RSpec-Puppet is a testing framework used to write unit tests for Puppet manifests. It helps you validate that your Puppet code behaves as expected. This guide provides step-by-step instructions on how to set up and use RSpec-Puppet to test your Puppet modules.

**Prerequisites**

1. **Ruby**: Ensure Ruby is installed on your system.
2. **Puppet Development Kit (PDK)**: Although not strictly necessary, PDK simplifies setting up and managing Puppet modules and their dependencies.

**1. Install the Puppet Development Kit (PDK)**

PDK provides a streamlined way to work with Puppet modules and RSpec-Puppet.

**Install PDK**:

gem install pdk

**2. Create or Navigate to Your Puppet Module**

If you are starting from scratch, you can create a new module using PDK.

**Create a New Module**:

pdk new module my\_module

cd my\_module

If you have an existing module, navigate to its directory.

**3. Set Up RSpec-Puppet**

**PDK Initialization**:

If you used PDK to create your module, RSpec-Puppet is already included. Otherwise, you’ll need to manually set it up.

1. **Initialize the RSpec Configuration**:

**Install Bundler**:

gem install bundler

**Create a Gemfile**:

If a Gemfile doesn’t already exist, create one:

touch Gemfile

**Add RSpec-Puppet to the Gemfile**:

ruby

source 'https://rubygems.org'

gem 'rspec-puppet'

gem 'puppetlabs\_spec\_helper', '~> 2.0'

gem 'rspec'

gem 'beaker-rspec'

gem 'puppet-resource\_api', '~> 1.9'

// apt-get install ruby-dev (for troubleshooting)

**Install Dependencies**:

bundle install

1. **Set Up RSpec Configuration Files**:

Create the RSpec configuration files if they don't already exist.

**Create .rspec**:

echo '--format documentation' > .rspec

**Create spec/spec\_helper.rb**:

**File: spec/spec\_helper.rb**:

ruby

require 'puppetlabs\_spec\_helper/module\_spec\_helper'

require 'rspec-puppet'

RSpec.configure do |c|

c.before(:each) do

Puppet[:hieradata] = 'spec/fixtures/hieradata'

end

c.after(:each) do

RSpec::Puppet::Coverage.report!(100)

end

end

**Create spec/fixtures/manifests/site.pp**:

This file will be used to test your Puppet manifests.

puppet

node default {

include my\_module

}

**4. Write Your RSpec-Puppet Tests**

Write unit tests to ensure that your Puppet code behaves as expected.

**Create a Test for a Puppet Class**:

**File: spec/classes/my\_module\_spec.rb**:

require 'spec\_helper'

describe 'my\_module' do

context 'with default parameters' do

it { is\_expected.to compile }

it { is\_expected.to contain\_package('vim').with\_ensure('installed') }

end

# Add more tests as needed

end

**5. Run Your RSpec-Puppet Tests**

Run the tests to validate your Puppet module.

**Execute RSpec Tests**:

bundle exec rspec

This command will run all RSpec-Puppet tests and display the results.

**6. Code Coverage (Optional)**

Check code coverage to ensure that your tests cover all parts of your module.

**Generate Coverage Report**:

Add the following to your spec/spec\_helper.rb file if not already present:

ruby

require 'rspec-puppet-facts'

include RspecPuppetFacts

RSpec.configure do |c|

c.default\_facts = {

osfamily: 'Debian',

operatingsystem: 'Ubuntu',

operatingsystemrelease: '18.04',

}

end

Run the coverage report:

bundle exec rspec --format documentation --format Rspec::Puppet::Coverage::Formatter

**7. Linting and Validation**

Ensure your Puppet code adheres to best practices.

**Run Puppet Lint**:

bundle exec puppet-lint

**Validate Puppet Manifests**:

bundle exec puppet parser validate manifests/init.pp

**Summary**

**To use RSpec-Puppet**:

1. **Install PDK and RSpec-Puppet**: Ensure PDK and required gems are installed.
2. **Set Up RSpec Configuration**: Create necessary configuration files for RSpec-Puppet.
3. **Write Unit Tests**: Create tests for your Puppet manifests using RSpec-Puppet.
4. **Run Tests**: Use bundle exec rspec to execute your tests.
5. **Check Code Coverage**: (Optional) Verify that your tests cover all parts of your code.
6. **Lint and Validate**: Ensure your Puppet code follows best practices.