

Puppet RSpec

Why You Should, Too

Dennis Rowe

Senior DevOps Engineer
Cirrus Logic (cirrus.com)

[shr3kst3r](#) on Twitter, Github, and Freenode

- **Setup of Puppet RSpec**
- **Quick RSpec primer**
- **Review some RSpec code**
- **Opinions on testing**

**Testing is like free
kittens**



Not Testing?

TESTING IN THE AIR

GET OUT THE GAS MASKS

So, RSpec

[**https://github.com/rodjek/rspec-puppet**](https://github.com/rodjek/rspec-puppet)

[**http://rspec-puppet.com/tutorial/**](http://rspec-puppet.com/tutorial/)

[**https://github.com/puppetlabs/puppetlabs_spec_helper**](https://github.com/puppetlabs/puppetlabs_spec_helper)

First, Install RVM

- **RVM** = Ruby Version Manager
- **<https://rvm.io/>**
- **Allows us to manage ruby versions and gemsets**
- **Specify ruby version in repo/.rvmrc**
- **source ~/.rvm/scripts/rvm**

```
% curl -L https://get.rvm.io |  
bash -s stable --rails --  
autolibs=enabled
```

```
% rvm use ruby-1.9.3@puppet-  
camp-austin --install --create
```

```
% ruby -v
```

```
ruby 1.9.3p392 [x86_64-  
darwin12.2.1]
```

Gemfile Time

- **`http://rubygems.org`**
- **Specifies exactly what gems are used**
- **``bundle install`` installs the gems in our gemset**

Example Gemfile

```
source :rubygems
```

```
gem "puppet", "~> 3.1.1"
```

```
gem "puppetlabs_spec_helper", "~> 0.4.1"
```

Gemfile.lock

GEM

remote: <http://rubygems.org/>

specs:

diff-lcs (1.2.4)

factor (1.7.0)

hiera (1.2.1)

json_pure

json_pure (1.7.7)

puppet (3.1.1)

factor (~> 1.6)

RSpec Prep

- **mkdir -p spec/classes/puppet-camp-austin/**

- **Add Rakefile**

```
require 'rubygems'
```

```
require 'puppetlabs_spec_helper/rake_tasks'
```

- **Add spec/spec_helper.rb**

```
require 'rubygems'
```

```
require 'puppetlabs_spec_helper/module_spec_helper'
```


Setup **.fixtures.yml**

```
% cat .fixtures.yml
```

```
fixtures:
```

```
  symlinks:
```

```
    puppet-camp-austin: "#{source_dir}"
```

Quick Overview of RSpec Constructs

describe

- Expresses the concept of the **block**
- They can be **nested**
- The **first level describe should be a class, define, function, or host**

Example describe

```
describe "myfunction" do
  describe "it should do this" do
    ...
  end
  describe "it should also do this"
    ...
  end
end
```

lets

```
let(:title) { 'namevar' }
```

```
let(:params) {{ 'var1' => 'foo', ... }}
```

```
let(:node) { 'ahost.example.com' }
```

```
let(:facts) {{ :osfamily => 'Debian', ... }}
```

Matchers

- **include_class**
- **contain_{package, service, user, etc}**

should or should_not

```
it { should include_class('foobarz') }
```

```
it { should_not include_class('foobarz') }
```

```
it { should contain_package('fooz') }
```

```
it { should contain_user('barz') }
```

Now moving on

**So, let *us* add an
example class**

```
% cat manifests/example1.pp
```

```
class puppet-camp-austin::example1 {  
  
}
```

cat spec/classes/puppet-camp-austin/ example1_spec.rb

```
require 'spec_helper'
```

```
describe 'puppet-camp-austin::example1' do
```

```
  it 'should include git' do
```

```
    should contain_package('git')
```

```
  end
```

```
end
```

Let us run RSpec

```
% rake spec
```

F

Failures:

1) puppet-camp-austin::example1 should include git

Failure/Error: should contain_package('git')

expected that the catalogue would contain Package[git]

./spec/classes/puppet-camp-austin/example1_spec.rb:5:in
`block (2 levels) in <top (required)>'

Oh no! We Failed!

```
% cat manifests/example1.pp
```

```
class puppet-camp-austin::example1 {  
    package { 'git':  
        ensure => installed  
    }  
}
```

% rake spec

•

Finished in 0.11266 seconds

1 example, 0 failures

**Now we are cooking
with peanut oil!**

Happy Happy Happy

Deep Thoughts

**“It is what you don’t
expect that most
needs looking for.”**

– Neal Stephenson, Anathem

What should we test?

- **Non-obvious things**
- **Complex things**
- **Bugs**

Time for something
not obvious

Let us say we decided we
wanted to let a **puppet**
forge module install git
for us. Oh look, there is
a module called
puppetlabs/git!

Let us add the module to **.fixtures.yml**

```
% cat .fixtures.yml
```

```
fixtures:
```

```
  repositories:
```

```
    git: git://github.com/puppetlabs/puppetlabs-git.git
```

```
  symlinks:
```

```
    puppet-camp-austin: "#{source_dir}"
```

Add the git class

```
% cat manifests/example2.pp  
class puppet-camp-austin::example2  
{  
    include 'git'  
}
```

Add a test

```
% cat spec/classes/puppet-camp-austin/example2_spec.rb  
  
require 'spec_helper'  
  
describe 'puppet-camp-austin::example2' do  
  it 'should include git' do  
    should contain_package('git')  
  end  
end
```


Run the Tests (I am sure they will pass)

```
% rake spec
```

```
.F
```

Failures:

```
1) puppet-camp-austin::example2 should include git
```

```
Failure/Error: should contain_package('git')
```

```
expected that the catalogue would contain Package[git]
```

```
# ./spec/classes/puppet-camp-austin/example2_spec.rb:5:in  
`block (2 levels) in <top (required)>'
```

What?

**The git module
doesn't install git?**

Is this crazy??!

OK, let us go look

```
class git {  
  package { 'git-core':  
    ensure => installed,  
  }  
}
```

**Should you be testing
3rd party modules
this way?**

Probably not.

It depends.

**Has it burned you in
the past?**

It did me :(

Let us try this again

We write a test

```
describe 'puppet-camp-austin::example3' do
  describe 'operating system is Debian' do
    let(:facts) { {:operatingsystem => 'Debian'} }

    it 'should include git' do
      should contain_package('git')
    end
  end
end

describe 'operating system is MINIX' do
  let(:facts) { {:operatingsystem => 'MINIX'} }

  it 'should notify a warning' do
    should contain_notify('git cannot be install for this package')
  end
end
end
```

**Maybe we go to
lunch, forget about
our test, come back,
write some code...**

This is the code we write

```
class puppet-camp-austin::example3 {  
    Package { ensure => "installed" }  
  
    case $operatingsystem {  
        'RedHat', 'CentOS': { package {'git': } }  
        /^(Debian|Ubuntu)$/:{ package {'git': } }  
        default:             { package {'git-core': } }  
    }  
}
```

Test it!

```
rspec ./spec/classes/puppet-camp-  
austin/example3_spec.rb:15 #  
puppet-camp-austin::example3  
operating system is MINIX should  
notify a warning
```

D'oh!

Let us add that notify

```
class puppet-camp-austin::example3 {  
    Package { ensure => "installed" }  
  
    case $operatingsystem {  
        'RedHat', 'CentOS': { package {'git': } }  
        /^(Debian|Ubuntu)$/:{ package {'git': } }  
        default:             { notify {'git cannot be install for this package': } }  
    }  
}
```

It works!

% rake spec

..

Finished in 0.13768 seconds

2 examples, 0 failures

In Retrospect

Puppet + RSPEC

Rocks

**Why are you not
doing it!!!?**

**You should test
interesting things**

**Don't be scared to
break it!**

**My contrived examples
are at [https://
github.com/shr3kst3r/
puppet-camp-austin](https://github.com/shr3kst3r/puppet-camp-austin)**

**Questions and maybe
some answers?**