Himalayan Peaks of Testing Data Pipelines

Talk about quality of pipelines

Pasha Finkelshteyn, JetBrains

Pasha Finkelshteyn

me

Developer 🥑 for Big Data @ JetBrains

🐦 @asm0di0
🐘 @asm0dey@fosstodon.org

Data processing

kafka

topic: actions

modify data

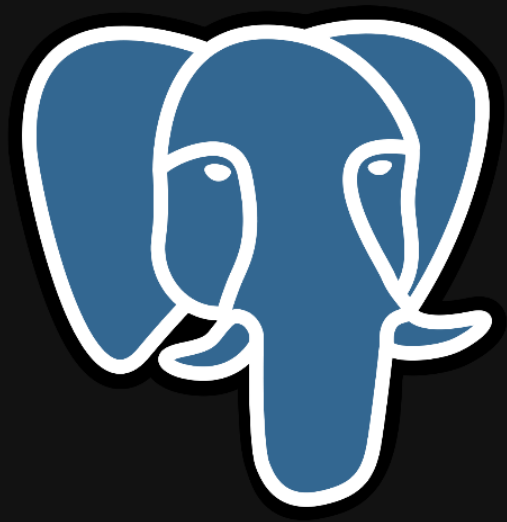


table: actions

Data lake?

Bronze → Silver → Gold

```
{
  "data": {
    "pk": 2,
    "aa": "text"
  },
  "op": "c",
  "ts_ms":
"1580390884335"
}
```

pk	aa	datetime
2	text	2020-01-01

data	aa_count
2020-01-01	231

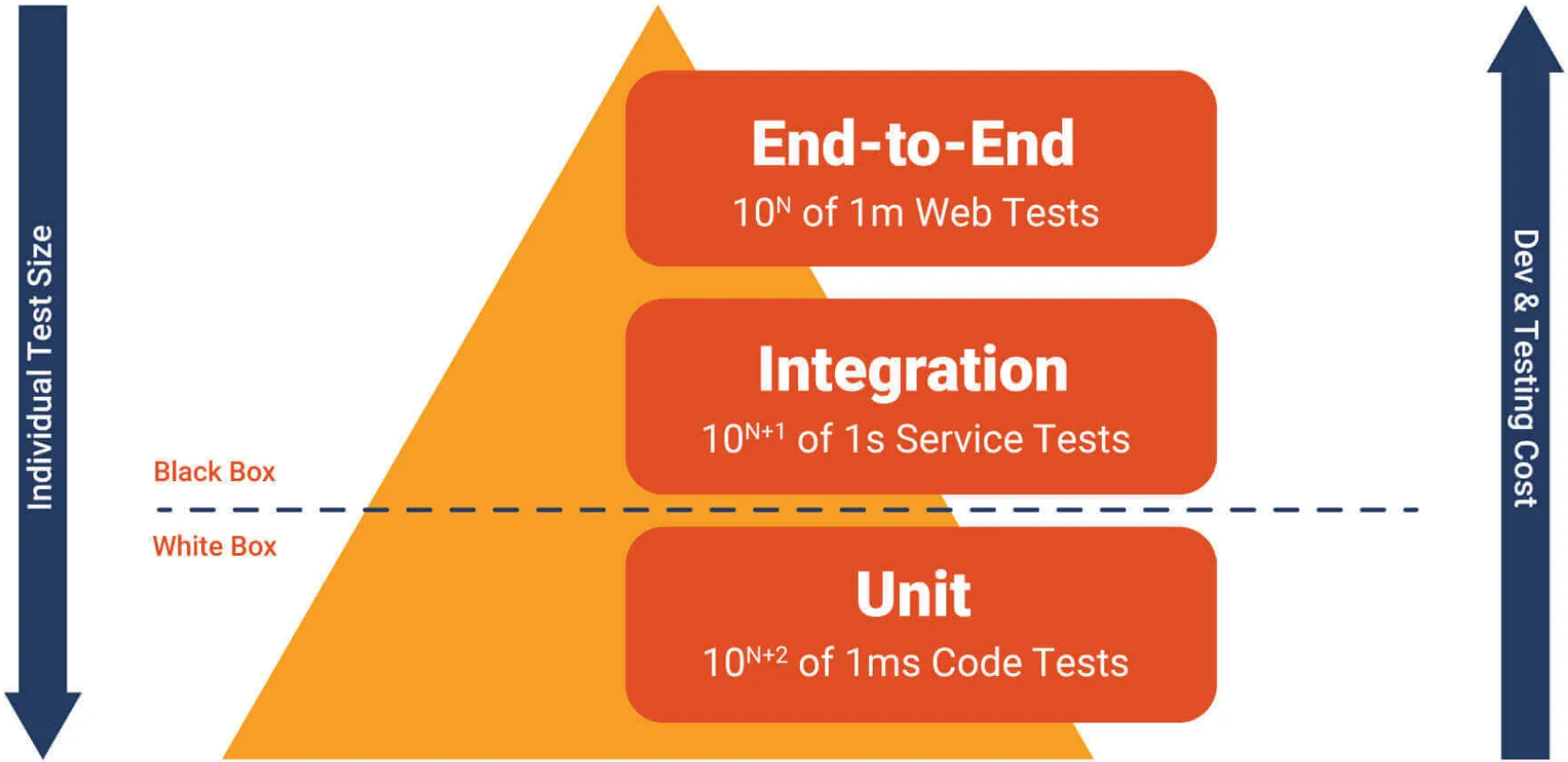
Who needs pipelines

- Data Scientists
- Data Analytics
- Marketing
- PO

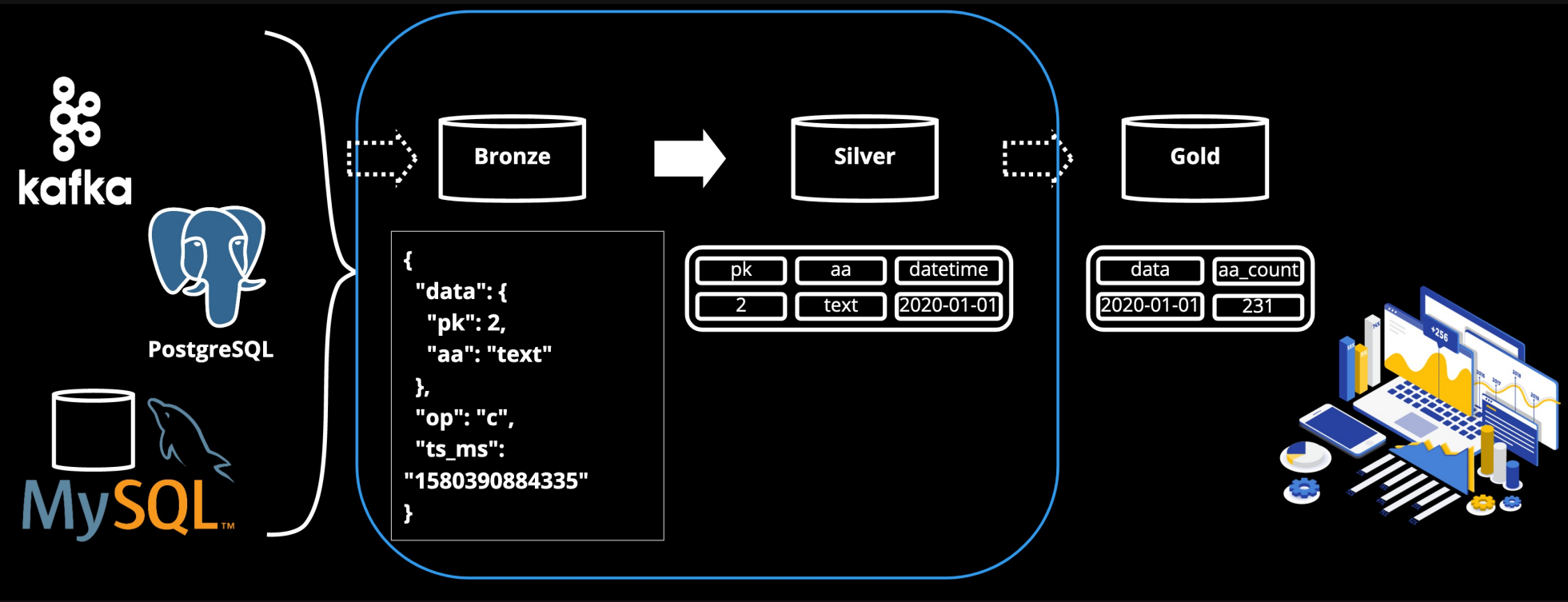
It have to be tested!

Pyramid of testing?

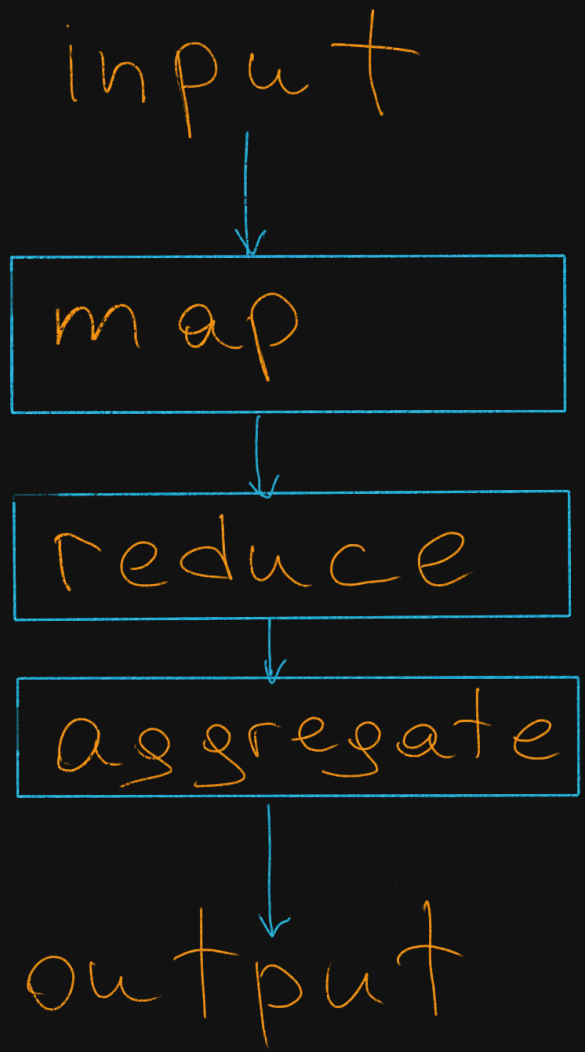
Pyramid of testing?


Individual Test Size
End-to-End
10^N of 1m Web Tests
Integration
10^N+1 of 1s Service Tests
Black Box
White Box
Unit
10^N+2 of 1ms Code Tests
Dev & Testing Cost

Pyramid of testing. Unit


kafka
PostgreSQL
MySQL
Bronze
Silver
Gold
{
"data": {
"pk": 2,
"aa": "text"
},
"op": "c",
"ts_ms":
"1580390884335"
}
pk
aa
datetime
2
text
2020-01-01
data
aa_count
2020-01-01
231

Typical pipeline



Typical pipeline

```
StructType schema = new StructType(new StructField[]{
  new StructField("pk", DataTupes.LongType, false, MetaData.empty()),
  new StructField("aa", DataTupes.StringType, false, MetaData.empty()),
})

spark.read
  .schema(schema)
  .csv(/* path */)
  .map(/* mapper */)
  .write
  .parquet("path")
```

Unit testing of pipeline

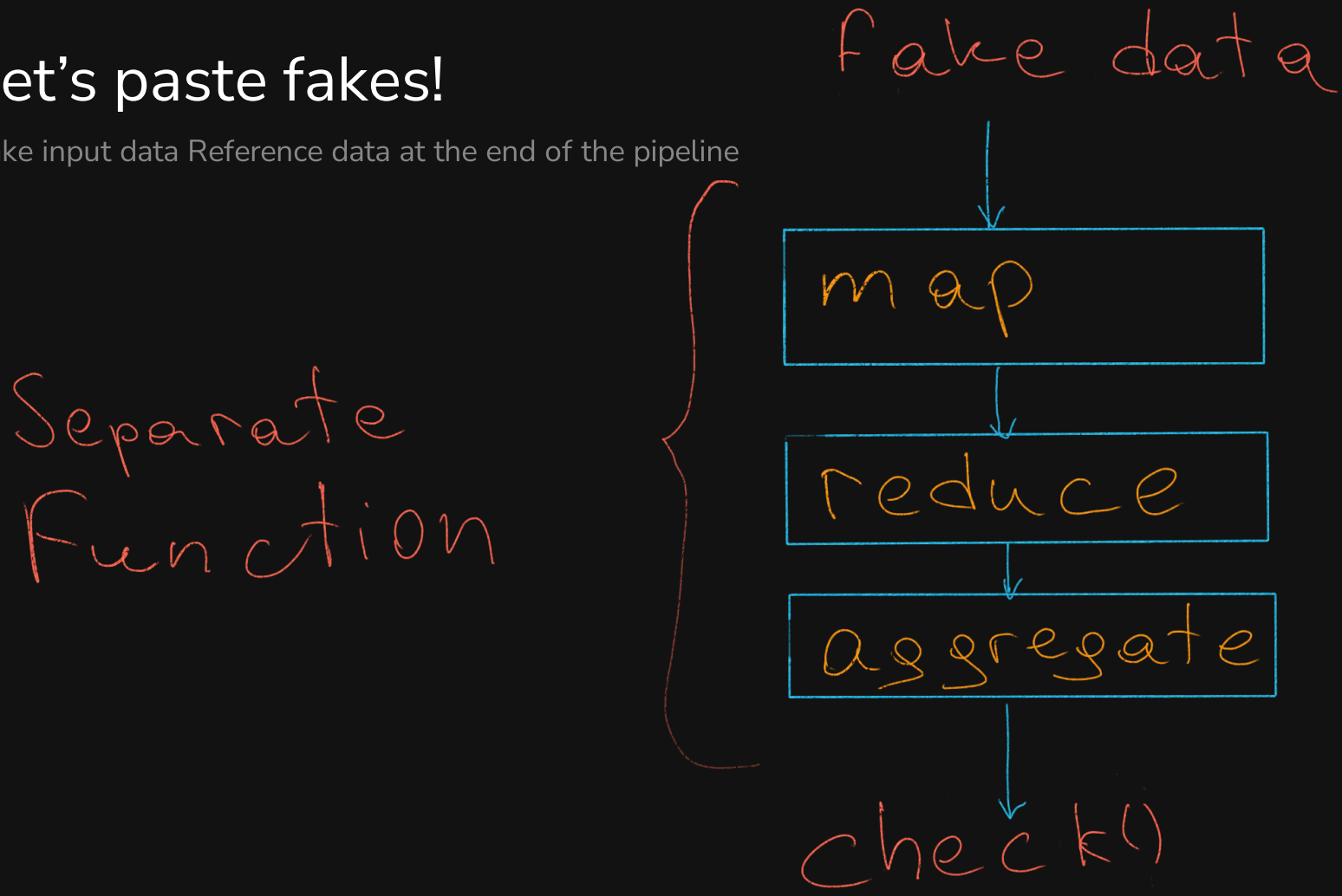
What may we test here?

A pipeline should transform data correctly!

Correctness is a business term

Let’s paste fakes!

Fake input data Reference data at the end of the pipeline

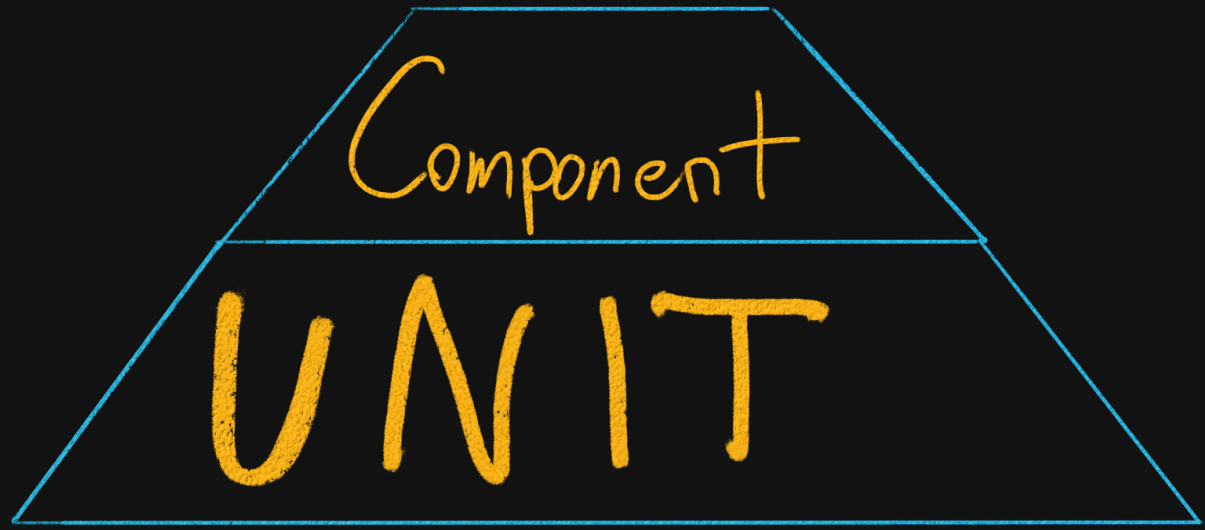


Tools

holdenk/spark-testing-base ← Tools to run tests

MrPowers/spark-daria ← tools to easily create test data

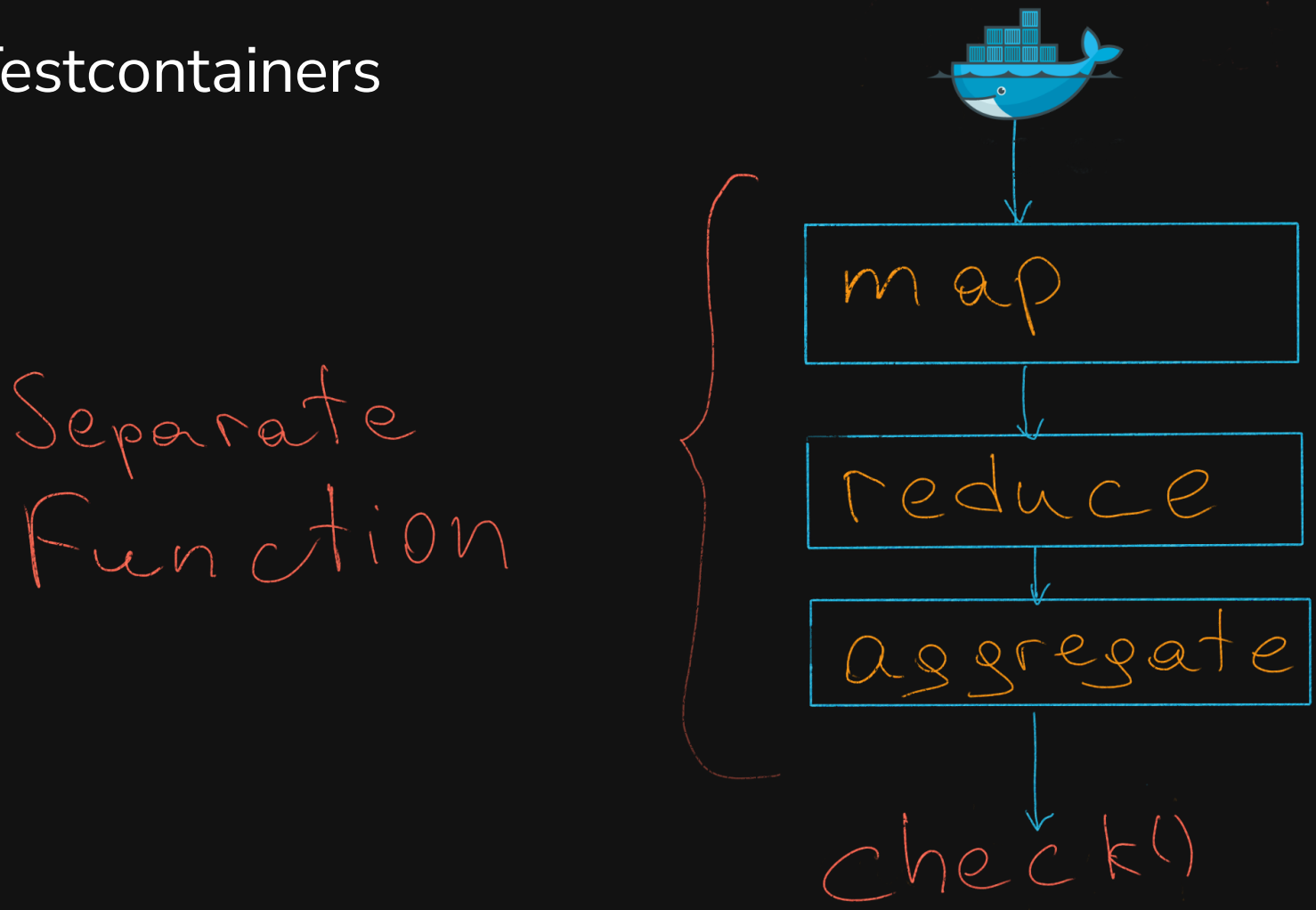
Component testing



Testcontainers

TESTCONTAINERS

Testcontainers



TestContainers

Supported languages:

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

TestContainers

```
from testcontainers postgres import PostgresContainer
import sqlalchemy

postgres_container = PostgresContainer("postgres:13")
with postgres_container as postgres:
  e = sqlalchemy.create_engine(postgres.get_connection_url())
  result = e.execute("SELECT version()")
  version, = result.fetchone()
print(version)
```