Lessons we learned developing infrastructure software at Box

April 2018, Aleksandr Kuzminsky
Today’s agenda

1. Branching model
2. Tests
3. Dependencies
4. Artifacts
Git Flow: Develop and Master Branches
Git Flow: Feature Branches
Git Flow: Release Branches
Git Flow: Hotfix Branches
Security and Compliance on direct git-push
Box flow: based on git-flow

```sh
# git flow feature start innodb-fast
...
# git flow feature publish innodb-fast
```

- ✒️ PR ➔ develop branch
- ✔️ CI must pass
- 👀 CR is optional
Box flow: develop branch protection

- Protect this branch
- Require status checks to pass before merging
- Include administrators
Box flow: release

# git flow release start 1.2.3
# bumpversion patch
# git flow release publish 1.2.3

- 📝 PR ➔ develop, PR ➔ master
- ✔️ CI tests must pass
- 👀 CR is mandatory
- 🚫 Comment fixes go to release/1.2.3
Box flow: master branch protection

- Protect this branch
- **Require pull request reviews before merging**
- Require status checks to pass before merging
- Include administrators
Box flow: hotfixes

# git flow hotfix start 1.2.4
# bumpversion patch
# git flow release publish 1.2.4

- 📝 PR ➔ develop, PR ➔ master
- ✔️ CI tests must pass
- 🕵️ CR is mandatory
- 🌀 Comment fixes go to hotfix/1.2.4
Tests
Unit tests

• Test Driven Development
  • For critical classes
  • For bugs
• pytest for computations and files
• unittest.mock for external resource and code
  • MySQL instances, APIs, already tested code
Tests hierarchy

```
- tests
  - development
  - unit
    - cli
    - client
      - magpie_client
        - __init__.py
        - test_atomic_open.py
        - test_get.py
        - test_get_url_from_srv.py
        - test_init.py
        - test_post.py
        - test_random_srv_record.py
        - test_set_config.py
        - test_view_config.py
        - __init__.py
        - conftest.py
  - proxy
```
Tests hierarchy
Tests hierarchy
Tests hierarchy

Methods
pytest
def test_multiply():
    assert multiply(2, 3) == 6
pytest: write test that fails

```python
from pydata import multiply

def test_multiply():
    assert multiply(2, 3) == 6

def multiply(x, y):
    return 0
```

```
$ pytest test/unit/test_demo.py
>     assert multiply(2, 3) == 6
E     assert 0 == 6E
+   where 0 = multiply(2, 3)
test/unit/test_demo.py:5: AssertionError
```
pytest: fix implementation

```python
from pydata import multiply

def test_multiply():
    assert multiply(2, 3) == 6

def multiply(x, y):
    return x * y
```

```
$ pytest test/unit/test_demo.py
test/unit/test_demo.py::test_multiply PASSED [100%]
```
pytest: parameterized tests

```python
@pytest.mark.parametrize('x, y, result', [
    (2, 3, 6),
    (2, 2, 4)
])
def test_multiply(x, y, result):
    assert multiply(x, y) == result
```
@pytest.mark.parametrize('x, y', [(None, 1), (1, None), (None, None)])
def test_multiply(x, y):
    with pytest.raises(ValueError):
        multiply(x, y)
To mock or not to mock (for unit tests)

- **Mock:**
  - API calls
  - External code calls
  - Database queries
  - Anything external
    - already tested code

- **No Mock:**
  - Calculations
  - Objects manipulations
  - Anything that works with CPU, memory, files
pytest: mock API calls

```python
@mock.patch('pydata.call_api')
def test_multiply(m_api):
    multiply(2, 3)
    m_api.assert_called_once_with(
        "http://x.com/x=2&y=3"
    )
```

```python
def multiply(x, y):
    result = call_api(
        "http://x.com/x=%d&y=%d"
        % (x, y)
    )
    return result
```
Further watching

Michael Tom-Wing, Christie Wilson - Introduction to Unit Testing in Python with Pytest - PyCon 2016
https://youtu.be/UPanUFVFFzY

Bay PIGgies March 2017: Testing in Layers
https://youtu.be/pCrW_LiSb34
Dependencies
Artifacts: Local PyPI repo

![PyPI Server](https://pypi.org/project/pypiserver/)
$ cat ~/.pip/pip.conf
[global]
extra-index-url = https://pypi.box.com/simple
Packages
RPM Packages with Omnibus

- **Full-stack installers – RPM with no dependencies**
  - Includes Python+libs
  - No dependency hell
  - Doesn’t break existing packages
  - Python/modules version flexibility
  - [https://github.com/chef/omnibus](https://github.com/chef/omnibus)

- Store in your Yum Repo of choice
Thank you!

Also, We’re Hiring!
Picture credits

- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow