Testing, CI, CD

Why testing?

- Ensure that code behaves as expected.
- Avoid breaking changes.

assert

unittest

Unit testing

Testing individual units of a software.

Writing tests

- After writing code.
- Before writing code (aka Test-Driven Development or TDD).

A good test case

- Focuses on a small and specific functionality.
- Independent from other test cases.
- Representative.

Good practices

- Explore testing framework(s).
- Run test suite before and after making changes.
- Replicate bugs with test cases.

Testing Django Apps

Testing Models

Testing Views

Testing Frontend (Selenium)

`unittest.TestCase` API

`setUp`, `setupClass`

`tearDown`, `tearDownClass`

`assertEqual`, `assertNotEqual`

`assertTrue`, `assertFalse`

`assertIn`, `assertNotIn`

`assertIs`, `assertIsNot`

`assertRaises`

`...`

Continuous Integration and Deployment (CI/CD)

Continuous Integration

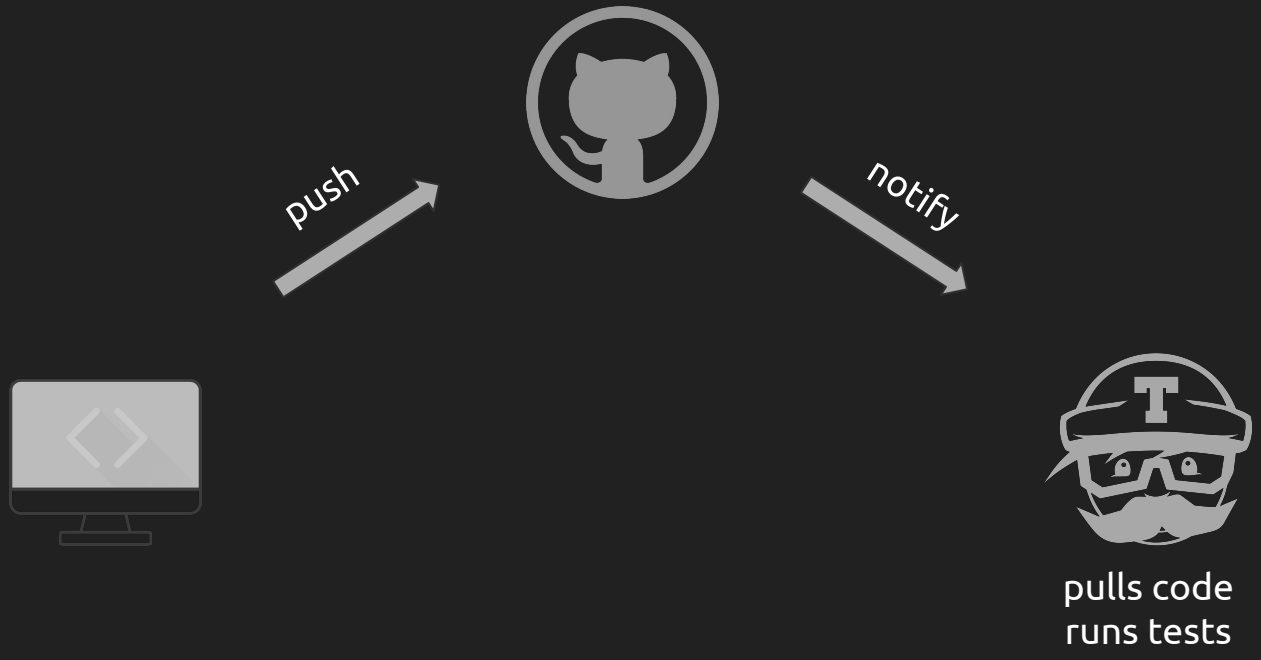
Automatically testing code to verify its integrity.

Continuous Integration Tools

- CircleCI
- Codeship
- Travis CI
- ...

Travis CI

Workflow



.travis.yml

.travis.yml

- A YAML file.
- Specifies what Travis CI should do upon getting notified.
- Example format:

```
key1: value1
key2: value2
key3:
  - item1
  - item2
key4:
  nested_key: value
...
```

.travis.yml

```
language: python
python: 3.6
install: pip install -r requirements.txt
script: python manage.py test
...
```

.travis.yml

```
...
services: postgresql
...
```

Continuous Deployment

Automatically deploying code after integrating it.

.travis.yml

```
...
deploy:
  provider: heroku
  api_key: $HEROKU_API_KEY
  app: kzidane-airline
  run: python manage.py migrate
  on: master
```