Introduction to Testing Chef

Cookbook development workflow

DCAST – March 2015

https://github.com/nathenharvey/dcast-testing-infrastructure

CHEF

Nathen Harvey

- nharvey@chef.io
- @nathenharvey

- Co-organizer – DevOps DC

- http://bit.ly/farmer-nathen (explicit)
- http://bit.ly/farmer-nathen-sfw

CHEF

What is Chef?

Leader in next generation of automation solutions

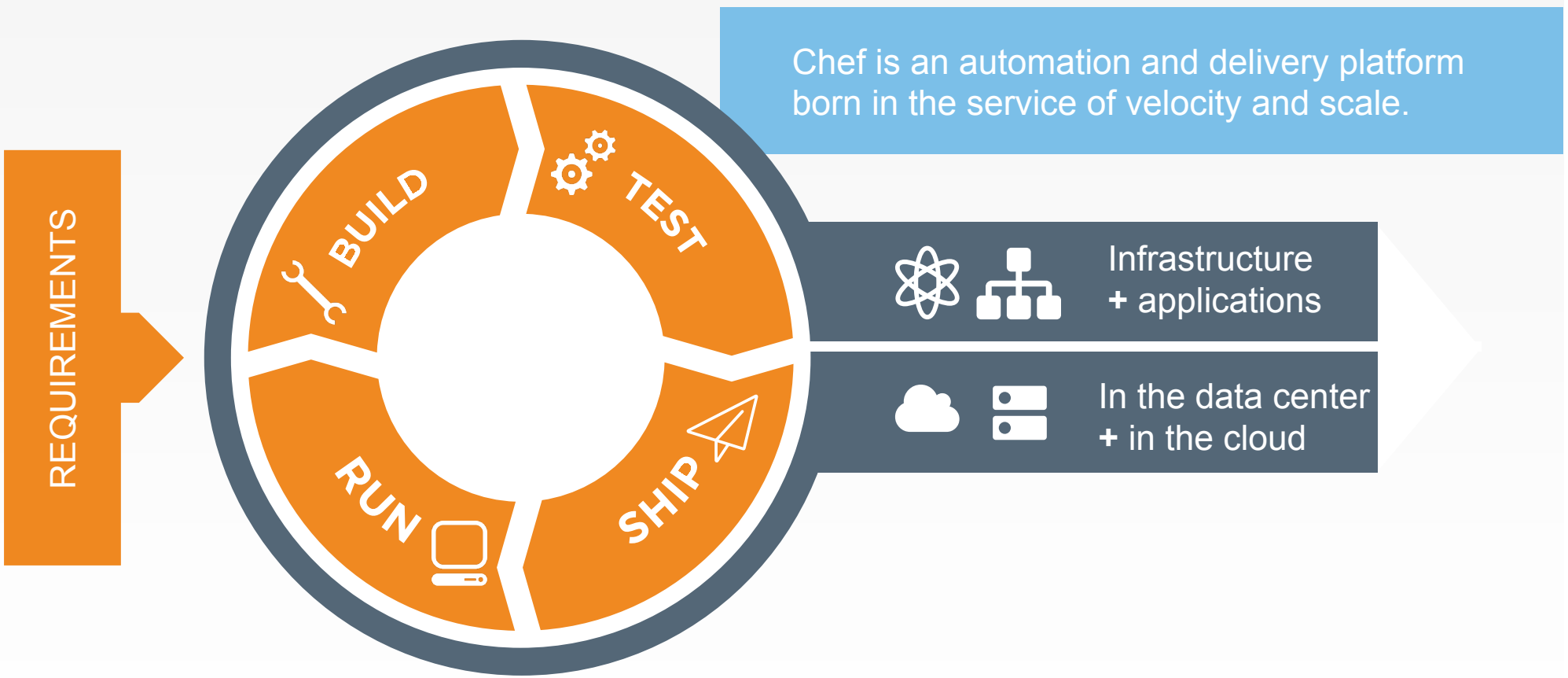
Leading the DevOps movement
– Alignment between Infrastructure and Apps

Key partners: Amazon Web Services, Microsoft,
IBM, HP, Accenture, PWC, VMWare

Offices in Seattle, San Francisco, Atlanta, and London

CHEF

The New Face of Business



CHEF

How Chef Works

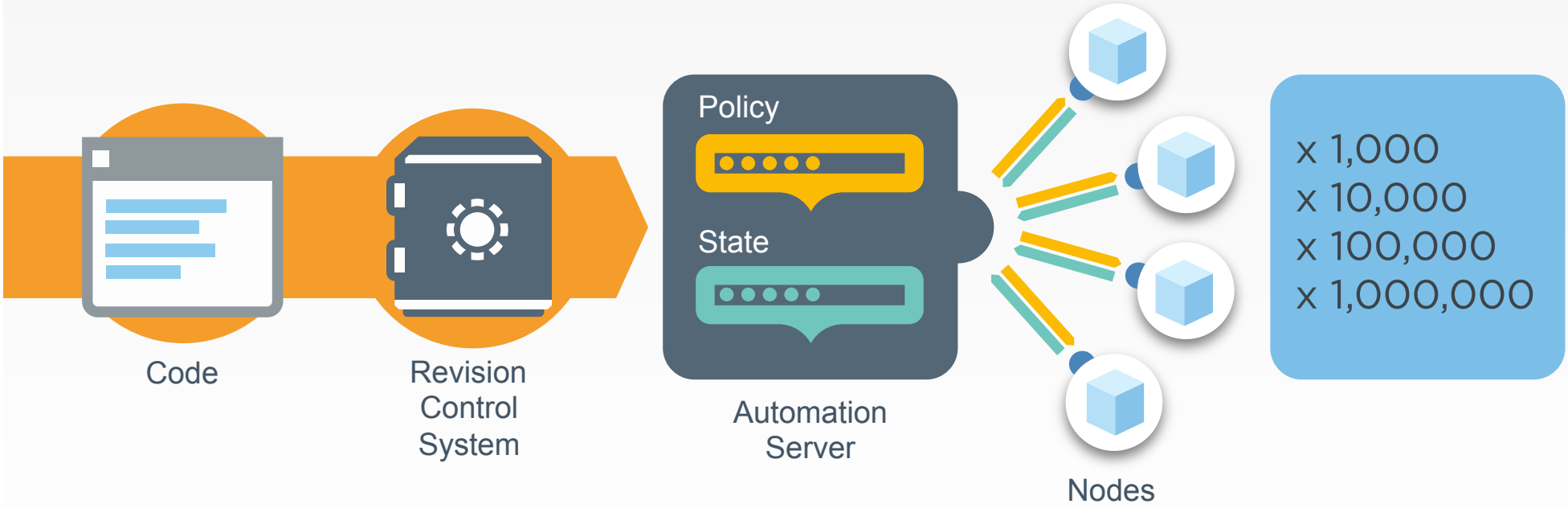
- Define reusable resources and infrastructure state as code
- Manages deployment and on-going automation
- Community content available for all common automation tasks

```
package “httpd” do
    action :install
end

template “/etc/httpd/httpd.conf” do
    source “httpd.conf.erb”
    mode “0755”
    owner “root”
    group “root”
end

service “httpd” do
    action :start, :enable
done
```

```
windows_feature ‘IIS-WebServerRole’ do
    action :install
end

windows_feature ‘IIS-ASPNET’ do
    action :install
end

iis_pool ‘FooBarPool’ do
   runtime_version “4.0”
   action :add
end

iis_site ‘FooBarLTDSite’ do
   protocol :http
   port 80
   path “C:\\FooBarSite”
   action :add :start
end
```

CHEF

Chef Architecture



CHEF

Resources

Fundamental building blocks

Resources

- Piece of the system and its desired state

Resources - Package

Package that should be installed

```
package "mysql-server" do
  action :install
end
```

Resources - Service

Service that should be running and restarted on reboot

```
service "iptables" do
  action [ :start, :enable ]
end
```

Resources - Service

File that should be generated

```
file "/etc/motd" do
  content "Property of Chef Software"
end
```

Resources - Cron

Cron job that should be configured

```
cron "restart webserver" do
  hour '2'
  minute '0'
  command 'service httpd restart'
end
```

Resources - User

User that should be managed

```
user "nginx" do
  comment "Nginx user <nginx@example.com>"
  uid 500
  gid 500
  supports :manage_home => true
end
```

Resources - DSC

DSC resource that should be run

```
dsc_script 'emacs' do
  code <<-EOH
  Environment 'texteditor'
  {
    Name = 'EDITOR'
    Value = 'c:\\emacs\\bin\\emacs.exe'
  }
  EOH
end
```

Resources – Registry Key

Registry key that should be created

```
registry_key "HKEY_LOCAL_MACHINE\
\SOFTWARE\\Microsoft\\Windows\
\CurrentVersion\\Policies\\System"
do
  values [{
    :name => "EnableLUA",
    :type => :dword,
    :data => 0
    }]
  action :create
end
```

Test and Repair

Resources follow a test and repair model

```
package "vim"
```

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

Test Is vim installed?

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

Test Is vim installed?

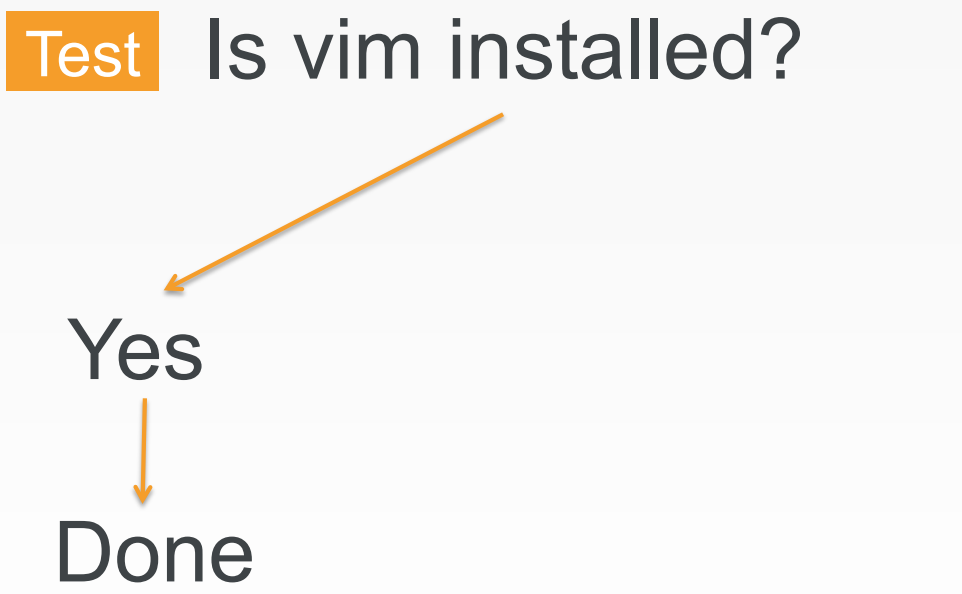
Yes

CHEF

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

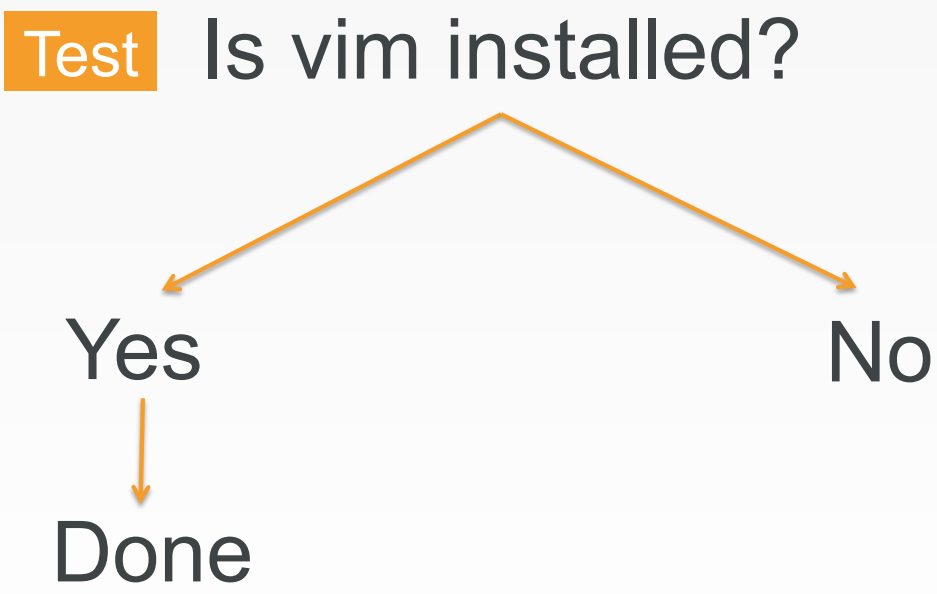


CHEF

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

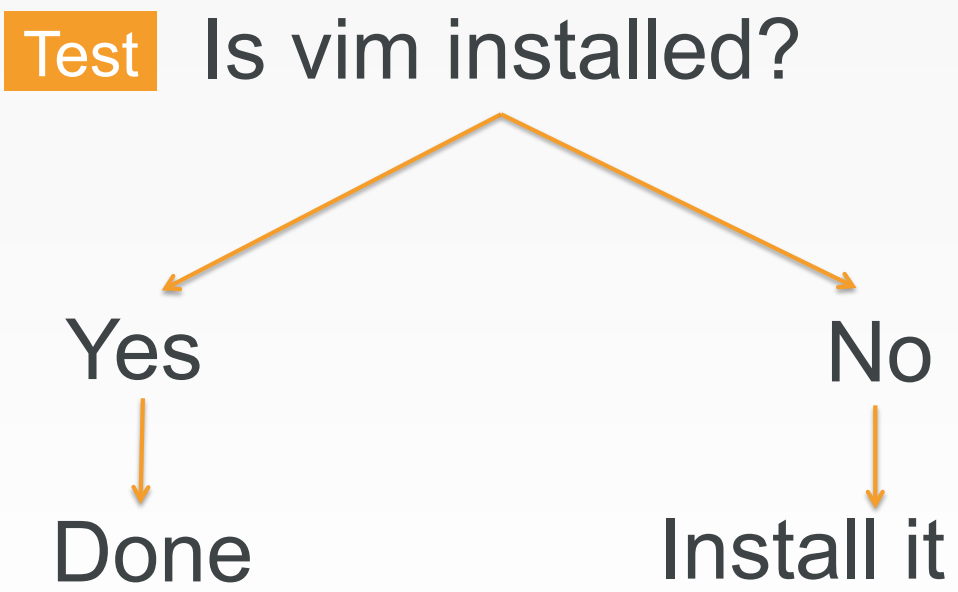


CHEF

Test and Repair

Resources follow a **test** and repair model

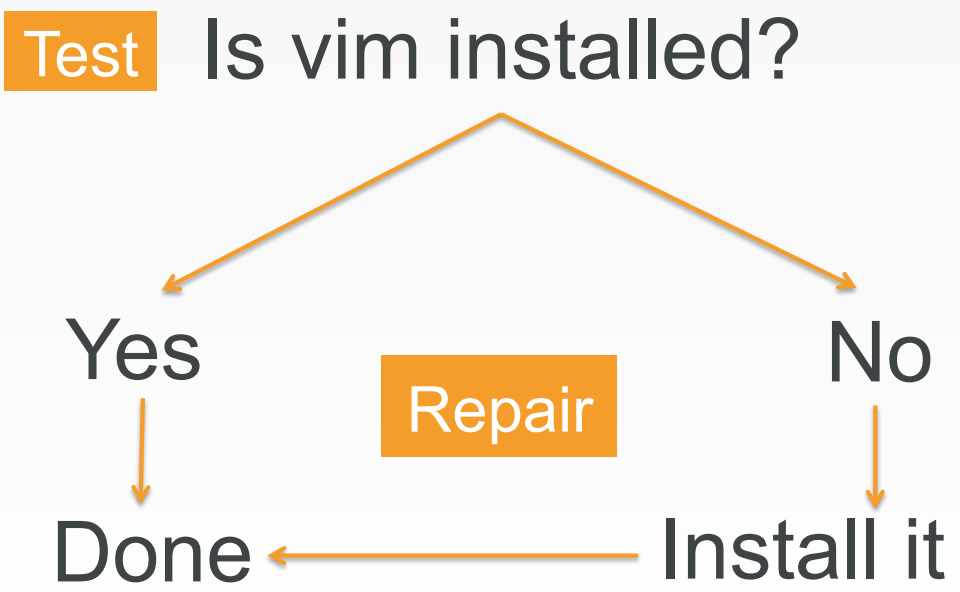
```
package "vim"
```



Test and Repair

Resources follow a **test** and **repair** model

```
package "vim"
```



CHEF

Resources

- package
- template
- service
- directory
- user
- group
- dsc_script
- registry_key
- powershell_script
- cron
- mount
- route
- …and more!

Testing Chef Cookbooks

CHEF

Our process

- Write policy
- Apply policy
- Verify policy

- Not bad for the simple case, will quickly get untenable

Faster Feedback

- Speed-up the feedback loops with automated testing.
- Have confidence in your changes before you run them in production

Chef Testing

- Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code follow our style guide?

Test-driving infrastructure

- We are going to use a relatively simple scenario
- We are going to explore many facets of testing
- We are going to follow a test-first, test-driven model

Our Scenario

- We want a custom home page available on the web.

Create an apache cookbook

```
$ chef generate cookbook apache
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[/home/chef/chef-repo/cookbooks/apache] action create
    - create new directory /home/chef/chef-repo/cookbooks/apache
    - restore selinux security context
  * template[/home/chef/chef-repo/cookbooks/apache/metadata.rb] action create_if_missing
    - create new file /home/chef/chef-repo/cookbooks/apache/metadata.rb
    - update content in file /home/chef/chef-repo/cookbooks/apache/metadata.rb from none to 4c0e2d
    (diff output suppressed by config)
    - restore selinux security context
  * template[/home/chef/chef-repo/cookbooks/apache/README.md] action create_if_missing
    - create new file /home/chef/chef-repo/cookbooks/apache/README.md
    - update content in file /home/chef/chef-repo/cookbooks/apache/README.md from none to 5c3d3a
    (diff output suppressed by config)
    - restore selinux security context
  * cookbook_file[/home/chef/chef-repo/cookbooks/apache/chefignore] action create
 …
```

Create an apache cookbook

```
$ cd apache
```

Chef client success status

- Requirements to verify chef-client success:
  - A target server running the same OS as production

Chef client success status

- Requirements to verify chef-client success:
  - A target server running the same OS as production
  - A chef-client with access to the cookbook

Test Kitchen

- Test harness to execute code on one or more platforms
- Driver plugins to allow your code to run on various cloud and virtualization providers
- Includes support for many testing frameworks
- Included with ChefDK

Test Matrix

- Two operating systems

ubuntu-12.04
centos-6.4

Test Matrix

- Two operating systems
- One recipe

	default
ubuntu-12.04	apache::default
centos-6.4	apache::default

Test Matrix

- Two operating systems
- Two recipes

	default	ssl
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl

Test Matrix

- Three operating systems
- Two recipes

	default	ssl
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl
ubuntu-14.04	apache::default	apache::ssl

Configuring the Kitchen

OPEN IN EDITOR: apache/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    attributes:
```

SAVE FILE!

CHEF

.kitchen.yml

- driver - virtualization or cloud provider

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    attributes:
```

.kitchen.yml

- provisioner - application to configure the node

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    attributes:
```

.kitchen.yml

- platforms - target operating systems

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    attributes:
```

.kitchen.yml

- suites - target configurations

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    attributes:
```

.kitchen.yml

	default
ubuntu-12.04	apache::default
centos-6.4	apache::default

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4


suites:
  - name: default
    run_list:
      - recipe[apache::default]
```

.kitchen.yml

	default	ssl
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
  - name: ssl
    run_list:
      - recipe[apache::ssl]
```

.kitchen.yml

	default	ssl
ubuntu-12.04	apache::default	apache::ssl
centos-6.4	apache::default	apache::ssl
ubuntu-14.04	apache::default	apache::ssl

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4
  - name: ubuntu-14.04

suites:
  - name: default
    run_list:
      - recipe[apache::default]
  - name: ssl
    run_list:
      - recipe[apache::ssl]
```

Move to the apache cookbook directory

```
$ cd ~/chef-repo/cookbooks/apache
```

List the Test Kitchens

```
$ kitchen list
```

```
Instance             Driver   Provisioner  Last Action
default-ubuntu-1204  Vagrant  ChefZero     <Not Created>
default-centos-65    Vagrant  ChefZero     <Not Created>
```

Create the kitchen

```
$ kitchen create
```

```
-----> Starting Kitchen (v1.3.1)
-----> Creating <default-centos-64>...
       Step 0 : FROM centos:centos6
        ---> 68eb857ffb51
       Step 1 : RUN yum clean all
        ---> Running in cdf3952a3f18
       Loaded plugins: fastestmirror
       Cleaning repos: base extras libselinux updates
       Cleaning up Everything
        ---> b1cccd25ce55
       Removing intermediate container cdf3952a3f18
       Step 2 : RUN yum install -y sudo openssh-server openssh-clients which curl
        ---> Running in 9db69ace459d
       Loaded plugins: fastestmirror
```

Use AWS for test instances

OPEN IN EDITOR: `.kitchen.yml`

```
---
driver:
  name: ec2
  aws_access_key_id: <%= ENV['AWS_ACCESS_KEY_ID'] %>
  aws_secret_access_key: <%= ENV['AWS_SECRET_ACCESS_KEY'] %>
  aws_ssh_key_id: <%= ENV['AWS_KEYPAIR_NAME'] %>
  ssh_key: <%= ENV['AWS_KEY_PATH'] %>
  availability_zone: <%= ENV['AWS_AVAILABILITY_ZONE'] %>


provisioner:
  name: chef_zero

platforms:
  # - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
    run_list:
      - recipe[apache::default]
    attributes:
```

SAVE FILE!

CHEF

Chef client success status

- Requirements to verify chef-client success:
  - A target server running the same OS as production
  - A chef-client with access to the cookbook

Go to the right place

```
$ cd ~/chef-repo/cookbooks/apache
```

Apply our policy

```
$ kitchen converge
```

```
-----> Starting Kitchen (v1.3.1)
-----> Converging <default-centos-64>...
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 3.2.3...
       Removing non-cookbook files before transfer
       Preparing validation.pem
       Preparing client.rb
-----> Installing Chef Omnibus (install only if missing)
       downloading https://www.chef.io/chef/install.sh
         to file /tmp/install.sh
       trying curl...
```

Chef Testing

- Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

Chef Testing

- ✓ Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

Verifying node state

Serverspec

CHEF

Chef Testing

- ✓ Did chef-client complete successfully?
- Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

Manually inspect the test node

```
$ kitchen login
```

```
kitchen@localhost's password:
```

Manually inspect the test node

```
$ curl http://localhost
```

```
curl: (7) couldn't connect to host
```

Serverspec

- Write tests to verify your servers
- Not dependent on Chef
- Defines many resource types
  - package, service, user, etc.
- Works well with Test Kitchen
- http://serverspec.org/

CHEF

Move to the proper directory

```
$ cd ~/chef-repo/cookbooks/apache
```

Default location for tests

- Test Kitchen will look in the `test/integration` directory for test-related files

Write a Serverspec test

OPEN IN EDITOR: test/integration/default/serverspec/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do

  # Serverspec examples can be found at
  # http://serverspec.org/resource_types.html

  it 'does something' do
    skip 'Replace this with meaningful tests'
  end

end
```

SAVE FILE!

CHEF

Generic Expectation Form

```
describe "<subject>" do
  it "<description>" do
    expect(thing).to eq result
  end
end
```

Run the serverspec test

```
$ kitchen verify
```

```
 apache::default
         does something (PENDING: Replace this with meaningful tests)

       Pending: (Failures listed here are expected and do not affect your suite's status)

         1) apache::default does something
            # Replace this with meaningful tests
            # /tmp/busser/suites/serverspec/default_spec.rb:8


       Finished in 0.00113 seconds (files took 1.16 seconds to load)
       1 example, 0 failures, 1 pending

       Finished verifying <default-centos-64> (0m9.16s).
-----> Kitchen is finished. (0m10.67s)
```

How would you test our criteria?

- We want a custom home page available on the web.

What is success?

- Package is installed?
- Page is displayed?
- What else?

Verify package is installed

OPEN IN EDITOR: test/integration/default/serverspec/default_spec.rb

```
require 'serverspec'
set :backend, :exec

describe "apache" do
  it "is awesome" do
    expect(true).to eq true
  end

  it "is installed" do
    expect(package("httpd")).to be_installed
  end
end
```

SAVE FILE!

CHEF

Exercise the test

```
$ kitchen verify
```

```
 apache
         is awesome
         is installed (FAILED - 1)

       Failures:

         1) apache is installed
            Failure/Error: expect(package("httpd")).to
be_installed
              expected Package "httpd" to be installed
              /bin/sh -c rpm\ -q\ httpd
              package httpd is not installed
```

Test is failing, make it pass

- Test-driven development involves
  - Write a test to verify something is working
  - Watch the test fail
  - Write just enough code to make the test pass
  - Repeat

Update our cookbook

OPEN IN EDITOR: ~/chef-reop/cookbooks/apache/recipes/default.rb

```
package "httpd"
```

SAVE FILE!

CHEF

Converge the node again

```
$ kitchen converge
```

```
-----> Converging <default-centos-64>...
       Preparing files for transfer
       Resolving cookbook dependencies with Berkshelf 3.1.5...
       Removing non-cookbook files before transfer
       Transfering files to <default-centos-64>
       [2014-11-10T09:20:26+00:00] INFO: Starting chef-zero on host localhost, port 8889
with repository at repository at /tmp/kitchen
         One version per cookbook

       [2014-11-10T09:20:26+00:00] INFO: Forking chef instance to converge...
       Starting Chef Client, version 11.16.4
       [2014-11-10T09:20:27+00:00] INFO: *** Chef 11.16.4 ***
       [2014-11-10T09:20:27+00:00] INFO: Chef-client pid: 571
 ...
```

Exercise the test

```
$ kitchen verify
```

```
 apache
         is awesome
         is installed

       Finished in 0.48165 seconds (files took 1.05
seconds to load)
       2 examples, 0 failures
       Finished verifying <default-centos-64>
(0m5.64s).
-----> Kitchen is finished. (0m11.84s)
```

What else will you test?

- Is the service running?
- Is the port accessible?
- Is the expected content being served?

- Make sure everything works from a fresh kitchen, too!

Extend the Serverspec test

OPEN IN EDITOR: test/integration/default/serverspec/default_spec.rb

```
describe 'apache' do
  it "is installed" do
    expect(package 'httpd').to be_installed
  end

  it "is running" do
    expect(service 'httpd').to be_running
  end

  it "is listening on port 80" do
    expect(port 80).to be_listening
  end

  it "displays a custom home page" do
    expect(command("curl localhost").stdout).to match /hello/
  end
end
```

SAVE FILE!

CHEF

Verify the kitchen

```
$ kitchen verify
```

```
apache
        is installed
        is running
        is listening on port 80
        displays a custom home page

      Finished in 0.3968 seconds
      4 examples, 0 failures
      Finished verifying <default-centos-64> (0m4.25s).
```

Kitchen Workflow

- `kitchen create`
- `kitchen converge`
- `kitchen verify`
- `kitchen destroy`

- All at once with `kitchen test`

Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

Even Faster Feedback

ChefSpec

CHEF

Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

This is too slow!

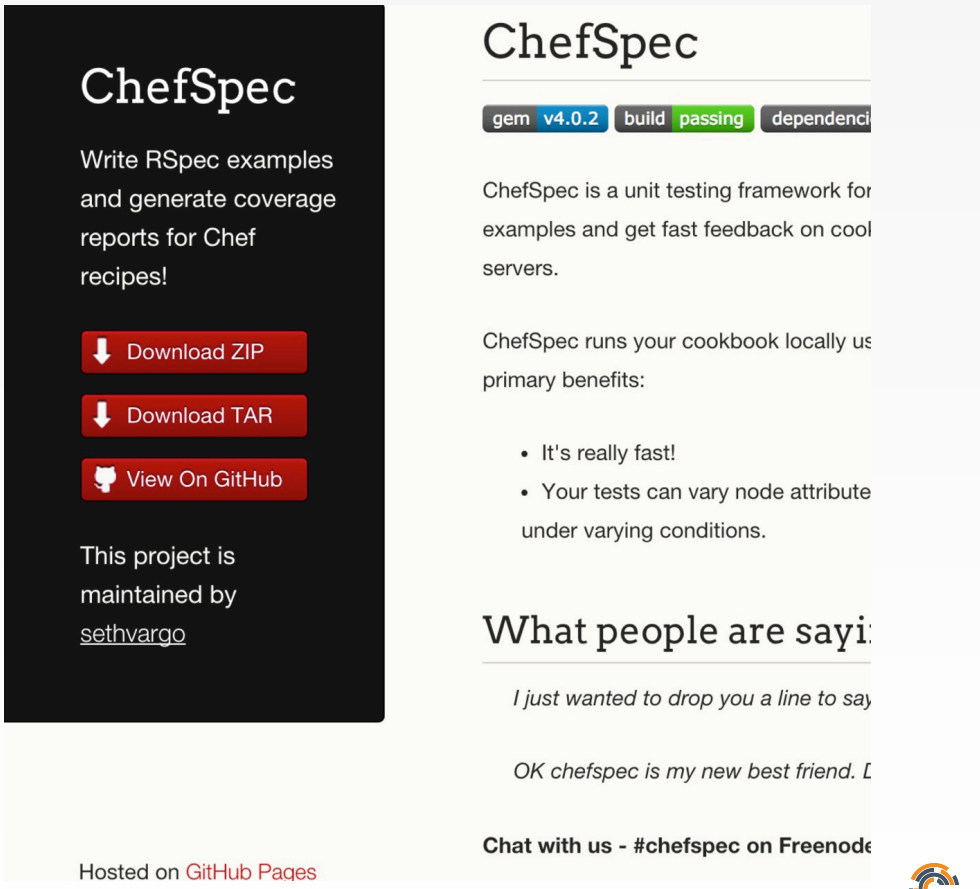
- To test our code, we need to spin up a test kitchen, converge a node, execute some tests.
- Our simple test case takes about 2 minutes to fully execute.

Properly configured resources

- We need a way to verify that the resources in our recipes are properly configured
- We want to get faster feedback

ChefSpec

- Test before you converge
- Get feedback on cookbook changes without the need for target servers



http://sethvargo.github.io/chefspec/

Write a ChefSpec test

OPEN IN EDITOR: spec/unit/default.rb

```
require 'spec_helper'

describe 'apache::default' do

  context 'When all attributes are default, on an unspecified platform' do

    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      chef_run # This should not raise an error
    end

  end
end
```

SAVE FILE!

CHEF

Run the ChefSpec tests

```
$ rspec spec
```

```
.

Finished in 0.00865 seconds (files took 5.5 seconds to load)
1 example, 0 failures
```

Chef Testing

- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- ✓ Are the resources properly defined?
- Does the code following our style guide?

Clean code

Follow best practices, avoid mistakes

CHEF

Foodcritic

- Check cookbooks for common problems
- Style, correctness, deprecations, etc.
- Included with ChefDK

http://www.foodcritic.io/

Change our recipe

OPEN IN EDITOR: recipes/default.rb

```
package_name = "httpd"

package "#{package_name}"

service "httpd" do
  action :start
end

template "/var/www/html/index.html" do
  source "index.html.erb"
end
```

SAVE FILE!

CHEF

Run Foodcritic

```
$ foodcritic .
```

```
FC002: Avoid string interpolation
where not required: ./recipes/
default.rb:7
```

Chef Testing

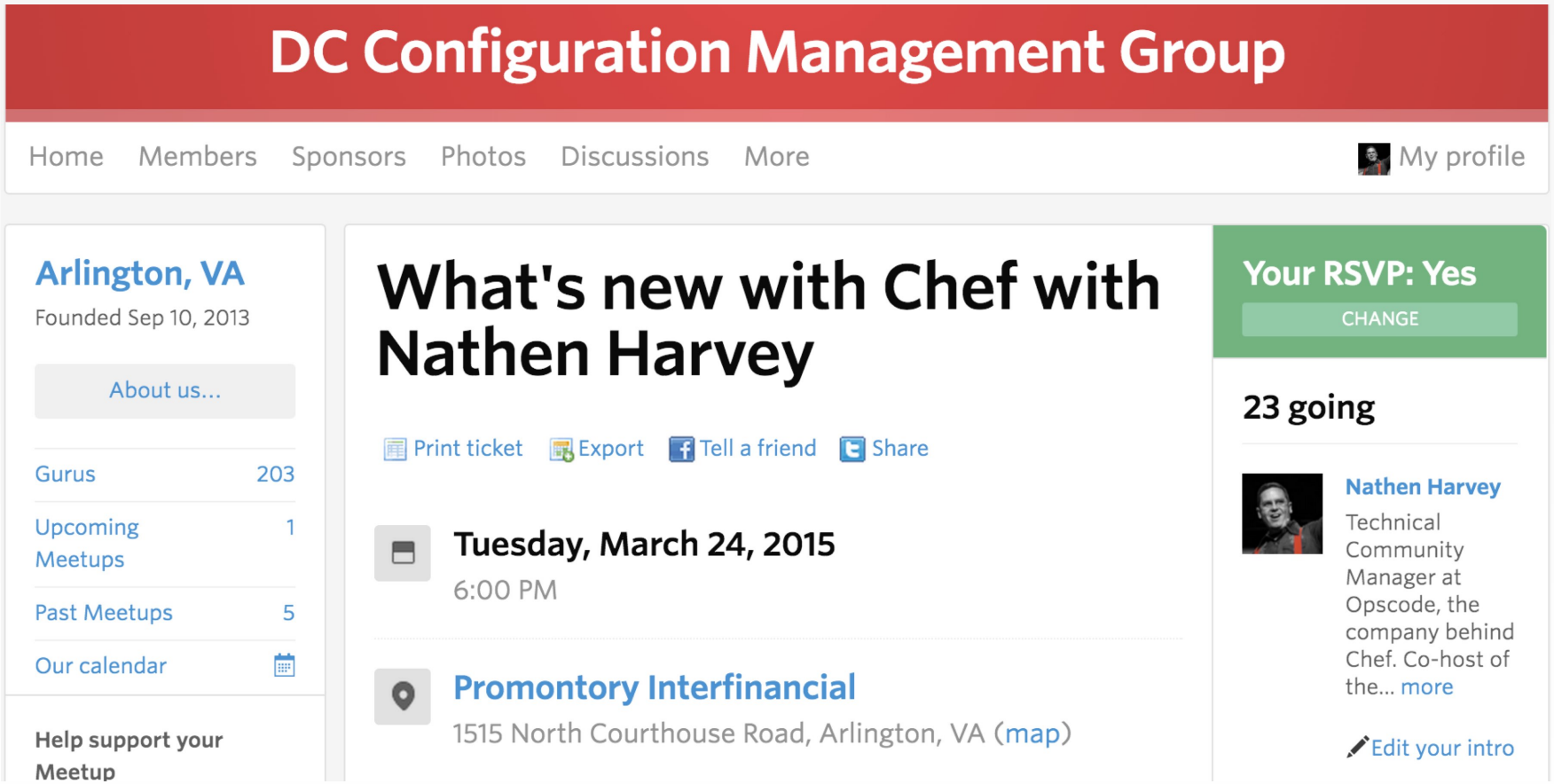
- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- ✓ Are the resources properly defined?
- ✓ Does the code following our style guide?

ChefConf

http://chefconf.com

CHEF

DC Configuration Management Group



http://www.meetup.com/DC-Configuration-Management-Group/events/220951825/

CHEF

DevOps DC Meetup

http://www.meetup.com/DevOpsDC/

CHEF

Upcoming Chef Training

- Chef Intermediate Topics – April 28-29
- Chef Fundamentals – April 30-May 1

- Discount code: MEETUP

DevOpsDays DC

- June 11-12
- USPTO, Alexandria

- Now Open
  - Registration
  - CFP
  - Sponsorships

http://www.devopsdays.org/events/2015-washington-dc/

CHEF

Thank you!

- What questions can I answer for you?

- nharvey@chef.io
- @nathenharvey

- http://bit.ly/farmer-nathen (explicit)
- http://bit.ly/farmer-nathen-sfw

CHEF

CHEF™

CODE CAN