From Monitoring to Automated Testing of your Infrastructure Code

Jesse Reynolds, Puppet
SREcon Singapore 2018
@jessereynolds
Professional Services Engineer at Puppet
Lives in Adelaide
Works everywhere but Adelaide
Infracode?
Writing tests has been a best practice in software engineering for a while now...
Infrastructure code is software, so you should write tests for it too.
Why bother?
Let me tell you a story...
There was this bank.
Both classes get applied.
All machines now have 1.1.1.1 and that’s not how you TCP/IP.
What went wrong?
What can be done to prevent this happening again?
Testing to the rescue!
OK, what do we need?
Provision a machine
Apply your code to the machine
Execute the tests
Examine the exit status
zero good
non-zero bad
The test harness

Time to make some choices
Welcome to Kitchen - KitchenCI

Infrastructure Code Deserves Tests Too

What is Kitchen?
Kitchen provides a test harness to execute infrastructure code on one or more platforms in isolation.

A driver plugin architecture is used to run code on various cloud providers and virtualization technologies such as Vagrant, Amazon EC2, and Docker. Read more

Many testing frameworks are supported out
Kitchen Puppet

gem version 3.3.2 downloads 151,016 total build passing

kitchen-puppet

A Test Kitchen Provisioner for Puppet

The providers supports both puppet apply and puppet agent clients and puppet bolt.
---

driver:
  name: vagrant

provisioner:
  name: puppet_apply
  manifests_path: /repository/puppet_repo/manifests
  modules_path: /repository/puppet_repo/modules-mycompany
  hiera_data_path: /repository/puppet_repo/hieradata

platforms:
- name: nocmUbuntu-12.04
  driver_plugin: vagrant
  driver_config:
    box: nocmUbuntu-12.04
    box_url: http://puppet-vagrant-boxes.puppetlabs.com/ubuntu-server-12042-x64-vbox4210-nocm.box

suites:
- name: default
require 'spec_helper'

if os[:family] == 'ubuntu'
  describe '/etc/lsb-release' do
    it "exists" do
      expect(file('/etc/lsb-release')).to be_file
    end
  end
end

if os[:family] == 'redhat'
  describe '/etc/redhat-release' do
    it "exists" do
      expect(file('/etc/redhat-release')).to be_file
    end
  end
end
A Test Kitchen Provisioner for Ansible.

The provisioner works by passing the Ansible repository based on attributes in `kitchen.yml` and calling `ansible-playbook`. It installs Ansible on the server and runs `ansible-playbook` using host `localhost`.

It has been tested against the Ubuntu 12.04/14.04/16.04, Centos 6/7 and Debian 6/7/8 boxes running in XUbuntu.
beaker / docs / README.md

 Staten's library connected to Muppet blog.

https://github.com/puppetlabs/beaker/blob/master/docs/README.md
RSpec tests for your servers configured by CFEngine, Puppet, Ansible, Itamae or anything else.

About V2

Serverspec/Specinfra v2 has been just released. See the document about v2.

About

With Serverspec, you can write RSpec tests for checking your servers are configured correctly.

Serverspec tests your servers’ actual state by executing command locally, via SSH, via WinRM, via Docker API and so on. So you don’t need to install any agent softwares on your servers and can use any configuration management tools, Puppet, Ansible, CFEngine, Itamae and so on.
Behaviour Driven Development for Ruby.
Making TDD Productive and Fun.
RSPEC-PUPPET
RSpec test framework for your Puppet manifests

Getting started is easy!
(The rest is a bit more difficult though)
Install the rspec-puppet gem and run `rspec-puppet-init` from inside your module to set everything up.

```
$ gem install rspec-puppet
$ cd your-module
$ rspec-puppet-init
```

Then continue on to the tutorial!

View on GitHub
Bats: Bash Automated Testing System

Bats is a TAP-compliant testing framework for Bash. It provides a simple way to verify that the UNIX programs you write behave as expected.

A Bats test file is a Bash script with special syntax for defining test cases. Under the hood, each test case is just a function with a description.

```bash
#!/usr/bin/env bats

test "addition using bc" { 
```
Test Anything Protocol

TAP, the Test Anything Protocol, is a simple text-based interface between testing modules in a test harness. TAP started life as part of the test harness for Perl but now has implementations in C, C++, Python, PHP, Perl, Java, JavaScript, and others.

Here’s what a TAP test stream looks like:

```
1..4
  ok 1 - Input file opened
  not ok 2 - First line of the input valid
  ok 3 - Read the rest of the file
  not ok 4 - Summarized correctly # TODO Not written yet
```

Testing with TAP

- Testing with TAP - How to run TAP based tests in your language of choice
“Reasons” to not write tests...
"It takes too long to write tests"
"Who is going to run them anyway?"
New team member joins

... or you join a new team
You can actually go holiday
Could you re-use your monitoring checks for infacode CI testing?
What’s your organisation doing?
Thank you for listening!

@jessereynolds