Introduction

● Who am I:
  ○ Remi Duraffort
  ○ OSS developer since 2007
    ■ VLC, v8, LAVA, ...
  ○ LAVA maintainer

● Some ideas about testing in/with python
  ○ Based on my own experience
  ○ LAVA, lavafed, meta-lava, ReactOBus, Artifactorial, KissCache, ...

● Some recommendations
Code format and static analysis
Code format and static analysis

- bandit
- black
- dodgy
- pep8
- prospector
- pycodestyle
- pyflakes
- pylint
- vulture
- ...

...
Static analysis/formatting

- bandit
- black
- dodgy
- pep8
- prospector
- pycodestyle
- pyflakes
- pylint
- vulture
- ...

- Basic static analyzer
- Reliable
- Code formatting tool
- Becoming the standard
- Don’t bother about formatting anymore
- Useless, except if called with the right arguments
- `pylint3 --disable=all --enable=elif,exceptions,stdlib,imports,variables,string,string_constant,logging,newstyle,classes --disable=C0411,C0412,E0401,E0611,R0201,R0401,W --enable=W0404,W1505,W0611`
Unit tests
Which framework

- unittest
- nose
- nose2
- pytest
- ...

Other frameworks might include:

- PyCharm
- Eclipse
- IntelliJ IDEA
- Visual Studio
- Sublime Text
- Atom
- VS Code
import unittest

class TestStringMethods(unittest.TestCase):
    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

if __name__ == '__main__':
    unittest.main()
def test_upper():
    assert 'foo'.upper() == 'FOO'
Which framework

- unittest
- nose
- nose2
- pytest
pytest features

- Just use assert
  - assertEqual
  - assertIsNone
  - assertFalse
  - ...

- **Capture stdout/stderr**
  - Print during tests
  - Visible only when failing
    - By category
  - Test stdout/stderr content

- **Capture logging**

```python
def test_myoutput(capsys):
    print("hello")
sys.stderr.write("world\n")
captured = capsys.readouterr()
assert captured.out == "hello\n"
assert captured.err == "world\n"

print("next")
captured = capsys.readouterr()
assert captured.out == "next\n"
```
pytest features

- **Plugins**
  - Django
  - Asyncio
  - Coverage

- **Monkey patching**
  - Some functions might depend on global settings or external resources
    - Network access
    - Looking at the PATH
    - ...
  - Replace functions/builtins/environment
    - with own implementation/values
  - Examples:
    - Replace network access by a call to a static function
    - Always return the same results (success/error)
    - Do not depend on external conditions
      - Reliable/reproducible/faster
pytest features

```python
import pytest
import requests

import app

class MockResponse:
    @staticmethod
    def json():
        return {"mock_key": "mock_response"}

@pytest.fixture
def mock_response(monkeypatch):
    """Requests.get() mocked to return {'mock_key': 'mock_response'}."""
    def mock_get(*args, **kwargs):
        return MockResponse()
    monkeypatchsetattr(requests, "get", mock_get)

def test_get_json(mock_response):
    result = app.get_json("https://fakeurl")
    assert result["mock_key"] == "mock_response"
```
pytest features

```python
monkeypatchsetattr(obj, name, value, raising=True)
monkeypatchdelattr(obj, name, raising=True)
monkeypatchsetitem(mapping, name, value)
monkeypatchdelitem(obj, name, raising=True)
monkeypatchsetenv(name, value, prepend=False)
monkeypatchdelenv(name, raising=True)
monkeypatchsyspath_prepend(path)
monkeypatchchdir(path)
```

- Changes will be reverted after the test
monkey patch use cases

- HTTP/XML-RPC calls
- Environment variables
  - PATH, ...
- Signals
  - Test code using signal.alarm
- Time
  - Making time.sleep(N) a nop
  - Or the test would take a long time
- Suprocess
  - Subprocess.check_output
    - Return the output without running the process
    - Raise an error
      - Might be sometime difficult to reproduce error conditions
- File system access
  - Calling open(“/etc/passwd”) should be mocked!
monkey patching filesystem

real_open = open

def monkey_open(path, *args):
    if path == "/etc/lava-server/dispatcher-config/health-checks/qemu.yaml":
        return real_open(str(tmpdir / "qemu.yaml"), *args)
    if path == "/etc/lava-server/dispatcher-config/health-checks/docker.yaml":
        raise FileNotFoundError()
    if path == "/etc/lava-server/dispatcher-config/health-checks/docker2.yaml":
        raise PermissionError("permission denied", "permission denied")
    return real_open(path, *args)

monkeypatch.setitem(__builtins__, "open", monkey_open)
System mocking
Integration testing

- Test the full system
  - Errors that unit tests can’t detect
  - Test performances
  - Stress test
  - Fault injection
  - ...

- Use cases:
  - Is LAVA able to run 500 jobs in parallel?
  - Can LAVA boot a set of specific boards?
  - ...

Running 500 jobs in parallel?

- **Hardware:**
  - 500 boards
  - ~100 servers
  - Many people to plug everything
  - Some board and server **will** fail
- **Not reliable**

- **Mock some part of the system**
  - Use only one server
  - Mock the right part
    - Keep the test effective
Running 500 jobs in parallel?

Users

server

dispatcher 1

dispatcher 2

dispatcher N

tftp&nfs server

Power control

Serial relay
Running 500 jobs in parallel?
Running 500 jobs in parallel?
Running 500 jobs in parallel?
Running 500 jobs in parallel?
Running 500 jobs in parallel?

- Looks like lava-run
  - Same command line
  - Handle signals
  - Return value
  - Sending logs
  - Right format
  - Similar speed
- Not using CPU/RAM/IO/...
- Every services running normally
Chaos engineering
Chaos engineering

“The intent was to move from a development model that assumed no breakdowns to a model where breakdowns were considered to be inevitable”

- Introduce errors on a production system
- Monitor the system
  - Self healing?
  - Resilient?
- Chaos Monkey:
  - Stop servers randomly
- ...
- Invented and used by Netflix
Questions?
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org
pytest features

- Exceptions
  - Assert exception raised
  - Test content

```python
with pytest.raises(FileNotFoundError) as exc:
    function_that_should_raise()
assert exc.strerror == "No such file or directory"
assert exc.errno == 2
```
pytest features

- **Fixtures**
  - Setup, tear down, ...
  - Autouse if needed

```python
import pytest

def setup():
    return dict(hello="world")

def test_function(setup):
    assert setup["hello"] == "world"
```