Ansible, Integration Testing, and You. Catching configuration problems before they happen.

—

...With a little help from the Chef community.

Who Am I?

Bob Killen / @mrbobbytables

Senior Research Cloud Administrator

http://arc-ts.umich.edu

Why bother testing?

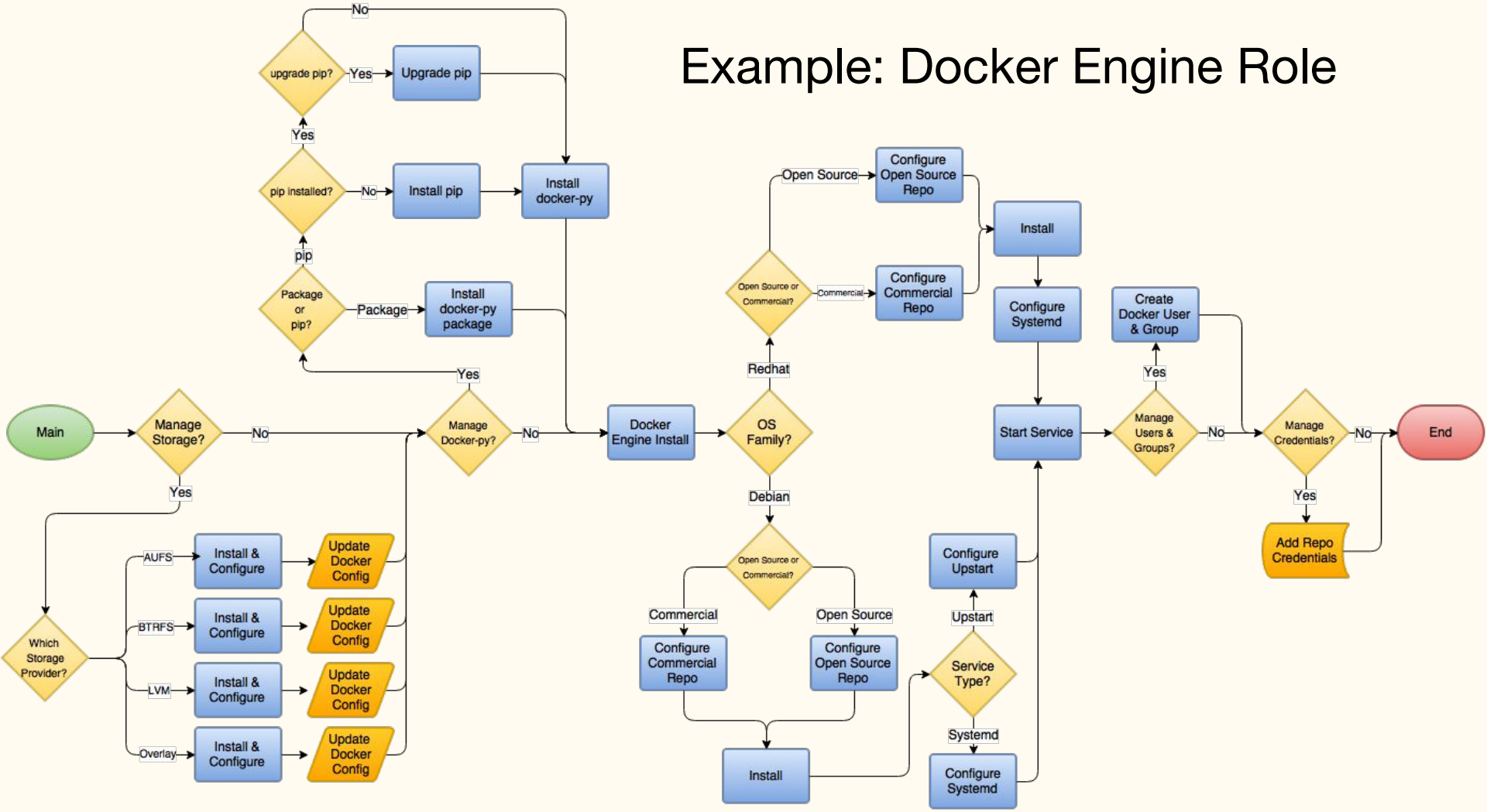
Ansible is an ordered, fail-fast system. Why would further testing be needed?

Complexity

Common Examples:

- Does it work with sysVinit? How about Systemd?
- Are there any issues between minor distribution releases? CentOS 7.0 vs Centos 7.1 or Ubuntu 14.04.4 vs 14.04.5.
- Supporting multiple optional or conflicting features?
- Complex dependencies?
- Requires working with other live service?

Example: Docker Engine Role



Types of Testing Applicable to Ansible

- **Static Code Analysis**
  - Performs Code Analysis without executing any code itself.
  - Examples:
    - Ansible Syntax check (`ansible-playbook --syntaxcheck`).
    - **Ansible-lint** - Performs checks on playbooks for practices and behavior that could be improved.

- **Unit Test**
  - Smallest possible ‘unit’ of code is tested in isolation.
  - Within Ansible, this would be classified as a task requisite or assertion. A service ‘should’ be started. A file ‘should’ contain this string. When it is not encountered, fail the task.
  - Example:
    - Ansible Check Mode (`ansible-playbook --check`).

Types of Testing Applicable to Ansible (cont.)

- **Integration Testing**
  - Builds upon unit-tests and tests them as a group or module. Within Ansible, this would be a role.
  - Verifies all units executed and produced the desired outcome.
  - Usually tested with a functional version of external dependencies.

- **Acceptance Testing**
  - Similar to Integration testing, Acceptance testing builds upon Integration testing. The intent is to test end-to-end, say an entire playbook of multiple roles.
  - If at all possible, simulate end-user usage of the deployed system.

What Makes for a Good Testing Framework?

- Fast.
- Easily Repeatable.
- Modular.
- Support testing against multiple distributions or environments.
- Ability to be integrated into a CI/CD pipeline.

Test Kitchen!

http://kitchen.ci/

- Originally developed by Opscode in late 2012. 1.0 released in 2013, and is still actively being maintained and developed.
- Extremely simple workflow based of a yaml DSL.
- Large plugin ecosystem supporting roughly 50 drivers and over 20 different testing frameworks.
- Easy to integrate with a variety of CI systems using native Ruby tools.
- FAST FAST FAST!

Getting Started with Test-Kitchen

1. Install Ruby 2.0 or greater (suggestion: install a ruby version manager such as rvm, rbenv, or churby)
2. Install bundler and test-kitchen `gem install bundler test-kitchen`
3. If intending to test local, install your local testing endpoint such as vagrant, or docker:
    - Vagrant - https://www.vagrantup.com/
    - Docker - https://www.docker.com/
4. Create a file in the root of your project titled `Gemfile`.
5. Edit the file and add the needed drivers or plugins you intend to use.
6. Execute the command `bundle install`. This will install the dependencies and manage them for the project.

```
source 'https://rubygems.org'

group :test do
  gem 'kitchen-ansible'
  gem 'kitchen-ec2'
  gem 'kitchen-vagrant'
  gem 'kitchen-verifier-shell'
  gem 'net-ssh'
  gem 'rake'
  gem 'rspec'
  gem 'serverspec'
  gem 'test-kitchen'
end
```

Test Kitchen Components

- **Driver -** Supplies test-kitchen with the information regarding the endpoint where it will execute actions. This includes things such as: Docker, Dsc (windows), Ec2, Google (gce), Openstack, ssh, Vagrant and many others.
- **Platforms -** The systems or environments intended to test against. Examples: CentOS 7.2, Ubuntu 16.04, and Windows 10.
- **Provisioner -** The application or framework that will be used to configure the system. Ansible, Bash, Chef, Puppet, Salt, Windows DSC fall into this category.
- **Verifier* -** Informs test-kitchen of how it will perform a test action. Inspec, serverspec, and ssh are examples of verifiers.
- **Suite -** Describes the specific tests that should be executed against a platform. If using a verifier, supplies the suite specific verification configuration. Test suites include such frameworks as Bats, Cucumber, Inspec, Rspec and Serverspec.
- **Instance -** An instance is a combination of a suite and platform, and functions as the testable unit.

* Verifier is currently optional depending on testing method.

.kitchen.yml

```yaml
---

driver:
  name: vagrant
  vagrantfile_erb: Vagrantfile_local.erb

provisioner:
  name: ansible_playbook
  hosts: all
  role_name: test-role
  sudo_command: sudo -E -H
  require_ansible_repo: false
  require_ansible_source: true
  require_ansible_omnibus: true
  require_chef_for_busser: false
  require_ruby_for_busser: false
  ansible_verbose: true
  ansible_verbosity: warn
  idempotency_test: true

platforms:
  - name: centos-7
    driver_config:
      box: bento/centos-7.2
      box_url: https://atlas.hashicorp.com/bento/boxes/centos-7.2
  - name: debian-8
    driver_config:
      box: bento/debian-8.5
      box_url: https://atlas.hashicorp.com/bento/debian-8.5
  - name: ubuntu-1604
    driver_config:
      box: bento/ubuntu-16.04
      box_url: https://atlas.hashicorp.com/bento/ubuntu-16.04

verifier:
  name: shell

suites:
  - name: test-1
    provisioner:
      playbook: tests/vagrant/test-1.yml
    verifier:
      command: bundle exec rspec -c -f d -I serverspec spec/test_spec.rb
  - name: test-2
    provisioner:
      playbook: tests/vagrant/test-2.yml
    verifier:
      command: bundle exec rspec -c -f d -I serverspec spec/test_spec.rb
```

Example Vagrant Kitchen Config

- **Driver** - vagrant (custom vagrantfile supplied)
- **Provisioner -** ansible_playbook
  - https://github.com/neillturner/kitchen-ansible
- **Platforms -**
  - Centos 7.2
  - Debian 8.5
  - Ubuntu 16.04
- **Verifier -** shell (executes shell command)
- **Suites -**
  - test-1
  - test-2

```
muninn:test-role bob$ kitchen list
Instance             Driver   Provisioner      Verifier  Transport  Last Action
test-1-centos-7      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-debian-8      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-ubuntu-1604   Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-centos-7      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-debian-8      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-ubuntu-1604   Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
```

.kitchen.cloud.yml

```yaml
---

driver:
  name: ec2
  aws_ssh_key_id: <%= ENV['AWS_SSH_KEY_ID'] %>
  security_group_ids: [ "<%= ENV['AWS_SGROUP_ID'] %>" ]
  region: <%= ENV['AWS_REGION'] %>
  availability_zone: <% ENV['AWS_AVAILABILITY_ZONE'] %>
  instance_type: <%= ENV['AWS_INSTANCE_TYPE'] || 't2.micro' %>
  associate_public_ip: true
  require_chef_omnibus: false

transport:
  ssh_key: <%= ENV['KITCHEN_SSH_KEY'] %>
  connection_timeout: 60
  connection_retries: 10
  max_ssh_sessions: 4

provisioner:
  name: ansible_playbook
  hosts: all
  role_name: test-role
  sudo_command: sudo -E -H
  require_ansible_repo: false
  require_ansible_source: true
  require_ansible_omnibus: true
  require_chef_for_busser: false
  require_ruby_for_busser: false
  ansible_verbose: true
  ansible_verbosity: warn
  idempotency_test: true


# Default for centos-7 image is to persist the root volume.
# The below overrides that.
platforms:
  - name: centos-7
    driver:
      block_device_mappings:
        - device_name: /dev/sda1
          ebs:
            volume_type: standard
            volume_size: 8
            delete_on_termination: true
  - name: debian-8
  - name: ubuntu-16.04

verifier:
  name: shell

suites:
  - name: test-1
    provisioner:
      playbook: tests/cloud/test-1.yml
    verifier:
      command: KITCHEN_SSH_KEY=<%= ENV['KITCHEN_SSH_KEY'] %> PLAYBOOK=tests/cloud/test-1.yml bundle exec rspec -c -f d -I serverspec
  - name: test-2
    provisioner:
      playbook: tests/cloud/test-2.yml
    verifier:
      command: KITCHEN_SSH_KEY=<%= ENV['KITCHEN_SSH_KEY'] %> PLAYBOOK=tests/cloud/test-2.yml bundle exec rspec -c -f d -I serverspec
```

Example AWS ec2 Kitchen Config

- **Driver** - ec2
- **Provisioner -** ansible_playbook
- **Platforms -**
  - Centos 7.2
  - Debian 8.5
  - Ubuntu 16.04
- **Verifier -** shell (executes shell command)
- **Suites -**
  - test-1
  - test-2

```
muninn:test-role bob$ kitchen list
Instance             Driver  Provisioner      Verifier  Transport  Last Action
test-1-centos-7      Ec2     AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-debian-8      Ec2     AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-ubuntu-1604   Ec2     AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-centos-7      Ec2     AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-debian-8      Ec2     AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-ubuntu-1604   Ec2     AnsiblePlaybook  Shell     Ssh        <Not Created>
```

Test Kitchen Lifecycle

- Create
- Converge
- Verify
- Destroy

Test Kitchen Lifecycle - Create

- Creates the instance(s) as supplied by the `kitchen create` command.
- The VM is spun up, container started, or ssh endpoint contacted.

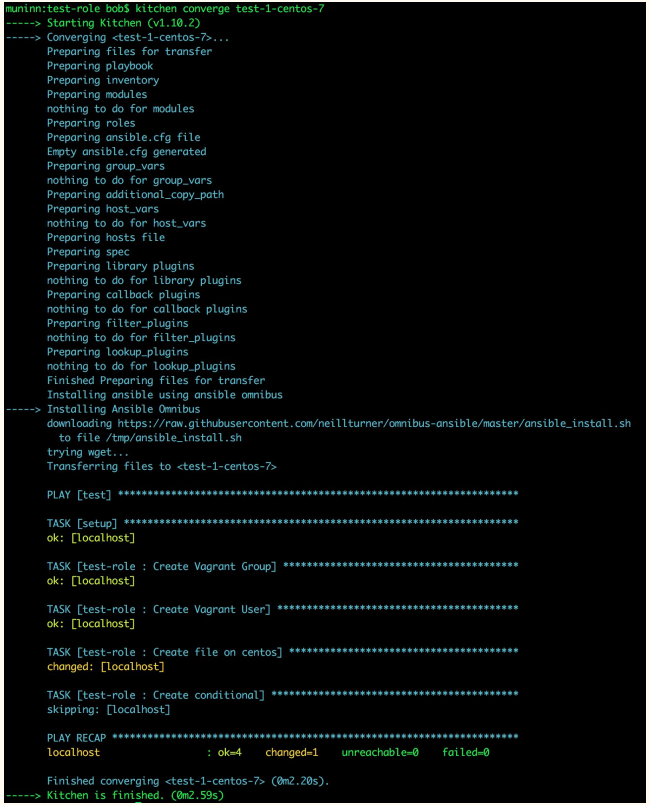
```
muninn:test-role bob$ kitchen list
Instance            Driver   Provisioner      Verifier  Transport  Last Action
test-1-centos-7     Vagrant  AnsiblePlaybook  Shell     Ssh        Created
test-1-debian-8     Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-ubuntu-1604  Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-centos-7     Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-debian-8     Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-ubuntu-1604  Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
```

```
muninn:test-role bob$ kitchen create test-1-centos-7
-----> Starting Kitchen (v1.10.2)
-----> Creating <test-1-centos-7>...
       Bringing machine 'default' up with 'virtualbox' provider...
       ==> default: Importing base box 'bento/centos-7.2'...
==> default: Matching MAC address for NAT networking...
       ==> default: Checking if box 'bento/centos-7.2' is up to date...
       ==> default: A newer version of the box 'bento/centos-7.2' is available! You currently
       ==> default: have version '2.2.6'. The latest is version '2.2.9'. Run
       ==> default: `vagrant box update` to update.
       ==> default: Setting the name of the VM: kitchen-test-role-test-1-centos-7_default_1470790900452_10487
       ==> default: Clearing any previously set network interfaces...
       ==> default: Preparing network interfaces based on configuration...
           default: Adapter 1: nat
       ==> default: Forwarding ports...
           default: 22 (guest) => 2222 (host) (adapter 1)
       ==> default: Booting VM...
       ==> default: Waiting for machine to boot. This may take a few minutes...
           default: SSH address: 127.0.0.1:2222
           default: SSH username: vagrant
           default: SSH auth method: private key
           default: Warning: Remote connection disconnect. Retrying...
           default:
           default: Vagrant insecure key detected. Vagrant will automatically replace
           default: this with a newly generated keypair for better security.
           default:
           default: Inserting generated public key within guest...
           default: Removing insecure key from the guest if it's present...
           default: Key inserted! Disconnecting and reconnecting using new SSH key...
       ==> default: Machine booted and ready!
       ==> default: Checking for guest additions in VM...
       ==> default: Setting hostname...
       ==> default: Mounting shared folders...
           default: /tmp/vagrant-cache => /Users/bob/.vagrant.d/cache/bento/centos-7.2
       ==> default: Configuring cache buckets...
       ==> default: Machine not provisioned because `--no-provision` is specified.
       [SSH] Established
       Vagrant instance <test-1-centos-7> created.
       Finished creating <test-1-centos-7> (0m40.24s).
-----> Kitchen is finished. (0m40.61s)
```

Test Kitchen Lifecycle - Converge

- Executes a converge against the node(s) as supplied by the `kitchen converge` command.
- The converge process happens in 3 stages.
  - Provisioning dependencies are installed (install ansible)
  - Local files (ansible role) are copied to the node
  - Execute an action as supplied by the test-kitchen config. For Ansible, this would be executing the supplied playbook.

```
muninn:test-role bob$ kitchen list
Instance             Driver   Provisioner      Verifier  Transport  Last Action
test-1-centos-7      Vagrant  AnsiblePlaybook  Shell     Ssh        Converged
test-1-debian-8      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-ubuntu-1604   Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-centos-7      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-debian-8      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-ubuntu-1604   Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
```



Test Kitchen Lifecycle - Verify

```
muninn:test-role bob$ kitchen verify test-1-centos-7
-----> Starting Kitchen (v1.10.2)
-----> Setting up <test-1-centos-7>...
       Finished setting up <test-1-centos-7> (0m0.00s).
-----> Verifying <test-1-centos-7>...
       [Shell] Verify on instance=#<Kitchen::Instance:0x007fd09a886d60> with state={:hostname=>
"127.0.0.1", :port=>"2222", :username=>"vagrant", :ssh_key=>"/Users/bob/projects/ansible/roles/
test-role/.kitchen/kitchen-vagrant/kitchen-test-role-test-1-centos-7/.vagrant/machines/default/
virtualbox/private_key", :last_action=>"setup"}

TEST ROLE
  File "/tmp/distrib"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644

Finished in 0.09321 seconds (files took 0.77889 seconds to load)
4 examples, 0 failures

       Finished verifying <test-1-centos-7> (0m1.36s).
-----> Kitchen is finished. (0m1.75s)
```

- Executes a verify action against the node(s) as supplied by the `kitchen verify` command.
- Verify actions are dependant on both the testing framework and verifier.

```
muninn:test-role bob$ kitchen list
Instance            Driver   Provisioner      Verifier  Transport  Last Action
test-1-centos-7     Vagrant  AnsiblePlaybook  Shell     Ssh        Verified
test-1-debian-8     Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-ubuntu-1604  Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-centos-7     Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-debian-8     Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-ubuntu-1604  Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
```

Test Kitchen Lifecycle - Destroy

```
muninn:test-role bob$ kitchen destroy
-----> Starting Kitchen (v1.10.2)
-----> Destroying <test-1-centos-7>...
       ==> default: Forcing shutdown of VM...
       ==> default: Destroying VM and associated drives...
       Vagrant instance <test-1-centos-7> destroyed.
       Finished destroying <test-1-centos-7> (0m4.01s).
-----> Destroying <test-1-debian-8>...
       Finished destroying <test-1-debian-8> (0m0.00s).
-----> Destroying <test-1-ubuntu-1604>...
       Finished destroying <test-1-ubuntu-1604> (0m0.00s).
-----> Destroying <test-2-centos-7>...
       Finished destroying <test-2-centos-7> (0m0.00s).
-----> Destroying <test-2-debian-8>...
       Finished destroying <test-2-debian-8> (0m0.00s).
-----> Destroying <test-2-ubuntu-1604>...
       Finished destroying <test-2-ubuntu-1604> (0m0.00s).
-----> Kitchen is finished. (0m4.40s)
```

- Simply destroys the VM, container etc as supplied by the `kitchen destroy` command.
- If no instance is supplied, will destroy all.

```
muninn:test-role bob$ kitchen list
Instance             Driver   Provisioner      Verifier  Transport  Last Action
test-1-centos-7      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-debian-8      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-1-ubuntu-1604   Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-centos-7      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-debian-8      Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
test-2-ubuntu-1604   Vagrant  AnsiblePlaybook  Shell     Ssh        <Not Created>
```

Serverspec - TDD for Infrastructure

http://serverspec.org

- Based off Ruby Rspec and SpecInfra.
- Configuration Management System Agnostic.
- Supports a wide range of Operating Systems Including AIX, BSD (OSX included), Linux, Windows, and other offshoots such as SmartOS.
- Easy to understand and quick to write tests.
- 40 Resource types*, and easily extendable.

* List of available Resource types http://serverspec.org/resource_types.html

Serverspec Examples

```
describe iptables do
  it { is_expected.to have_rule('-P INPUT ACCEPT').with_table('mangled').with_chain('INPUT') }
end
```

```
describe user('bob') do
  it { is_expected.to exist }
  it { is_expected.to belong_to_group 'wheel' }
  it { is_expected.to have_uid 999 }
  it { is_expected.to have_login_shell '/bin/bash' }
end
```

```
describe 'The webserver service' do
  subject { service('nginx') }
  it { is_expected.to be_enabled }
  it { is_expected.to be_running }
  it { is_expected.to be_running.under('supervisord') }
end
```

```
describe file('/etc/passwd') do
  it { is_expected.to exist }
  it { is_expected.to be_file }
  it { is_expected.to be_owned_by 'root' }
  it { is_expected.to be_grouped_into 'root' }
  it { is_expected.to be_mode 644 }
  its(:contents) { is_expected.to match(/^bob:x:.*/) }
end
```

```
describe command('cat /etc/passwd') do
  its(:exit_status) { is_expected.to eq 0 }
  its(:stdout) { is_expected.to match(/^bob:x:.*/) }
end
```

Role Content

tasks/main.yml

```yaml
---
# tasks file for test-role

- name: Create Vagrant Group
  group:
    name: vagrant
    state: present

- name: Create Vagrant User
  user:
    name: vagrant
    state: present

- name: Create file
  copy:
    content: '{{ ansible_os_family|lower }}'
    dest: /tmp/distrib
    owner: vagrant
    group: vagrant
    mode: '0644'

- name: Create conditional file
  copy:
    content: '{{ playbook_var }}'
    dest: /tmp/conditional
    owner: vagrant
    group: vagrant
    mode: '0644'
  when: ansible_distribution|lower == 'ubuntu'
```

tests/vagrant/test-1.yml

```yaml
---

- name: test
  hosts: all
  connection: local
  gather_facts: true
  roles:
    - 'test-role'
  tags:
   - 'test'
  vars:
    playbook_var: test1
```

tests/vagrant/test-2.yml

```yaml
---

- name: test
  hosts: all
  connection: local
  gather_facts: true
  roles:
    - 'test-role'
  tags:
   - 'test'
  vars:
    playbook_var: test2
```

Test-Role Spec File and (Successful) Output

```
require 'serverspec'
require 'spec_helper'


Rspec.describe 'TEST ROLE' do
  describe file('/tmp/distrib') do
    it { should exist }
    it { should be_owned_by 'vagrant' }
    it { should be_grouped_into 'vagrant' }
    it { should be_mode 644 }
  end

  if os[:family] == 'ubuntu'
    describe file('/tmp/conditional') do
      it { is_expected.to exist }
      it { is_expected.to be_owned_by 'vagrant' }
      it { is_expected.to be_grouped_into 'vagrant' }
      it { is_expected.to be_mode 644 }
      its(:content) { is_expected.to match(/^test[0-9]/) }
    end
  end
end
```

```
muninn:test-role bob$ kitchen verify test-1-ubuntu-1604
-----> Starting Kitchen (v1.11.0)
-----> Verifying <test-1-ubuntu-1604>...
       [Shell] Verify on instance=#<Kitchen::Instance:0x007ffc4b1a8a20> with state={:hostname=>"127.0.0.1",
:port=>"2222", :username=>"vagrant", :ssh_key=>"/Users/bob/projects/ansible/roles/test-role/.kitchen/kitchen
-vagrant/kitchen-test-role-test-1-ubuntu-1604/.vagrant/machines/default/virtualbox/private_key", :last_actio
n=>"verify"}

TEST ROLE
  File "/tmp/distrib"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644
  File "/tmp/conditional"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644
    content
      should match /^test[0-9]/

Finished in 0.12793 seconds (files took 0.5032 seconds to load)
9 examples, 0 failures

       Finished verifying <test-1-ubuntu-1604> (0m1.15s).
-----> Kitchen is finished. (0m1.57s)
```

Test-Role Spec File and (Unsuccessful) Output

```
require 'serverspec'
require 'spec_helper'


Rspec.describe 'TEST ROLE' do
  describe file('/tmp/distrib') do
    it { should exist }
    it { should be_owned_by 'vagrant' }
    it { should be_grouped_into 'vagrant' }
    it { should be_mode 644 }
  end

  if os[:family] == 'ubuntu'
    describe file('/tmp/conditional') do
      it { is_expected.to exist }
      it { is_expected.to be_owned_by 'vagrant' }
      it { is_expected.to be_grouped_into 'vagrant' }
      it { is_expected.to be_mode 644 }
      its(:content) { is_expected.to match(/^test[0-9]/) }
    end
  end
end
```

```
muninn:test-role bob$ kitchen verify test-1-ubuntu-1604
-----> Starting Kitchen (v1.11.0)
-----> Verifying <test-1-ubuntu-1604>...
       [Shell] Verify on instance=#<Kitchen::Instance:0x007fc211d859c0> with state={:hostname=>"127.0.0.1",
:port=>"2222", :username=>"vagrant", :ssh_key=>"/Users/bob/projects/ansible/roles/test-role/.kitchen/kitchen
-vagrant/kitchen-test-role-test-1-ubuntu-1604/.vagrant/machines/default/virtualbox/private_key", :last_actio
n=>"verify"}

TEST ROLE
  File "/tmp/distrib"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644
  File "/tmp/conditional"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644
    content
      should match /^test[0-9]/ (FAILED - 1)

Failures:

  1) TEST ROLE File "/tmp/conditional" content should match /^test[0-9]/
     On host `127.0.0.1'
     Failure/Error: its(:content) { is_expected.to match(/^test[0-9]/) }
       expected "test\n" to match /^test[0-9]/
       Diff:
       @@ -1,2 +1,2 @@
       -/^test[0-9]/
       +test
       sudo -p 'Password: ' /bin/sh -c cat\ /tmp/conditional\ 2\>\ /dev/null\ \|\|\ echo\ -n
       test

     # ./spec/test_spec.rb:19:in `block (3 levels) in <top (required)>'

Finished in 0.16176 seconds (files took 0.60554 seconds to load)
9 examples, 1 failure

Failed examples:

rspec ./spec/test_spec.rb:19 # TEST ROLE File "/tmp/conditional" content should match /^test[0-9]/

>>>>>> ------Exception-------
>>>>>> Class: Kitchen::ActionFailed
>>>>>> Message: 1 actions failed.
>>>>>>     Verify failed on instance <test-1-ubuntu-1604>.  Please see .kitchen/logs/test-1-ubuntu-1604.log
for more details
>>>>>> ----------------------
>>>>>> Please see .kitchen/logs/kitchen.log for more details
>>>>>> Also try running `kitchen diagnose --all` for configuration
```

Including Serverspec Tests with Roles

1. Create a folder within the root of the role directory called `spec`.
2. Create a file in this directory titled `spec_helper.rb`. The spec helper file acts as a helper to be included in that instructs serverspec how to communicate with the instance. With test-kitchen, these variables will be passed as environment variables and prefixed with `KITCHEN_`.
3. Create your serverspec test file(s) and call it `<name>_spec.rb`.
4. Within your spec file, include the spec file with the following:
   - `require 'spec_helper'`
5. Update the kitchen config to include the shell verifier, and then give it the command to execute the serverspec test. Example:

```
bundle exec rspec -c -f d -I serverspec
```

spec_helper.rb

```
require 'serverspec'

set :backend, :ssh

options = Net::SSH::Config.for(host)
options[:host_name] = ENV['KITCHEN_HOSTNAME']
options[:user]      = ENV['KITCHEN_USERNAME']
options[:port]      = ENV['KITCHEN_PORT']
options[:keys]      = ENV['KITCHEN_SSH_KEY']
options[:paranoid]  = false

set :host,        options[:host_name]
set :request_pty, true
set :ssh_options, options
set :env, LANG: 'C', LC_ALL: 'C'

RSpec.configure do |c|
  c.disable_monkey_patching!
end
```

This tells rspec to load serverspec and execute the test located at `spec/test_spec.rb`. For a specific list of rspec commands, execute `bundle exec rspec --help`.

Integrating Serverspec with Ansible - AnsibleSpec

- What AnsibleSpec is -
  - A ruby gem (`ansible_spec`) that acts as an Ansible Config Parser.
  - Makes Ansible variables available for use in Serverspec tests.
  - Supports host patterns, ranges, and dynamic inventory sources.
  - Works with multiple roles and spec files.
  - Designed to validate deployments.

- What AnsibleSpec isn’t -
  - It cannot render jinja.
  - It does have some open issues.
  - It does not play well with test-kitchen.*

Installing and Configuring Ansiblespec

- Add `ansible_spec` to your `Gemfile`, and execute `bundle install`.
- The command `ansiblespec-init` will create a default spec and Rakefile.
- Set 3 environment variables: `PLAYBOOK`, `INVENTORY`, and `HASH_BEHAVIOUR`.
  - `PLAYBOOK` - Path to the Ansible playbook you wish to use.
  - `INVENTORY` - Path to the Ansible Inventory associated with the playbook.
  - `HASH_BEHAVIOUR` - The merge behaviour for Ansible (options: `replace` or `merge`)
- These variables may also be set in an AnsibleSpec dot file (`.ansiblespec`)

That’s It!

Using AnsibleSpec with Serverspec

- Ansible Variables are accessed via the `property` hash.

```
require 'serverspec'
require 'spec_helper'


RSpec.describe 'TEST ROLE' do
  describe file('/tmp/distrib') do
    it { should exist }
    it { should be_owned_by 'vagrant' }
    it { should be_grouped_into 'vagrant' }
    it { should be_mode 644 }
  end

  if os[:family] == 'ubuntu'
    describe file('/tmp/conditional') do
      it { is_expected.to exist }
      it { is_expected.to be_owned_by 'vagrant' }
      it { is_expected.to be_grouped_into 'vagrant' }
      it { is_expected.to be_mode 644 }
      its(:content) { is_expected.to match(property['playbook_var']) }
    end
  end
end
```

```
muninn:test-role bob$ kitchen verify test-1-ubuntu-1604
-----> Starting Kitchen (v1.10.2)
-----> Verifying <test-1-ubuntu-1604>...
       [Shell] Verify on instance=#<Kitchen::Instance:0x007f81c21a8f60> with state={:hostname=>"127.0.0.1",
 :port=>"2222", :username=>"vagrant", :ssh_key=>"/Users/bob/projects/ansible/roles/test-role/.kitchen/kitch
en-vagrant/kitchen-test-role-test-1-ubuntu-1604/.vagrant/machines/default/virtualbox/private_key", :last_ac
tion=>"verify"}

TEST ROLE
  File "/tmp/distrib"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644
  File "/tmp/conditional"
    should exist
    should be owned by "vagrant"
    should be grouped into "vagrant"
    should be mode 644
    content
      should match "test1"

Finished in 0.12755 seconds (files took 0.55372 seconds to load)
9 examples, 0 failures

       Finished verifying <test-1-ubuntu-1604> (0m1.15s).
-----> Kitchen is finished. (0m1.52s)
```

Faking AnsibleSpec for use with Test-Kitchen

```
source 'https://rubygems.org'

group :test do
  gem 'deep_merge'
  gem 'kitchen-ansible'
  gem 'kitchen-ec2'
  gem 'kitchen-vagrant'
  gem 'kitchen-verifier-shell'
  gem 'net-ssh'
  gem 'rake'
  gem 'rspec'
  gem 'serverspec'
  gem 'test-kitchen'
  gem 'yarjuf'
end
```

```
suites:
  - name: test-1
    provisioner:
      playbook: tests/vagrant/test-1.yml
    verifier:
      command: PLAYBOOK=tests/vagrant/test-1.yml bundle exec rspec -c -f d -I serverspec
```

Ansiblespec's behaviour can be simulated with a slightly more complicated spec helper.

```
require 'deep_merge'
require 'serverspec'
require 'net/ssh'
require 'yaml'

def merge_vars(vars, hash, behaviour)
  if hash.is_a?(Hash)
    if behaviour.casecmp('merge')
      vars.deep_merge!(hash)
    else
      vars.merge!(hash)
    end
  end
  vars
end

set :backend, :ssh

options = Net::SSH::Config.for(host)
options[:host_name] = ENV['KITCHEN_HOSTNAME']
options[:user]      = ENV['KITCHEN_USERNAME']
options[:port]      = ENV['KITCHEN_PORT']
options[:keys]      = ENV['KITCHEN_SSH_KEY']
options[:paranoid]  = false

set :host,        options[:host_name]
set :request_pty, true
set :ssh_options, options
set :env, LANG: 'C', LC_ALL: 'C'


if ENV['PLAYBOOK']
  playbook = YAML.load_file(ENV['PLAYBOOK'].to_str)
  role_defaults = YAML.load_file('defaults/main.yml')
  role_vars = YAML.load_file('vars/main.yml')
  hash_behaviour = ENV['HASH_BEHAVIOUR'] || 'replace'
  vars = {}
  vars = merge_vars(vars, role_defaults, hash_behaviour)
  playbook.each do |play|
    play.key?('vars') && (vars = merge_vars(vars, play['vars'], hash_behaviour))
  end
  vars = merge_vars(vars, role_vars, hash_behaviour)
  set_property vars
end
```

Automating Test-Kitchen with Rake

- Rake is like make, but for Ruby.
- It uses a ‘Rakefile’ to define tasks.
- Tasks are written in standard Ruby syntax, no XML or other dependencies required.

```
require 'rake'

namespace :group1 do

  desc 'this describes task 1'
  task :task1 do
    puts 'this is task 1'
  end

  desc 'this describes task 2'
  task :task2 do
    puts 'this is task 2'
  end
end
```

```
muninn:test-role bob$ bundle exec rake --tasks
rake group1:task1  # this describes task 1
rake group1:task2  # this describes task 2
muninn:test-role bob$ bundle exec rake group1:task1
this is task 1
muninn:test-role bob$ bundle exec rake group1:task2
this is task 2
```

Rake and Test Kitchen

```ruby
require 'rake'
require 'kitchen/rake_tasks'


namespace :integration do

  namespace :vagrant do

    @loader = Kitchen::Loader::YAML.new(local_config: '.kitchen.yml')
    config = Kitchen::Config.new(loader: @loader)
    Kitchen.logger = Kitchen.default_file_logger

    desc 'Execute test suite 1'
    task :'task-1' do
      instances = config.instances.select { |obj| obj.suite.name == 'test-1' }
      instances.each do | instance |
        instance.test
      end
    end

    desc 'Execute test suite 2'
    task :'task-2' do
      instances = config.instances.select { |obj| obj.suite.name == 'test-2' }
      instances.each do | instance |
        instance.test
      end
    end

  end
end
```

- The kitchen config (`.kitchen.yml`) is loaded and logger defined.
- The tasks are defined and instances filtered by suite name.
- For each instance, the test-kitchen action `test` will be called.

The rake task can then be called and that specific test suite will be executed on all defined platforms.

```
muninn:test-role bob$ bundle exec rake --tasks
rake integration:vagrant:task-1  # Execute test suite 1
rake integration:vagrant:task-2  # Execute test suite 2
```

Speeding up Rake Tasks with Concurrency

- Concurrency is achieved by spawning each instance in a separate thread.
- Threads are managed via the `concurrency` variable being passed at run-time as a environment variable.
- Moving task execution to the `task_runner` function has the added bonus of cleaning up the code.

```
muninn:test-role bob$ concurrency=3 bundle exec rake integration:vagrant:task-1
-----> Cleaning up any prior instances of <test-1-centos-7>
-----> Cleaning up any prior instances of <test-1-debian-8>
-----> Cleaning up any prior instances of <test-1-ubuntu-1604>
-----> Destroying <test-1-centos-7>...
-----> Destroying <test-1-debian-8>...
-----> Destroying <test-1-ubuntu-1604>...
       Finished destroying <test-1-centos-7> (0m0.00s).
-----> Testing <test-1-centos-7>
       Finished destroying <test-1-debian-8> (0m0.00s).
-----> Creating <test-1-centos-7>...
       Finished destroying <test-1-ubuntu-1604> (0m0.00s).
-----> Testing <test-1-debian-8>
-----> Testing <test-1-ubuntu-1604>
-----> Creating <test-1-debian-8>...
-----> Creating <test-1-ubuntu-1604>...
```

```ruby
require 'rake'
require 'kitchen/rake_tasks'

def task_runner(config, suite_name, action, concurrency)
  task_queue = Queue.new
  instances = config.instances.select { |obj| obj.suite.name =~ /^#{suite_name}$/ }
  instances.each { |i| task_queue << i }
  workers = (0...concurrency).map do
    Thread.new do
      begin
        while instance = task_queue.pop(true)
          instance.send(action)
        end
      rescue ThreadError
      end
    end
  end
  workers.map(&:join)
end


namespace :integration do

  namespace :vagrant do

    @loader = Kitchen::Loader::YAML.new(local_config: '.kitchen.yml')
    config = Kitchen::Config.new(loader: @loader)
    Kitchen.logger = Kitchen.default_file_logger

    concurrency = (ENV['concurrency'] || '1').to_i

    desc 'Execute test suite 1'
    task :'task-1' do
      task_runner(config, 'test-1', 'test', concurrency)
    end

    desc 'Execute test suite 2'
    task :'task-2' do
      task_runner(config, 'test-2', 'test', concurrency)
    end

    desc 'Destroy all Instances'
    task :destroy do
      task_runner(config, '.*', 'destroy', concurrency)
    end
  end
end
```

Rake, Test-Kitchen, and AWS

```ruby
desc 'Execute all test suites using the Cloud Provider'
task :cloud do
  @loader = Kitchen::Loader::YAML.new(local_config: '.kitchen.cloud.yml')
  config = Kitchen::Config.new(loader: @loader)
  concurrency = (ENV['concurrency'] || '10').to_i
  task_runner(config, '.*', 'test', concurrency)
end

namespace :cloud do
  @loader = Kitchen::Loader::YAML.new(local_config: '.kitchen.cloud.yml')
  config = Kitchen::Config.new(loader: @loader)

  desc 'Execute test suite 1'
  task :'task-1' do
    task_runner(config, 'test-1', 'test', concurrency)
  end

  desc 'Execute test suite 2'
  task :'task-2' do
    task_runner(config, 'test-2', 'test', concurrency)
  end

  desc 'Destroy all Instances'
  task :destroy do
    task_runner(config, '.*', 'destroy', concurrency)
  end
end
```

```
muninn:test-role bob$ bundle exec rake --tasks
rake integration:cloud            # Execute all test suites using the Cloud Provider
rake integration:cloud:destroy    # Destroy all Instances
rake integration:cloud:task-1     # Execute test suite 1
rake integration:cloud:task-2     # Execute test suite 2
rake integration:vagrant          # Execute all test suites using the Vagrant Provider
rake integration:vagrant:destroy  # Destroy all Instances
rake integration:vagrant:task-1   # Execute test suite 1
rake integration:vagrant:task-2   # Execute test suite 2
```

Adding basic AWS support is as easy as requiring `'aws-sdk'` and loading the `.kitchen.cloud.yml`.

...However aws has it's own set of issues that should be addressed before using it in an automated fashion.

Rake, Test-Kitchen, and AWS Gotchas

- API Request limit triggers failed task.
- Tests can occasionally fail due to resource scarcity when using small instance types (`t2.micro`).
- Async SCP transfer errors can unexpectedly fail the test (error - `SCP upload failed (open failed (1))`)
- Instance ebs volumes without the flag `delete_on_termination: true` **WILL PERSIST** when an instance is deleted.
- When executing concurrent tests and a failure occurs, it's possible test-kitchen will not be aware of the instance, and it will **NOT** be possible for test-kitchen to delete through standard means.

All can be dealt with….

Failure due to API Request Limit

- “Request limit exceeded.” Error.
- Decrease concurrency count (suggested value: 8)
- Open PR to wrap commands in a retryable block:
  - https://github.com/test-kitchen/kitchen-ec2/pull/305

Failing due to Instance Resource Starvation...

- `t2.micro` systems start with 30 CPU Credits (1 credit == 1 core @ 100% for 1 minute), and have a baseline performance of 10%.
- **Creating over 100 instances in a 24 hour period WILL consume the initial CPU Credit.**
- Use CloudWatch to monitor your t2 instance credits with the following benchmarks.
  - `CPUCreditUsage`
  - `CPUCreditBalance`
- Use an appropriate instance size.
- Purchase more instance credits.

Instance type	Initial CPU credit*	CPU credits earned per hour	Base performance (CPU utilization)	Maximum earned CPU credit balance***
t2.nano	30	3	5%	72
t2.micro	30	6	10%	144
t2.small	30	12	20%	288
t2.medium	60	24	40%**	576
t2.large	60	36	60%**	864

T2 instance description: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/t2-instances.html
CloudWatch Aggregating Stats: http://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/GetSingleMetricAllDimensions.html

Failures with SCP Concurrency

- Tune the transport setting `max_ssh_sessions` in the kitchen config file (defaults to `9`).
  - **ONLY** available in test-kitchen 1.9.2+
- Increase the value of `maxSessions` in instance `sshd_config` if customizing your own AMI.

```
transport:
  ssh_key: <%= ENV['KITCHEN_SSH_KEY'] %>
  connection_timeout: 60
  connection_retries: 10
  max_ssh_sessions: 4
```

Volumes Persisting Deletes

- Add the `delete_on_termination: true` flag to the ebs volume config.
- Must be done for the public CentOS AMI.

```
platforms:
  - name: centos-7
    driver:
      block_device_mappings:
        - device_name: /dev/sda1
          ebs:
            volume_type: standard
            volume_size: 8
            delete_on_termination: true
```

Instances Persisting After a Failure

There are multiple methods of destroying objects in AWS after a failure.

...However only **ONE** way has proven to be truly reliable..


I SAY

...by Destroying Everything in the Security Group

```
desc 'Destroys ALL cloud instances and volumes in the security group.'
task :'sgroup-destroy' do
  aws_creds = Aws::Credentials.new(ENV['AWS_ACCESS_KEY_ID'], ENV['AWS_SECRET_ACCESS_KEY'])
  ec2_client = Aws::EC2::Client.new(region: ENV['AWS_REGION'], credentials: aws_creds)
  ec2 = Aws::EC2::Resource.new(client: ec2_client)
  instance_filter = [
    { name: 'instance.group-id', values: [ENV['AWS_SGROUP_ID']] },
    { name: 'instance-state-name', values: %w(running pending) }
  ]
  destroy_instances = []
  destroy_volumes = []
  ec2_instances = ec2.instances(filters: instance_filter)
  ec2_instances.each do |instance|
    destroy_instances.push(instance.instance_id)
    instance.volumes.each do |volume|
      destroy_volumes.push(volume.volume_id)
    end
    puts "Sending Terminate request for instance: #{instance.instance_id}"
    instance.terminate
  end
  unless destroy_instances.empty?
    ec2_client.wait_until(:instance_terminated, instance_ids: destroy_instances) do |w|
      w.before_attempt do
        puts 'Polling for instance termination...'
      end
    end
  end
  destroy_volumes.each do |volume_id|
    begin
      puts "Deleting volume: #{volume_id}"
      ec2_client.delete_volume(volume_id: volume_id)
    rescue Aws::EC2::Errors::InvalidVolumeNotFound
      next
    end
  end
  unless destroy_volumes.empty?
    ec2_client.wait_until(:volume_deleted, volume_ids: destroy_volumes) do |w|
      w.before_attempt do
        puts 'Polling for volume deletion...'
      end
    end
  end
```

1. Filters instances for any in the security group that are in a `running` or `pending` state.
2. Generate a list of volumes attached to those instances.
3. Sends signal to instance to terminate.
4. Poll until all instances are destroyed.
5. Iterate through volumes gathered earlier and send signal to destroy.
6. Poll until all volumes are destroyed.

Bringing it all Together...CI/CD Integration

- Many different CI/CD systems, and all integrate differently. TravisCI, Jenkins, CircleCI, Drone.io, Concourse etc.
- Common components among all of them for working with test-kitchen:
  - Set environment variables - AWS variables
  - Execute command - Rake Command
  - Signal Success/Fail - End Result

TravisCI

https://travis-ci.org/

- Only integrates with GitHub.
- Free for Public Repos.
- Ansible Galaxy's default.
- Incredibly easy to use.
- Tests branches and Pull Requests.
- Execution controlled via `.travis.yml` file located at root of git repo.
- Caveats:
  - No advanced reporting. Either pass/fail by Exit Code.
  - Builds running more than 50 minutes will be killed.
  - Log limit size of 4MB, if log is going to be larger a wrapper script (example: https://gist.github.com/roidrage/5238585) is required.

Travis File Config

```
---
language: ruby
rvm:
  - 2.3.0

env:
  global:
    - AWS_INSTANCE_TYPE=t2.micro
    - KITCHEN_SSH_KEY=provision.key

before_install:
  - bundle install

script:
  - bundle exec rake integration:cloud

after_script:
  - bundle exec rake integration:cloud:sgroup-destroy
```

- https://docs.travis-ci.com/
- Environment variables that are not considered secret can be placed in travis file without encryption
- Most items are done as lists and not key/value pairs.

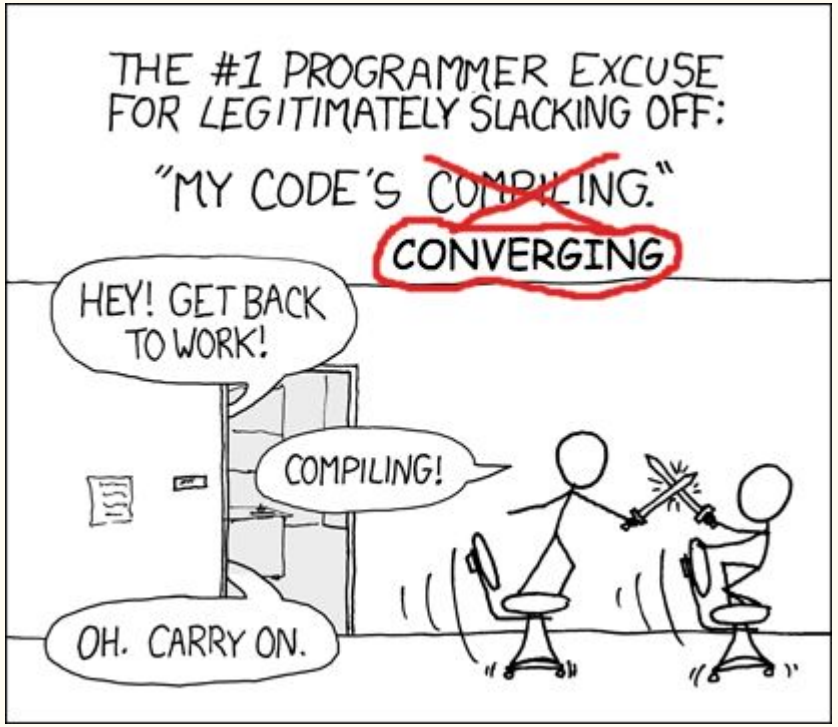
Adding Secrets to Travis

Travis generates a pair of RSA keys on a per repo basis that can be used to store secrets in a public repo securely.

1. Install and Configure travis CLI - https://github.com/travis-ci/travis.rb
2. Add secrets to the travis config with: `travis encrypt <env var name>=value --add` . This should include `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, `AWS_SSH_KEY_ID`, `AWS_SGROUP_ID`, `AWS_REGION`, and `AWS_AVAILABILITY_ZONE`.
3. Encrypting files is similar. To encrypt the AWS ssh key, use the following: `travis encrypt-file <filename> --add`. This will create a new encrypted file ending with the `.enc` extension.

BE CERTAIN YOU DO NOT ADD YOUR UNENCRYPTED FILES TO THE GIT REPO!

```
language: ruby
rvm:
- 2.3.0
env:
  global:
  - AWS_INSTANCE_TYPE=t2.micro
  - KITCHEN_SSH_KEY=provision.key
  - secure: Mf7MyJU3RfasOksQP3GVqdkfApsS01TJVVWuo/C70fqAMgwtzgzLzu6d+JyDfNsyFd9reSskMzVe4FNkzYB2vbmVpzTfvgckZqJodHsZSWG8ECzskjqRl5jSng2b5xk3guX3V1XpMnZI4zazw5BxALD89MIPCVRn9GMHZFYq8QYFPWQWu28Dl0rtzsE20HF2Fw9CCTSZiy2PuA0Kzyq523aETDgCnSDl9tsWF+cpYXDqbBZxkzCAaFIUgZdT9Lfh5dqNhbClCL3SJmHFc4P34lXGO4b1somH8h0H5lQ/MrAOkAkh/Zd8szIgVWodf024zxYx70DHRkL5V0ZO4A1xPOKlHy7hC+6stiBimTMey0+8Teu6tS2n/xp9rEFG+NDyMVdLMhni+4L1grTx8+DCqStE+YAejBlTyawYWgQxELFTolamHRG0b4GQaLEQZz1cAmgxxusN5RAmTNnijGNVQ2lY4rJAbj44wOa8nl3A1McKFBR9OznpYXizlj1ET1PC6S+uvSvhFOu0y6yO7xADEnhG2egHjDM5siYvnalnjBA4q1ATrtwgHpFwNjGG2Gl7gj22VdpPCvrsjr27wE3/w+cSJpVnkUhT6v8I/LGxfGRNZLpUNg6zfPx9sG0m7EDU874YPKLxuuIdw9JfaGsNHRifr6ysW2M42Ybn672h0OAc=
  - secure: kzrItFz4Vcax1ytD2MGwVKqg8VPr0DIih9rqY1FhuGlbTcf8iqCtHvarYtTfIxniguLmh6OVUVfjckSHyta15oxQNy1dR8CmE/dHrnwDCsdthlGWP7jVEtBOk7FMmPHSfZcKzvSMcyazJydOB1gV6jths65AE1PnHqG4Muf0g+HCrPXRgGyU3FFUqpGqIy+wz3ukyXY/UrWrbtUvIH0U326g5SgORZf7rEXZPRkoVDR2XP3NuRX4wbpMaQ2IGV5DrII3NcWKgM8rhyr7X44zX1P4hArfPKhw0FKKCAGvFRhf3gSvB2qfFrUXNVl+lL8qnyILt8FgvNIZOX51OLdEK7aHr5vXwVZBq1oBUQu+bR22mqF6ydVnu77n5ILq1hWwAor4jAqTRzkC0nqvlPPTz3GpoAzEWZ3vqBLr/7HpmXoea/sJuX9XlmjO/tczwIdivQ4WvAc/R2e5PD80FQVRA4zfvIVL3TnNr/X2b/o0NjOkm1y2x2P7iLG1xg8K1VujQizi2RCno+umkk+v88LOjU7mVZ4xAFzPKhDyS8aG5DMRxJXMBc3g7oaheei1Q/p14WAgSS1zUJsFIUkEEol7sB/wM2czGmHl5b5xL/Mv1grlnUghtlsbmUMXm7hLHOPAVGsrz6zF8zgFaqLa3+qCxvmRdsptPNhu8jNeu/gl0x4=
  - secure: RmNkiLV3li7bRxi3ii5y0pXdylg51twl45VVt5EVaDUtUNux7ur+tdJiRRJ82fEolJo3z+l+sH5TQgVgY7MeaHycPFpBs2ecpdD5lWNxlw5Wt9DH0IPtk1WI0BUEwqnp/Zrlr0gS9MIagXxHgS+Il55cMq8tRYDmLG+935KnoQwaH+U7g6Qtau0TdnQxj3rs7TSDO/heo+nas9C/px7wvhnUgOrAAfQqJEah8GcjiqRuTiHnhKrIpI+b8+BA3vpsG9HMSpB5GQrYyguCGwv4aANtN+VSSWEYUWdgNGvYLny5m9o4IAOviwNk0Yk/SrI7pNbGG7JQw57MkkN4KG+XP1Lp56vuUrpN6X3jB35RC0zYoYlN/oK1JIg4qFxTZTtKG70FTXFBzrSqRF5YUuVXpoDLTpwJGZeTbXJmqUTsWarMs1nP/kPtX9+yvPQMUjYNV52iWg46DTscKwMXWp0Nqeo7LNpVaU2wlLpke6qBtUGUsr78+Ebbi2zHhJ9dqu+cSzi3/671cwR6VAXEQ7kwjNAesxWPX5WHfe2c11ndTmxoYy5PcAxuJnbqZMieoR8SjhhnjcHv7Dm9EMCKgOfr6YEP5hXzE7kMsxP0ricNrqFSzfvH126hSXts7uGU7PBNLhtBkFGdJVSwYEhXa/wqVOAC9vy4oua1SNEa7vapC+I=
  - secure: VpS2VSImSFbVOGAdKtBpmm6m5S7WKhfqSrsJILdHnWB6HUsf9JUXr6gpB0sK/cyyOY2Uxkm0OvOWBbRn2Sy+0wn6vt+u/7OIWVpcE0GYGRB/u3meuReukJnHkDpDV1eDk6oiEgyA3Efhj5eckQNeVXY+799yppZRHRMq5AUkDrGWAekS1oOp4/VsWHOJ8R1vOsmKSqlLfmYDgr14yCOs6Eq/TQzfJxCfJ8Ijf1RpFjmH9vIF967z7yp6pTQZlAcweF7TNGjlftKEbKjPMl4gia3Ht8LvVJ+UC7Nn7aHh20Ps1V7225JXBRM80qhPtg8Yut1JqnhXLlwbsOZfWWjgJoizKh/TGYpIuSsssCJoKomSgQxKOC7YbuQD9hfOMULset7WEWfA7gXYIBO+kfqqSmQwlZl02swx2JXiJvYtA4fXJ5xpkgSpToHHtfz1TtcqARfAxR9WnMZrniBUcKBkxJ3+qHvlyg4Gt+VSMung7wByhq8plJHZ0XjbzaBfpp6Giqd+teHIamu3u9qogVRvO7cwul+KLHr4DdDluuZGSLUHOfDAeCLCRLCoD4mddM0JBjPDgA1xYCAnjAjESyB2xg6kgOZ62Hr/RzEpaDCQsy+xSQ4mHWyCnEKGCbO0xcvT2whQ1NrMlsC5kaPEAzGqfqaNwYIjfkqzpYCeAsThoaA=
  - secure: lHrqG0efbsJf5rSmz3GxcFpUURqUq9x1JCCTRUYEGf7n4mSAl3/XVFOeR1gybKVfAkeOVDdybGKFvvxcyjN2y4C/gkPpCXJhR9zAViLtq4LIil9TNx6qV6MfJpJmYbWng+eVWvaeThFyclLxIMHa8fnmGy93coT7MdxhxVPzBwty3ocS3IZOgAfsFX5SIfcePIud+sp1SxDiMWnfk2ZVQkirHkl4bV51ufXRikomtRliS/deal7On9iy4oeRwEsjzceICeh7A9A4zYG6S9NZozri0xQIZmrgP2M8AqFD6xpz/hdrRfq9EVXbcxAtJKsy2fJpsH7+p5+2K9c3TyKSi4eOm7Tbj4Rm01TIH3rYw8wBsuSXc427Y+Yl4LELRXSxMpE5U2SIFpkOhH1K1FCZYSL9D9ZsJIFmHH3MKo46aMrMpKIN8mLZi3OM8UJ8xtJ/+jMLM94FHWV/btIA/P/J1S1FMj5AW77ymr/m8XMztqkk1VmQ3U3llh1ZobYsP+AKlpZylcAmUTblx6O435kyLnGEyBbFIwoS4d8rxp/0qCw4jmLz7C5Fh8pAidurD01Su3rMKgv0HOHJ+LJsM0jCG5f9hXPrlvt8x4FNrRZx9mqDrd6CNepxV/op1vtv7lPelmyKAJAcQeLiCXjnEFRX71nk2UparC+UQyivfBgSs0Q=
  - secure: dLyzUDp9a5S5Z8+fCx+GX1pxvjoyjbLVrV+tfJ1ekmQKFSbCg2kftvoVJ06HckxAsgk2hyZFYaQr5hoYEg2k8LoVeyOJJgFVef+7/wEoGN3l92plajLFt9Jjsqn0jjCJjexncI1p3Cb7TaU+NifzBuWkOwH3D3KVao/Ei0jFkVxetG8fdMQ0KL8q8FH1DlWD0yurBOhRYcoZktFG27kfZ1eQRTH1TWbEusf1I5KLJHgOTzIh5F/K2IcCwL704qzSy7v6xwLURY+1/XtRq+DpUYhAH37uT1ai7xyjs/LGy2NB2qcDFs2vRv56eAxF5N2KDcieLFpSzzb4mdT+bPHhRLdZ5EKbKyVyDw2I8GsUHFx7XqJxUG4xd34yiqvns64XUTs6Yvx76ls60+3GzVkgE5yLIMBK9J1wwecvvI4XFzsLdXOiMojaW2NJ1mx4S0YEgYirVV6WPP4qBmN7rzkENaiULKLRSxbjORoC3Fzyx1I/SvRa85+BzbHAxnGrf9Fl+DFmG2xT2vTouV0EZxtWMBPXedEmKCvho/nJAmAm6ElM2M4CgYYmFujK16CGrduZFGNNBPW1RELT2Heu14VsHKsod88OwK0HnVLMPk7TngP7mb4AxrhxGUQQk5l+euu/FCaJ4ieX8GwxVmZxd8R+TW/RNQNeZ/O9IFr+U3rfWjI=
before_install:
- openssl aes-256-cbc -K $encrypted_c927e4b32ed8_key -iv $encrypted_c927e4b32ed8_iv
  -in provision.key.enc -out provision.key -d
- bundle install
script:
- bundle exec rake integration:cloud
after_script:
- bundle exec rake integration:cloud:sgroup-destroy
```



Now give Travis some time to work...

TravisCI Build Results

mrbobbytables / test-role

build passing

Current | Branches | Build History | Pull Requests | More options

✓ master Bob Killen	fix test in spec file	#2 passed e511b7b	27 min 43 sec 26 minutes ago
✗ master Bob Killen	added travis support	#1 failed 709523c	30 min 24 sec about an hour ago

TEST ALL THE THINGS!