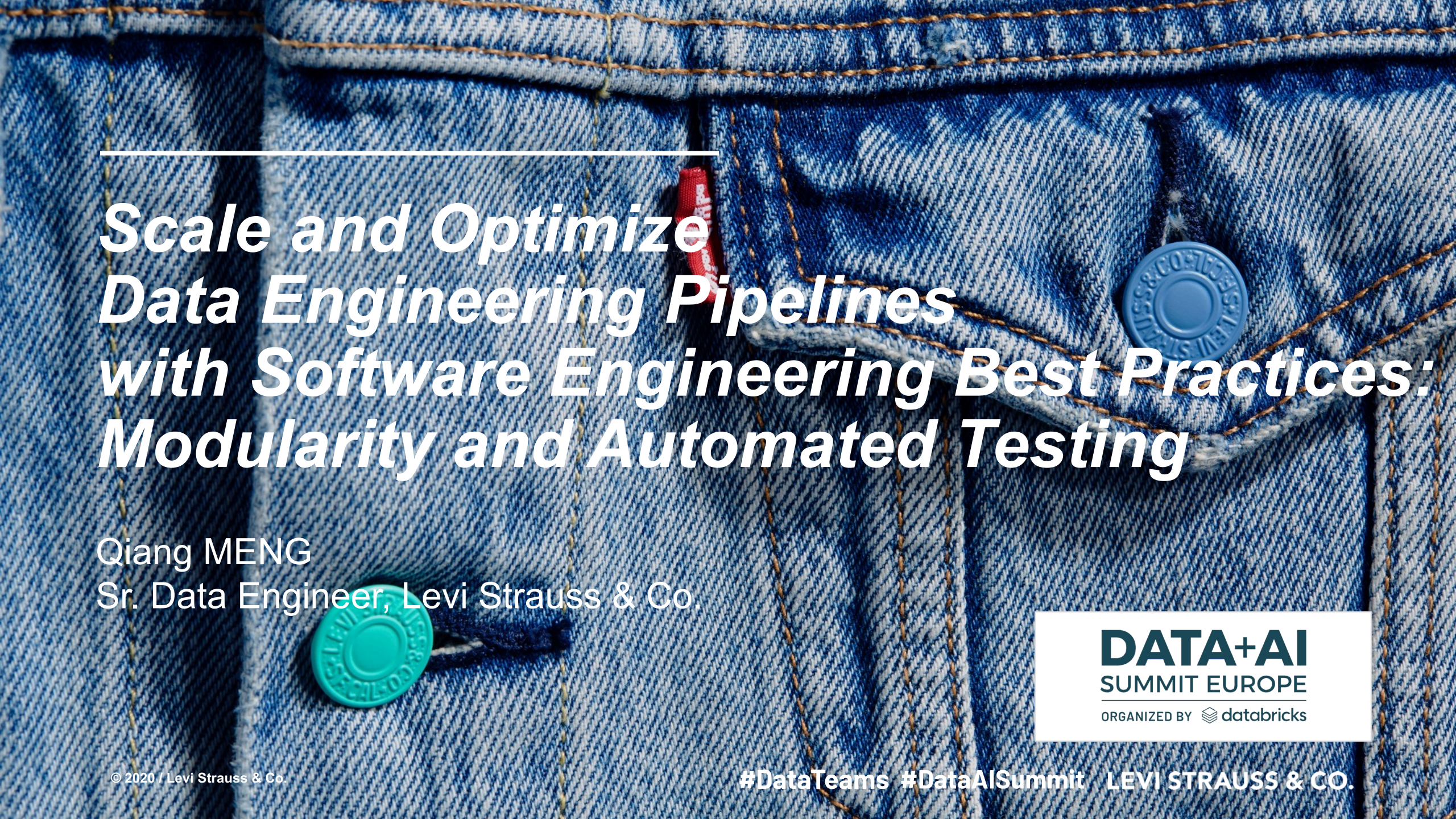


Scale and Optimize Data Engineering Pipelines with Software Engineering Best Practices: Modularity and Automated Testing

Qiang MENG
Sr. Data Engineer, Levi Strauss & Co.

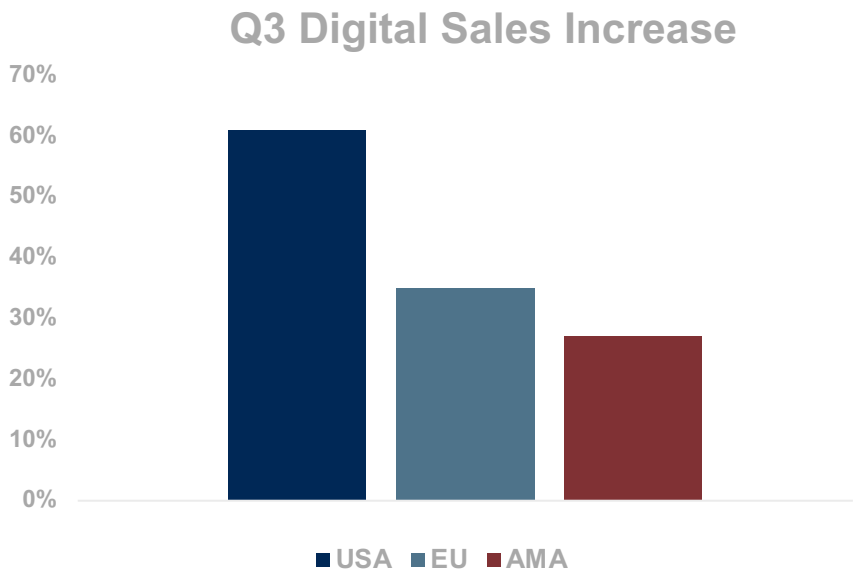
DATA+AI SUMMIT EUROPE
ORGANIZED BY databricks

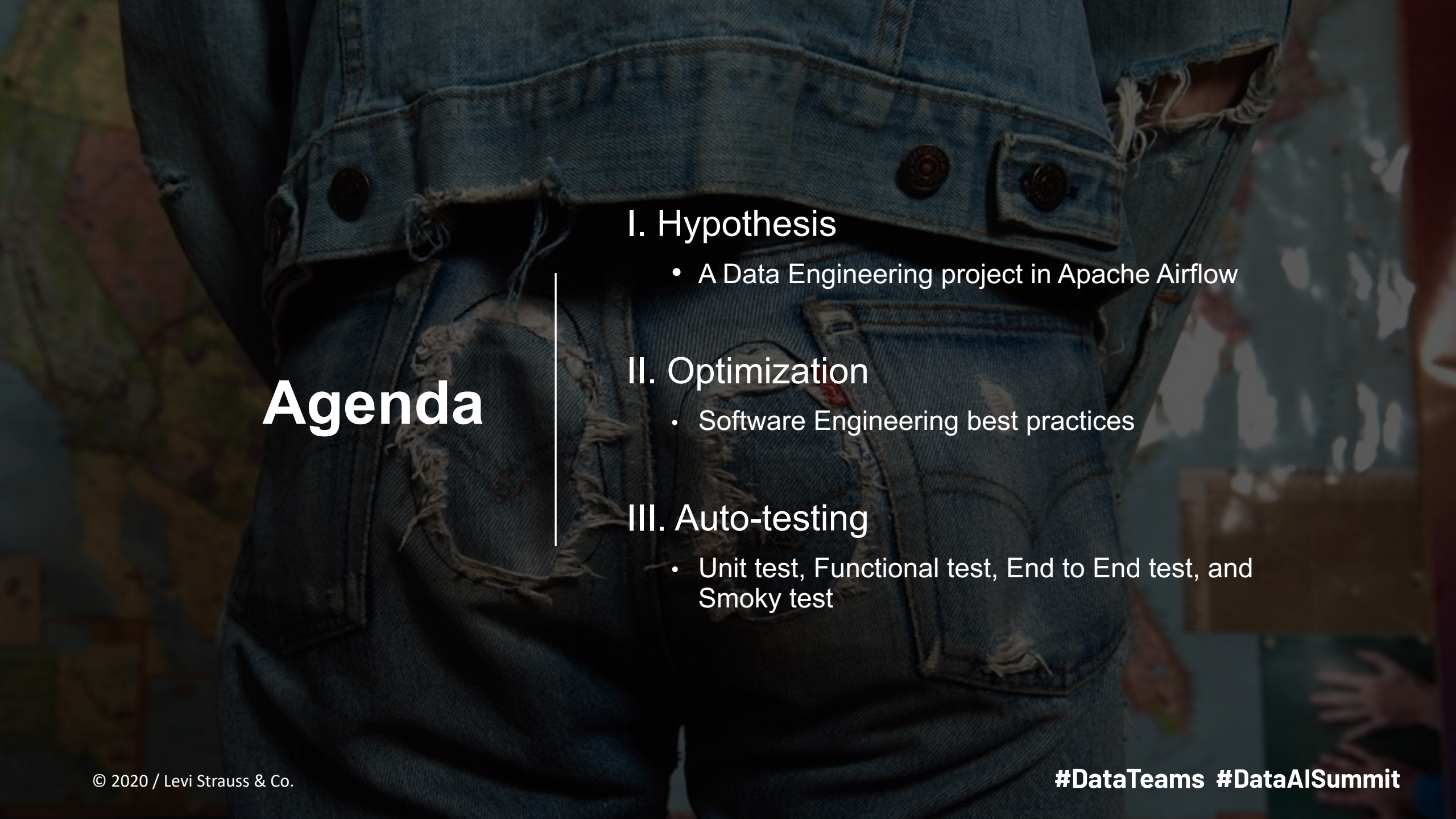
Qiang MENG

- **Levi Strauss & Co., Belgium**
  - Databricks – AWS – Airflow – Spark - Dataiku
- **H&M Group, Sweden**
  - Databricks – Azure – Airflow – Spark - Hive
- **Softbank Robotics, France**
  - AWS – Luigi – Spark

#DataTeams #DataAISummit

Levi's Data, Analytics & AI





Agenda

I. Hypothesis

- A Data Engineering project in Apache Airflow

II. Optimization

- Software Engineering best practices

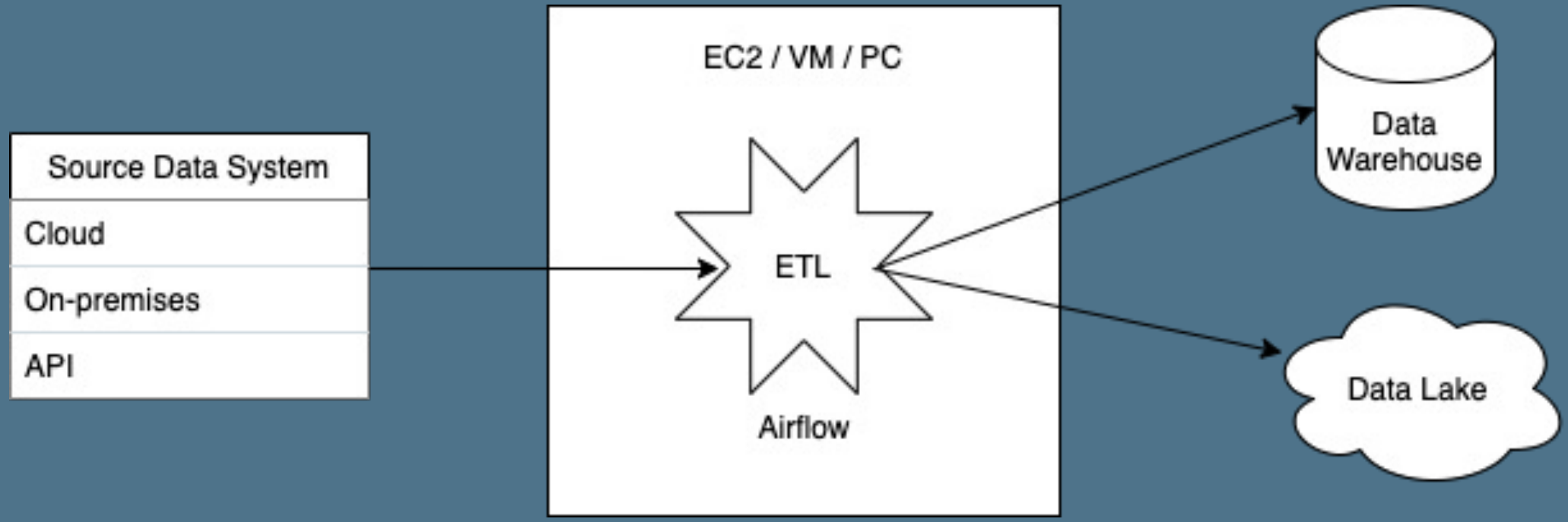
III. Auto-testing

- Unit test, Functional test, End to End test, and Smoky test

I. Hypothesis

Hypothesis

Data Engineering – Cloud - Airflow - TB-level - Batch



Hypothesis

Dev Environments

- Local – Dev – Preprod – Prod

Sample Data

- ETL in local / Dev

Code Versioning – GIT

- feature - dev - release - hotfix – master

Conventions

- DB, Schema, Table, Column, Pipeline

Container-orchestration

- Docker, K8S, …

Spark clusters

- Databricks, EMR, …

Data Versioning

- Delta Lake, …

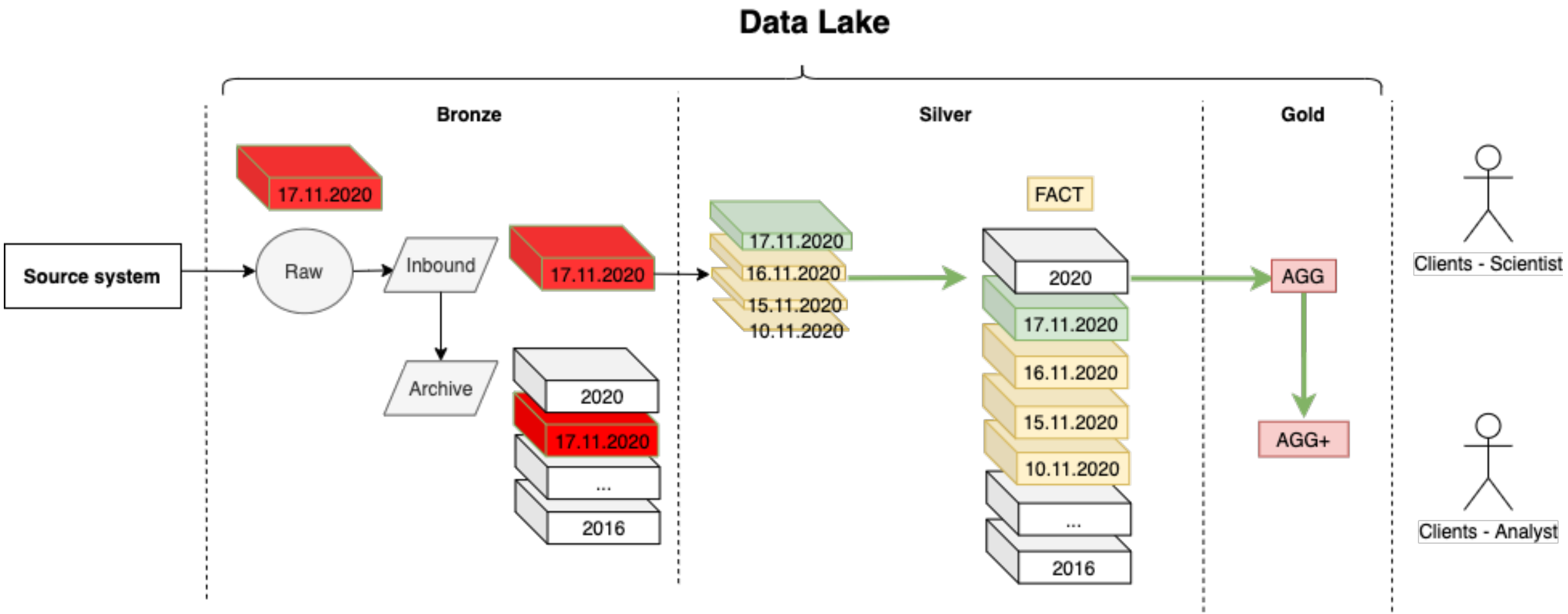
CICD

Airflow

- **airflow**
  - dags
  - operators
  - …
- **utils**
- **modules**
- **docs**
- **tests**
- **others**

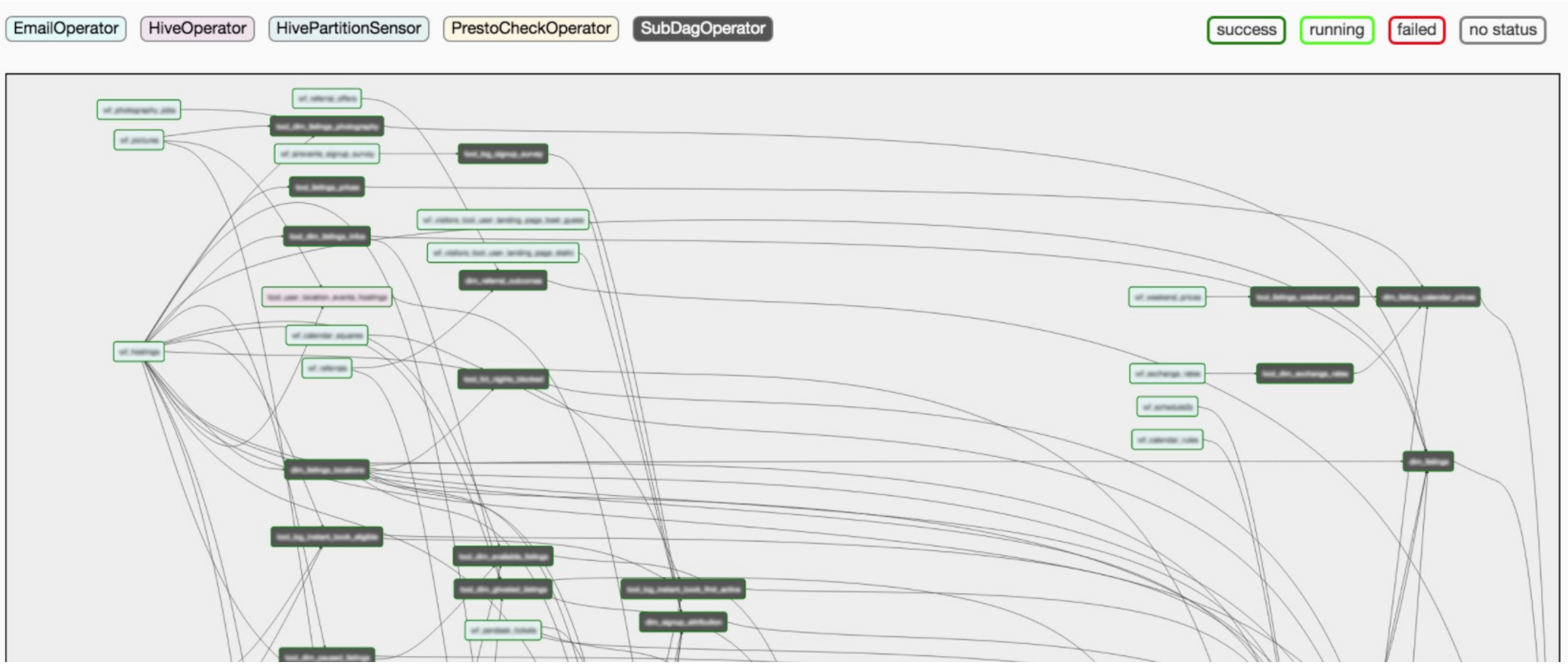
- **airflow**
  - dags
  - perators
  - …
- **others**

II. Optimization



A Big Data ETL

Optimization - Airflow DAG



Optimization - Airflow DAG

A simple template of tasks – Data Feeds Based

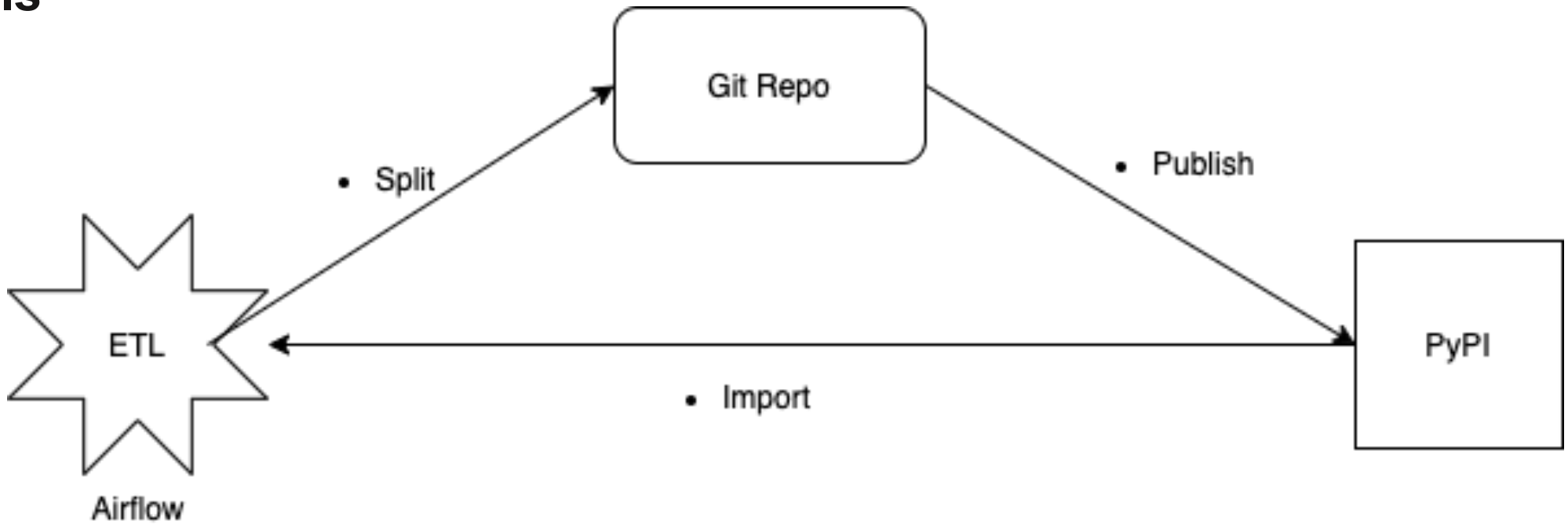
Optimization - Airflow Operator

Do the real work

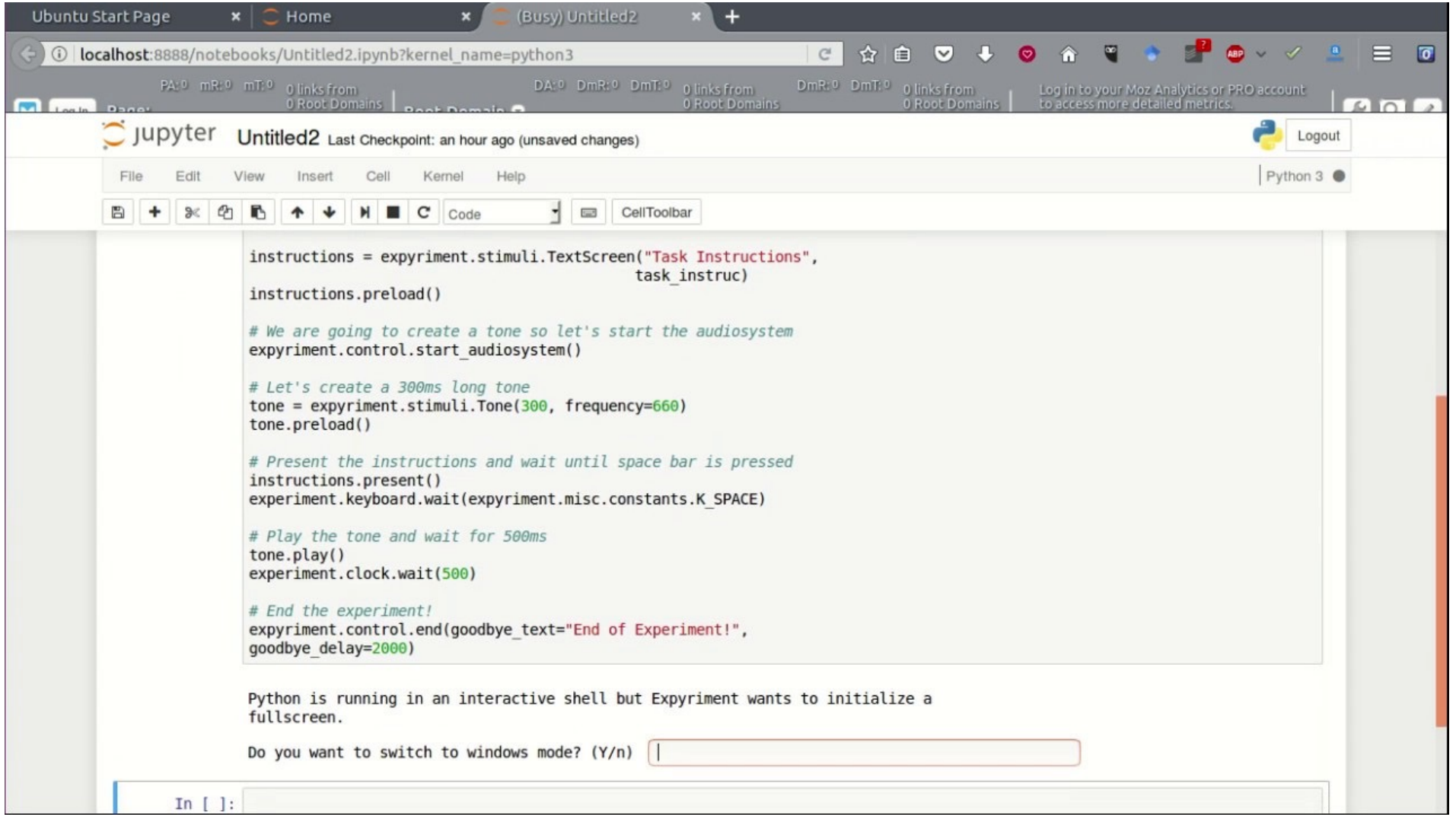
- **Default Operators**
  - Keep as they are
- **Customized Operators**
  - Hive
  - Spark
  - Deduplication
  - Data Validation
  - Data Profiling
  - …

Optimization - Modules and Utils

- **Modules**
- **Utils**

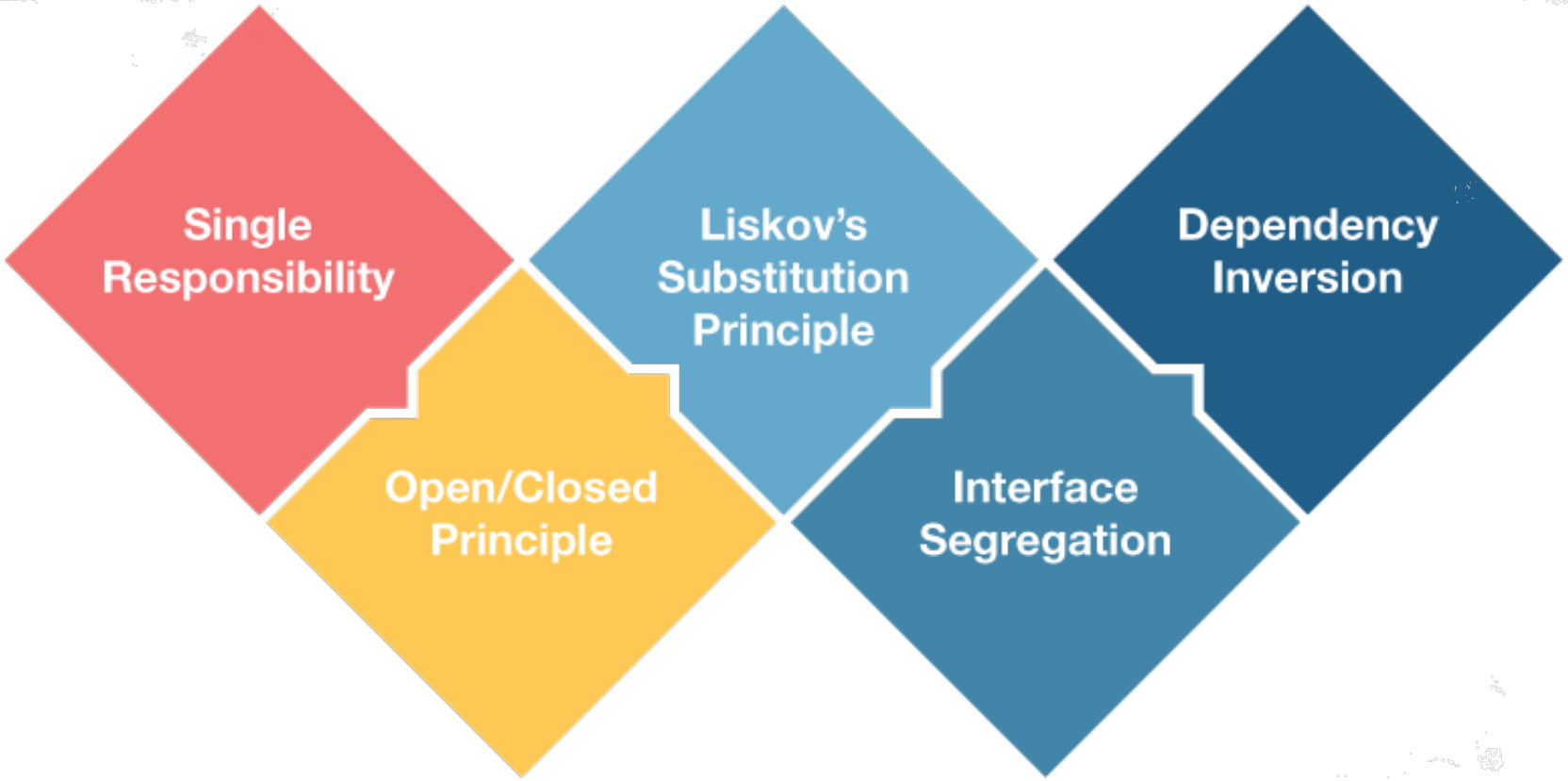


Optimization - OOP



```
class Vehicle:
    vtype = ''

    def start(self, x):
        self.vtype = x
        print('The', self.vtype, 'started')

    @staticmethod
    def cleanVehicle():
       print('Cleaning the vehicle')

#Auto class extends Vehicle class
class Auto(Vehicle):
    nun = 2

    def open(self, x):
        self.num = 4
        print(self.num, 'doors were opened')

auto1 = Auto()
auto1.start("car")
auto1.open(4)
```

Optimization - OOP

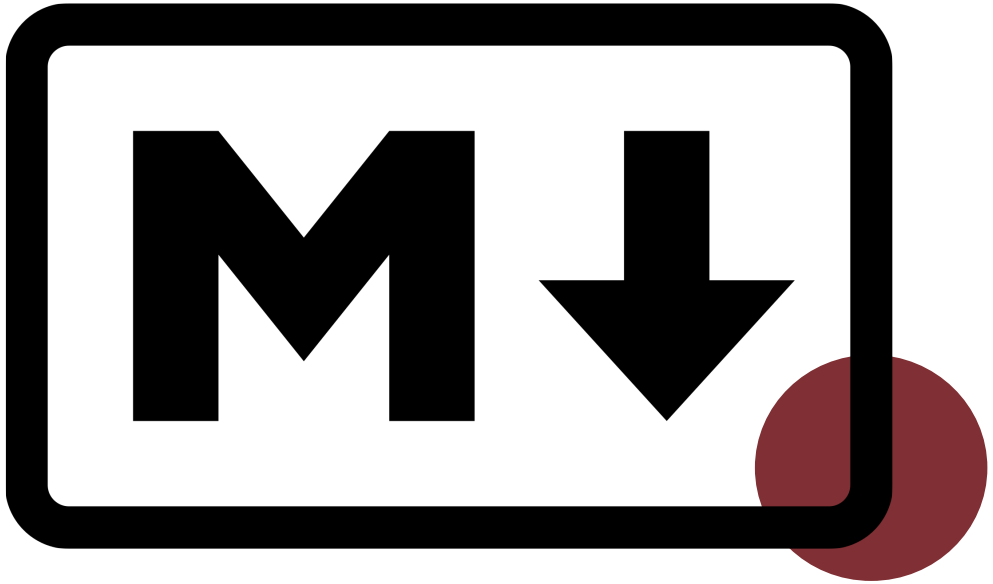


Optimization - Design Patten

Optimization - Formatting

Optimization - Docs

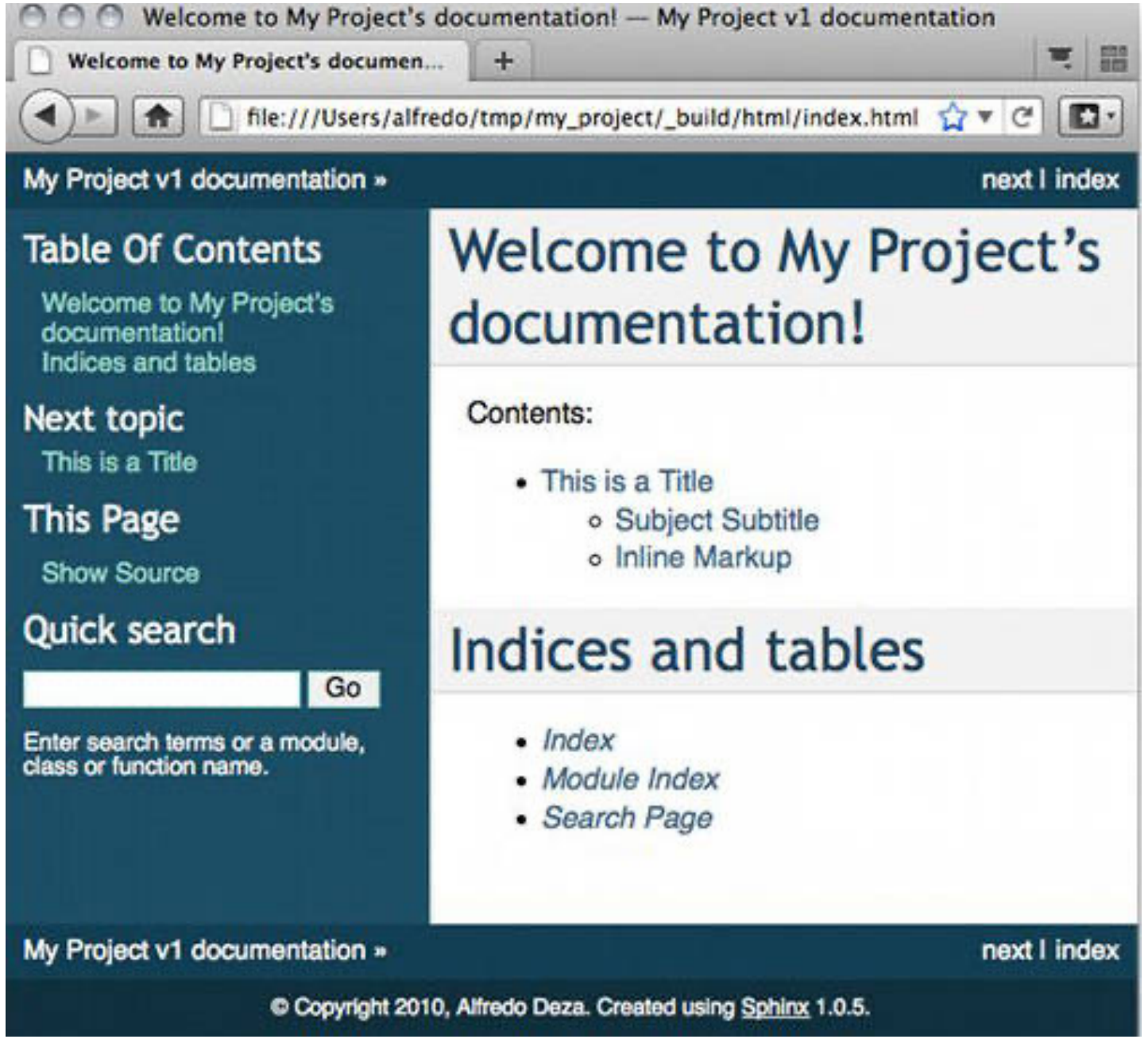
- Docstring - Sphinx
- Data Catalog – Markdown



Docstring

```
# -*- coding: utf-8 -*-
"""Example Google style docstrings.

This module demonstrates documentation as specified by the `Google Python
Style Guide`_. Docstrings may extend over multiple lines. Sections are created
with a section header and a colon followed by a block of indented text.

Example:
    Examples can be given using either the ``Example`` or ``Examples``
    sections. Sections support any reStructuredText formatting, including
    literal blocks::

        $ python example_google.py

Section breaks are created by resuming unindented text. Section breaks
are also implicitly created anytime a new section starts.

Attributes:
    module_level_variable1 (int): Module level variables may be documented in
        either the ``Attributes`` section of the module docstring, or in an
        inline docstring immediately following the variable.

        Either form is acceptable, but the two should not be mixed. Choose
        one convention to document module level variables and be consistent
        with it.

Todo:
    * For module TODOs
    * You have to also use ``sphinx.ext.todo`` extension

.. _Google Python Style Guide:
   http://google.github.io/styleguide/pyguide.html

"""
```

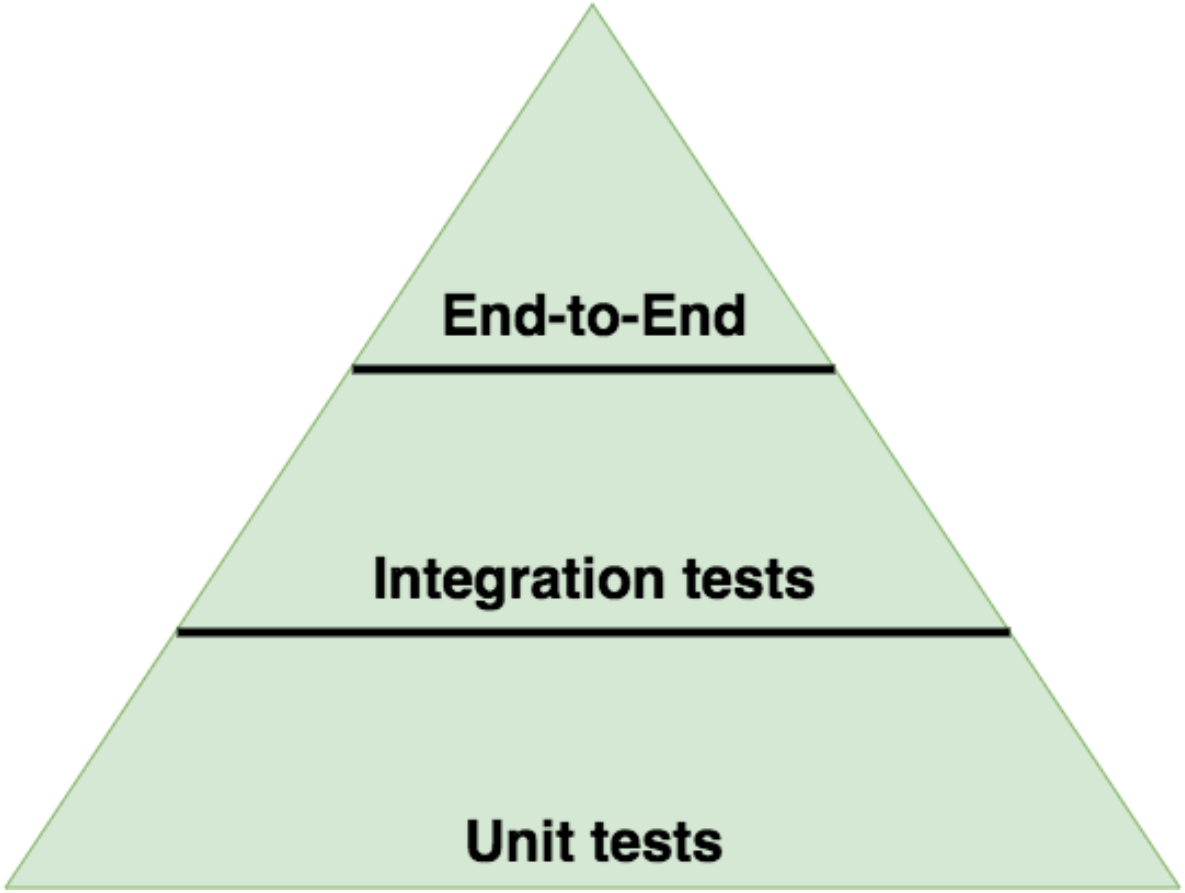
Auto Data Catalog with Markdown



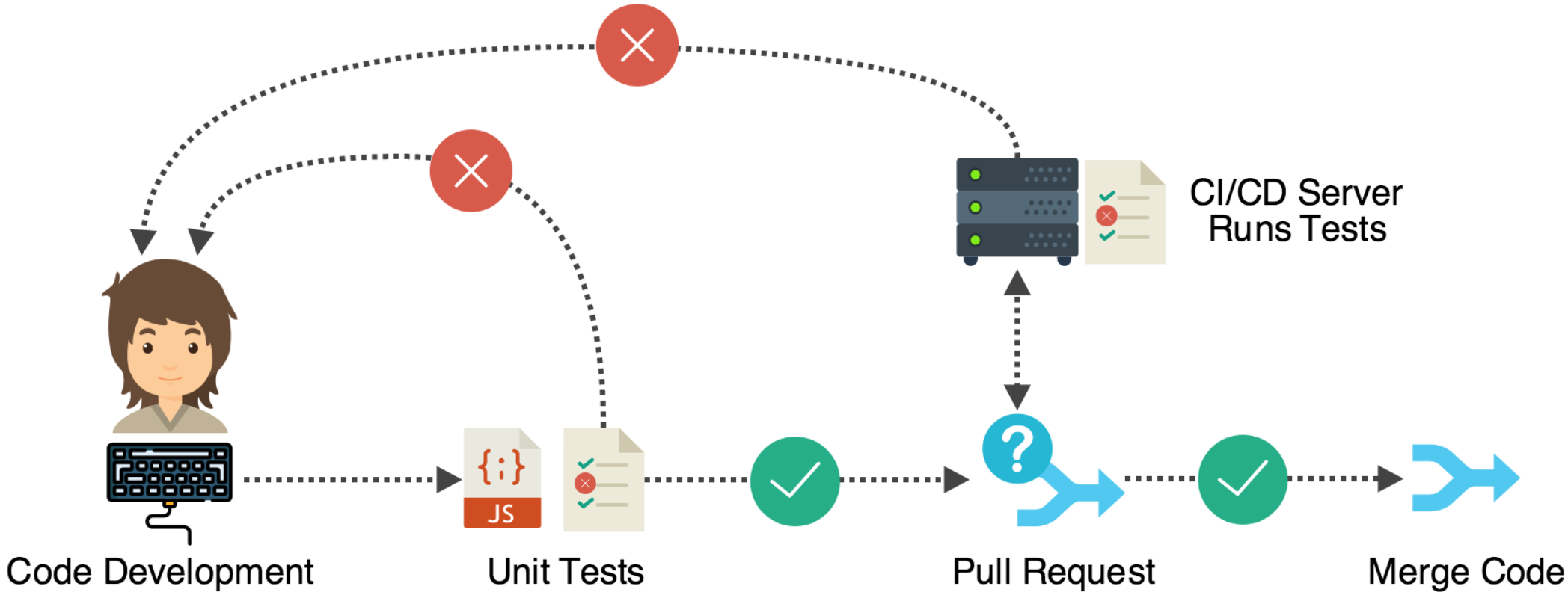
III. Auto-testing

Auto-Testing

- Unit tests
- Integration tests
- End-to-End tests
- Smoky tests

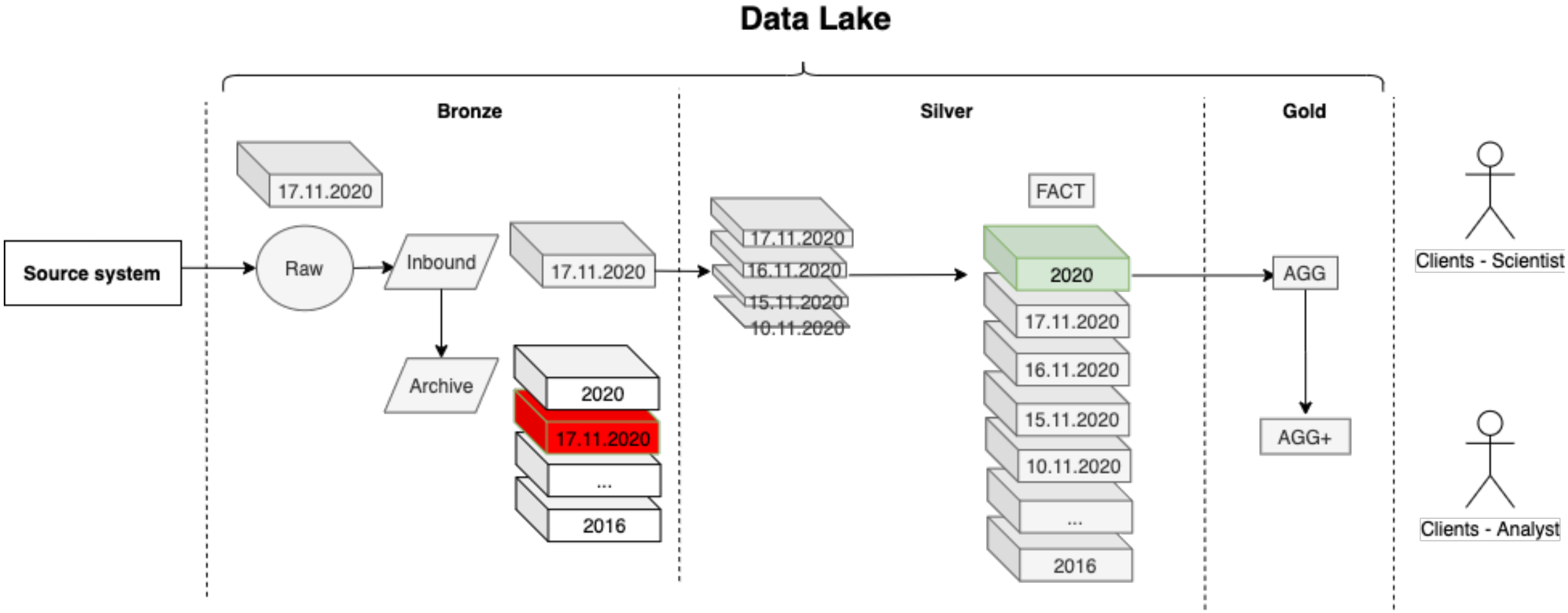


Unit Tests



Integration Tests

End-to-End Tests



Smoke Tests

THANK YOU

Feedback

Your feedback is important to us.

Don’t forget to rate and review the sessions.