BCG GAMMA

nibble

Advanced Testing Training

An introduction to Software testing
in a Machine Learning context

October 2020

About us

Florent Piétot

Data Scientist & Machine Learning Engineer

linkedin.com/in/florentpietot

Edouard Théron

Software & Machine Learning Engineer

linkedin.com/in/edouardtheron

nibble

Agenda

I. Context and general presentation

II. Focus on End-to-End Testing

III. Focus on Integration Testing

IV. A few words about Performance Testing

V. Focus on Unit Testing

VI. Code Refactoring principles

VII. Introduction to Continuous Integration

Agenda

I. Context and general presentation

II. Focus on End-to-End Testing

Day 1

III. Focus on Integration Testing

IV. A few words about Performance Testing

Day 2

V. Focus on Unit Testing

Day 3

VI. Code Refactoring principles

VII. Introduction to Continuous Integration

Day 4

Agenda

I. Context and general presentation

II. Focus on End-to-End Testing

Day 1

III. Focus on Integration Testing

IV. A few words about Performance Testing

V. Focus on Unit Testing

VI. Code Refactoring principles

VII. Introduction to Continuous Integration

Part I — Context and general presentation

- Introduction
- Different types of testing
- The Testing Pyramid of Software
- Extra Challenges for Machine Learning Projects

Part I — Context and general presentation

- **Introduction**
- Different types of testing
- The Testing Pyramid of Software
- Extra Challenges for Machine Learning Projects

Definition

Software testing is an organizational process within software development in which ***business-critical software is verified for correctness, quality, and performance****.*

[...]

It is used to ***ensure that business systems*** *and* ***product features behave*** *correctly* ***as expected.***

From *https://www.atlassian.com/continuous-delivery/software-testing*

Why is software testing necessary?

Why is software testing necessary?

Perpetual evolution

- New Features
- Maintenance
- New environment
- ...

Why is software testing necessary?

Perpetual evolution

- New Features
- Maintenance
- New environment
- ...

Even more applicable for ML

- Real-world data evolve by nature
- Technology and tools evolve quickly

Testing exists to improve process and product quality.

How to test a software?

There are two kinds of people in this World

Clint Eastwood, The Good, the Bad and the Ugly

There are two kinds of people in this World

Those who do manual testing...

- ✖ Error prone
- ✖ Unreliable
- ✖ Low-value task
- ✖ Costly

… And those who have automated their tests.

- ✓ Tracked in text files (git)
- ✓ Shareable
- ✓ Reusable
- ✓ Improvable

Part I — Context and general presentation

- Introduction
- **Different types of testing**
- The Testing Pyramid of Software
- Extra Challenges for Machine Learning Projects

Here is a subjective shortlist of most commons tests…

- End-to-end tests
- Integration tests
- Unit tests
- Performance tests
- Load tests
- ...

- End-to-end tests
- Integration tests
- Unit tests

- Performance tests
- Load tests

- ...

- End-to-end tests
- Integration tests
- Unit tests

Functional tests

- Performance tests
- Load tests

Acceptance tests

- ...

- **End-to-end tests**
- Integration tests
- Unit tests
- Performance tests
- Load tests
- ...

- **End-to-end tests**
- Integration tests
- Unit tests
- Performance tests
- Load tests
- ...

❏ “Black-box” testing

❏ Validate the functionality of a system as a whole

❏ Must be as close as possible to the final user behavior

❏ Fully integrated

- **End-to-end tests**
- Integration tests
- Unit tests
- Performance tests
- Load tests
- ...

❏ “Black-box” testing

❏ Validate the functionality of a system as a whole

❏ Must be as close as possible to the final user behavior

❏ Fully integrated

Scenario

Your prediction app running in production:

- accepts a JSON payload over an HTTP endpoint,
- fetch a serialized model stored in an S3 bucket,
- run a prediction call with data bundled in the JSON payload,
- return an HTTP response with the prediction result.

What are your suggestions of good end-to-end tests?

- End-to-end tests
- **Integration tests**
- Unit tests
- Performance tests
- Load tests
- ...

- End-to-end tests
- **Integration tests**
- Unit tests
- Performance tests
- Load tests
- ...

❏ Check interface points between subsystems

- Front-end app can communicate with back-end
- Back-end can exchange data with database server
- ...

❏ “Black box-ish”: API documentation should be enough

- End-to-end tests
- **Integration tests**
- Unit tests
- Performance tests
- Load tests
- ...

❏ Check interface points between subsystems
  - Front-end app can communicate with back-end
  - Back-end can exchange data with database server
  - ...

❏ “Black box-ish”: API documentation should be enough

Scenario

Your ML prediction system needs to:

- download a fitted model from a Cloud storage
- store its prediction results and prediction context in a relational database, running in a separate environment

What are your suggestions of good integration tests?

- End-to-end tests
- Integration tests
- **Unit tests**
- Performance tests
- Load tests
- ...

- End-to-end tests
- Integration tests
- **Unit tests**
- Performance tests
- Load tests
- ...

❏ Check tiny portions of the code base at a time: **units**

❏ "White box": you need to have access to the source code

❏ Often requires *mocking*, *monkey-patching* or *stubbing*: avoid putting integration between different units in the testing path.

- End-to-end tests
- Integration tests
- **Unit tests**
- Performance tests
- Load tests
- ...

❑ Check tiny portions of the code base at a time: **units**

❑ “White box”: you need to have access to the source code

❑ Often requires *mocking*, *monkey-patching* or *stubbing*: avoid putting integration between different units in the testing path.

Scenario

A given function of a Feature Engineering pipeline builds a new array based on two different columns of a pandas DataFrame.

What are your suggestions of good unit tests?

- End-to-end tests
- Integration tests
- Unit tests
- Performance tests
- Load tests
- ...

Acceptance tests

- End-to-end tests
- Integration tests
- Unit tests
- **Performance tests**
- Load tests
- ...

- End-to-end tests
- Integration tests
- Unit tests
- **Performance tests**
- Load tests
- ...

❏ Performance may refer to:
  - Processing time
  - Memory / CPU / bandwidth usage
  - ...

- End-to-end tests
- Integration tests
- Unit tests
- **Performance tests**
- Load tests
- ...

❏ Performance may refer to:
  - ○ Processing time
  - ○ Memory / CPU / bandwidth usage
  - ○ ...

<u>Scenario</u>

Think about a Machine Learning project you've worked on.

What acceptance criterias would you label as "performance"?

- End-to-end tests
- Integration tests
- Unit tests
- Performance tests
- **Load tests**
- ...

❏ Check that the system keeps behaving as expected under a heavy load

❏ Definition of “heavy” greatly depends on projects

  ○ 1/1k/1M/1b requests per minute

  ○ 1MB/GB/TB/PB transiting data per day

  ○ ...

❏ May easily overlap with “performance”

This is not an exhaustive list.

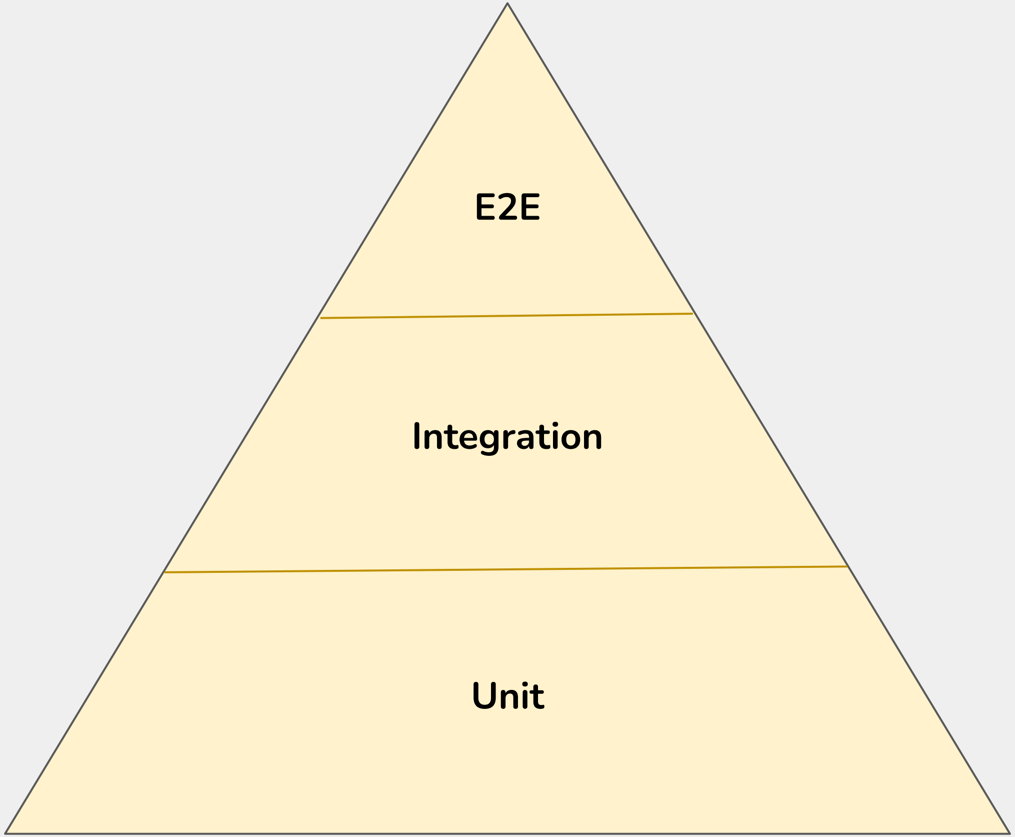
There is no consensus about namings, nor meanings.

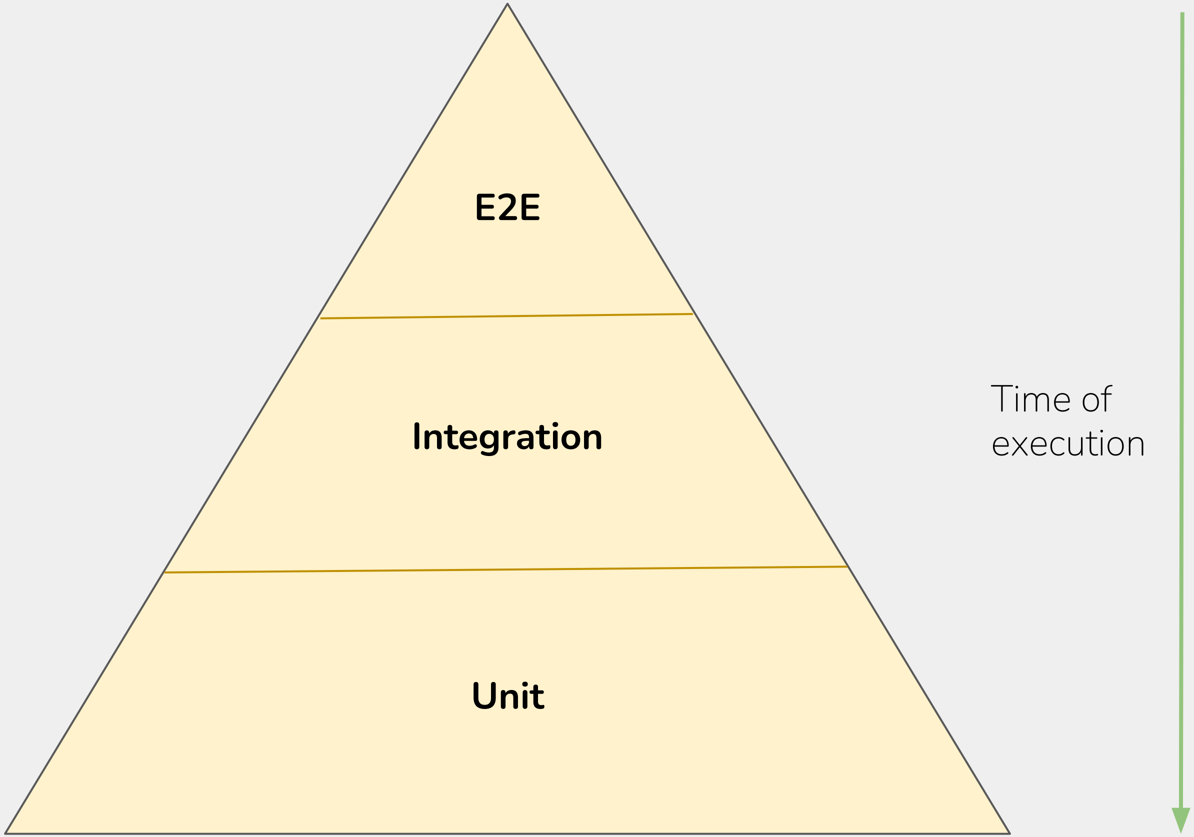
Putting the right label can be difficult.

What matters is
what the test do, not
the label you put on it.

Part I — Context and general presentation

- Introduction
- Different types of testing
- **The Testing Pyramid of Software**
- Extra Challenges for Machine Learning Projects


E2E
Integration
Unit

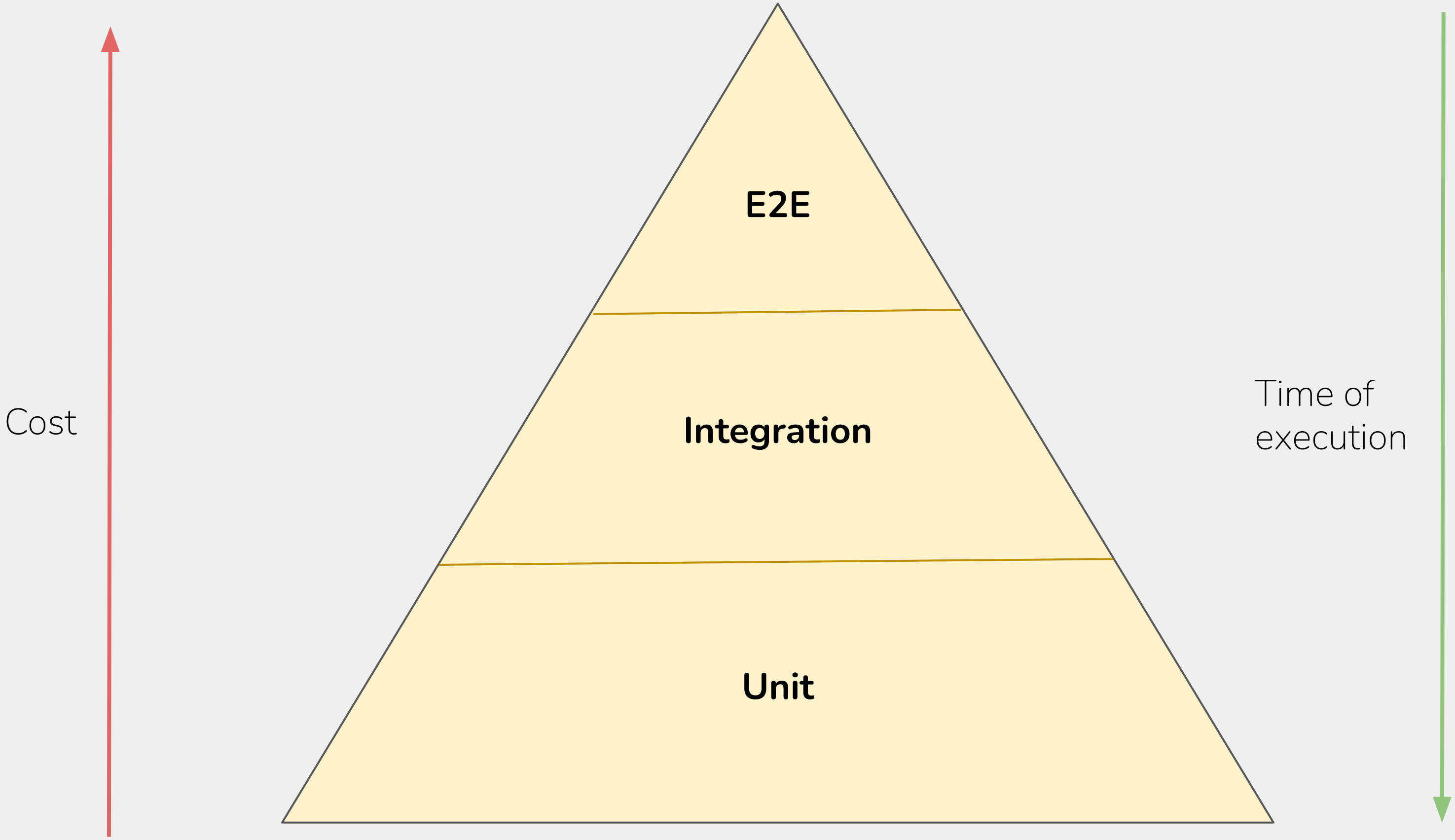


E2E

Integration

Unit

Time of execution



E2E

Integration

Unit

Cost

Time of execution

The **less integration**,
the **faster** a test run…

The **less integration**,
the **faster** a test run…

…and the **cheaper** it is
to process &
implement.

Faster tests are 99% of the time **easier to write**.

Orders of magnitude for a regular project

- > 100s of unit tests
- 10s of integration tests
- < 10 of end-to-end tests

First Part — Contextualization

- Introduction
- Different types of testing
- **The Testing Pyramid of Software**
- Extra Challenges for Machine Learning Projects

First Part — Contextualization

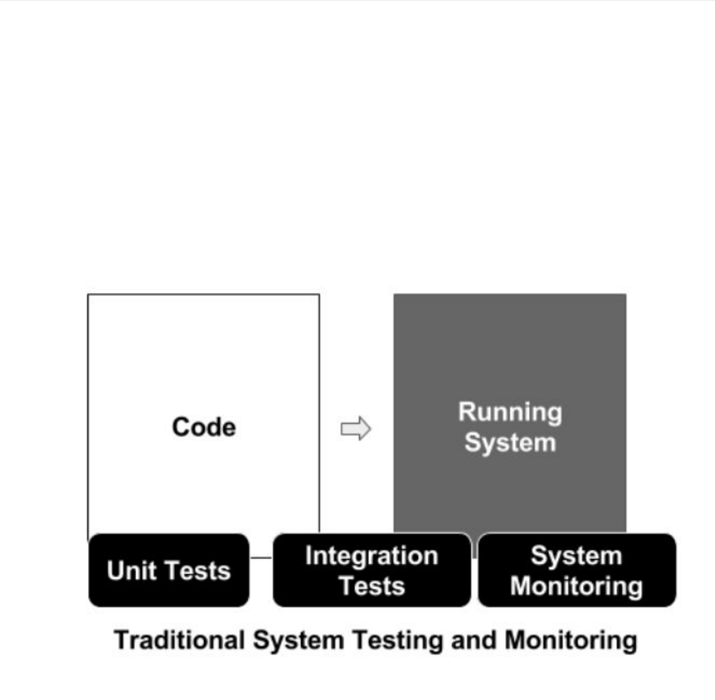
- Introduction
- Different types of testing
- The Testing Pyramid of Software
- **Extra Challenges for Machine Learning Projects**

Reliability of ML projects is much more difficult to grasp.

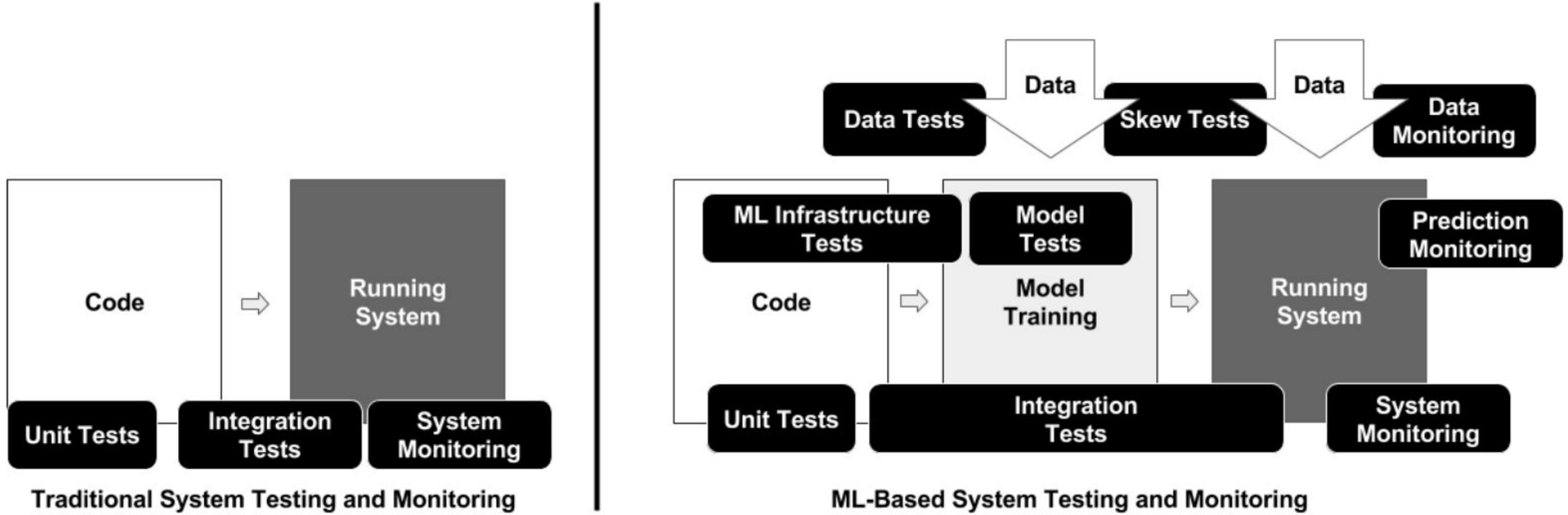
Reliability of ML projects is much more difficult to grasp.

Data and **Models** are non-deterministic objects…

Complex to specify with rules that can be easily translated to languages understood by machines.



Credits: *A Rubric for ML Production Readiness and Technical Debt Reduction*
Eric Breck et al., Google Inc.



Credits: *A Rubric for ML Production Readiness and Technical Debt Reduction*
Eric Breck et al., Google Inc.

Let’s list some of those extra challenges…

Does the model you want to deploy in production perform better than the current one?

What is the cost of this improvement?

processing time, computing resources etc...

Provided that a new model performs better than the current one, **what is the business value added by this performance gain?**

Is it worth the extra cost?

Is the data used to train/retrain your models still compatible with the pipeline?

schema, data types, data distribution…

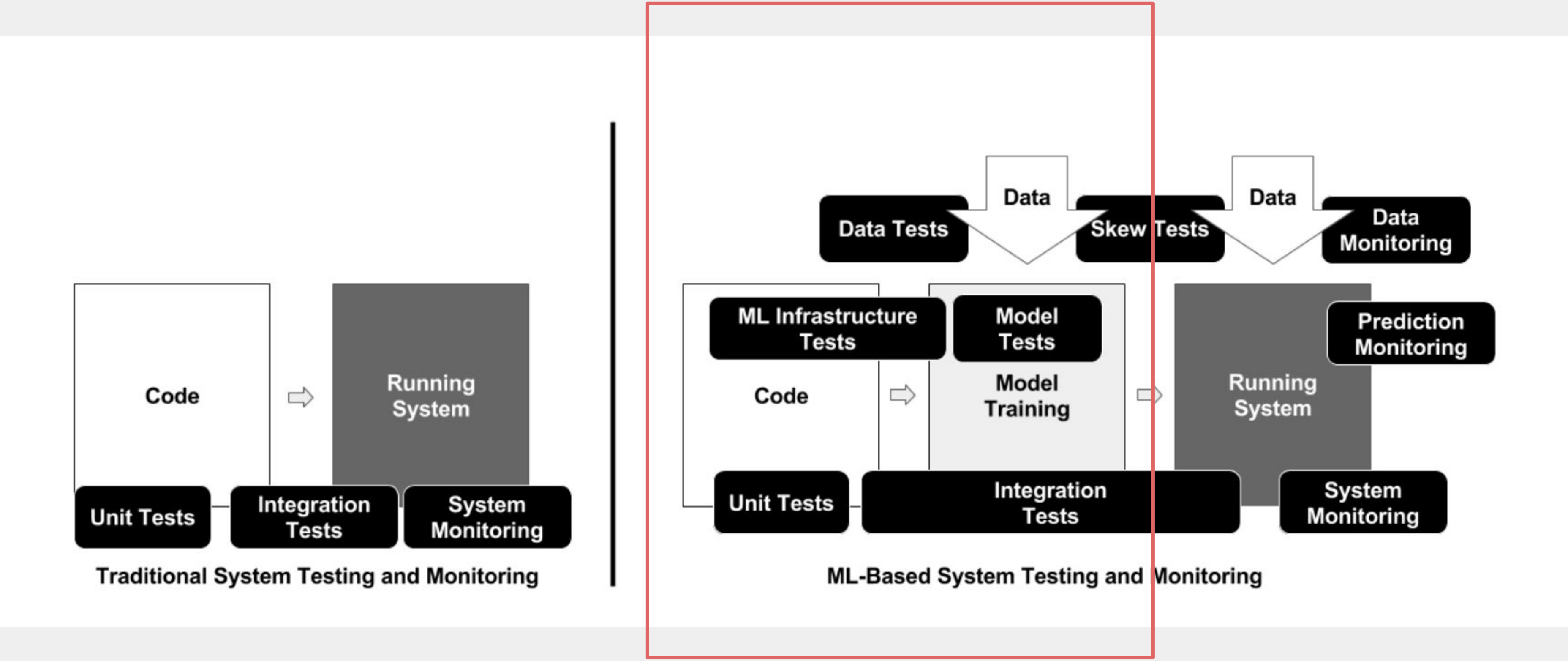
How to check that data leakage was not introduced during the last model optimization?

Your face right now...

“Do we have access to recipes to answer those questions?”

“Do we have access to recipes to answer those questions?”

No.



Traditional System Testing and Monitoring

ML-Based System Testing and Monitoring

Our focus for the coming days (roughly)

First Part — Contextualization

- Introduction
- Different types of testing
- The Testing Pyramid of Software
- **Extra Challenges for Machine Learning Projects**

First Part — Contextualization

- Introduction
- Different types of testing
- The Testing Pyramid of Software
- Extra Challenges for Machine Learning Projects

And now, for something completely different.

John Cleese, Monty Python

Part II — Focus on End-to-End testing

- Why / What— Purposes, usage and attributes
- How — Tools to use to write them
- Presentation of today’s workshop

Part II — Focus on End-to-End testing

- **Why / What — Purposes, usage and attributes**
- How — Tools to use to write them
- Presentation of today’s workshop

E2E Testing — Why & What

- Highest level of functional testing
- Check the behavior **at the final user level** (black box)
- Ideally **run in a dedicated environment**, close to production
- Fully integrated
- **Heavy to set up, slow to run**

80% of the work is

- Designing one or several test scenario
- Preparing test data or any other input
- Preparing a testing environment (web server, database credentials…)

Part II — Focus on End-to-End testing

- Why / What — Purposes, usage and attributes
- **How — Tools to use to write them**
- Presentation of today’s workshop

Have you ever done e2e testing?

E2E Testing — Tools

- Selenium to automate User Interface actions
- For systems without UI:
  - Bash / shell scripts
  - Python
  - JavaScript
  - … or any flexible scripting language!
- Continuous Integration servers to run them nightly

E2E Testing — Tools

- Selenium to automate User Interface actions
- For systems without UI:
  - Bash / shell scripts
  - Python
  - JavaScript
  - … or any flexible scripting language!
- Continuous Integration servers to run them nightly

Suggested tools for today

End-to-end test must be **as close as possible** to the final user perspective!

REMINDER

Example

End-to-end test of the webservice https://ipecho.net/plain

This service replies with the public IP address of the requester (ie, your browser, your terminal, or whatever web client you use).

We can test this service using dozens of different technologies. These examples use **Bash** and **Python**.

```bash
#!/bin/bash


SERVICE_URL=https://ipecho.net/plain

response=$(curl -s $SERVICE_URL)
echo "My IP address is $response"
```

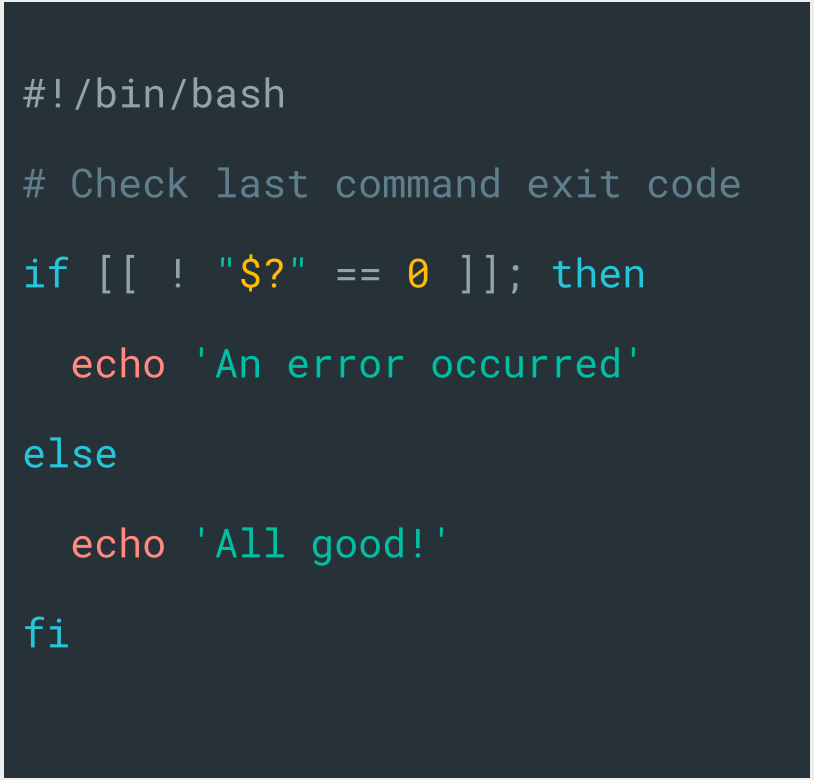
```python
#!/usr/bin/env python

import re
import requests

SERVICE_URL = 'https://ipecho.net/plain'
IPV4_REGEX = r'^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$'


resp = requests.get(SERVICE_URL)

assert resp.status_code == 200
assert re.fullmatch(IPV4_REGEX, resp.text) is not None
```

Refresher on basics
&
Useful tips

A *running program* is seen by the OS as a **process**

Exit codes

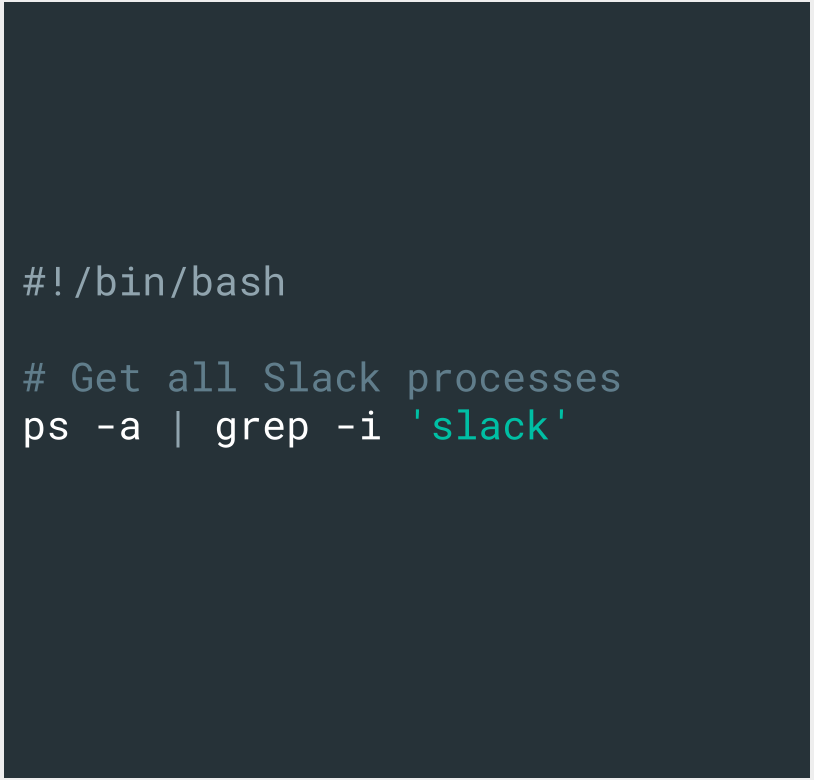
- 0 → success
- 1-255 → an error occurred



```bash
#!/bin/bash
# Check last command exit code
if [[ ! "$?" == 0 ]]; then
  echo 'An error occurred'
else
  echo 'All good!'
fi
```

List all running processes

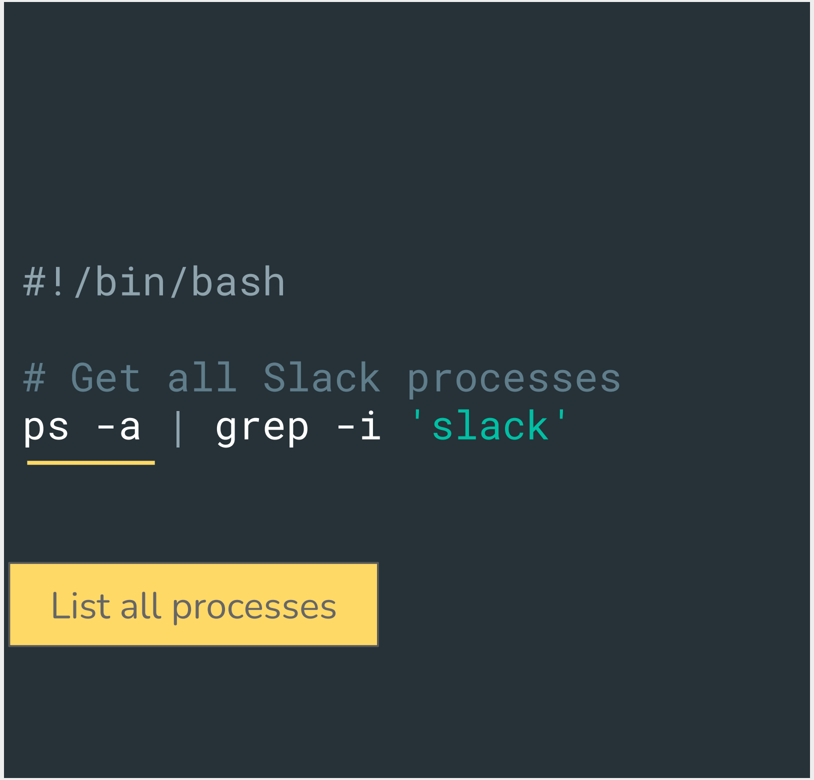
… and filter only processes of interest, by name.



```bash
#!/bin/bash

# Get all Slack processes
ps -a | grep -i 'slack'
```

List all running processes

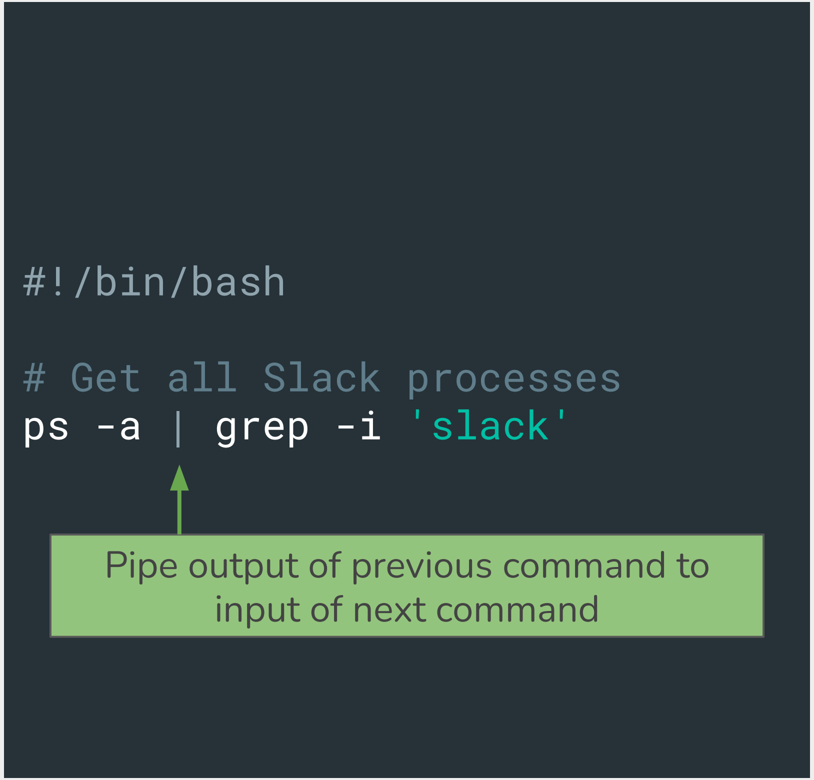
... and filter only processes of interest, by name.



```bash
#!/bin/bash

# Get all Slack processes
ps -a | grep -i 'slack'
```

List all running processes

... and filter only processes of interest, by name.

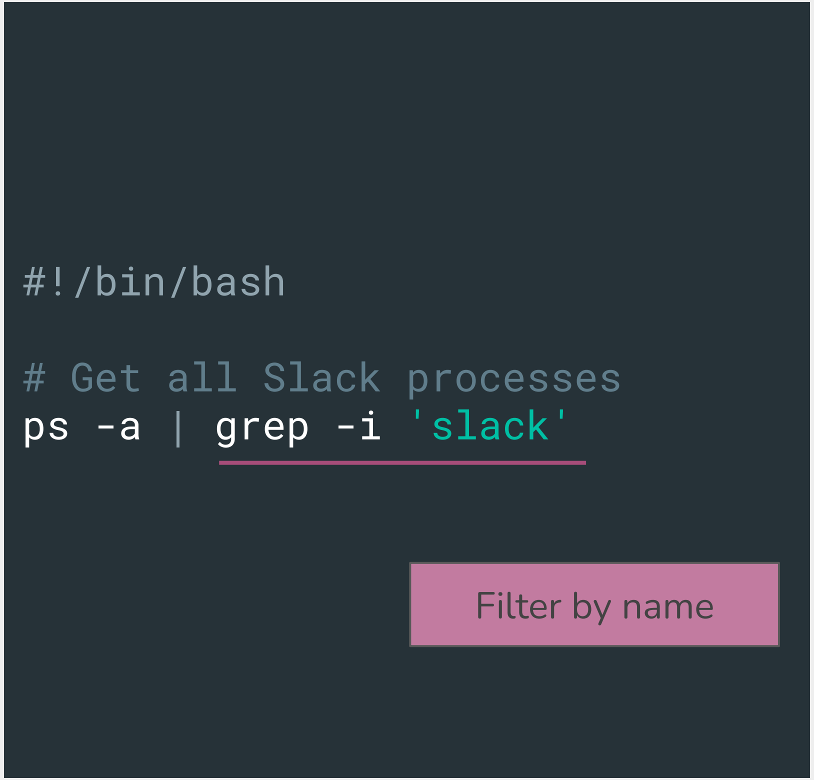


```bash
#!/bin/bash

# Get all Slack processes
ps -a | grep -i 'slack'
```

Pipe output of previous command to input of next command

List all running processes

... and filter only processes of interest, by name.

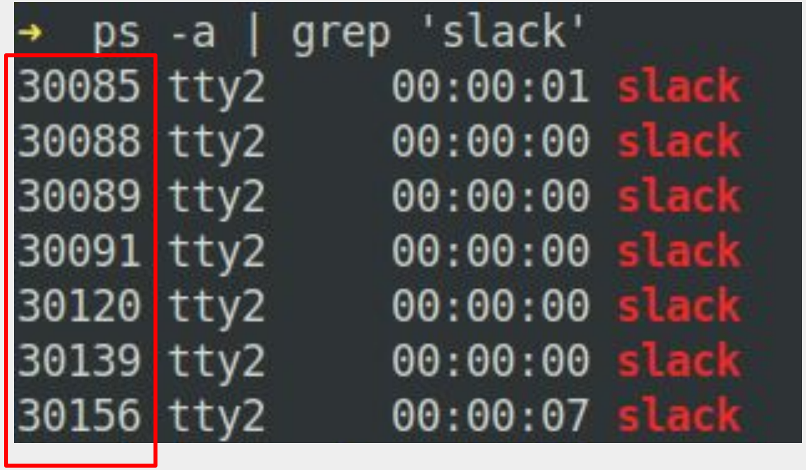


```bash
#!/bin/bash

# Get all Slack processes
ps -a | grep -i 'slack'
```

Filter by name

Processes have number(ID)

This is how an Operating System can identify them.



```
➜  ps -a | grep 'slack'
30085 tty2     00:00:01 slack
30088 tty2     00:00:00 slack
30089 tty2     00:00:00 slack
30091 tty2     00:00:00 slack
30120 tty2     00:00:00 slack
30139 tty2     00:00:00 slack
30156 tty2     00:00:07 slack
```

Kill a process using its number

The **-9** parameter is used for "force-kill"

```
#!/bin/bash

kill -9 30085 30088 ...
```

Navigate in directories and list content

$ → reference a variable

&& → execute if previous succeeded

```
#!/bin/bash

DIRNAME=/home/george

cd $DIRNAME && ls -la
```

Manage processes using Python

The `subprocess` module allows to do any shell command you would do in a terminal or a bash script.

```python
import subprocess

cmd = 'ping -c 4 google.com'
exit_code = subprocess.call(cmd.split())

if exit_code:
    print("Failed to reach Google.com")
else:
    print("Google.com is reachable.")
```

https://docs.python.org/3/library/subprocess.html

Part II — Focus on End-to-End testing

- Why / What — Purposes, usage and attributes
- How — Tools to use to write them
- **Presentation of today's workshop**

Predict point outcomes of tennis matches

A simple app controlled by a CLI, with **4 main components**:

`train` — generate features, and fit a model

`deploy` — deploy a fitted model “to production”

`predict` — run prediction using the currently deployed model

`serve` — expose the prediction service through an HTTP REST API

Print out the CLI documentation

```bash
#!/bin/bash

python -m src.main --help
```

Run a training experiment

data/ directory must contain **australian_open.csv.**

Experiments are then logged in **output/** directory, with their unique IDs.

```
#!/bin/bash


python -m src.main --train
```

Deploy a model

AWS credentials must be set in a `.env` file at the project’s root directory.

Also, at least one experiment must have run.

```
#!/bin/bash

python -m src.main --deploy
```

Run a prediction

Sample data need to be fed as prediction input!

```bash
#!/bin/bash



python -m src.main \
    --predict \
    --input=...
```

Serve predictions from HTTP

Endpoint is accessible at http://localhost:5000/predict

```
#!/bin/bash



python -m src.main --serve
```

Problem

Code base is not tested…

Problem

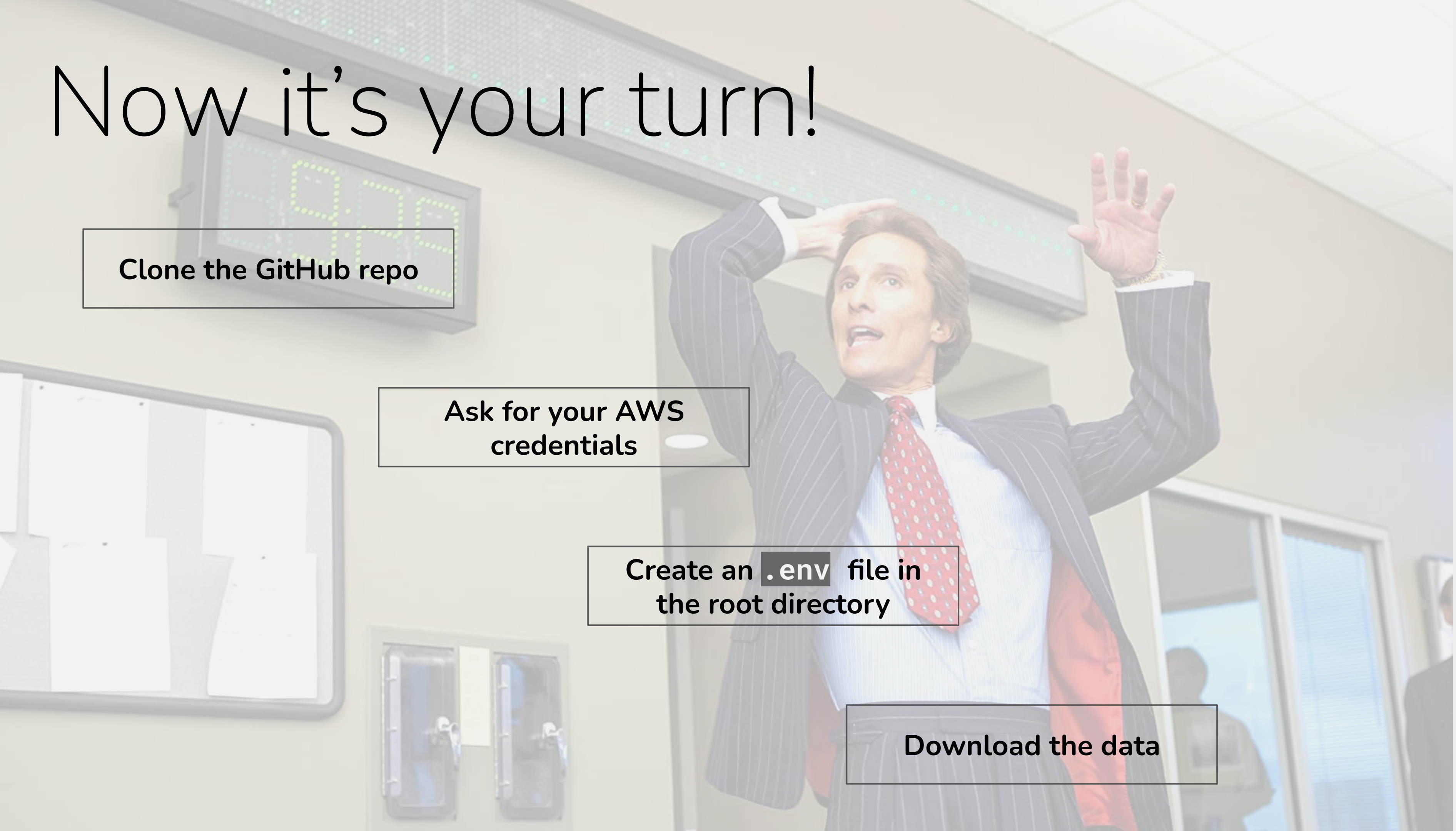
Code base is not tested...

Goal

Implement a robust test coverage that will greatly improve the value of the project.

First step: write some end-to-end tests

Now it’s your turn!



Clone the GitHub repo

Ask for your AWS credentials

Create an `.env` file in the root directory

Download the data

Let’s do a demo together…