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

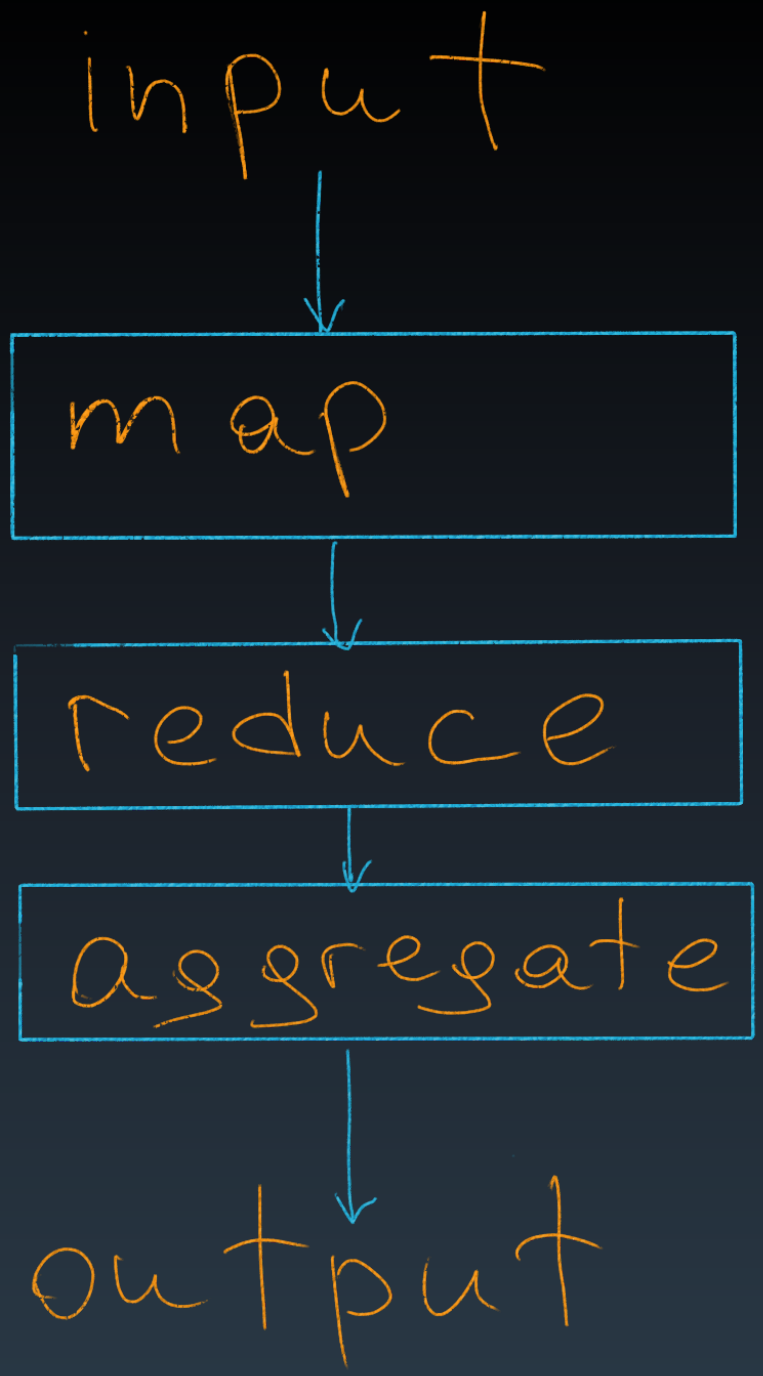
QA of pipeline

QA ≠ QC

QA is about processes and not only about software quality.

Pyramid of testing. Unit

Typical pipeline



Unit testing of pipeline

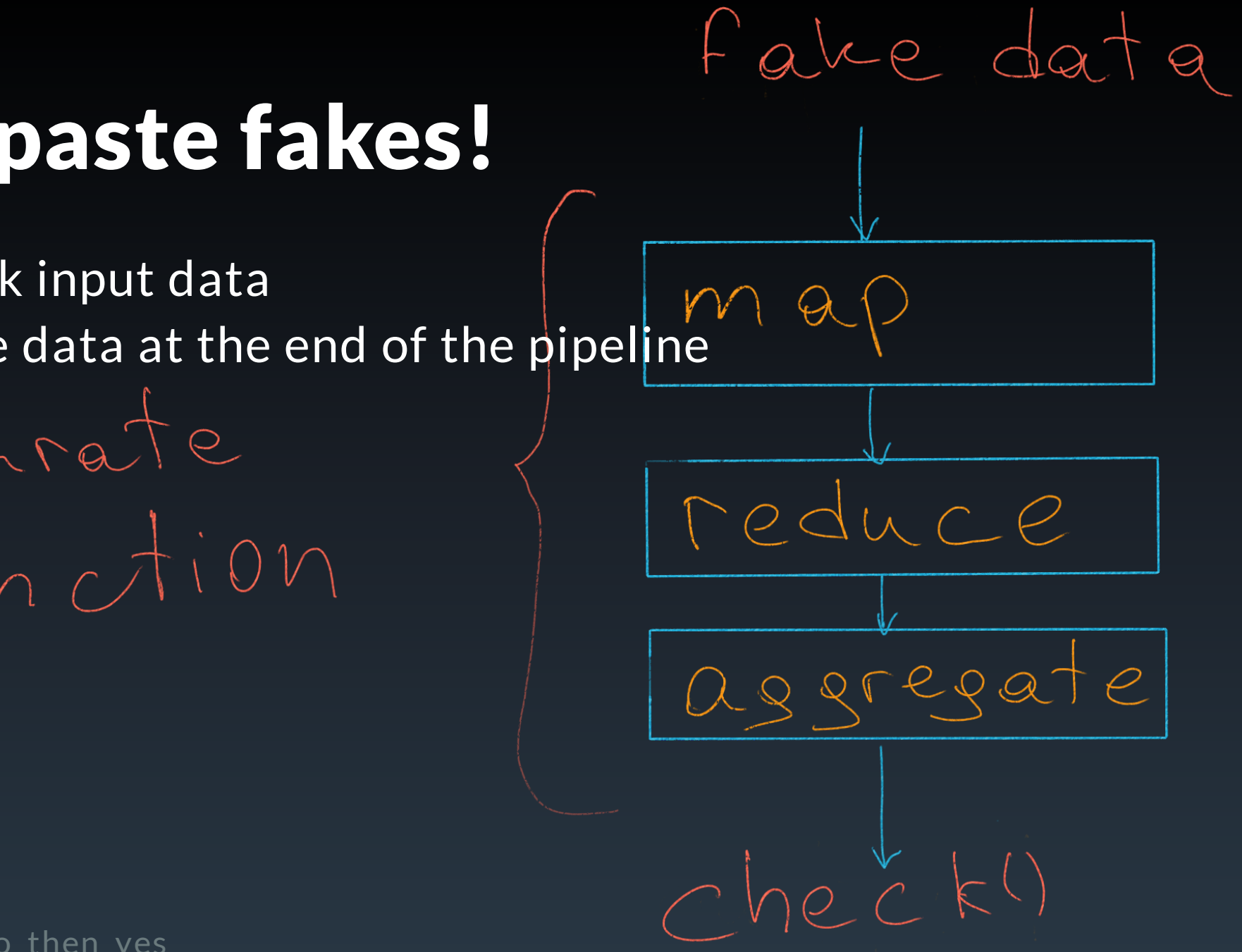
What may we test here?

A pipeline should transform data correctly!

Correctness is a business term

Let's paste fakes!

Fake/mock input data
Reference data at the end of the pipeline



Tools

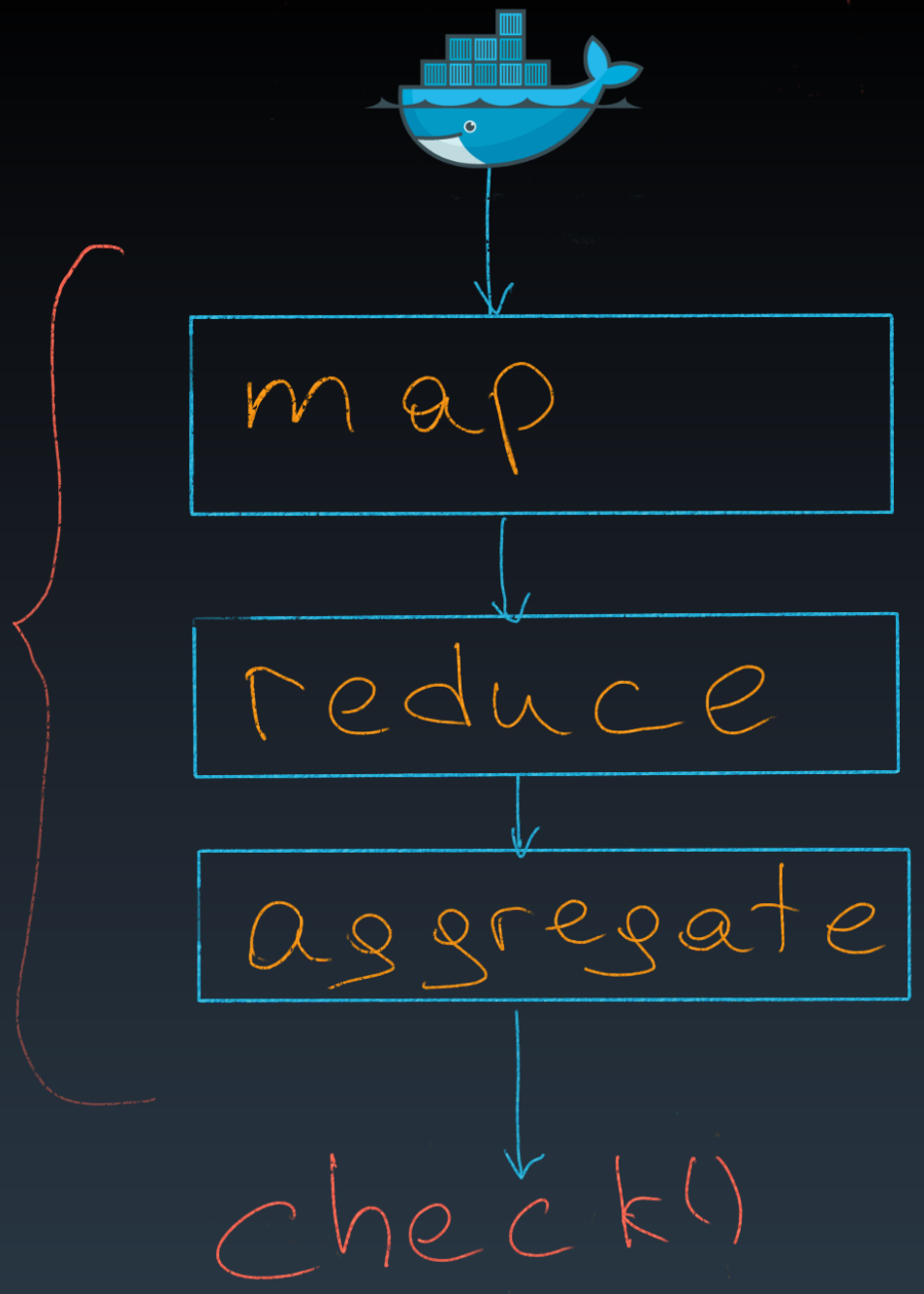
holdenk/spark-testing-base ← Tools to run tests
MrPowers/spark-daria ← tools to easily create test data

Component testing

TESTCONTAINERS

TestContainers

Separate Function



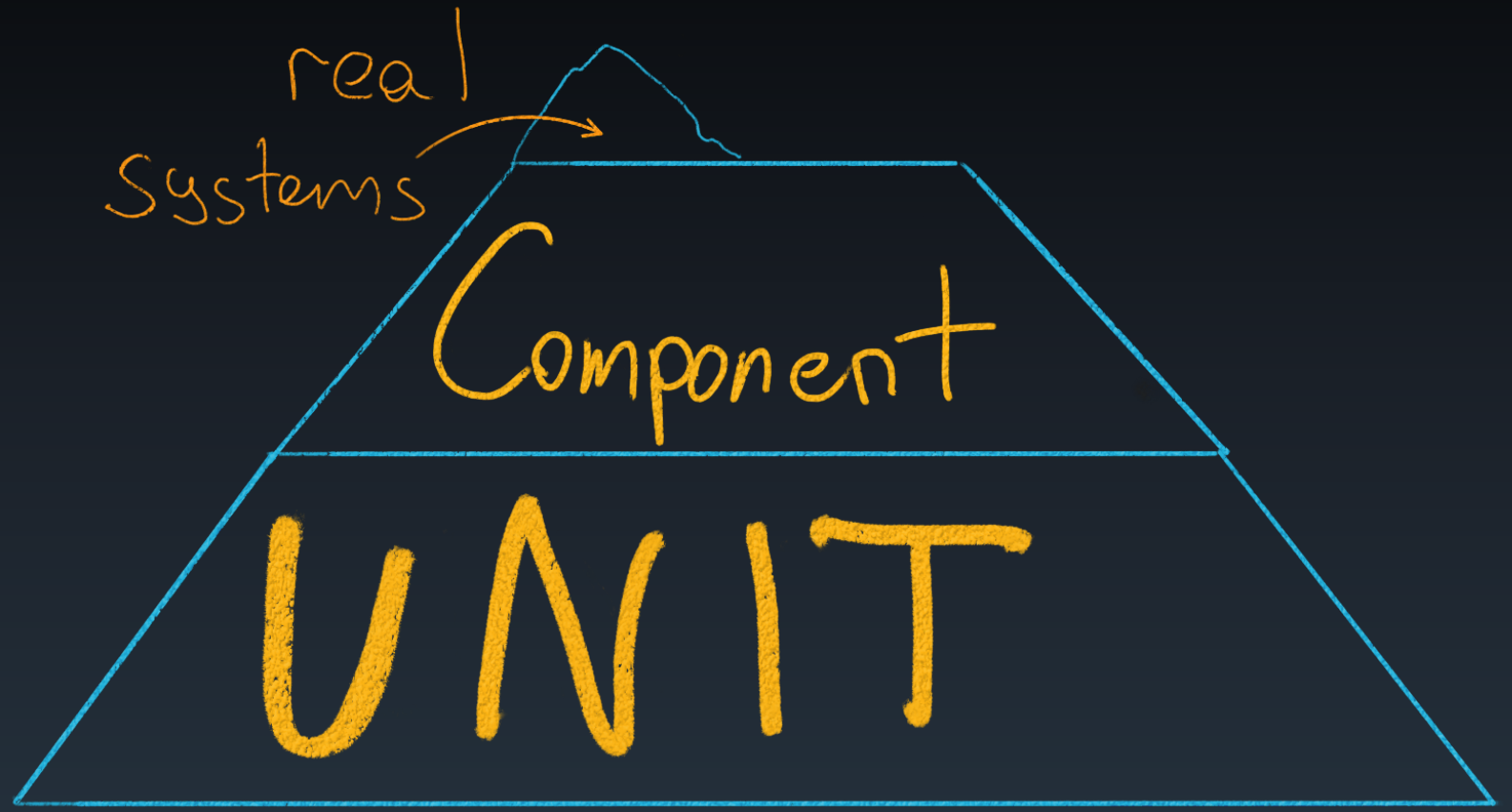
TestContainers

Supported languages:

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

Test Containers

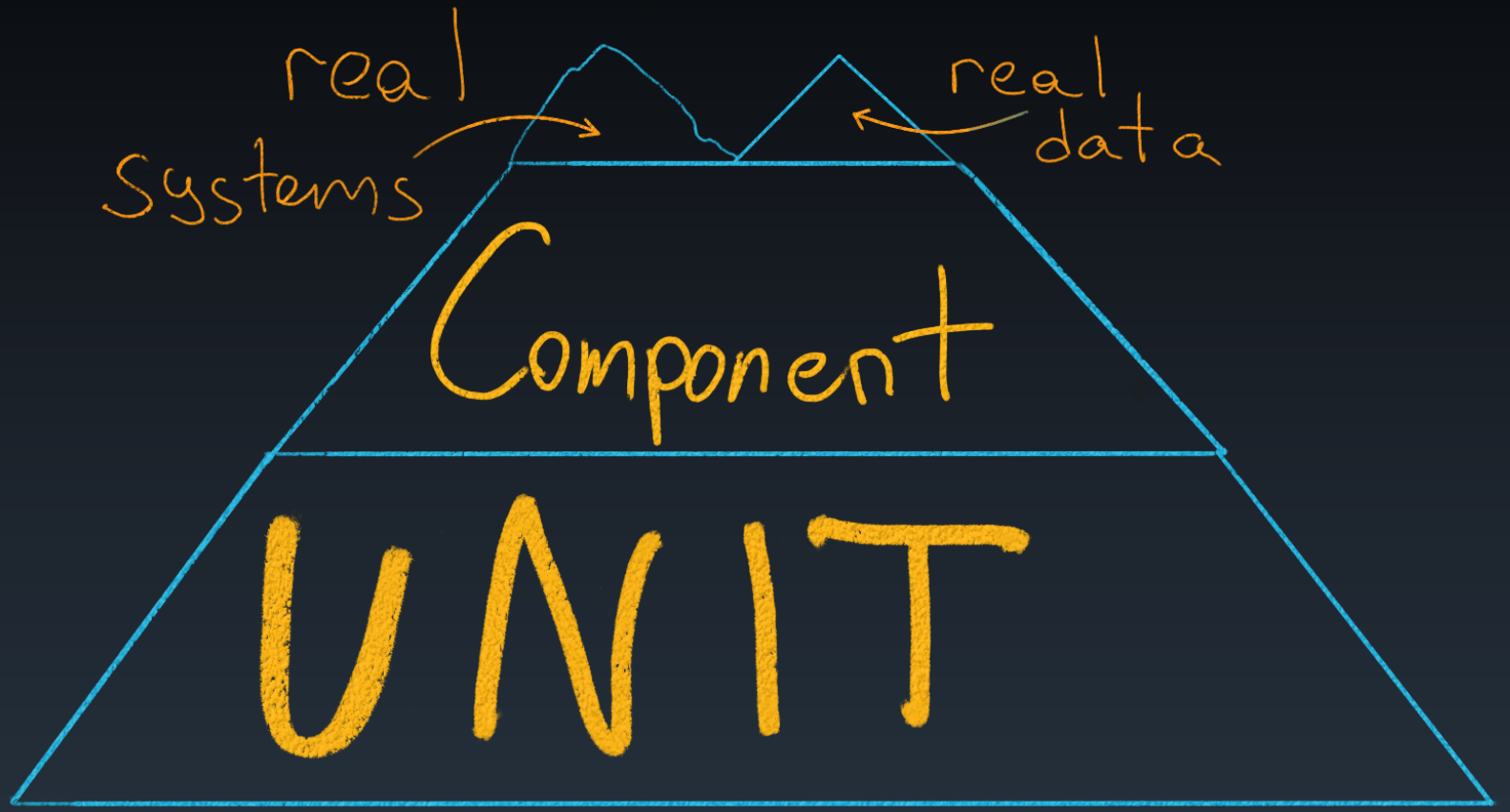
```
import sqlalchemy
from testcontainers.mysql import MySqlContainer

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
Component
UNIT

Real systems

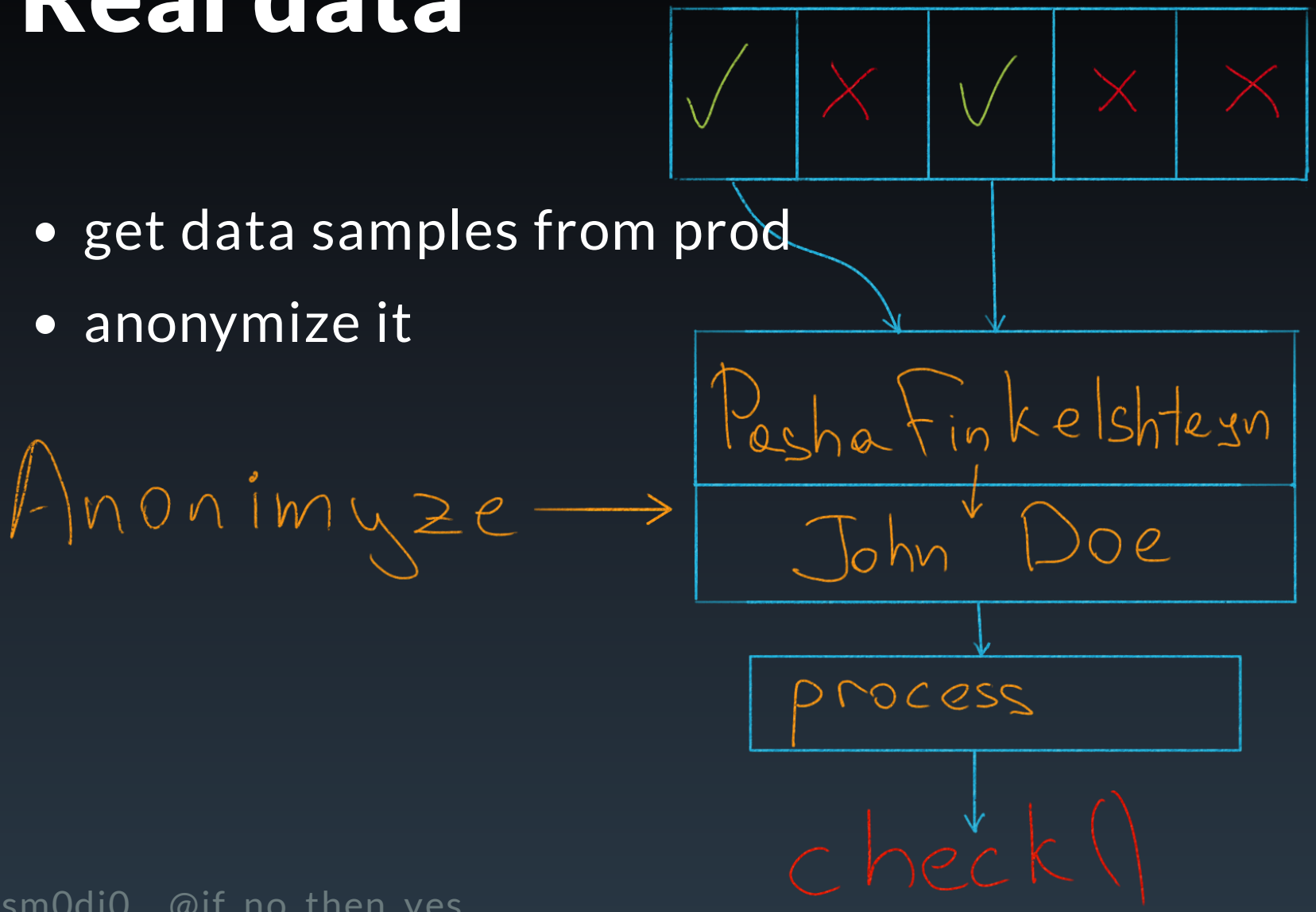
Why are component tests not enough?

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

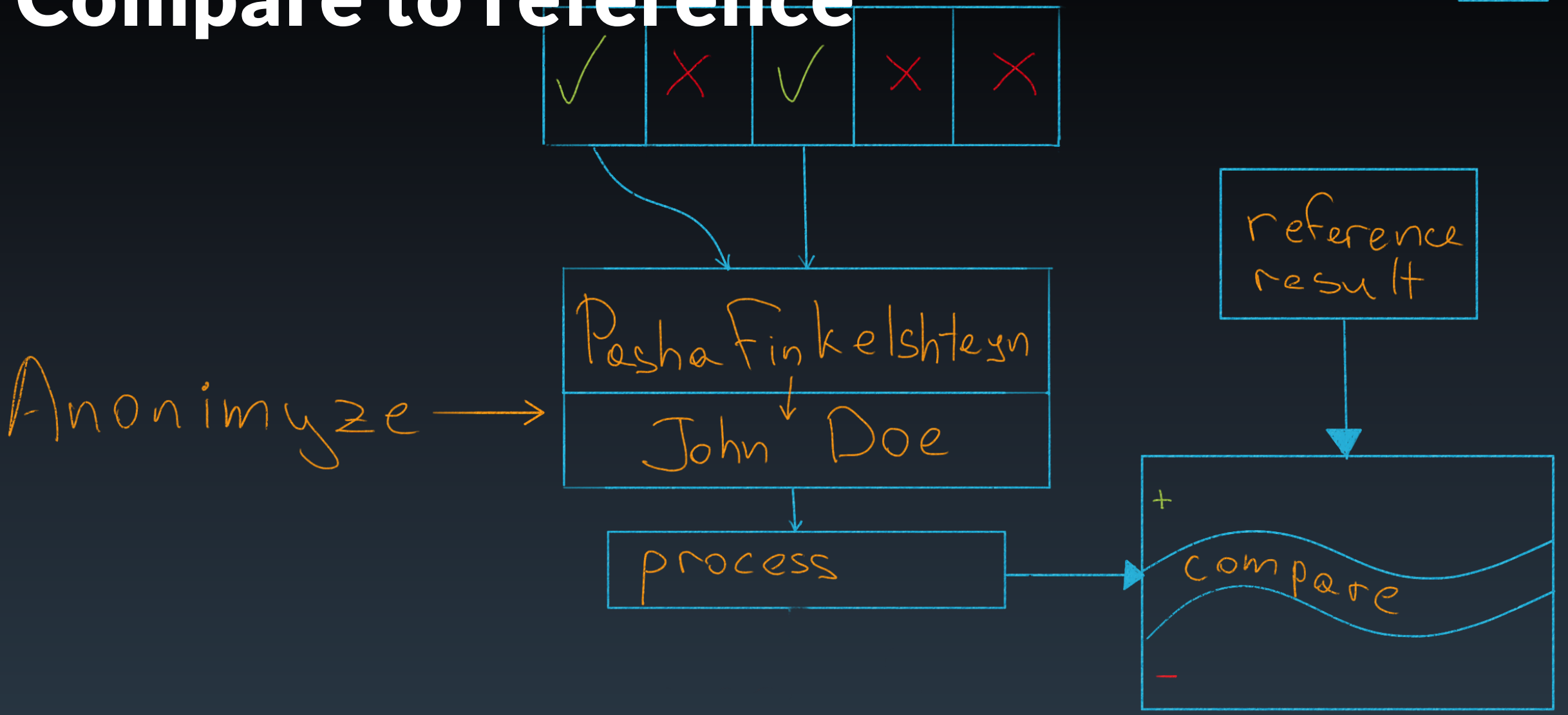

real
Systems
real
data
Component
UNIT

Real data

- get data samples from prod
- anonymize it



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

```
from pyspark.sql.types import Row, StructType
from datetime import datetime

schema = {
  "type": "struct",
  "fields": [
    {"name": "Id", "type": "long", "nullable": False, "metadata": {}},
    {"name": "SaleDate", "type": "timestamp", "nullable": False, "metadata": {}},
    {"name": "Country", "type": "string", "nullable": False, "metadata": {}},
  ]
}

table_rows = [
        Row(1, datetime(2021, 1, 1, 10, 0, 0), "RU" ),
        Row(2, datetime(1000, 1, 1, 10, 0, 0), "KZ"),
        Row(2, datetime(2018, 1, 1, 10, 0, 0), "AU"),
        Row(2, datetime(2019, 1, 1, 10, 0, 0), ""),
    ]

sample_df = spark.createDataFrame(table_rows, StructType.fromJson(schema))
```

Great expectations

```
from great_expectations.dataset.sparkdf_dataset import SparkDFDataset

ge_sample_df = SparkDFDataset(sample_df)
ge_sample_df.expect_column_values_to_be_in_set("Country", ["RU", "KZ"])
```

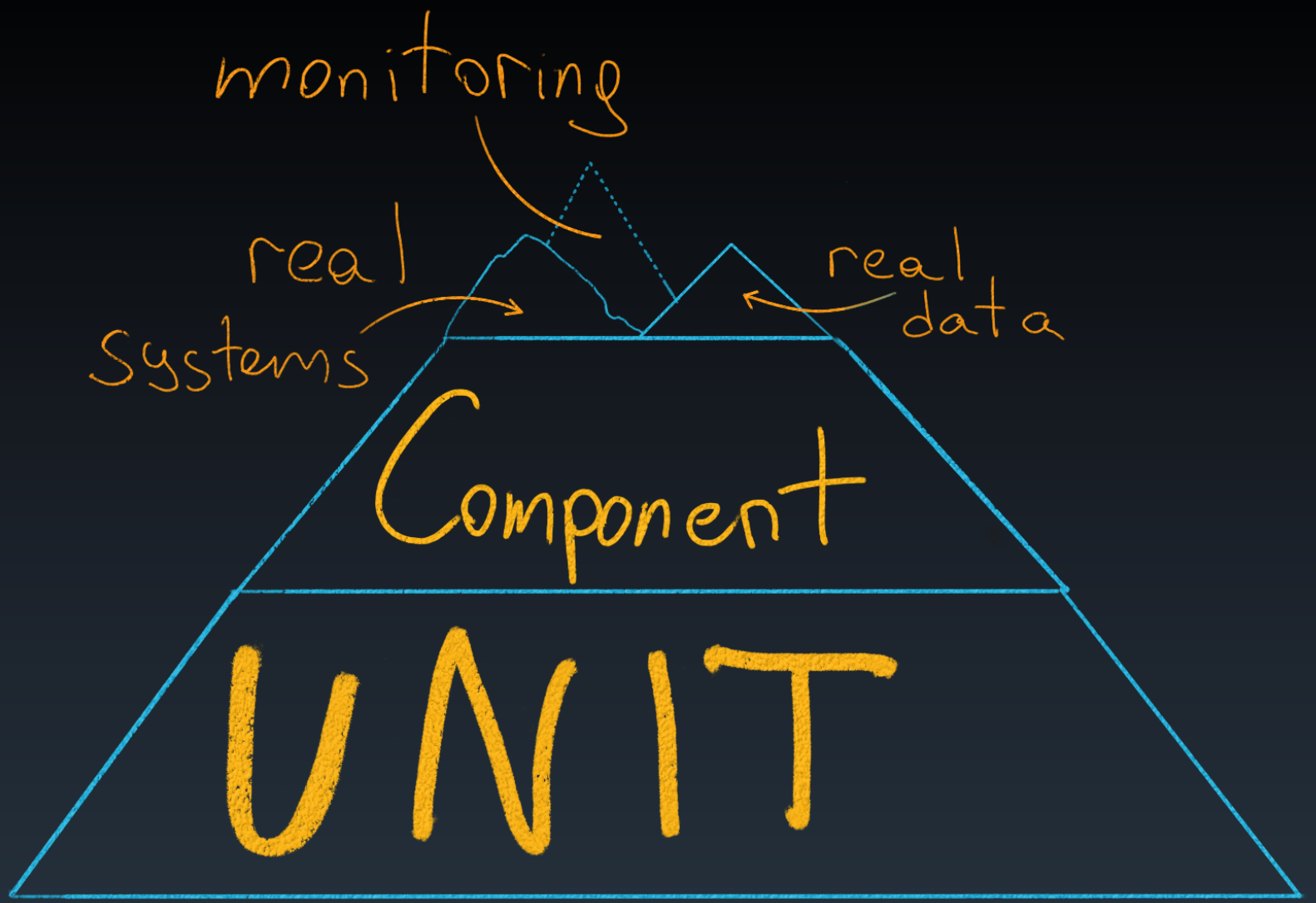
Great expectations


```
{
  "result": {
    "element_count": 4,
    "unexpected_count": 2,
    "unexpected_percent": 50.0,
    "partial_unexpected_list": ["AU",""]
  },
  "success": false,
  "expectation_config": {
    "kwargs": {
      "column": "Country",
      "value_set": ["RU", "KZ"]
    }
  }
}
```

Python Deequ

```
# No Spark 3.0 support yet
from pydeequ.checks import *
from pydeequ.verification import *

check = Check(spark, CheckLevel.Warning, "Country Check")
checkResult =(
  VerificationSuite(spark)
    .onData(sample_df)
    .addCheck(
        check.isContainedIn("Counrty", ["RU","KZ"]))
   .run()
)
checkResult_df = VerificationResult.checkResultsAsDataFrame(spark, checkResult)
checkResult_df.show()
```

Testing data quality at scale with PyDeequ


monitoring
real
Systems
real
data
Component
UNIT

Monitoring

Why?

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

Monitoring

What?

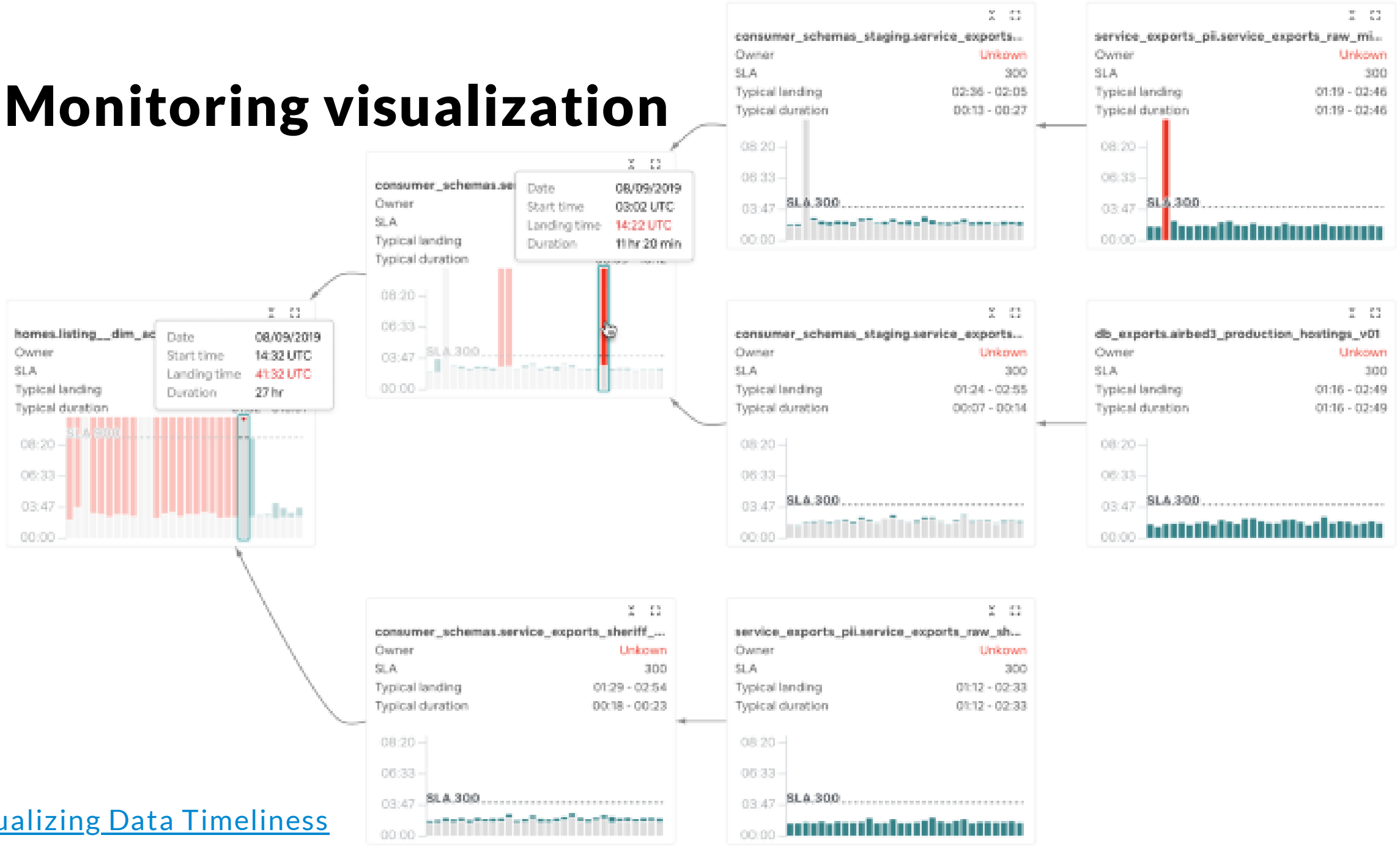
- data volumes
- counters
- time
- dead letter queue monitoring

Monitoring

How?

- use Listeners
- use data aggregations

Monitoring visualization



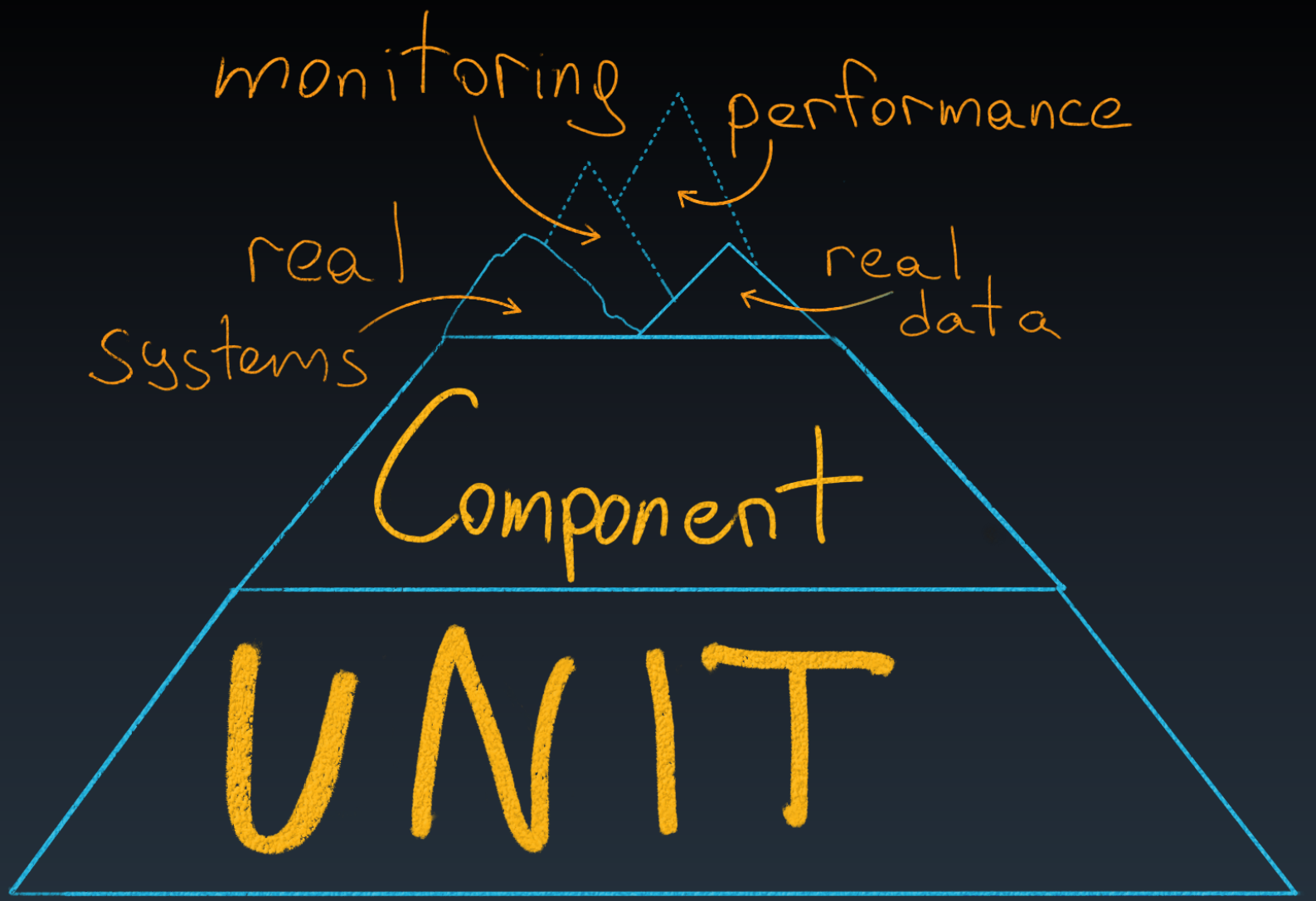
[Visualizing Data Timeliness]

End-to-End tests

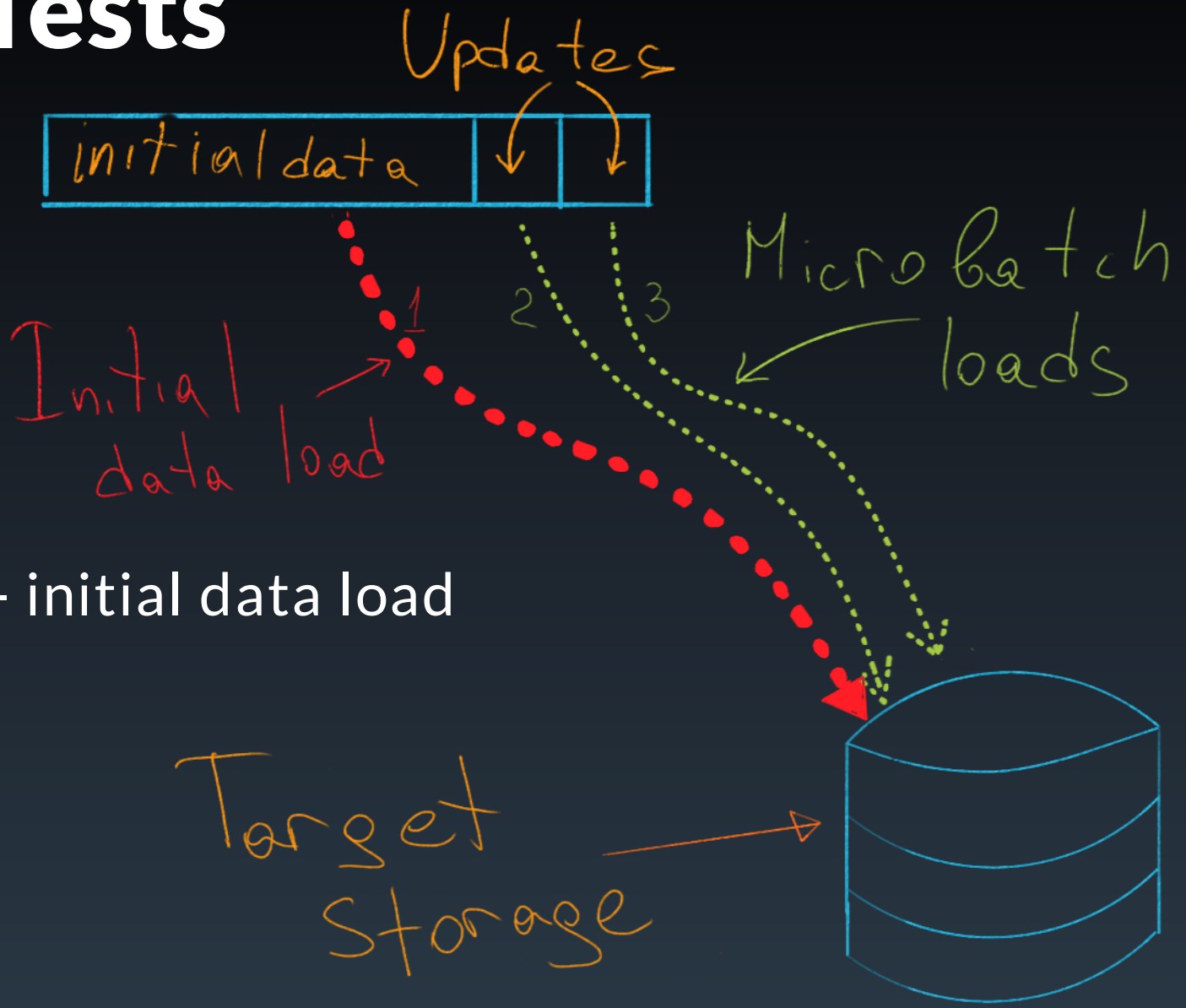
Compare with reports, old DWH

Multiple dimensions:

- data
- data latency
- performance, scalability


monitoring
performance
real
Systems
real
data
Component
UNIT

Performance Tests



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

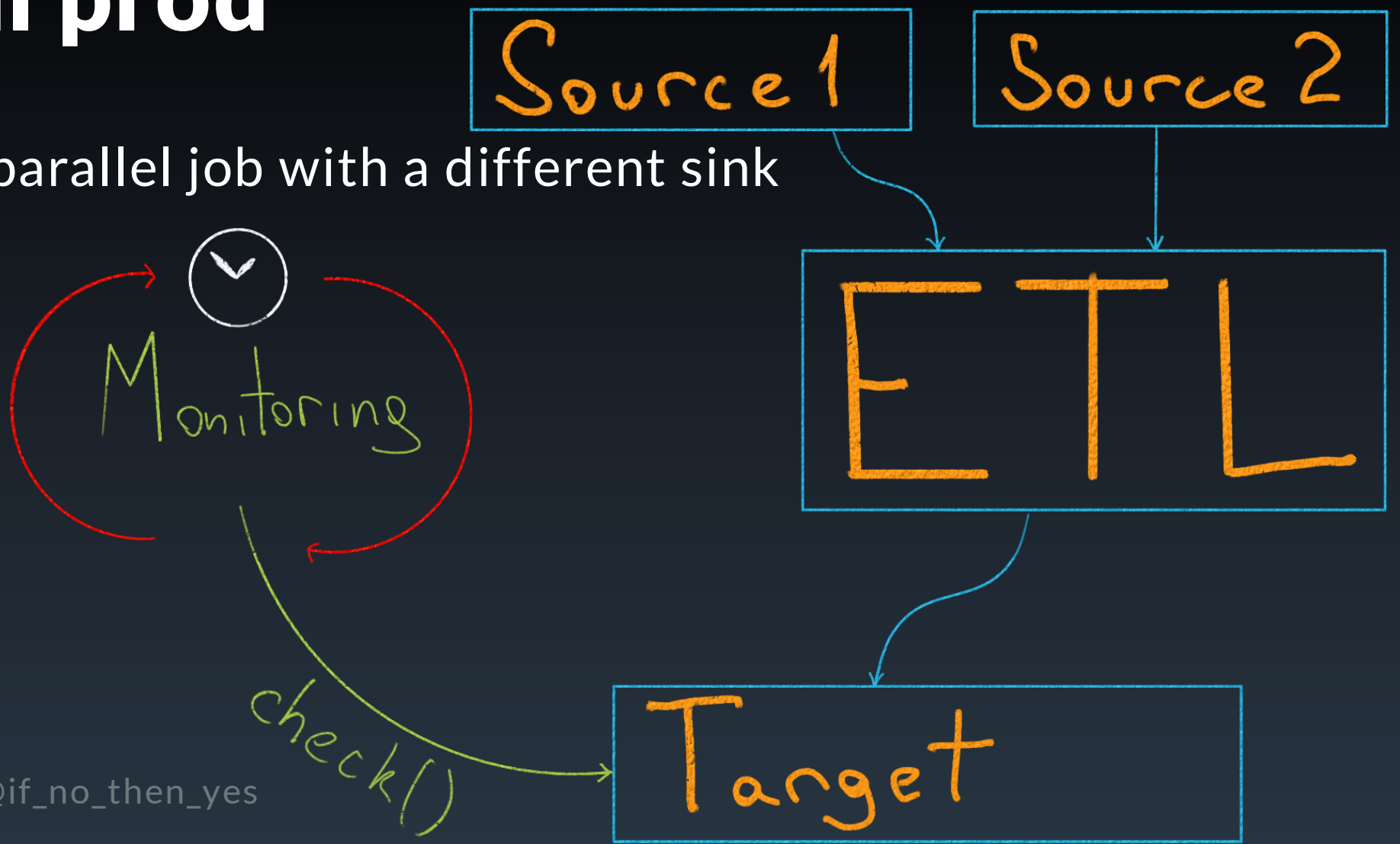
How to apply?

Real data, real system

Deploy full data backup on stage env, anonymize it 🤑

Real prod

Run a parallel job with a different sink



@asm0di0 @if_no_then_yes

[Using production data for testing in a post GDPR world]