facebook

“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

...and some of those bugs are really bad

```
#!/bin/bash
tmpdir=$(mktemp -d)

#... do some stuff

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

```
#!/bin/bash
tmpdir=$(mktemp -d)
#...
#... do some stuff
#...
# Cleanup and exit
rm -rf /*
exit 0
```




Golden.ie

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!

```
1  #!/bin/bash
2
3  tmpdir=$(mktemp -d)
4
5  # ... do some stuff
6  write_to_file > "$tmpdir/output"
7
8  # Cleanup and exit
9  rm -rf "$tmp/*"
```

ShellCheck Output

```
$ shellcheck myscript

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

$
```

Shellcheck

http://shellcheck.net

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

Mocks: taming external dependencies

```
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`

```
11  require_relative '../libraries/timers.rb'
12
13  describe FB::Systemd::Calendar::Every do
14    it 'generates hours' do
15      expect(FB::Systemd::Calendar.every(7).hours).to eq('0/7:0:0')
16      expect(FB::Systemd::Calendar.every(15).hours).to eq('0/15:0:0')
17    end
18
19    it 'generates minutes' do
20      expect(FB::Systemd::Calendar.every(7).minutes).to eq('*:0/7:0')
21      expect(FB::Systemd::Calendar.every(15).minutes).to eq('*:0/15:0')
22    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


FREE

People come and go, but a good test lasts forever


25
糖霜多拿滋
25
蜜糖多拿滋
35
Plain Old Ring
蜜糖波堤
蜜糖抹茶波堤
NT$35
抹茶多拿滋
NT$30
抹茶歐蕾香
NT$35
雙層抹茶波堤
NT$40
天使抹茶波堤
NT$40

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