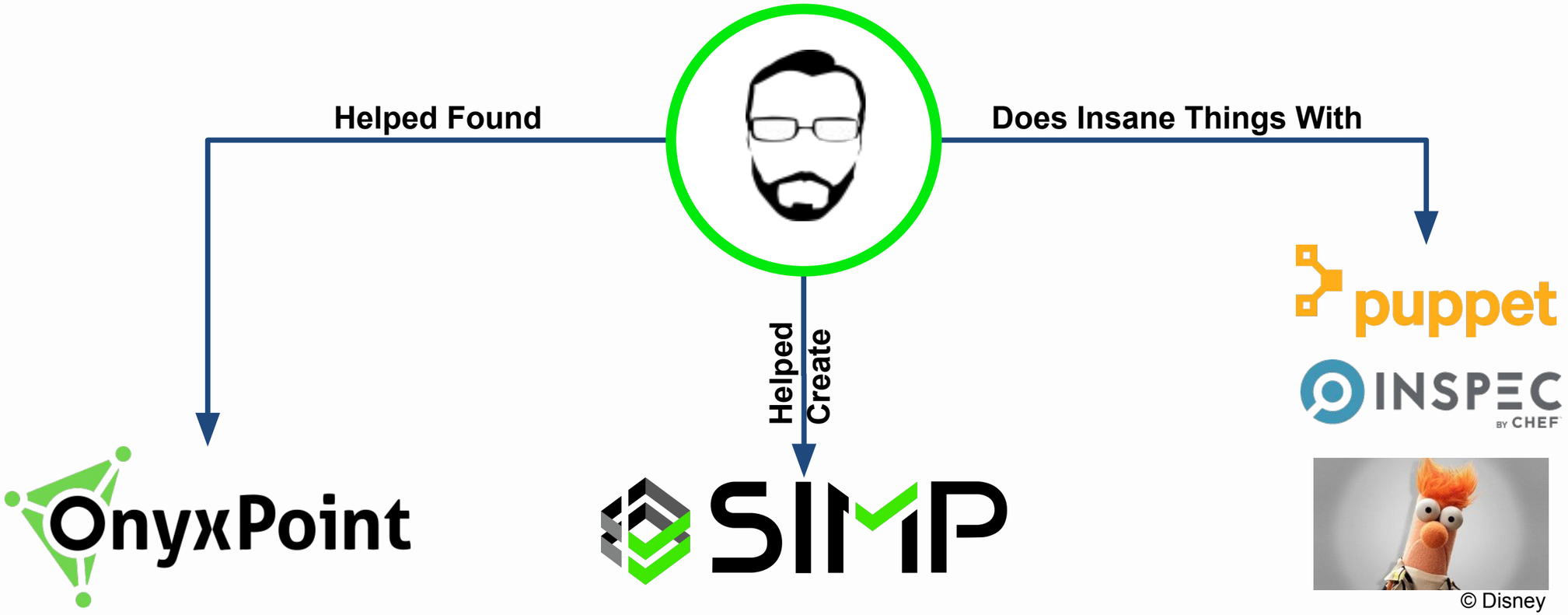
Multi-Node Acceptance Tests for Fun and Profit

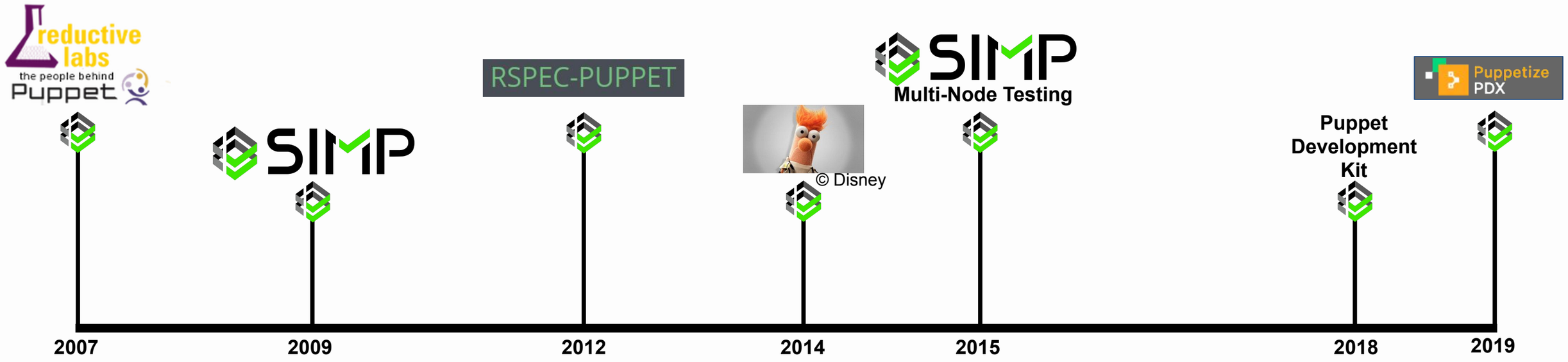
Trevor Vaughan
VP Engineering - Onyx Point, Inc.
SIMP Product Lead

Introduction



© Disney

How Did We Get Here?



NOTE: This is a SIMP-perspective timeline but generally mirrors public awareness.

Why Acceptance Testing is Important

Spec (Unit) Testing

- Example: ***rspec-puppet***
- Answers:
  - Does it have syntax errors?
  - Does it have logic errors?
  - Does it have artifact errors?
    - Templates
    - File content
    - Etc...

Acceptance Testing

- Level One
  - Examples: ***Beaker, Litmus, Test Kitchen***
  - Answers:
    - Does it work on a **single representative system** scenario?
- Level Two
  - Examples: ***Beaker***
  - Answers:
    - Does it work in complex networked representative systems scenarios?

Beaker - Why I Like It

Handles:

- Puppet or non-Puppet
- Vagrant
  - Docker
  - VirtualBox
  - libvirt
- AWS
- GCE
- Azure
- VMWare
- OpenStack
- Bare Metal Connections

© Disney

Straightforward Components:

- ***nodesets***
  - Host group configurations
  - VM-specific configurations
  - Data layer (with templating support!)
- ***Tests***
  - It's just RSpec
  - You can use *pry*
- ***Plugin System***
  - Modular
  - Add what you need, share as gems
    - *simp-beaker-helpers*

Exposes host data for all hosts in your tests!

~ A Vagrant of Vagrants ~

Beaker - What Could Use Some Work

- The documentation
  - Yep, it's pretty abysmal
  - Not consolidated
  - Out of date in many cases
  - The API documentation is the source of truth
  - You can also use one of the many SIMP repositories for inspiration

- Code cruft
  - Sprawling
  - Consistency issues
  - **Still a solid functional base and the only thing that appears to work well with multi-host setups**

- Community
  - Need more people adding features and providing core patches!

© Disney

Setting Up Beaker With PDK

Puppet Development Kit

"The shortest path to better modules"

Beaker

The shortest path to functional multi-node testing

WHO LIKES CONSTANT WALLS OF CODE?!

WHO LIKES CONSTANT WALLS OF CODE?!

(hint, hopefully you)

Setting up a PDK Project

```
# Install PDK

[demo@host ~]$ sudo yum install pdk

# Create a new test module

[demo@host ~]$ pdk new module demo-multi_node

# Answer the questions

[demo@host ~]$ cd multi_node

# The resulting module

[demo@host multi_node]$ ls

.vscode/  data/  examples/  files/  manifests/  spec/  tasks/  templates/  .fixtures.yml  .gitattributes
.gitignore  .gitlab-ci.yml  .pdkignore  .puppet-lint.rc  .rspec  .rubocop.yml  .travis.yml  .yardopts
CHANGELOG.md  Gemfile  Gemfile.lock  README.md  Rakefile  appveyor.yml  hiera.yaml  metadata.json
```

Beaker Prep: Install VirtualBox and Vagrant

HashiCorp Vagrant

VirtualBox

You MUST use Vagrant from vagrantup.com!

Beaker Prep: Update Your Gems

```
# Update Gemfile.local

[demo@host multi_node]$ cat Gemfile.local

group :development do
  gem "simp-beaker-helpers", [ ">= 1.16.1", "< 2.0" ], require: false
end

# Install the new gems

[demo@host multi_node]$ pdk bundle update

pdk (INFO): Using Ruby 2.5.3
pdk (INFO): Using Puppet 6.8.1
<lots of info on gems getting installed>
```

simp-beaker-helpers was created to provide “the missing bits” to make Beaker easy to use for the SIMP project.

There is nothing SIMP specific about it but namespacing is good!

Beaker Prep: Update Your Fixtures

```
[demo@host multi_node]$ cat .fixtures.yml

# An example
---
fixtures:
  forge_modules:
     simplib: "simp/simplib"
     stdlib: "puppetlabs/stdlib"

[demo@host multi_node]$ pdk bundle exec rake spec_prep

pdk (INFO): Using Ruby 2.5.3
pdk (INFO): Using Puppet 6.8.1
I, [2019-09-29T19:52:06.347805 #21546]  INFO -- : Creating symlink from spec/fixtures/modules/multi_node to /home/demo/multi_node
Notice: Preparing to install into /home/demo/multi_node/spec/fixtures/modules ...
/home/demo/multi_node/spec/fixtures/modules
└── simp-simplib (v3.15.3)
Notice: Preparing to install into /home/demo/multi_node/spec/fixtures/modules ...
/home/demo/multi_node/spec/fixtures/modules
└── puppetlabs-stdlib (v6.1.0)
```

This is the same **.fixtures.yml** that *rspec-puppet* uses!

You must include all dependencies for your tests.
This may include files that are not used by the module under test, **but that's OK!**

You may need to clean out ***spec/fixtures/modules*** manually occasionally.

Beaker Prep: Add Main Acceptance Helper

```
[demo@host multi_node]$ cat spec/spec_helper_acceptance.rb

require 'beaker-rspec'

if File.file?(File.join(__dir__, 'spec_helper_acceptance_local.rb'))
  require 'spec_helper_acceptance_local'
end
```

This simply mirrors the functionality of the existing *spec_helper.rb* since we expect PDK to smash this file at some point.

Beaker Prep: Add Beaker Acceptance Helper

```
[demo@host multi_node]$ cat spec/spec_helper_acceptance_local.rb

require 'simp/beaker_helpers'
include Simp::BeakerHelpers

# Install puppet on our hosts
hosts.each { |host| install_puppet } unless ENV['BEAKER_provision'] == 'no'

RSpec.configure do |c|
  # simp-beaker-helpers magic to fix some common Linux issues
  fix_errata_on(hosts)

  # Ensure that our test output is human readable (nobody likes dots)
  c.formatter = :documentation

  c.before :suite do
    # Install modules and dependencies from spec/fixtures/modules
    copy_fixture_modules_to( hosts )
  end
end
```

Beaker Nodesets

Create a Beaker Test - Define Test Nodes

```
[demo@host multi_node]$ mkdir -p spec/acceptance/nodesets

[demo@host multi_node]$ cat spec/acceptance/nodesets/default.yml

HOSTS:
  el7:
    roles:
      - default
    platform: el-7-x86_64
    hypervisor: vagrant
    box: centos/7

CONFIG:
  log_level: verbose
  type: aio
  puppet_environment: puppet6
  vagrant_memsize: 256
  vagrant_cpus: 1
```

Host definitions

Short hostname

Arbitrary tags

Poorly documented magic

Vagrant stuff

Global configuration - Applies to all hosts

Tell us ALL THE THINGS (mainly for debugging)

MOAR MAGIC

The version of **modern** puppet you want to use

Global Vagrant stuff

A Simple Test

```
[demo@host multi_node]$ cat spec/acceptance/00_test_spec.rb

require 'spec_helper_acceptance'

test_name "Single Node Test"

describe 'gathering puppet facts' do
  hosts.each do |host|
    context "on #{host}" do
      it do
        result = on(host, 'facter -p').output.strip.lines

        expect(result.grep(/Error/).grep(/Facter/)).to be_empty
      end
    end
  end
end
```

DO NOT USE *serverspec* - It is not multi-host safe.

Stick to pure *rspec* and *beaker-rspec* as shown in the example here.

A Simple Test - Executed!

```
[demo@host multi_node]$ pdk bundle exec rake beaker

<LOTS OF INFORMATION>

Single Node Test
== checking prepped modules from .fixtures.yml
  -- (use BEAKER_spec_prep=no to disable)
  -- 0 modules need to be prepped
  == all fixture modules present

gathering puppet facts
  on el7
    should be empty
Destroying vagrant boxes
==> el6: Forcing shutdown of VM...
==> el6: Destroying VM and associated drives...
==> el7: Forcing shutdown of VM...
==> el7: Destroying VM and associated drives...

Finished in 14.4 seconds (files took 3 minutes 49.6 seconds to load)
1 examples, 0 failures
```

Eye Break

A Multi-Node Test

```
[demo@host multi_node]$ cat spec/acceptance/nodesets/default.yml

HOSTS:
  el7:
    roles:
      - default
    platform: el-7-x86_64
    hypervisor: vagrant
    box: centos/7

  el6:
    platform: el-6-x86_64
    hypervisor: vagrant
    box: centos/6

CONFIG:
  log_level: verbose
  type: aio
  puppet_environment: puppet6
  vagrant_memsize: 256
  vagrant_cpus: 1
```

A Multi-Node Test - Executed!

```
[demo@host multi_node]$ pdk bundle exec rake beaker

<LOTS OF INFORMATION>

Multi-Node Test
== checking prepped modules from .fixtures.yml
  -- (use BEAKER_spec_prep=no to disable)
  -- 0 modules need to be prepped
  == all fixture modules present

gathering puppet facts
  on el7
    should be empty
  on el6
    should be empty
Destroying vagrant boxes
==> el6: Forcing shutdown of VM...
==> el6: Destroying VM and associated drives...
==> el7: Forcing shutdown of VM...
==> el7: Destroying VM and associated drives...

Finished in 28.2 seconds (files took 4 minutes 8.9 seconds to load)
2 examples, 0 failures
```

Server/Client Tests

Server/Client Test

```
[demo@host multi_node]$ cat spec/acceptance/nodesets/default.yml

HOSTS:
  el7:
    roles:
      - default
      - server
    platform: el-7-x86_64
    hypervisor: vagrant
    box: centos/7

  el6:
    roles:
      - client
    platform: el-6-x86_64
    hypervisor: vagrant
    box: centos/6

CONFIG:
  type: aio
  puppet_environment: puppet6
  vagrant_memsize: 256
  vagrant_cpus: 1
```

New role **'server'**

New role **'client'**

We made a Puppet module
Let's do something with it!

Puppet Class *multi_node*

```
[demo@host multi_node]$ cat manifests/init.pp

class multi_node (
  Boolean $is_server = false,
  Boolean $is_client = false
) {
  if $is_server { include multi_node::server }
  if $is_client { include multi_node::client }

  package { 'nc': ensure => 'installed' }
}
```

Puppet Class *multi_node::server*

```
[demo@host multi_node]$ cat manifests/server.pp

class multi_node::server (
  Simplib::Port        $port,
  Stdlib::Absolutepath $output_file
){

  service { 'firewalld': ensure => stopped, enable => false }
  service { 'iptables': ensure => stopped, enable => false }

  file { '/usr/local/bin/nc_listen':
    mode    => '0755',
    content => "#!/bin/bash\n\n nc -l ${nc_port} > ${output_file} &\n"
  }
}
```

Puppet Class *multi_node::client*

```
[demo@host multi_node]$ cat manifests/client.pp

class multi_node::client (
  Simplib::Host $server,
  Simplib::Port $port,
  String[1]     $message = 'This is a test'
) {

  file { '/usr/local/bin/nc_send':
    mode    => '0755',
    content => "#!/bin/bash\n\n echo '${message}' | nc ${server} ${port} -w 5\n"
  }

}
```

So, how do we know this thing actually works?

The Test - Setup and Variables

```
[demo@host multi_node]$ cat spec/acceptance/05_cross_node_spec.rb

require 'spec_helper_acceptance'
test_name "Cross-Node Test"
describe 'smash a file across' do
  let(:manifest){ 'include multi_node' }

  let(:test_port){ 12345 }
  let(:test_output_file){ '/tmp/test_file' }
  let(:test_message){ 'A special test message' }
  let(:server_fqdn){ fact_on(only_host_with_role(hosts, 'server'), 'fqdn') }

  let(:server_hieradata){{
    'multi_node::is_server'           => true,
    'multi_node::server::port'        => test_port,
    'multi_node::server::output_file' => test_output_file
  }}

  let(:client_hieradata){{
    'multi_node::is_client'       => true,
    'multi_node::client::server'  => server_fqdn,
    'multi_node::client::port'    => test_port,
    'multi_node::client::message' => test_message
  }}
```

```
el7:
  roles:
    - default
    - server
  platform: el-7-x86_64
  hypervisor: vagrant
  box: centos/7
```

The Test - The Basics

```
[demo@host multi_node]$ cat spec/acceptance/05_cross_node_spec.rb (continued)

  # Make sure things run cleanly without configuration
   hosts.each do |host|
     context "on #{host}" do
       it 'should run puppet' do
         apply_manifest_on(host, manifest, catch_failures: true)
       end

       it 'should be idempotent' do
         apply_manifest_on(host, manifest, catch_changes: true)
       end
     end
   end
```

'include multi_node'

The Test - Configure the Server

```
[demo@host multi_node]$ cat spec/acceptance/05_cross_node_spec.rb (continued)

  hosts_with_role(hosts, 'server').each do |host|
    context "on server #{host}" do
      it 'should set the hieradata' do
        set_hieradata_on(host, server_hieradata)
      end

      it 'should run puppet' do
        apply_manifest_on(host, manifest, catch_failures: true)
      end

      it 'should be idempotent' do
        apply_manifest_on(host, manifest, catch_changes: true)
      end
    end
  end
```

```
let(:server_hieradata){{
  'multi_node::is_server'           => true,
  'multi_node::server::port'        => 12345,
  'multi_node::server::output_file' => '/tmp/test_file'
}}
```

The Test - Configure the Client

```
[demo@host multi_node]$ cat spec/acceptance/05_cross_node_spec.rb (continued)

  hosts_with_role(hosts, 'client').each do |host|
    context "on client #{host}" do
      it 'should set the hieradata' do
        set_hieradata_on(host, client_hieradata)
      end

      it 'should run puppet' do
        apply_manifest_on(host, manifest, catch_failures: true)
      end

      it 'should be idempotent' do
        apply_manifest_on(host, manifest, catch_changes: true)
      end
    end
  end
```

```
let(:client_hieradata){{
  'multi_node::is_client'       => true,
  'multi_node::client::server'  => <server_fqdn>,
  'multi_node::client::port'    => 12345,
  'multi_node::client::message' =>
    'A special test message'
}}
```

Eye Break - Like the new SIMP Mascot?

The Test - Send the Test Message!

```
[demo@host multi_node]$ cat spec/acceptance/05_cross_node_spec.rb (continued)

  hosts_with_role(hosts, 'server').each do |host|
    context "on server #{host}" do
      it 'should start the listener' do
        on(host, '/usr/local/bin/nc_listen')
        sleep(2)
      end
    end
  end

  hosts_with_role(hosts, 'client').each do |host|
    context "on client #{host}" do
      it 'should send a message to the server' do
        on(host, '/usr/local/bin/nc_send')
      end
    end
  end
```

This script was created with **multi_node::server**

This script was created with **multi_node::client**

The Test - Did it work?!

```
[demo@host multi_node]$ cat spec/acceptance/05_cross_node_spec.rb (continued)

  hosts_with_role(hosts, 'server').each do |host|
    context "on server #{host}" do
      it 'should have received the test message' do
        content = file_contents_on(host, test_output_file)
        expect(content.strip).to eq(test_message)
      end
    end
  end
```

```
let(:test_output_file){ '/tmp/test_file' }
```

```
let(:test_message){ 'A special test message' }
```

The Test - Did it work?!

The Test - Did it work?!

```
[demo@host multi_node]$ pdk bundle exec rake beaker

<LOTS OF INFORMATION>

Multi-Node Test

smash a file across
  <TRUNCATED FOR SPACE>
  on server el7
    should set the hieradata
    should run puppet
    should be idempotent
  on client el6
    should set the hieradata
    should run puppet
    should be idempotent
```

```
<CONTINUED>

  on server el7
    should start the listener
  on client el6
    should send a message to the server
  on server el7
    should have received the test message

Destroying vagrant boxes
==> el6: Forcing shutdown of VM...
==> el6: Destroying VM and associated drives...
==> el7: Forcing shutdown of VM...
==> el7: Destroying VM and associated drives...

Finished in 1 minute 24.77 seconds (files took 3
minutes 22.7 seconds to load)
15 examples, 0 failures
```

But Wait, There's More!

AND NOW FOR MORE CODE!

Actually….Let’s Recap

Recap

We Now Know That Our Code:

- Functions on a Single Node
- Functions as a Server
- Functions as a Client
- Actually Processes Information Across the Network
  - **No Firewall Issues**
  - **No Functional Configuration Issues**

Debugging

- Set ***BEAKER_destroy=no*** or ***BEAKER_destroy=onpass***
  - Navigate to *.vagrant/beaker_vagrant_files/<nodeset>.yml/*
  - Run standard *vagrant* commands

```
[demo@host multi_node]$ cd .vagrant/beaker_vagrant_files/default.yml

[demo@host default.yml]$ vagrant status

vagrant status
Current machine states:

el7                       running (virtualbox)
el6                       running (virtualbox)

[demo@host default.yml]$ vagrant ssh el7

Last login: Mon Sep 30 01:59:35 2019 from 10.0.2.2
[vagrant@el7 ~]$
```

© Disney

Debugging - Cont'd

- Run *puppet apply* on any of the uploaded manifests
  - Or anything else that you need in order to debug!
  - Unfortunately, they aren't ordered by time so you have to dig

```
[demo@host default.yml]$ vagrant ssh el7

Last login: Mon Sep 30 01:59:35 2019 from 10.0.2.2
[vagrant@el7 ~]$ ls /tmp/apply_manifest.pp.*

/tmp/apply_manifest.pp.b6GQBQ
/tmp/apply_manifest.pp.CPLNW2
/tmp/apply_manifest.pp.FLy9jX

[vagrant@el7 ~]$ sudo /opt/puppetlabs/bin/puppet apply /tmp/apply_manifest.pp.b6GQBQ

Notice: Compiled catalog for el7.local in environment production in 0.05 seconds
Notice: Applied catalog in 0.41 seconds
```

© Disney

Additional Awesomeness - Making it Easier(ish)

- ***simp-beaker-helpers*** ruby gem
  - ***Suites***: Run completely independent tests in the same module space
    - SIMP uses this for compliance tests that need a “clean room” environment
    - Run them with ***pdk bundle exec rake beaker:suites***
  - YUM repos in nodesets
  - Helper methods for ease of use
    - ***copy_to***: use the fastest method possible for copying files to nodes
    - ***write_hieradata_to***: add hiera data to a node in the right spot
    - And more!
  - FIPS 140-2 kernel-level testing on RHEL-compatible systems
  - RHEL system registering and unregistering
  - InSpec and SCAP compliance testing
  - VM snapshot and restart support

https://github.com/simp/rubygem-simp-beaker-helpers

Additional Awesomeness - Examples!

- Examples for future reference from the SIMP GitHub
  - ***pupmod-simp-aide***
    - Simple and straightforward. Contains a compliance suite
  - ***pupmod-simp-rsyslog***
    - The beginning of it all: 4 systems, 4 suites, functional mayhem
  - ***pupmod-simp-simp***
    - Profile-level testing
      - 8 suites (including Windows)
      - Full puppet server and client setup
      - Firewalls, SELinux, and fun
  - ***simp-core***
    - Control repo-style example
    - Spins up all of SIMP using multiple methods
- Check the ***.gitlab-ci.yml*** files for how to run the tests!

https://github.com/simp

Contact Info

SEE ALSO

ABOUT ME

Trevor Vaughan
VP Engineering - Onyx Point, Inc.
tvaughan@onyxpoint.com
@peiriannydd OR @onyxpoint

PRESENTATION PROJECT

https://github.com/trevor-vaughan/puppetize_2019_multi_node_beaker

CONSULTING + TRAINING

http://www.onyxpoint.com
Puppet(8), GitLab(8), Automation(7), DevOps(2), Linux(8)