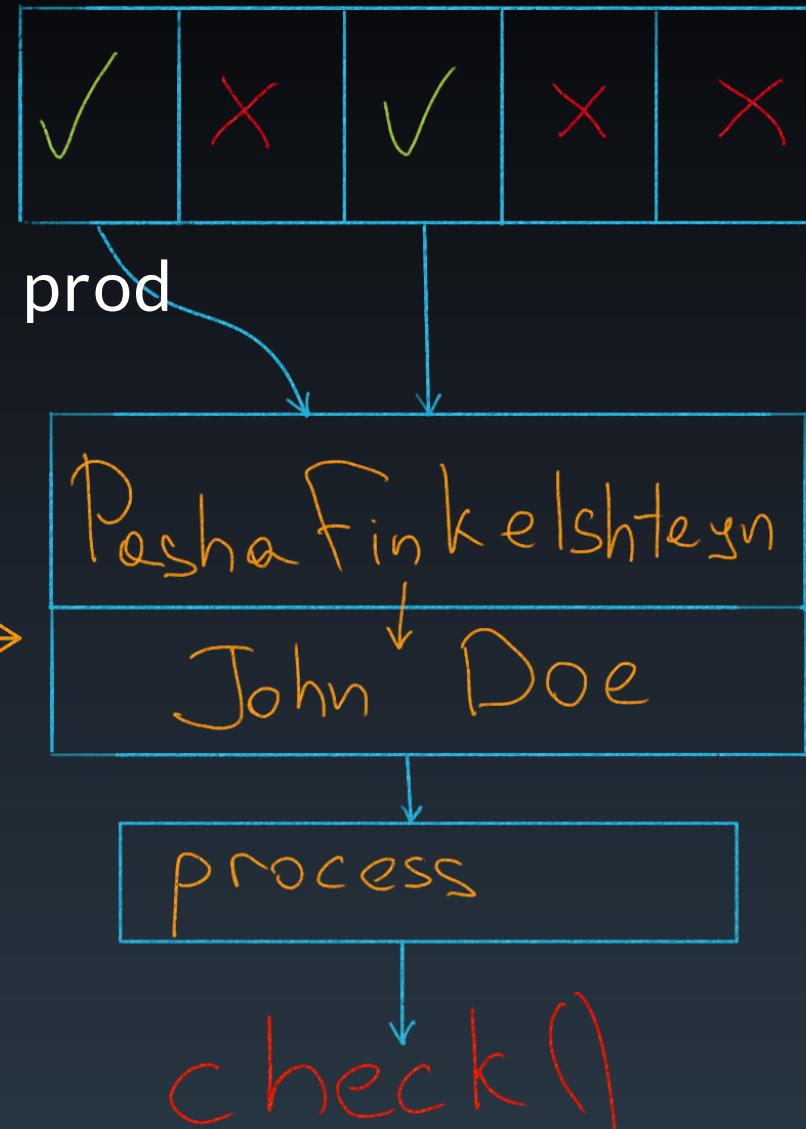
Why are component tests not enough?

- vendor lock tools (DB, processing, etc.)
- external error handling

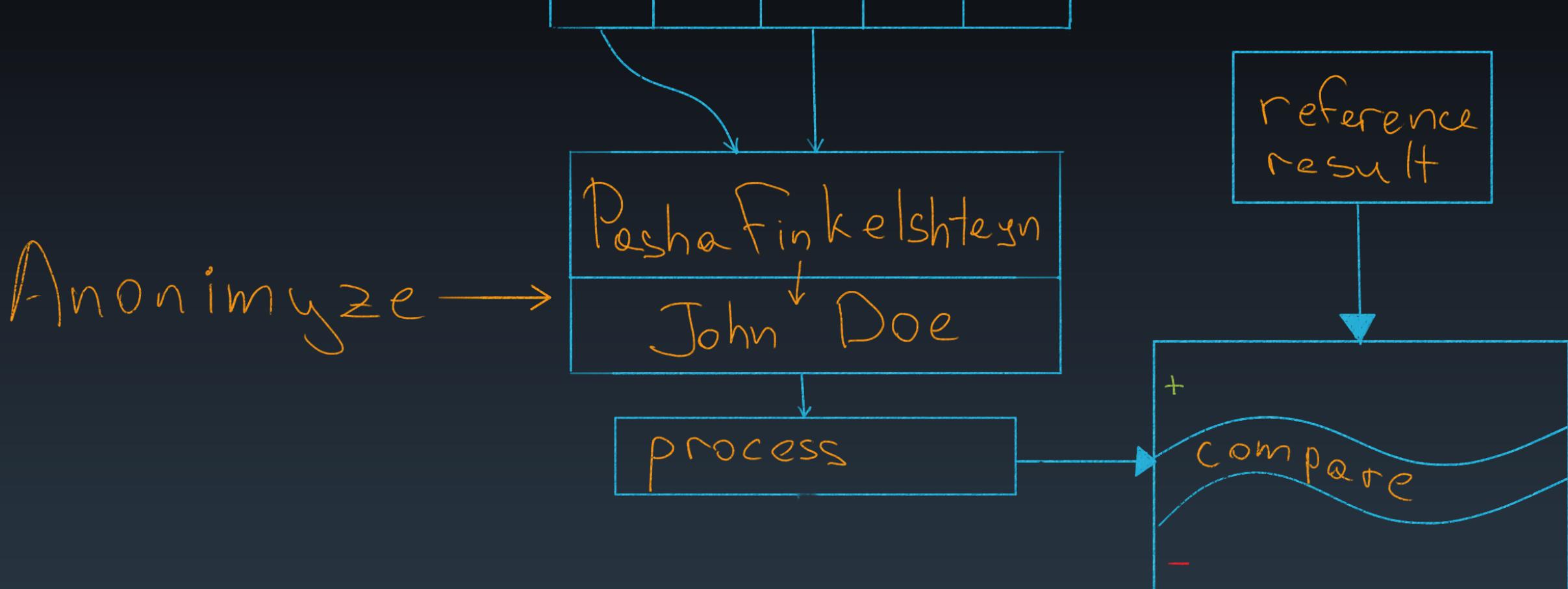
# Real data

- get data samples from prod
- anonymize it

Anonymize →



# Compare to reference



# Real data expectations

Test:

- ✓ no data
- ✓ valid data
- ✗ invalid data
- ✗ illegal data format

- Property-based testing

# Real data expectations. Tools:

- [great expectations](#),
- [Deequ](#)

# Monitoring

## Why?

- The only REAL testing is production
- Data tends to change over time

# Monitoring

What?

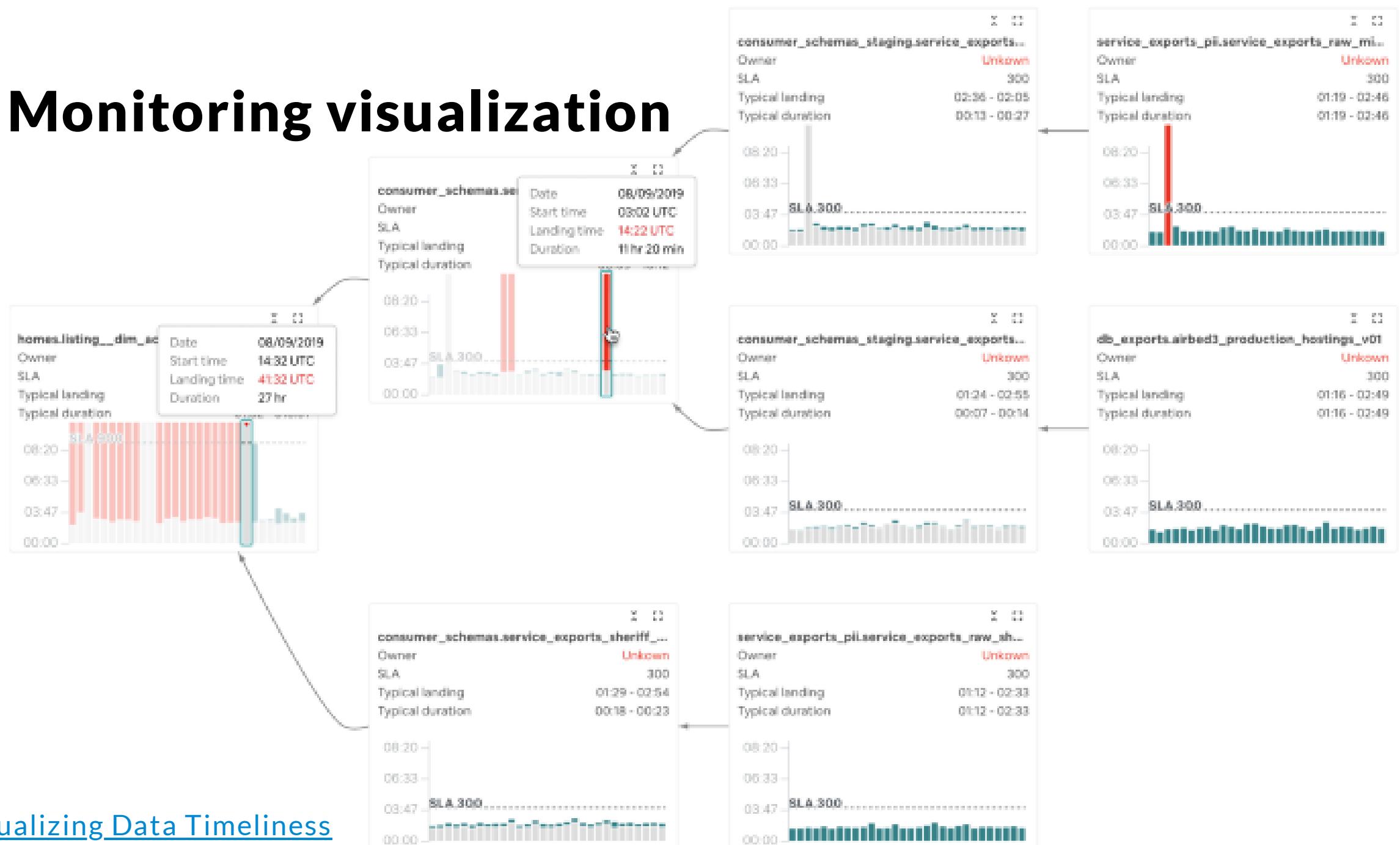
- data volumes
- counters
- time
- dead letter queue monitoring

# Monitoring

How?

- use Listeners
- use data aggregations

# Monitoring visualization

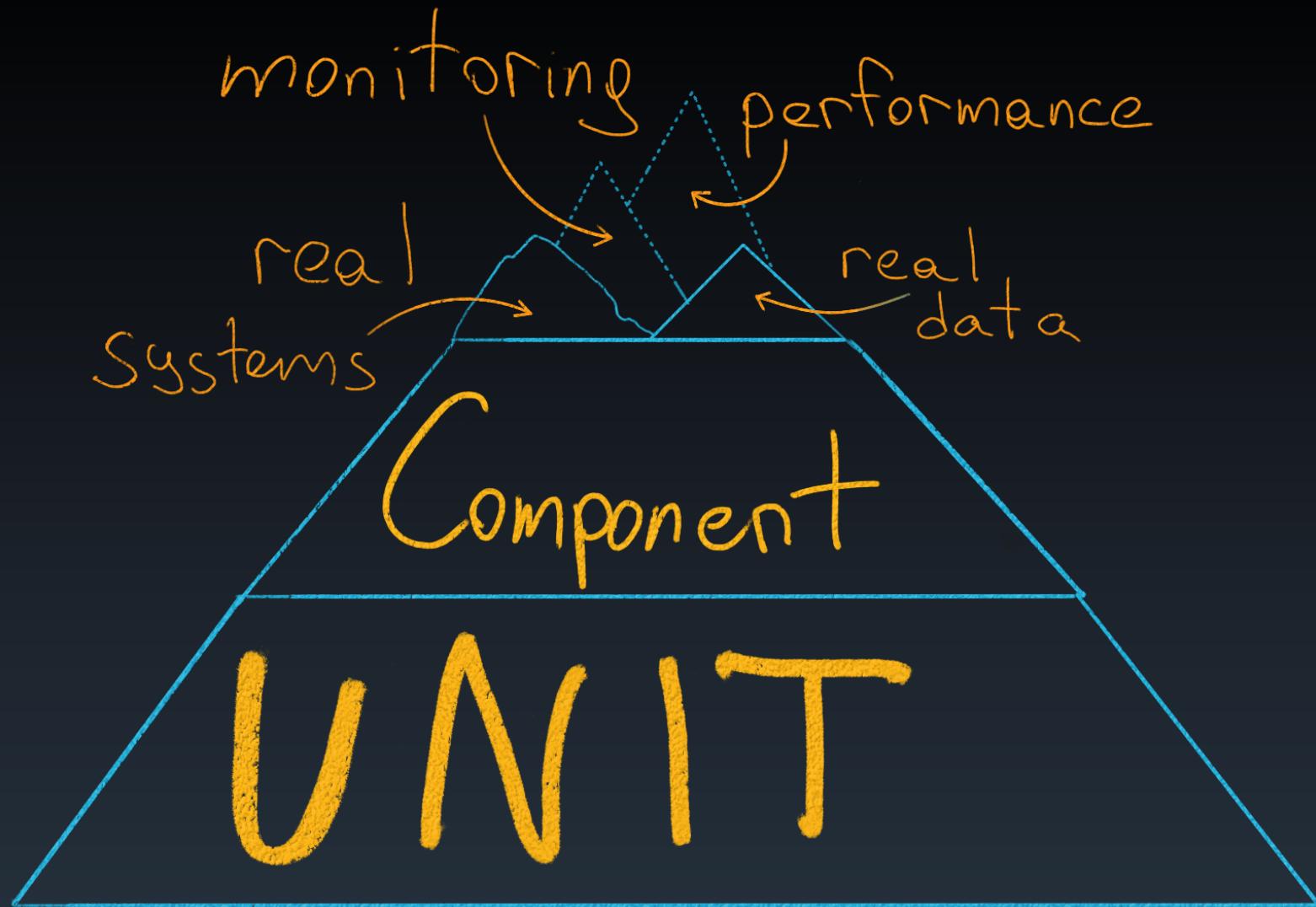


# End-to-End tests

Compare with reports, old DWH

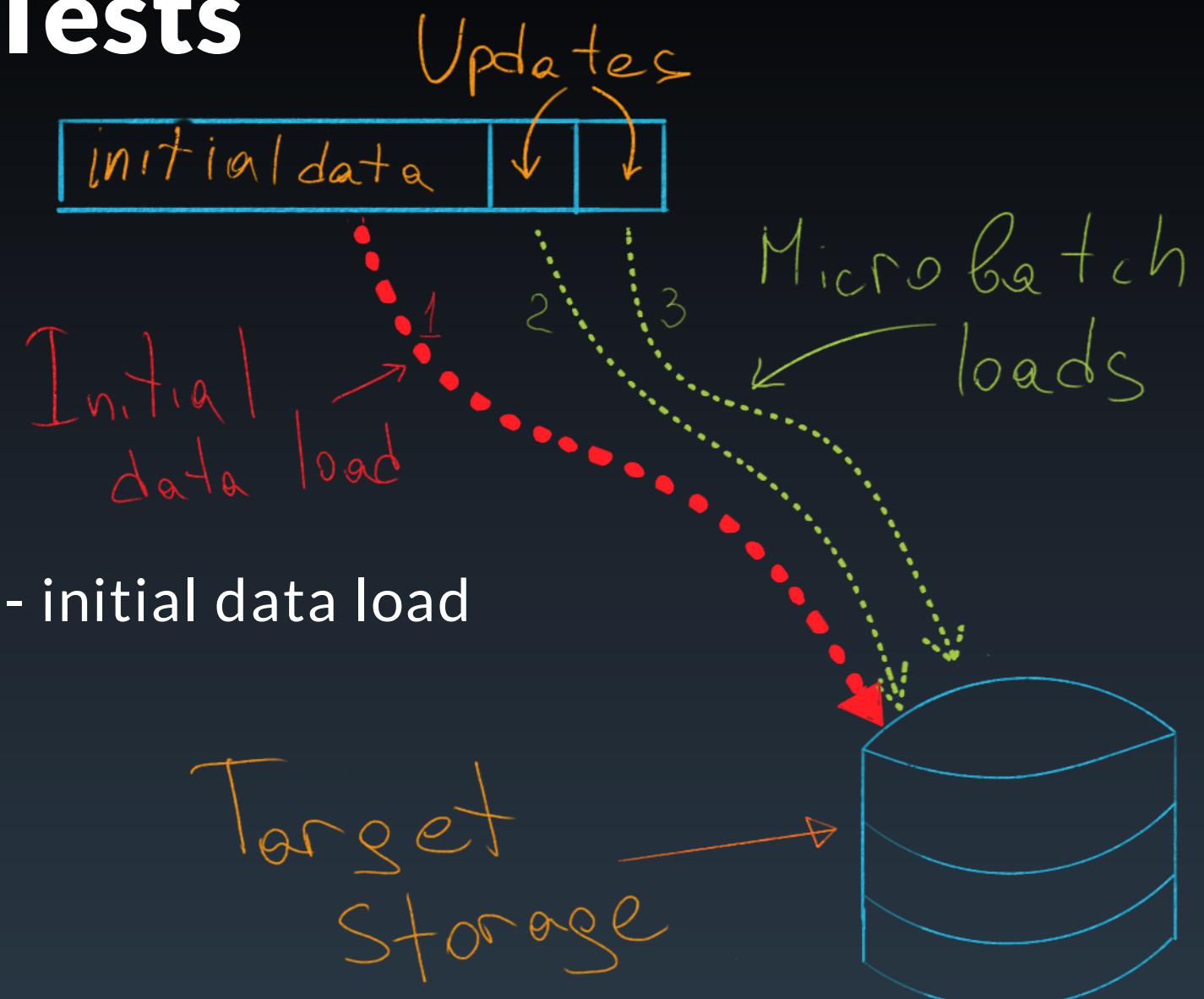
Multiple dimensions:

- data
- data latency
- performance, scalability



# Performance Tests

- Start with SLA
- Best performance test - initial data load



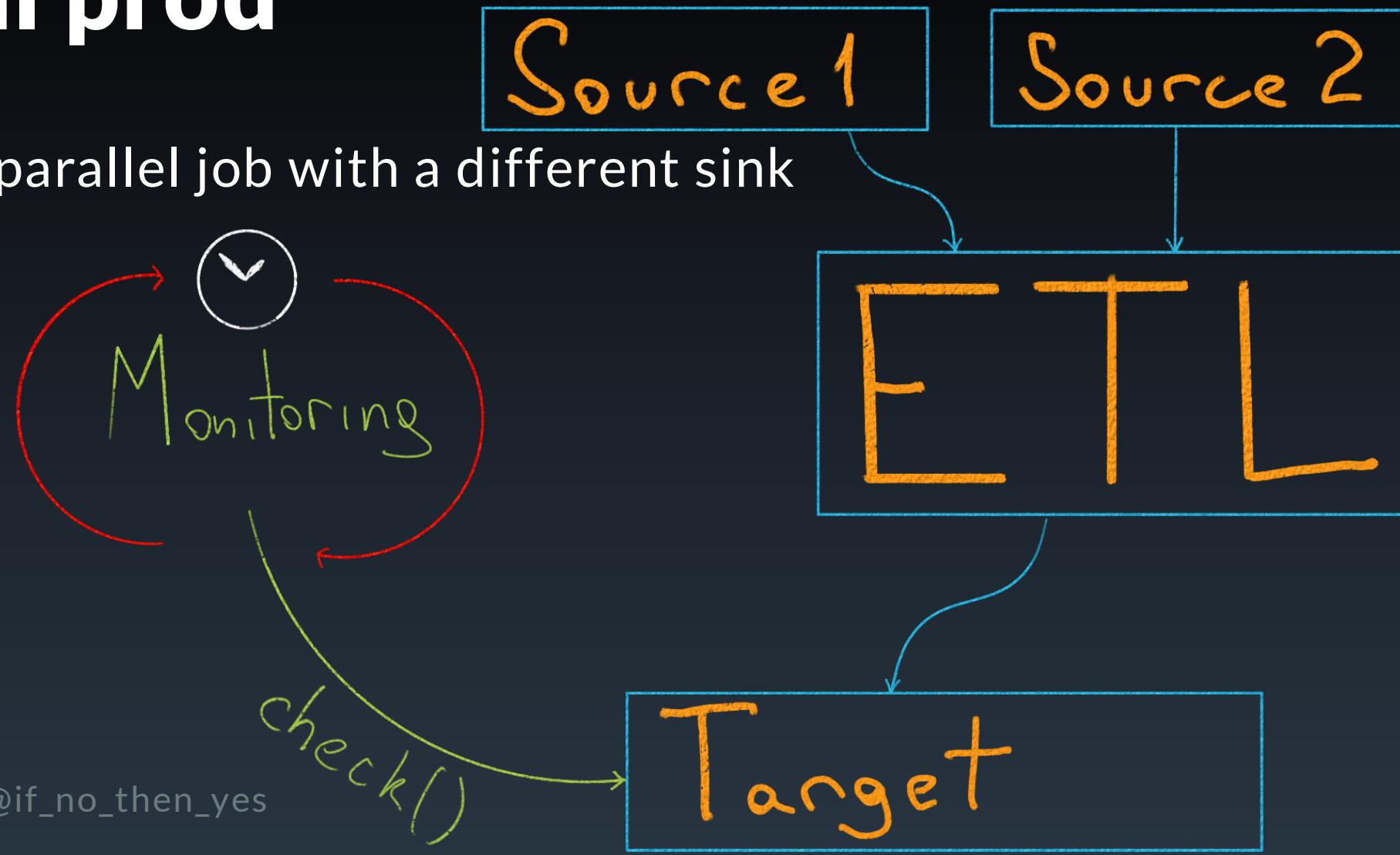
# How to apply?

# Real data, real system

Deploy full data backup on stage env, anonymize it 

# Real prod

Run a parallel job with a different sink



Using production data for testing in a post GDPR world