Cucumber-puppet

Nikolay Sturm

16.10.2010 @ DevOpsDays
Europe, Hamburg

The olden times

Our legacy configuration management system was inflexible but hard to break.

The new times

Puppet is like a programming language.

Code breaks

```
# puppetd --test
info: Retrieving plugin
info: Caching catalog for some.host
err: Could not run Puppet configuration client: Could
not find dependency Package[foo] for File[/etc/foo.conf]
at /puppet/production/modules/foo/manifests/init.pp:12
```

A cucumber feature [1/3]

```
Feature: Install inetd
  In order to run certain services
  The inetd service
  Must be installed

  Scenario: Setup inetd
    Given a node of class "inetd"
    When I compile the catalog
    Then package "inetd" should be "installed"
    Then there should be a resource "Service[inetd]"
    And the service should have "enable" set to "true"
    And the state should be "running"
    And the service should require "Package[inetd]"
```

A cucumber feature [2/3]

```
Feature: Install inetd
  In order to run certain services
  The inetd service
  Must be installed

  Scenario: Setup inetd
    Given a node of class "inetd"
    When I compile the catalog
    Then package "inetd" should be "installed"
    Then there should be a resource "Service[inetd]"
    And the service should have "enable" set to "true"
    And the state should be "running"
    And the service should require "Package[inetd]"
```

A cucumber feature [3/3]

```
Feature: Install inetd
  In order to run certain services
  The inetd service
  Must be installed

  Scenario: Setup inetd
    Given a node of class "inetd"
    When I compile the catalog
    Then package "inetd" should be "installed"
    Then there should be a resource "Service[inetd]"
    And the service should have "enable" set to "true"
    And the state should be "running"
    And the service should require "Package[inetd]"
```

Cucumber step definitions [1/2]

```
Given /^a node of class "([^\"]*)"$/ do |klass|
  @klass = klass
end

When /^I compile the catalog$/ do
  compile_catalog
end

Then /^package "([^\"]*)" should be "([^\"]*)"$/ do |p, s|
  steps %Q{
    Then there should be a resource "Package[#{p}]"
    And the state should be "#{s}"
  }
end
```

Cucumber step definitions [2/2]

```
Given /^a node of class "([^\"]*)"$/ do |klass|
  @klass = klass
end

When /^I compile the catalog$/ do
  compile_catalog
end

Then /^package "([^\"]*)" should be "([^\"]*)"$/ do |p, s|
  steps %Q{
    Then there should be a resource "Package[#{p}]"
    And the state should be "#{s}"
  }
end
```

Unit-testing a puppet class

```
Feature: Install inetd
  In order to run certain services
  The inetd service
  Must be installed

  Scenario: Setup inetd
    Given a node of class "inetd"
    When I compile the catalog
    Then package "inetd" should be "installed"
    Then there should be a resource "Service[inetd]"
    And the service should have "enable" set to "true"
    And the state should be "running"
    And the service should require "Package[inetd]"
```

Unit-testing a puppet class . . . not so good

```
Then package "inetd" should be
  "installed"

Then there should be a resource
  "Service[inetd]"
And the service should have
  "enable" set to "true"
And the state should be "running"

And the service should require
  "Package[inetd]"
```

```
class inetd {
  package { "inetd":
    ensure => installed,
  }

  service { "inetd":

    enable => true,

    ensure => running,

    require => Package["inetd"],
  }
}
```

A catalog policy to document rules

```
Feature: General catalog policy for all nodes
  In order to ensure applicability of a node's catalog
  As a manifest developer
  I want all catalogs to obey some general rules

  Scenario: Compile and verify catalog for my.host
    Given a node specified by "nodes/my.host.yaml"
    When I compile its catalog
    Then compilation should succeed
    And all resource dependencies should resolve
    And all files should have an owner, group, and mode
    And the node should have accounts for admins
```

A catalog policy operates on a real node's catalog

```
Feature: General catalog policy for all nodes
  In order to ensure applicability of a node's catalog
  As a manifest developer
  I want all catalogs to obey some general rules

  Scenario: Compile and verify catalog for my.host
    Given a node specified by "nodes/my.host.yaml"
    When I compile its catalog
    Then compilation should succeed
    And all resource dependencies should resolve
    And all files should have an owner, group, and mode
    And the node should have accounts for admins
```

A catalog policy deals with catalog wide properties

```
Feature: General catalog policy for all nodes
  In order to ensure applicability of a node's catalog
  As a manifest developer
  I want all catalogs to obey some general rules

  Scenario: Compile and verify catalog for my.host
    Given a node specified by "nodes/my.host.yaml"
    When I compile its catalog
    Then compilation should succeed
    And all resource dependencies should resolve
    And all files should have an owner, group, and mode
    And the node should have accounts for admins
```

A catalog policy avoids the duplication trap

```
Feature: General catalog policy for all nodes
  In order to ensure applicability of a node's catalog
  As a manifest developer
  I want all catalogs to obey some general rules

  Scenario: Compile and verify catalog for my.host
    Given a node specified by "nodes/my.host.yaml"
    When I compile its catalog
    Then compilation should succeed
    And all resource dependencies should resolve
    And all files should have an owner, group, and mode
    And the node should have accounts for admins
```

A catalog policy can be generated

```
Feature: General catalog policy for all nodes

  @host_one
  Scenario: Compile and verify catalog for my.host
    Given a node specified by "nodes/host.one.yaml"
    When I compile its catalog
    Then compilation should succeed

  @host_two
  Scenario: Compile and verify catalog for my.host
    Given a node specified by "nodes/host.two.yaml"
    When I compile its catalog
    Then compilation should succeed
```

Tags are your friends

```
$ cucumber-puppet --format progress \
        features/catalog/policy.feature
.........................................

25 scenarios (25 passed)
275 steps (275 passed)
0m50.679s
$ cucumber-puppet --format progress --tags @host_one \
        features/catalog/policy.feature
...........

1 scenario (1 passed)
11 steps (11 passed)
0m4.731s
```

And it actually finds mistakes

```
$ cucumber-puppet --format progress --tags @host_one \
        features/catalog/policy.feature
.........F-

(::) failed steps (::)

Service[inetd] cannot require Package[inetd], not in catalog. (RuntimeError)
features/catalog/policy.feature:75:in ‘And all resource dependencies should resolve’

Failing Scenarios:
cucumber features/catalog/policy.feature:65
# Scenario: Compile and verify catalog for host_one

1 scenario (1 failed)
11 steps (1 failed, 1 skipped, 9 passed)
0m4.877s
```

Contact:

Nikolay Sturm
sturm@erisiandiscord.de
@nistude
http://blog.nistu.de/

http://github.com/nistude/cucumber-puppet
http://projects.puppetlabs.com/projects/cucumber-puppet