“A Unit Test Would Have Caught This”
Fast and cheap testing for production engineers

Andrew Ryan, Facebook Inc.
SRECon Asia 2017
We all write buggy code sometimes...

def calc_5min_rate(pkts_now, pkts_5min):
    if pkts_now - pkts_5min > 0:
        rate = (pkts_now - pkts_5min) / 300
    elif pkts_now - pkts_5min < 0:
        rate = 0
    return rate

    But what if pkts_now == pkts_5min?

    NameError: name 'rate' is not defined
#!/bin/bash

tmpdir=$(mktemp -d)

#... do some stuff

# Cleanup and exit
rm -rf $tmp/*
exit 0

#... do some stuff

# Cleanup and exit
rm -rf /*
exit 0
Why Production Engineers don’t test

Actual lame excuses I’ve personally used

• “It’s just a shell script”
• “{I don’t have time/I don’t know how} to write tests”
• “I’m not a software engineer”
• “I ran it. It seems to work.”
• “My code has too many external dependencies”
Code we write as Production Engineers

1. **Shell scripts**
2. **Tooling** in Python/Perl/Ruby/Go/etc.
3. **Configuration management** (Chef, Puppet, Ansible, cfengine, etc.)
Testing Shell Scripts
The best way to test shell scripts is not to write them.
If you must write shell scripts (1)

Protect yourself at runtime

```
#!/bin/bash -ue
```

- **u**  Error on unset variables
- **e**  Exit if any command exits non-zero
If you must write shell scripts (2)

Lint!

```bash
#!/bin/bash

tmpdir=$(mktemp -d)

# ... do some stuff
write_to_file > "tmpdir/output"

# Cleanup and exit
rm -rf "tmp/*"
```

Shellcheck
http://shellcheck.net

```bash
$ shellcheck myscript
Line 9: rm -rf "tmp/*"
  ^-- SC2115: Use "${var:?}" to ensure this never expands to /* .
  ^-- SC2154: tmp is referenced but not assigned.
```

Lint!
If you must write shell scripts (3)

Unit test!

**BATS**
https://github.com/sstephenson/bats

**shunit2**
https://github.com/kward/shunit2
Testing in Python
Lint!

flake8
http://flake8.pycqa.org/

yapf
https://github.com/google/yapf

pylint
https://www.pylint.org
Unit testing

def calc_5min_rate(pkts_now, pkts_5min):
    if pkts_now - pkts_5min > 0:
        rate = (pkts_now - pkts_5min) / 300
    elif pkts_now - pkts_5min < 0:
        rate = 0
    return rate

class testCalcRate(unittest.TestCase):
    def test_calc_5min_rate(self):
        self.assertEqual(calc_5min_rate(3300, 300), 10) #PASS
        self.assertEqual(calc_5min_rate(300, 600), 0)   #PASS
        self.assertEqual(calc_5min_rate(600, 600), 0)   #FAIL
Unit testing changes the way you write code
def tmp_cleaner():
    subprocess.check_output(
        "rm -rf /tmp/foo*", shell=True
    )

@mock.patch('subprocess.check_output')
def test_tmp_cleaner(self, mock_check_output):
    tmp_cleaner()
    mock_check_output.assert_called_with(
        "rm -rf /tmp/foo*", shell=True
    )
Testing in Chef
Configuration *is* code
Lint!

rubocop: general Ruby linter
https://github.com/bbatsov/rubocop

foodcritic: Chef-specific Ruby style
http://www.foodcritic.io
Unit tests for Chef cookbooks

Examples: https://github.com/facebook/chef-cookbooks

```ruby
require_relative '../libraries/timers.rb'

describe FB::Systemd::Calendar::Every do
  it 'generates hours' do
    expect(FB::Systemd::Calendar.every(7).hours).to eq('0/7:0:0')
    expect(FB::Systemd::Calendar.every(15).hours).to eq('0/15:0:0')
  end

  it 'generates minutes' do
    expect(FB::Systemd::Calendar.every(7).minutes).to eq('*:0/7:0')
    expect(FB::Systemd::Calendar.every(15).minutes).to eq('*:0/15:0')
  end
end
```
Test on one or more servers

taste-tester: Test on 1-2 servers
https://github.com/facebook/taste-tester

Gate changes by host/rack/etc.

if condition_met
    # apply new chef policy
end
Closing thoughts
People come and go, but a good test lasts forever
各种口味的甜甜圈展示在货架上，每种甜甜圈都有相应的标签和价格。货架上标有价格为NT$35、NT$40、NT$45等。不同的口味和形状吸引着顾客的注意。背景中可以看到一些装饰和顾客的身影。
Where to go from here

Integration with code review tools
Phabricator, Review Board, etc.

Integration with CI tools
Jenkins, Travis CI, KitchenCI, Serverspec, etc.
facebook