What is unit testing

Arthur Freyman

golovast@gmail.com

www.mindend.com

What is unit testing

- Break down the project into small pieces
- Test them independently

What is unit testing + devops

- Infrastructure testing (AWS, Openstack, Azure, etc)
- Configuration management (Ansible, Chef, Puppet, Salt)
- Scripts/tools
- Other products

Why write them?

- Larger teams/complex systems
- Future proofing (somewhat)
- The rest of the organization.
- Test your logic

When to write them?

- Better as you go. More difficult later
- Tons of formal systems
  - TDD
  - BDD
  - ATDD
- Your code will be better.

Examples (Python)

- AWS script example:

```
import boto.ec2
def create_instance(options, region, key, sec_groups, subnets):
    conn = boto.ec2.connect_to_region(region)
    try:
        reservation = conn.run_instances(image_id=options.OS,
                        security_group_ids=sec_groups, subnet_id=subnets,
                        instance_type=options.size, min_count=options.Count,
                        max_count=options.Count, key_name=key, dry_run=False)
        return reservation
    except Exception as e:
        print "Couldn't launch instance because of:"
        print e
```

Examples (Python)

- Unit test example:

```
import create_instance
import unittest
from moto import mock_ec2
args = {}
sec_groups = 'sg-1234'
subnets = None
args.Key = 'test_key'
args.Size = 't3.medium'
args.Count = 1
args.OS = 'ami-1234'

@mock_ec2
def test_create_instance():
    conn = boto.ec2.connect_to_region('us-west-1')
    create_instance(args, 'us-west-1', args.Key, sec_groups, subnets)
    reservations = conn.get_all_instances()
    assert len(reservations) == 1
```

Examples (Python)

- Mocks – when resource unavailable, unsuitable, slow, etc.
- Stub – kind of like a mock, but can’t verify methods. Usually canned answers.
- Fake – Class that implements an interface but has fixed data and no logic.

Examples (Python)

- Script:

```
import os
dir_path = "/tmp/"
file_name = "text.txt"

def search_for_file(dir_path, file_name):
    file_list = []
    for f in os.listdir(dir_path):
        if os.path.isfile(os.path.join(dir_path, f)) and f == file_name:
            file_list.append(file_name)
    if not file_list:
        return “not found”
    else:
        return file_name

search_and_remove_file(dir_path, file_name)
```

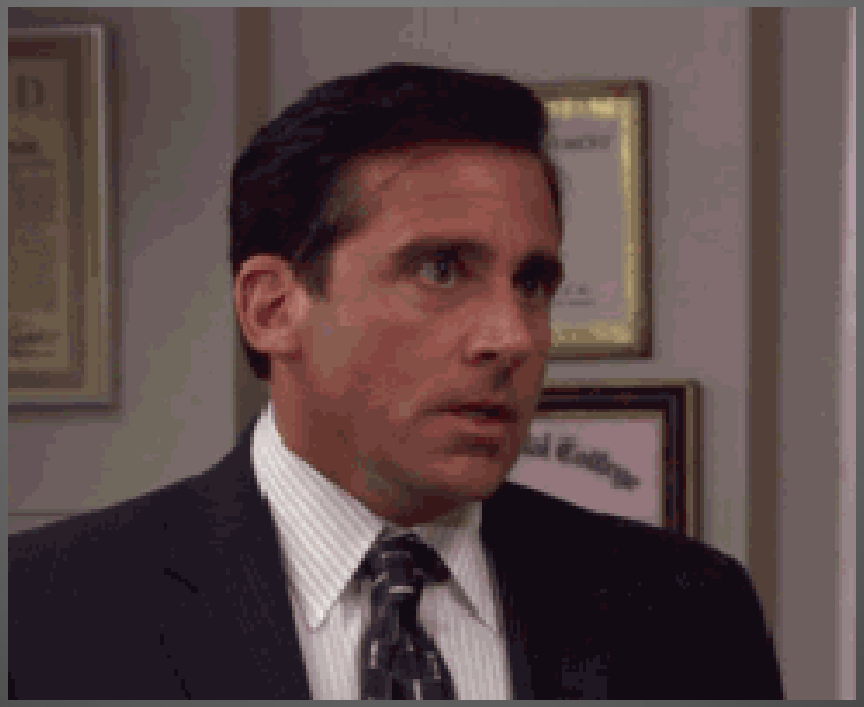
Examples (Python)

- Unit Test Example:

```
import search_and_remove_file
import unittest
from mock import patch

dir_path = "/tmp/"
file_name = "text.txt"

class MyTests (unittest.TestCase):

    def test_file_found(self):
        with patch('os.listdir', return_value=['text2.txt']):
            test_value = search_for_file(dir_path,file_name)
            self.assertEqual(test_value, file_name)

    def test_not_found(self):
        with patch('os.listdir', return_value=['text4.txt']):
            test_value = search_for_file(dir_path,file_name)
            self.assertEqual(test_value, "not found")

if __name__ == '__main__':
    unittest.main()
```

Examples (Shell)

- I am not going to do it. ☺
  - But check out bats (https://github.com/sstephenson/bats) if you have to do it. Also roundup and shunit2.



Examples (Ansible)

- Version your roles and maybe playbooks.
- Design your workflow
  - Branching
  - Builds
- Test your expectations
  - Files
  - Services
  - Packages
  - Users
  - Multiple variations.

Examples (Ansible)

- Keep the roles/manifests/cookbooks/states small
  - Unix philosophy – do one thing and do it well
  - Keep it modular
  - DRY
- Some java app
  - Bad: single role for configuring the system, installing prereqs and pushing the app
  - Good:
    - Role/jvm
    - Role/tomcat
    - Role/users
    - Role/sys_config
    - Role/code_deploy

Examples (Ansible)

- Complex role/cookbook/manifest/state
- Nginx as an example:
  - Could install with different modules (compiled and defaults)
  - From source/package
  - Different versions
  - Default sites with different configurations (proxy, etc)
- Validate as much as you can.
  - Especially complex conditionals.
  - Templating engines
  - Multiple sets of testing vars

Examples (Ansible + Friends)

- Ansible-lint (https://github.com/willthames/ansible-lint)
  - Defaults are good, but add your own things.
- Syntax check
  - Ansible-playbook –syntax-check –list-tasks –i '127.0.0.1' test.yml
- Assert module
- Serverspec, inspec, rolespec, testinfra, goss, chefspec, ansible_spec, test-kitchen
- Test playbooks

Examples (Ansible + Friends)

- Template validations (native tools)

```
vars:
- validate_conf: /usr/sbin/named-checkconf
tasks:
- name: install named.conf
action: tempalte src=named.conf.j2 dest=/etc/named.conf  validate = '{{ validate_conf }} %s'
```

- Other validators:
  - /usr/bin/nginx –t –c nginx.conf
  - Apachectl configtest
  - /usr/sbin/sshd –t
  - /usr/sbin/squid -k check
  - /usr/libexec/mysqld --defaults-file=/root/test-my.cnf --verbose --help 1>/dev/null
  - syslogd -f /etc/rsyslog.testing.conf -d

Examples (Ansible + Friends)

- Testinfra

```
def nginx_present(Package):
    nginx = Package(“nginx”)
    assert nginx.is_installed

def nginx_running_enabled(Service):
    nginx = Service(“nginx”)
    assert nginx.is_running
    assert nginx.is_enabled
```

Examples (Ansible + Friends)

- Serverspec

```
describe user('nginx') do
  it { should exist }
  it { should belong_to_group 'nginx'}
end

describe process('nginx') do
  it { should be_running }
  its(:user) { should eq 'nginx' }
end

describe port (80) do
  it { should be_listening }
end

describe "Server Configuration" do
  it 'should response with right status code' do
    uri = URI('http://localhost')
    http = Net::HTTP.new(uri.host)
    request = Net::HTTP::Get.new(uri)
    response = http.request(request)
    expect(response.code).to eq('200')
  end
end
```

Examples (Ansible + Friends)

- Goss – new tool, written in go.
  - Builds yaml files (sort of automatically)
  - ‘goss autoadd nginx’

```
package:
  nginx:
    installed: true
    versions:
    - 1.4.1-3ubuntu1.3
      nginx = Package(“nginx”)
      assert nginx.is_installed
service:
  nginx:
  enabled:true
  running:false
```

The End