Introduction to Chef

Testing your automation code

Nathen Harvey - @nathenharvey

github.com/nathenharvey/surge_introduction_to_chef

WiFi: surge2015 login: surge2015 password: surge2015

Chef Fundamentals by Chef Software, Inc. is licensed under a
Creative Commons Attribution-ShareAlike 4.0 International License.

v1.2.0

CHEF

Prerequisites

- Install Chef DK - downloads.getchef.com/chef-dk
- Have an ssh client
- Have a good text editor (Atom, Sublime, vim, emacs)

- Git & GitHub Account (Optional)

Introductions

CHEF

Nathen Harvey

- Community Director, Chef
- Co-host of the Food Fight Show
- Co-organizer of DevOpsDays DC
- 6th year at Surge
- Occasional farmer – http://bit.ly/farmer-nathen
- Love Eggs – http://eggs.chef.io

- @nathenharvey / nharvey@chef.io

foodfightshow.org
FOOD FIGHT

CHEF

Lab Assistants

- Adam Leff
- Travis Spaulding
- YOU ☺

Hello!

- System Administrator?

Hello!

- System Administrator?
- Developer?

Hello!

- System Administrator?
- Developer?
- DevOp?

Hello!

- System Administrator?
- Developer?
- DevOp?
- Business Person?

Which version control system do your use?

Which version control system do your use?

- `cp foo foo.bak`

Which version control system do your use?

- `cp foo foo.bak`
- `cp foo{,.`date +%Y%m%d%H%M`}`

CHEF

Which version control system do your use?

- `cp foo foo.bak`
- `cp foo{,.`date +%Y%m%d%H%M`}`
- `cp foo{,.`date +%Y%m%d%H%M`-`$USER`}`

CHEF

Are you experienced?

- Experience with Infrastructure as Code or Configuration Management?

Are you experienced?

- Experience with Infrastructure as Code or Configuration Management?
- Experience with Chef?

Just an Introduction

- Today is just an Introduction to Chef
- We'll cover lots of topics but won't go too deep on any of them
- Any discussion that takes us too far off the path will be captured
  - We will return to these topics as time permits

Class Logistics

- Streaming LIVE to YouTube via Google+ Hangout
- Slides in the GitHub repository
- Code will be added to GitHub as we go

- But…we have a social contract…

Agenda

CC BY SA

CHEF

Agenda

- Overview of Chef
- Resources
- Recipes
- Cookbooks
- Ensuring desired state
- Infrastructure State
- Faster Feedback with Testing
- Wrap Up

Breaks!

- We'll take breaks when we need them
- We will break for lunch, but lunch is not provided

Prerequisites

- Install Chef DK - downloads.chef.io/chef-dk/
- Have an ssh client
- Have a good text editor (Atom, Sublime, vim, emacs)

- Git & GitHub Account (Optional)

Slides, Code, Questions, etc.

- github.com/nathenharvey/surge_introduction_to_chef

- Slides are available now (subject to change)
- Code will be added as we go
- Submit PRs for any questions or topics that you'd like to see covered

Building your policy

Resources and Recipes

v1.2.0

CC BY SA

CHEF

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
```

Resources

- Piece of the system and its desired state

- http://docs.chef.io/chef/resources.html

What's up with the card?

- http://bit.ly/chef-surge2015

- Login: chef
- Password: [REDACTED]

Login to your lab machine

```
$ ssh chef@54.164.75.30
```

```
The authenticity of host '54.165.227.226
(54.165.227.226)' can't be established.
RSA key fingerprint is c1:ec:ab:66:fb:22:4a:
8f:c2:c5:9b:26:77:f3:dd:b3.
Are you sure you want to continue connecting
(yes/no)? yes
Warning: Permanently added
'54.165.227.226' (RSA) to the list of known
hosts.
chef@54.165.227.226's password:
```

Welcome to your workstation

- ChefDK version 0.7.0 is installed
  - `chef --version`
- Chef user has passwordless sudo access
  - `sudo cat /etc/shadow`

Lab – What does the cow say?

- **Problem**: Our workstation is boring, I love cows
- **Success Criteria**: Carry on a conversation with Henry

CHEF

chef-apply

- chef-apply is an executable program that allows you to work with resources
- Is included as part of the ChefDK
- A great way to explore resources
- NOT how you'll eventually use Chef in production

Install cowsay

```
$ sudo chef-apply -e "package 'cowsay'"
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * yum_package[cowsay] action install
    - install version 3.03-8.2.amzn1 of package cowsay
```

Say hello

```
$ cowsay "hello, surge"
```

```
 _________________
< hello, surge>
 -----------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Resources

- Describe the desired state
- Do not need to tell Chef how to get there

- What happens if you re-run the chef-apply command?

Install cowsay again with chef-apply

```
$ sudo chef-apply -e "package 'cowsay'"
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * yum_package[cowsay] action install (up to date)
```

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?

CHEF

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

Test Is vim installed?

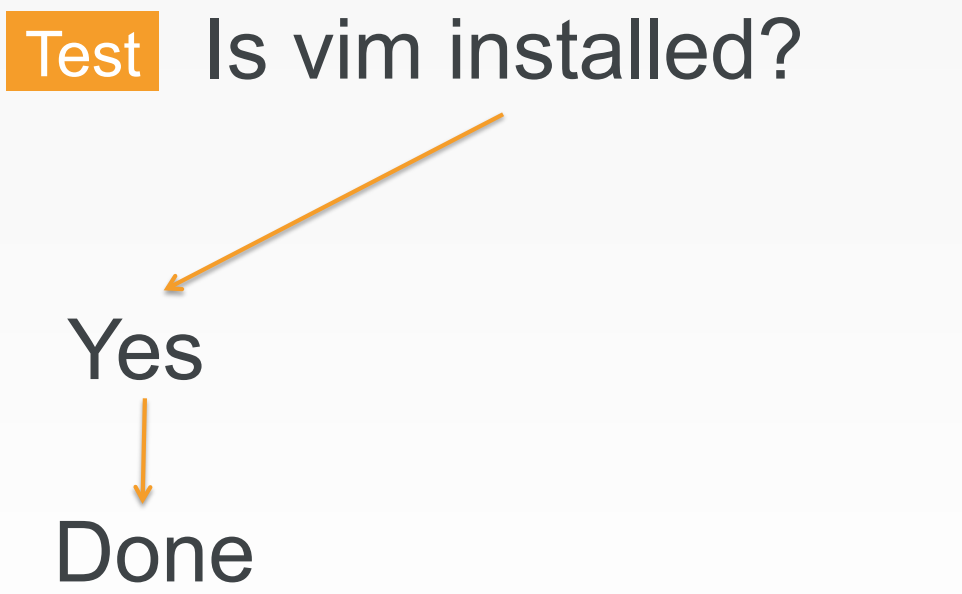
Yes

CHEF

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

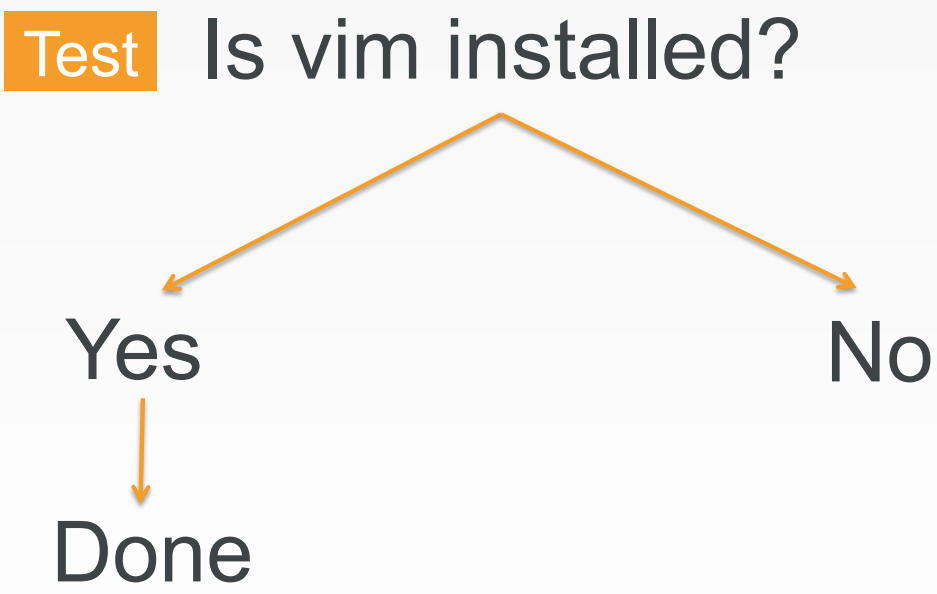


CHEF

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

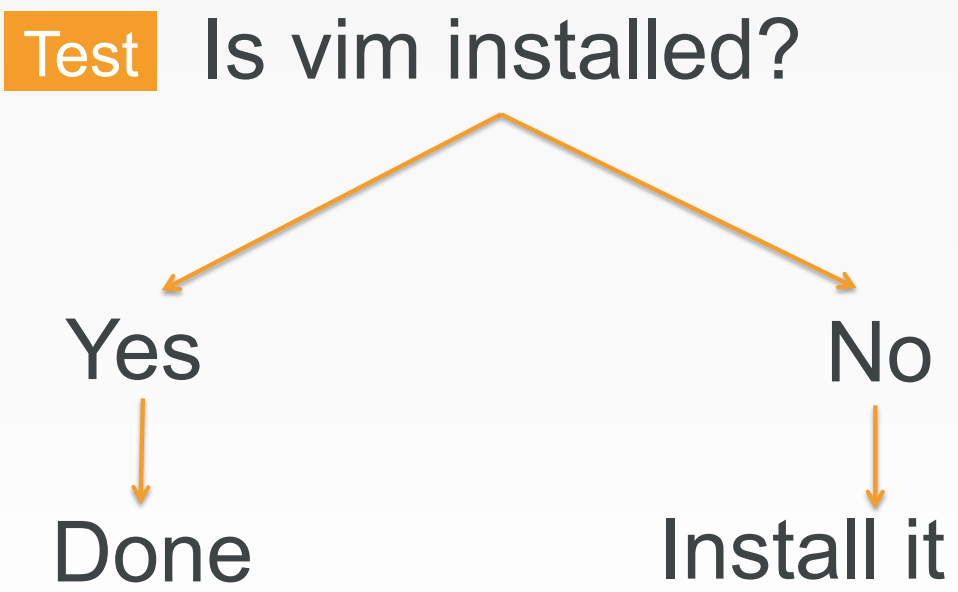


CHEF

Test and Repair

Resources follow a **test** and repair model

```
package "vim"
```

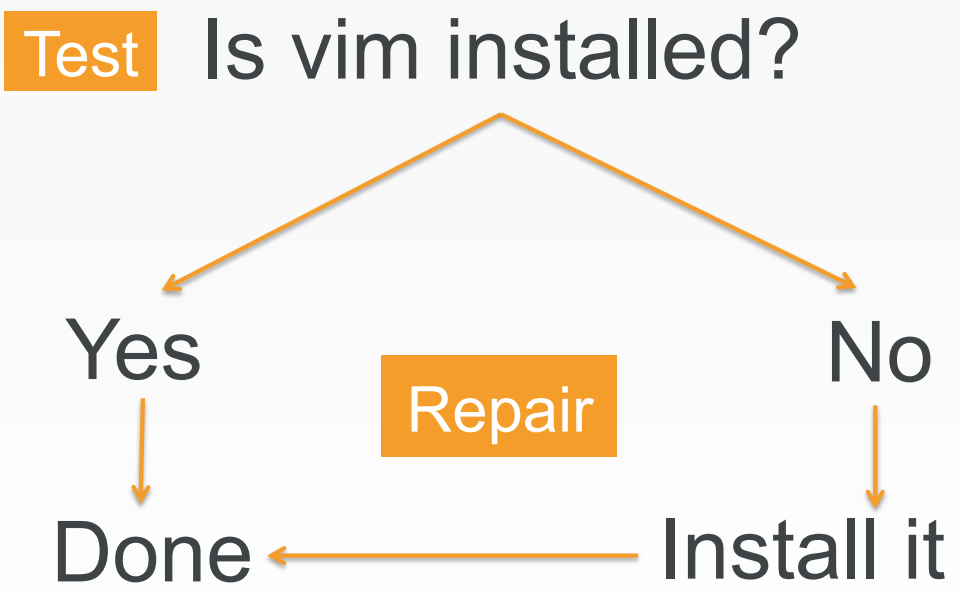


CHEF

Test and Repair

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

```
package "vim"
```



CHEF

Resources – Test and Repair

- Resources follow a test and repair model

- Resource currently in the desired state? (test)
  - Yes – Do nothing
  - No – Bring the resource into the desired state (repair)

Resources

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

Recipes

- Policy is defined as a collection of **resources** in **recipes**. There are lots of abstractions on top of this but **resources** are the basic building blocks.

Recipes

Collection of resources

CC BY SA

CHEF

Lab 2 – Serve our homepage

- **Problem**: We need a web server configured to serve up our home page
- **Success Criteria**: We see our home page in a web browser

What is our desired state?

- We see our home page in a web browser

What is required to meet this state?

- What resources will we need?

Recipe

- Collection of resources
- Recipes can:
  - Install and configure software components
  - Manage files
  - Deploy applications
  - Execute other recipes
  - And more…

Create learnchef/lab2 directory

```
$ mkdir -p ~/learnchef/lab2
```

Move to the learnchef/lab2 directory

```
$ cd ~/learnchef/lab2
```

Is there a web server?

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

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

Write a recipe for apache

OPEN IN EDITOR: ~/learnchef/lab2/apache.rb

```
package 'apache2'
```

SAVE FILE!

CHEF

Apply the apache recipe

```
$ sudo chef-apply apache.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * apt_package[apache2] action install
    - install version 2.4.7-1ubuntu4.5 of package apache2
```

chef-apply a recipe

- Now we are using chef-apply on a recipe, or a collection of resources

- Success?

Is it working?

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2014-03-19
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
```

Write a recipe for apache

OPEN IN EDITOR: ~/learnchef/lab2/apache.rb

```
package 'apache2'

service 'apache2' do
  action :start
end
```

SAVE FILE!

CHEF

The service resource

Actions

This resource has the following actions:

Action	Description
`:disable`	Use to disable a service.
`:enable`	Use to enable a service at boot.
`:nothing`	Default. Use to do nothing with a service.
`:reload`	Use to reload the configuration for this service.
`:restart`	Use to restart a service.
`:start`	Use to start a service (and keep it running until stopped or disabled).
`:stop`	Use to stop a service.

http://docs.getchef.com/resource_service.html

Resources

- Type (package, service, etc.)
- Name (“httpd”, “vim”, etc.)
- Actions
  - A description of the desired state of the resource
  - Defined as part of the resource

  - Will the action always be executed?

CHEF

Why did the package resource work?

Actions

This resource has the following actions:

Action	Description
`:install`	Default. Use to install a package. If a version is specified, use to install the specified version of a package.
`:upgrade`	Use to install a package and/or to ensure that a package is the latest version.
`:reconfig`	Use to reconfigure a package. This action requires a response file.
`:remove`	Use to remove a package.
`:purge`	Use to purge a package. This action typically removes the configuration files as well as the package. (Debian platform only; for other platforms, use the `:remove` action.)

http://docs.getchef.com/resource_package.html

CHEF

Why did the package resource work?

Actions

This resource has the following actions:

Action	Description
`:install`	Default. Use to install a package. If a version is specified, use to install the specified version of a package.
`:upgrade`	Use to install a package and/or to ensure that a package is the latest version.
`:reconfig`	Use to reconfigure a package. This action requires a response file.
`:remove`	Use to remove a package.
`:purge`	Use to purge a package. This action typically removes the configuration files as well as the package. (Debian platform only; for other platforms, use the `:remove` action.)

http://docs.getchef.com/resource_package.html

CHEF

Apply the apache recipe

```
$ sudo chef-apply apache.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * apt_package[apache2] action install (up to date)
  * service[apache2] action start (up to date)
```

Is it working?

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

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2014-03-19
    See: https://launchpad.net/bugs/1288690
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
```

Not there yet

- The web server is up and running
- The default home page is being displayed, not our home page

What resource should we use?

Resources: About Resources | Common Functionality — **Resources:** apt_package | bash | batch | breakpoint | chef_gem | chef_handler | cookbook_file | cron | deploy | directory | dpkg_package | dsc_script | easy_install_package | env | erl_call | execute | file | gem_package | git | group | http_request | ifconfig | link | log | mdadm | mount | ohai | package | powershell_script | registry_key | remote_directory | remote_file | route | rpm_package | ruby_block | script | service | subversion | template | user | yum_package | windows_package — **Single Page:** Resources and Providers

http://docs.getchef.com/

What resource should we use?

Resources: About Resources | Common Functionality — **Resources:** apt_package | bash | batch | breakpoint | chef_gem | chef_handler | cookbook_file | cron | deploy | directory | dpkg_package | dsc_script | easy_install_package | env | erl_call | execute | file | gem_package | git | group | http_request | ifconfig | link | log | mdadm | mount | ohai | package | powershell_script | registry_key | remote_directory | remote_file | route | rpm_package | ruby_block | script | service | subversion | template | user | yum_package | windows_package — **Single Page:** Resources and Providers

http://docs.getchef.com/

CHEF

What resource should we use?

Resources: About Resources | Common Functionality — **Resources:** apt_package | bash | batch | breakpoint | chef_gem | chef_handler | cookbook_file | cron | deploy | directory | dpkg_package | dsc_script | easy_install_package | env | erl_call | execute | file | gem_package | git | group | http_request | ifconfig | link | log | mdadm | mount | ohai | package | powershell_script | registry_key | remote_directory | remote_file | route | rpm_package | ruby_block | script | service | subversion | template | user | yum_package | windows_package — **Single Page:** Resources and Providers

http://docs.getchef.com/

CHEF

What resource should we use?

Resources: About Resources | Common Functionality — **Resources:** apt_package | bash | batch | breakpoint | chef_gem | chef_handler | cookbook_file | cron | deploy | directory | dpkg_package | dsc_script | easy_install_package | env | erl_call | execute | file | gem_package | git | group | http_request | ifconfig | link | log | mdadm | mount | ohai | package | powershell_script | registry_key | remote_directory | remote_file | route | rpm_package | ruby_block | script | service | subversion | template | user | yum_package | windows_package — **Single Page:** Resources and Providers

http://docs.getchef.com/

CHEF

What resource should we use?

Resources: About Resources | Common Functionality — **Resources:** apt_package | bash | batch | breakpoint | chef_gem | chef_handler | cookbook_file | cron | deploy | directory | dpkg_package | dsc_script | easy_install_package | env | erl_call | execute | file | gem_package | git | group | http_request | ifconfig | link | log | mdadm | mount | ohai | package | powershell_script | registry_key | remote_directory | remote_file | route | rpm_package | ruby_block | script | service | subversion | template | user | yum_package | windows_package — **Single Page:** Resources and Providers

http://docs.getchef.com/

CHEF

File Resource

- Use the file resource to manage files that are present on a node, including setting or updating the contents of those files.

File Resource - Actions

Actions

This resource has the following actions:

Action	Description
`:create`	Default. Use to create a file. If a file already exists (but does not match), use to update that file to match.
`:create_if_missing`	Use to create a file only if the file does not exist. (When the file exists, nothing happens.)
`:delete`	Use to delete a file.
`:touch`	Use to touch a file. This updates the access (atime) and file modification (mtime) times for a file.

http://docs.getchef.com/resource_file.html#actions

File Resource – content attribute

`content`	A string that is written to the file. The contents of this attribute will replace any previous content when this attribute has something other than the default value. The default behavior will not modify content.

http://docs.getchef.com/resource_file.html#actions

Write a recipe for apache

OPEN IN EDITOR: ~/learnchef/apache.rb

```
Package 'apache2'

service 'apache2' do
  action :start
end

file '/var/www/html/index.html' do
  content "<h1>hello world</h1>\n"
end
```

SAVE FILE!

CHEF

Re-apply the recipe

```
$ sudo chef-apply apache.rb
```

```
 -      </div>
    -     </div>
    -     <div class="validator">
    -     <p>
    -       <a href="http://validator.w3.org/check?uri=referer"><img
src="http://www.w3.org/Icons/valid-xhtml10" alt="Valid XHTML 1.0
Transitional" height="31" width="88" /></a>
    -     </p>
    -     </div>
    -   </body>
    -</html>
    -
    +<h1>hello world</h1>
```

CHEF

Is it working?

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

```
<h1>hello world</h1>
```

Scenario testing

- System administrator logs in to the server and changes the file manually.

Change the home page

OPEN IN EDITOR: /var/www/html/index.html

```
<h1>Hello, You!</h1>
```

SAVE FILE!

CHEF

Is it working?

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

```
<h1>Hello, You!</h1>
```

Re-apply the recipe

```
$ sudo chef-apply apache.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * apt_package[apache2] action install (up to date)
  * service[apache2] action start (up to date)
  * file[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 681590
to cdf94d
    --- /var/www/html/index.html        2015-09-23
10:19:30.541649829 +0000
    +++ /var/www/html/.index.html20150923-12116-1kyd5rv
2015-09-23 10:19:36.076415999 +0000
    @@ -1,2 +1,2 @@
    -<h1>hello you</h1>
    +<h1>hello world</h1>
```

CHEF

Key Learning

- Do not manually change files that are being managed by Chef
- Changes always start in the code repository

Separating data from policy

- Storing the home page content directly in the recipe feels wrong
- We can manage that content separately using a different resource
  - cookbook_file
  - remote_file
  - template

CHEF

Template resource

- An ERB template that is used to generate files based on the variables and logic contained within the template.

Problem!

- chef-apply is not capable of loading templates
- Templates do not belong in the recipe file
- It’s time to graduate to chef-client and a cookbook

Cookbooks

Recipes and supporting files

CC BY SA

CHEF

Cookbooks

- A cookbook is like a “package” for Chef configuration and policy
- Every cookbook will have a name & version
- A cookbook can contain
  - Recipes
  - Files & Templates
  - Data attributes
  - Libraries
  - And more…

Cookbooks

- Typically map 1:1 to a piece of software or functionality.
- For example
  - mysql
  - users
  - sercurity
  - apache
  - ntp

Infrastructure as Code

- Package code into distributable units
- Version the code and the packages
- Separate data and policy
  - The web server should have a home page
  - The home page should have specific content

Lab 3 – Manage the homepage content separately

- **Problem**: Storing the home page content in the recipe is unmanageable
- **Success Criteria**: The home page is stored in it's own file

chef

- chef is an executable command line tool for
  - generating cookbooks, recipes, and other things that make up your Chef code
  - ensuring RubyGems are downloaded properly for your development environment
  - verifying that all the components are installed and configured correctly
- Included with ChefDK

Lab 3 – Manage the homepage content separately

- **Problem**: Storing the home page content in the recipe is unmanageable
- **Success Criteria**: The home page is stored in it's own file

Lab 3 – Manage the homepage content separately

1. Create a chef-repo
2. Create an apache cookbook

What can `chef` generate?

```
$ chef generate --help
```

```
Usage: chef generate GENERATOR [options]

Available generators:
  app          Generate an application repo
  cookbook     Generate a single cookbook
  recipe       Generate a new recipe
  attribute    Generate an attributes file
  template     Generate a file template
  file         Generate a cookbook file
  lwrp         Generate a lightweight resource/provider
  repo         Generate a Chef policy repository
```

How do we generate a repo?

```
$ chef generate repo --help
```

```
Usage: chef generate repo NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright holder - defaults to 'The Authors'
    -m, --email EMAIL                Email address of the author - defaults to 'you@example.com'
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 - defaults to all_rights
    -p, --policy-only                Create a repository for policy only, not cookbooks
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the code_generator cookbook
        --generator-cookbook
```

Lab 3 – Manage the homepage content separately

1. Create a chef-repo
2. Create an apache cookbook

Go home!

```
$ cd ~
```

Create a chef-repo

```
$ chef generate repo chef-repo -p
```

```
Compiling Cookbooks...
Recipe: code_generator::repo
  * directory[/home/chef/chef-repo] action create
    - create new directory /home/chef/chef-repo
    - restore selinux security context
  * template[/home/chef/chef-repo/LICENSE] action create
    - create new file /home/chef/chef-repo/LICENSE
    - update content in file /home/chef/chef-repo/LICENSE from none to dbc1af
    (diff output suppressed by config)
    - restore selinux security context
  * cookbook_file[/home/chef/chef-repo/README.md] action create
    - create new file /home/chef/chef-repo/README.md
    - update content in file /home/chef/chef-repo/README.md from none to 767ead
    (diff output suppressed by config)
    - restore selinux security context
  * cookbook_file[/home/chef/chef-repo/Rakefile] action create
```

Commit this chef-repo to git

```
$ cd chef-repo
```

CHEF

Initialize repository

```
$ git init
```

```
Initialized empty Git repository
in /home/chef/chef-repo/.git/
```

Commit this chef-repo to git

```
$ git add .
```

Commit this chef-repo to git

```
$ git commit -m "Initial chef-repo"
```

```
[master (root-commit) 6774a70] Initial chef repo
 11 files changed, 388 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 LICENSE
 create mode 100644 README.md
 create mode 100644 Rakefile
 create mode 100644 certificates/README.md
 create mode 100644 chefignore
 create mode 100644 config/rake.rb
 create mode 100644 cookbooks/README.md
 create mode 100644 data_bags/README.md
 create mode 100644 environments/README.md
 create mode 100644 roles/README.md
```

Lab 3 – Manage the homepage content separately

✓ Create a chef-repo

2. Create an apache cookbook

Create an apache cookbook

```
$ chef generate cookbook --help
```

```
Usage: chef generate cookbook NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright holder - defaults to 'The Authors'
    -m, --email EMAIL                Email address of the author - defaults to 'you@example.com'
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 - defaults to all_rights
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the code_generator cookbook
        --generator-cookbook
```

Create an apache cookbook

```
$ cd cookbooks
```

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 94785f
    (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 new file /home/chef/chef-repo/cookbooks/apache/chefignore
    - update content in file /home/chef/chef-repo/cookbooks/apache/chefignore from none to 9727b1
    (diff output suppressed by config)
    - restore selinux security context
  * cookbook_file[/home/chef/chef-repo/cookbooks/apache/Berksfile] action create_if_missing
```

Create new git repo for this cookbook

```
$ cd apache
```

Create new git repo for this cookbook

```
$ git init
```

```
Initialized empty Git repository
in /home/chef/chef-repo/cookbooks/
apache/.git/
```

Commit the initial cookbook

```
$ git add .
```

Commit the initial cookbook

```
$ git commit -m "initial apache recipe, does nothing"
```

```
[master (root-commit) af2b629] initial apache
recipe, does nothing
 6 files changed, 144 insertions(+), 0 deletions(-)
 create mode 100644 .kitchen.yml
 create mode 100644 Berksfile
 create mode 100644 README.md
 create mode 100644 chefignore
 create mode 100644 metadata.rb
 create mode 100644 recipes/default.rb
```

Write the default recipe

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

```
#
# Cookbook Name:: apache
# Recipe:: default
#
# Copyright (c) 2014 The Authors, All Rights Reserved.
package 'apache2'

service 'httpd' do
  action :start
end

file '/var/www/html/index.html' do
  content "<h1>hello world</h1>\n"
end
```

SAVE FILE!

CHEF

Wait a tick!

- We wanted to move the contents of the home page to another file
- Let's change the `file` resource to a `template` resource

Replace the `file` resource

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

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

SAVE FILE!

CHEF

Template resource

- An ERB template that is used to generate files based on the variables and logic contained within the template.

Create a new template file

- Template files are ERB files
- The `chef` command includes a template generator

Let’s create a template!

```
$ chef generate template --help
```

```
Usage: chef generate template [path/to/cookbook] NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright holder - defaults to 'The Authors'
    -m, --email EMAIL                Email address of the author - defaults to 'you@example.com'
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 - defaults to all_rights
    -s, --source SOURCE_FILE         Copy content from SOURCE_FILE
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the code_generator cookbook
        --generator-cookbook
```

Go to the chef-repo

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

CHEF

Let’s create a template!

```
$ chef generate template cookbooks/apache/ index.html -s /var/www/html/index.html
```

```
Compiling Cookbooks...
Recipe: code_generator::template
  * directory[cookbooks/apache/templates/default] action create
    - create new directory cookbooks/apache/templates/default
    - restore selinux security context
  * file[cookbooks/apache/templates/default/index.html.erb]
action create
    - create new file cookbooks/apache/templates/default/
index.html.erb
    - update content in file cookbooks/apache/templates/default/
index.html.erb from none to cdf94d
    (diff output suppressed by config)
    - restore selinux security context
```

Check the template

```
$ cat cookbooks/apache/templates/default/index.html.erb
```

```
<h1>hello world</h1>
```

Lab 3 – Manage the homepage content separately

- ✓ Create a chef-repo
- ✓ Create an apache cookbook

Verify the changes

```
$ sudo chef-client -z -r "recipe[apache]"
```

```
[2015-09-23T10:24:16+00:00] WARN: No config file found or specified on command line, using command line options.
Starting Chef Client, version 12.4.1
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * apt_package[apache2] action install (up to date)
  * service[apache2] action start (up to date)
  * template[/var/www/html/index.html] action create (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/3 resources updated in 1.403112927 seconds
```

Verify the changes

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

```
<h1>hello world</h1>
```

Ensuring desired state

chef-client and run lists

CC BY SA

CHEF

chef-client

- chef-client is an executable
  - performs all actions required to bring the node into the desired state
  - typically run on a regular basis
    - daemon
    - cron
    - Windows service
- Included with ChefDK

chef-client modes

- In conjunction with a Chef Server
- Local mode (no Chef Server)

chef-client privileges

- Usually run with elevated privileges
  - root
  - sudo
  - Administrator
- Can run as a normal user

Desired State

- Policies describe the desired state
- Each node can follow a sub-set of policies
- The policies for each node are stored in a run list

Run List

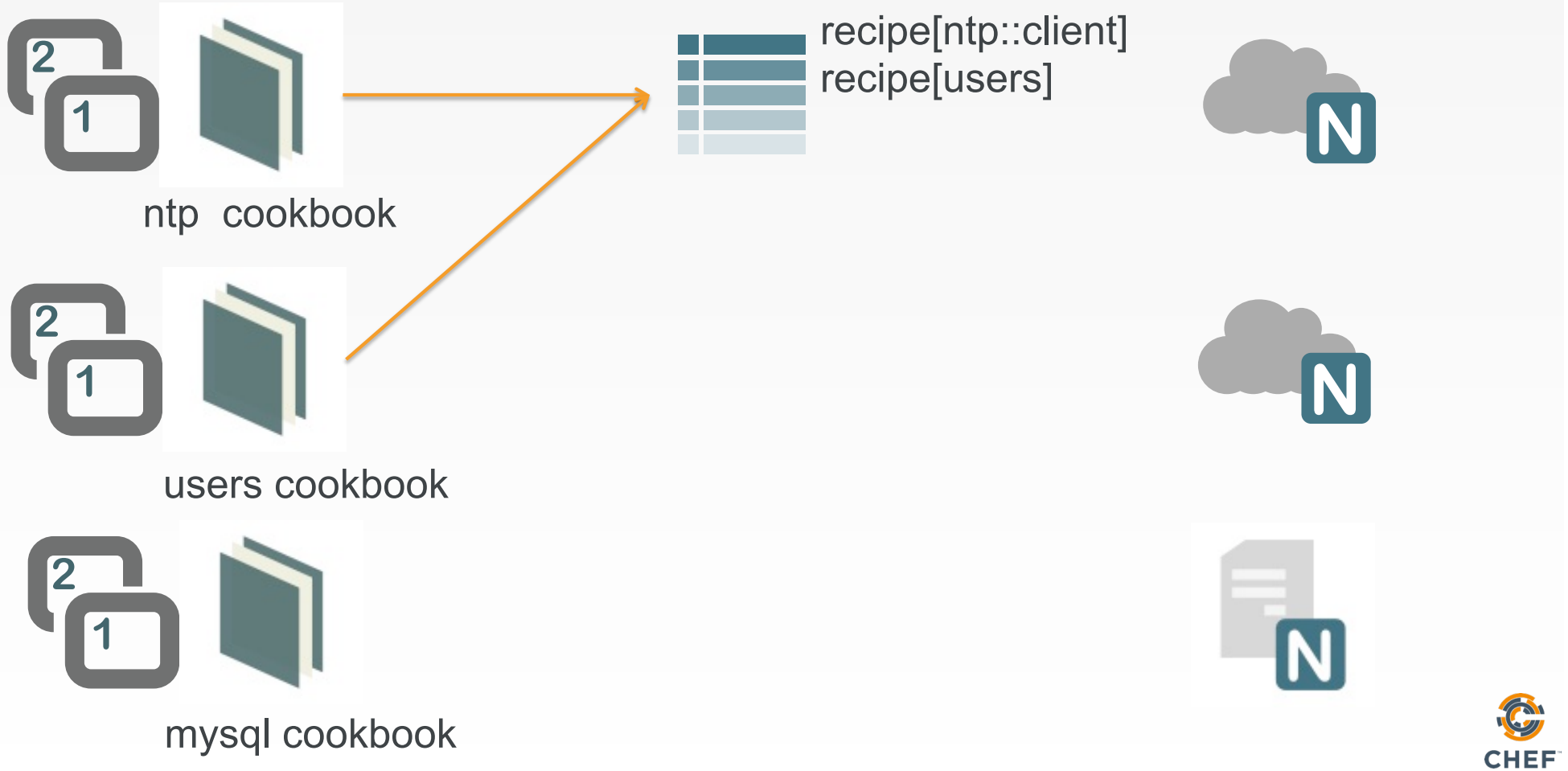
ntp cookbook

users cookbook

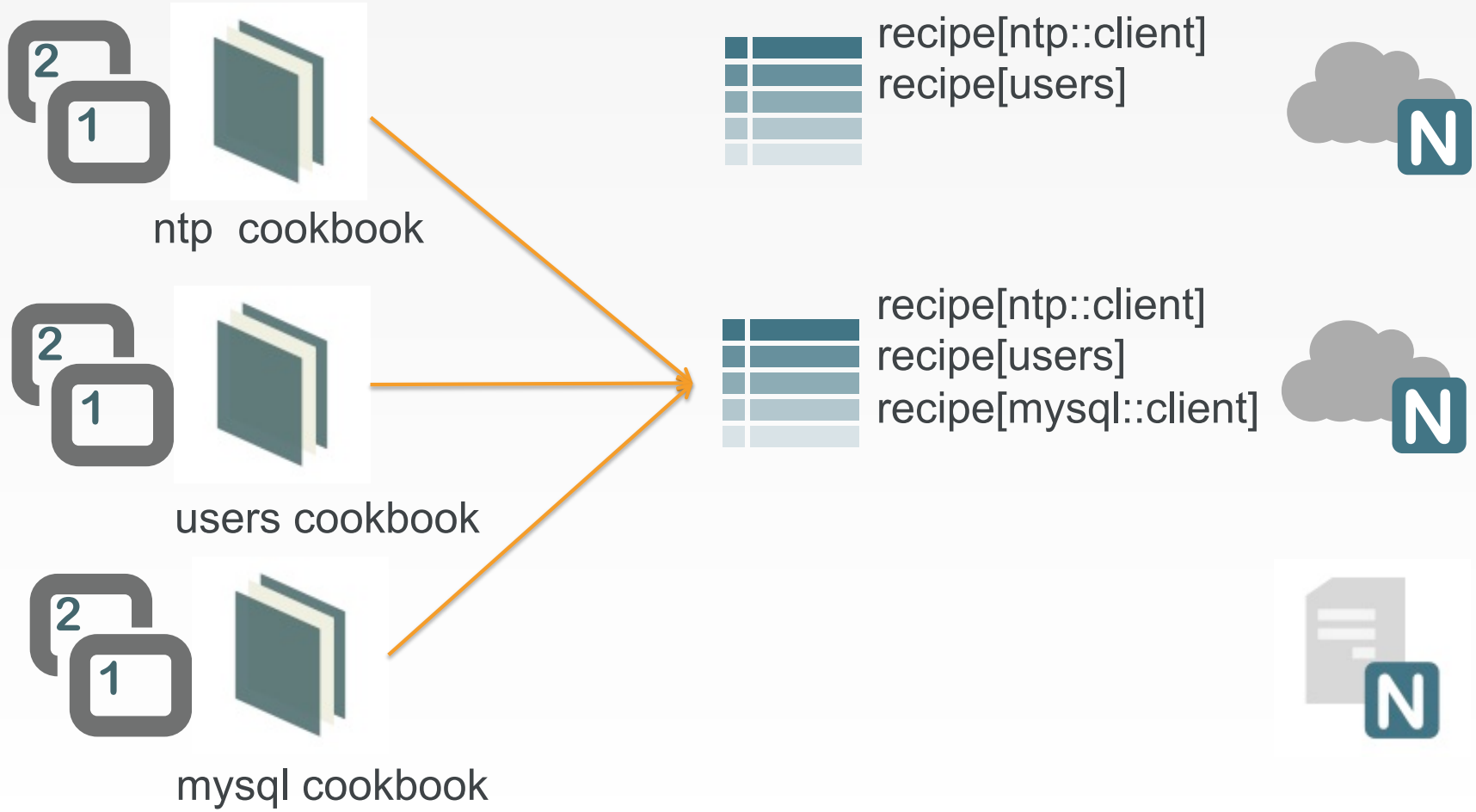
mysql cookbook

CHEF

Run List

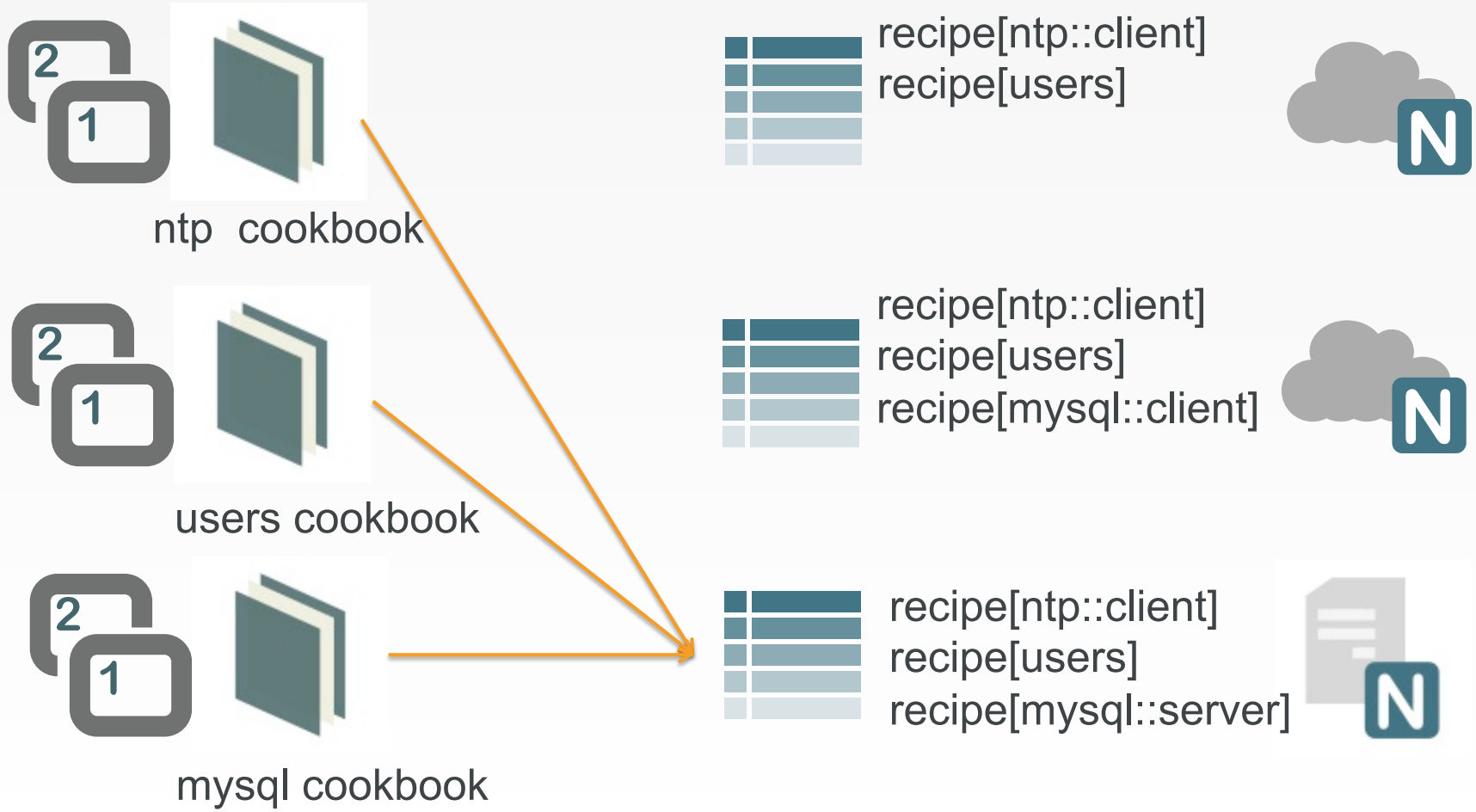

recipe[ntp::client]
recipe[users]
ntp cookbook
users cookbook
mysql cookbook

Run List



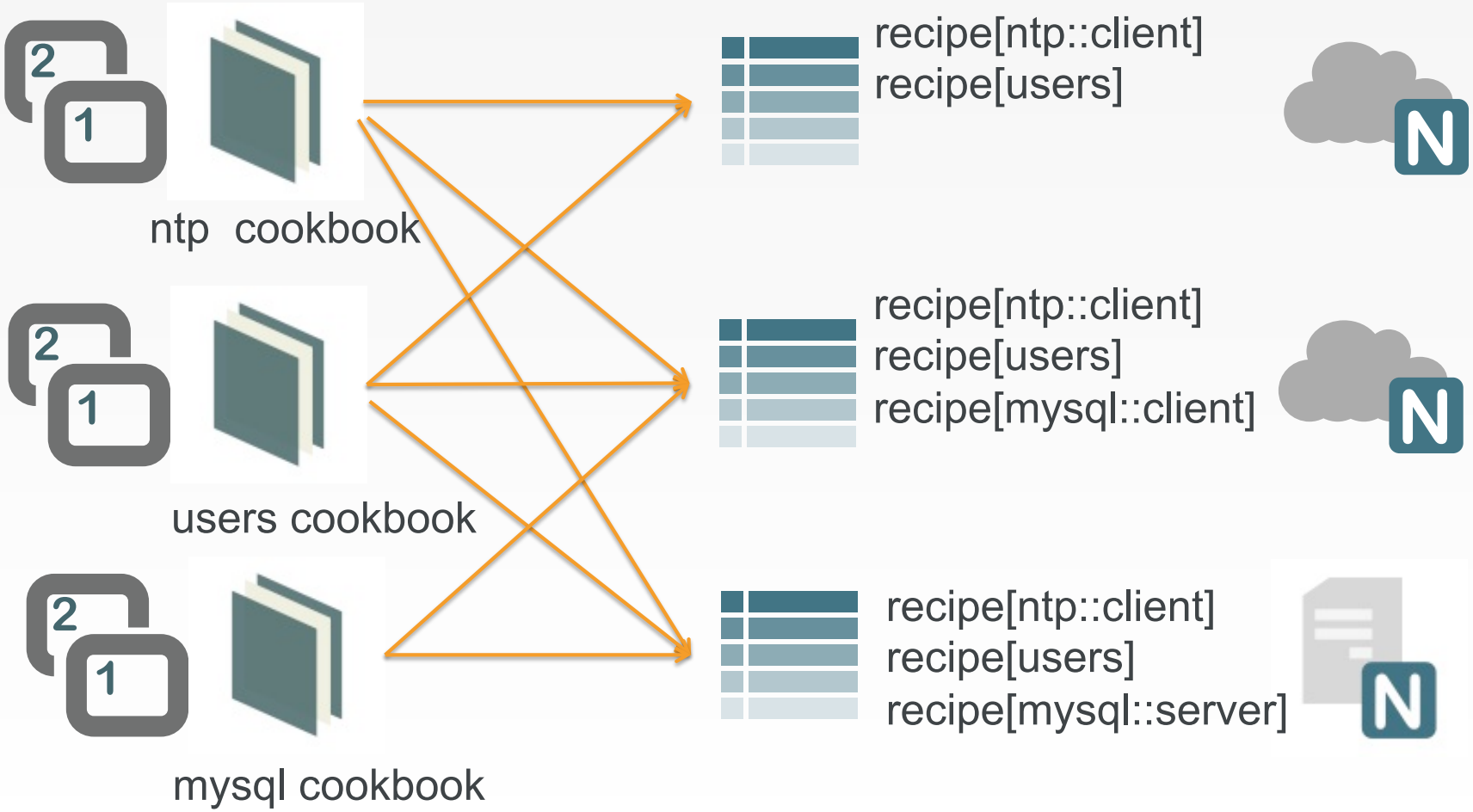
CHEF

Run List



CHEF

Run List



CHEF

Lab 4 – Verify the cookbook

- **Problem**: We need to test our new apache cookbook
- **Success Criteria**: The default recipe applies cleanly and our home page is visible

Run chef-client with a run list

```
$ sudo chef-client -z -r "recipe[apache]"
```

```
[2015-09-23T10:24:44+00:00] WARN: No config file found or specified on command line, using command line options.
Starting Chef Client, version 12.4.1
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * apt_package[apache2] action install (up to date)
  * service[apache2] action start (up to date)
  * template[/var/www/html/index.html] action create (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/3 resources updated in 1.423703943 seconds
```

Verify the home page

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

```
<h1>hello world</h1>
```

Infrastructure State

The node object

CHEF

Node Object

- A node is a physical, virtual, or cloud machine that is managed by the chef-client

- Node object is a data structure representing the state of a node

https://docs.getchef.com/chef_overview_nodes.html

CHEF

Node state

- chef-client saves the current state of the node
- Local mode – in the nodes directory
- Normal mode – on the Chef server

New directory: nodes

```
$ ls ~/chef-repo/nodes
```

```
ip-172-31-32-114.ec2.internal.json
```

Explore the node's .json file

```
$ less nodes/*.json
```


```
{
  "name": "ip-172-31-32-114.ec2.internal",
  "normal": {
    "tags": [

    ]
  },
  "automatic": {
    "languages": {
      "ruby": {
        "platform": "x86_64-linux",
        "version": "2.1.2",
        "release_date": "2014-05-08",
        "target": "x86_64-unknown-linux-gnu",
        "target_cpu": "x86_64",
        "target_vendor": "unknown",
        "target_os": "linux",
        "host": "x86_64-unknown-linux-gnu",
        "host_cpu": "x86_64",
        "host_os": "linux-gnu",
…
```


ohai

- A system profiler that is used to detect the state of nodes
  - Platform
  - Network
  - Memory
  - And much more
- Included with ChefDK

Run ohai

```
$ ohai | less
```


```
{
  "languages": {
    "ruby": {
      "platform": "x86_64-linux",
      "version": "2.1.2",
      "release_date": "2014-05-08",
      "target": "x86_64-unknown-linux-gnu",
      "target_cpu": "x86_64",
      "target_vendor": "unknown",
      "target_os": "linux",
      "host": "x86_64-unknown-linux-gnu",
      "host_cpu": "x86_64",
      "host_os": "linux-gnu",
      "host_vendor": "unknown",
      "bin_dir": "/opt/chefdk/embedded/bin",
      "ruby_bin": "/opt/chefdk/embedded/bin/ruby",
      "gems_dir": "/opt/chefdk/embedded/lib/ruby/gems/2.1.0",
      "gem_bin": "/opt/chefdk/embedded/bin/gem"
    },
 …
```


CHEF

Use ohai to find the platform

```
$ ohai platform
```

```
[
  "ubuntu"
]
```

Find the platform in the node object

```
$ grep '"platform"' nodes/*.json
```

```
 "platform": "x86_64-linux",
    "platform": "ubuntu",
```

Lab 5 – Read node attributes in our policy

- **Problem**: We want to see some node details on our home page.
- **Success Criteria**: Our home page includes additional information about the node

Display node attributes

OPEN IN EDITOR: cookbooks/apache/templates/default/index.html.erb

```
<h1>hello world</h1>

<p>
  This is a <%= node['platform'] %> <%= node['platform_version'] %> server.
  with <%= node['memory']['total'] %> RAM.
</p>
```

SAVE FILE!

CHEF

Display node attributes

OPEN IN EDITOR: cookbooks/apache/templates/default/index.html.erb

```
<h1>hello world</h1>

<p>
  This is a <%= node["platform"] %> <%= node["platform_version"] %> server.
  with <%= node["memory"]["total"] %> RAM.
</p>
```

SAVE FILE!

CHEF

Display node attributes

OPEN IN EDITOR: cookbooks/apache/templates/default/index.html.erb

```
<h1>hello world</h1>

<p>
  This is a <%= node["platform"] %> <%= node["platform_version"] %> server.
  with <%= node["memory"]["total"] %> RAM.
</p>
```

SAVE FILE!

CHEF

Display node attributes

OPEN IN EDITOR: cookbooks/apache/templates/default/index.html.erb

```
<h1>hello world</h1>

<p>
  This is a <%= node["platform"] %> <%= node["platform_version"] %> server.
  with <%= node["memory"]["total"] %> RAM.
</p>
```

SAVE FILE!

CHEF

Display node attributes

OPEN IN EDITOR: cookbooks/apache/templates/default/index.html.erb

```
<h1>hello world</h1>

<p>
  This is a <%= node["platform"] %> <%= node["platform_version"] %> server.
  with <%= node["memory"]["total"] %> RAM.
</p>
```

SAVE FILE!

CHEF

Verify the change

```
$ sudo chef-client -z -r "recipe[apache]"
```

```
 * service[apache2] action start (up to date)
  * template[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from cdf94d to 51b17b
    --- /var/www/html/index.html        2015-09-23 10:19:36.076415999 +0000
    +++ /tmp/chef-rendered-template20150923-12966-1muvnml       2015-09-23 10:26:04.779566767 +0000
    @@ -1,2 +1,8 @@
     <h1>hello world</h1>
    +
    +<p>
    +  This is a ubuntu 14.04 server.
    +  with 2048484kB RAM.
    +</p>
    +

Running handlers:
Running handlers complete
Chef Client finished, 1/3 resources updated in 1.390230459 seconds
```

Verify the change

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

```
<h1>hello world</h1>


<p>
  This is a ubuntu 14.04 server.
  with 2048484kB RAM.
</p>
```

Custom node attributes

- Ohai provides many attributes for the node object
- You may want to include your own custom attributes

Lab 6 - Create a custom node attribute

- **Problem**: We need a customized greeting on our home page.
- **Success Criteria**: Our home page includes a custom greeting.

Lab 6 - Create a custom node attribute

1. Create the attribute
2. Display the attribute on the home page

chef can generate attributes

```
$ chef generate attribute --help
```

```
Usage: chef generate attribute [path/to/cookbook] NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright holder - defaults to 'The Authors'
    -m, --email EMAIL                Email address of the author - defaults to 'you@example.com'
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 - defaults to all_rights
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the code_generator cookbook
        --generator-cookbook
```

Generate attribute

```
$ chef generate attribute cookbooks/apache/ default
```

```
Compiling Cookbooks...
Recipe: code_generator::attribute
  * directory[cookbooks/apache/attributes] action create
    - create new directory cookbooks/apache/attributes
  * template[cookbooks/apache/attributes/default.rb] action create
    - create new file cookbooks/apache/attributes/default.rb
    - update content in file cookbooks/apache/attributes/default.rb from none to e3b0c4
    (diff output suppressed by config)
```

Create an attribute

OPEN IN EDITOR: cookbooks/apache/attributes/default.rb

```
default['apache']['greeting'] = 'DC'
```

SAVE FILE!

CHEF

Update the home page

OPEN IN EDITOR: cookbooks/apache/templates/default/index.html.erb

```
<h1>hello <%= node['apache']['greeting'] %></h1>

<p>
  This is a <%= node['platform'] %> <%= node['platform_version'] %> server.
  with <%= node['memory']['total'] %> RAM.
</p>
```

SAVE FILE!

CHEF

Let’s look at the node

```
$ knife node list --local
```

```
WARNING: No knife configuration file found
ip-172-31-47-246.ec2.internal
```

Let’s look at the node

```
$ knife node show ip-172-31-47-246.ec2.internal --local
```

```
WARNING: No knife configuration file found
Node Name:   ip-172-31-47-246.ec2.internal
Environment: _default
FQDN:        ip-172-31-47-246.ec2.internal
IP:          54.173.239.129
Run List:    recipe[apache]
Roles:
Recipes:     apache::default
Platform:    ubuntu 14.04
Tags:
```

Let’s look at the node

```
$ knife node show ip-172-31-47-246.ec2.internal --local -a apache
```

```
WARNING: No knife configuration file found
ip-172-31-47-246.ec2.internal:
  apache:
```

Apply the change

```
$ sudo chef-client -z
```

```
 * template[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 51b17b to a74dd1
    --- /var/www/html/index.html        2015-09-23 10:26:04.779566767 +0000
    +++ /tmp/chef-rendered-template20150923-13358-f6f2s3        2015-09-23
10:29:40.691566767 +0000
    @@ -1,4 +1,4 @@
    -<h1>hello world</h1>
    +<h1>hello DC</h1>

     <p>
       This is a ubuntu 14.04 server.

Running handlers:
Running handlers complete
Chef Client finished, 1/3 resources updated in 1.420250472 seconds
```

Verify the change

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

```
<h1>hello DC</h1>


<p>
  This is a ubuntu 14.04 server.
  with 2048484kB RAM.
</p>
```

Let’s look at the node

```
$ knife node show ip-172-31-47-246.ec2.internal --local -a apache
```

```
WARNING: No knife configuration file found
ip-172-31-47-246.ec2.internal:
  apache:
    greeting: DC
```

Faster Feedback

Automate your testing

v1.2.0

CC BY SA

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 following our style guide?

Chef client success status

- Requirements to verify chef-client success:
  - A place to store the cookbook artifact

Chef client success status

- Requirements to verify chef-client success:
  - A place to store the cookbook artifact
  - A chef-client with access to the cookbook

Chef client success status

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

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

CHEF

Test Matrix

- Two operating systems

ubuntu-12.04
centos-6.4

CHEF

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: `cookbooks/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]
```

Lab 7 – Verify chef-client success

- **Problem**: Testing our cookbook takes too long
- **Success Criteria**: Our cookbook runs in test kitchen

Move to the apache cookbook directory

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

Update .kitchen.yml

OPEN IN EDITOR: .kitchen.yml

```
---
driver:
  name: docker
  use_sudo: false

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-14.04

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

SAVE FILE!

CHEF

List the Test Kitchens

```
$ kitchen list
```

```
Instance             Driver   Provisioner  Verifier  Transport  Last Action
default-ubuntu-1404  Docker   ChefZero     Busser    Ssh        <Not Created>
```

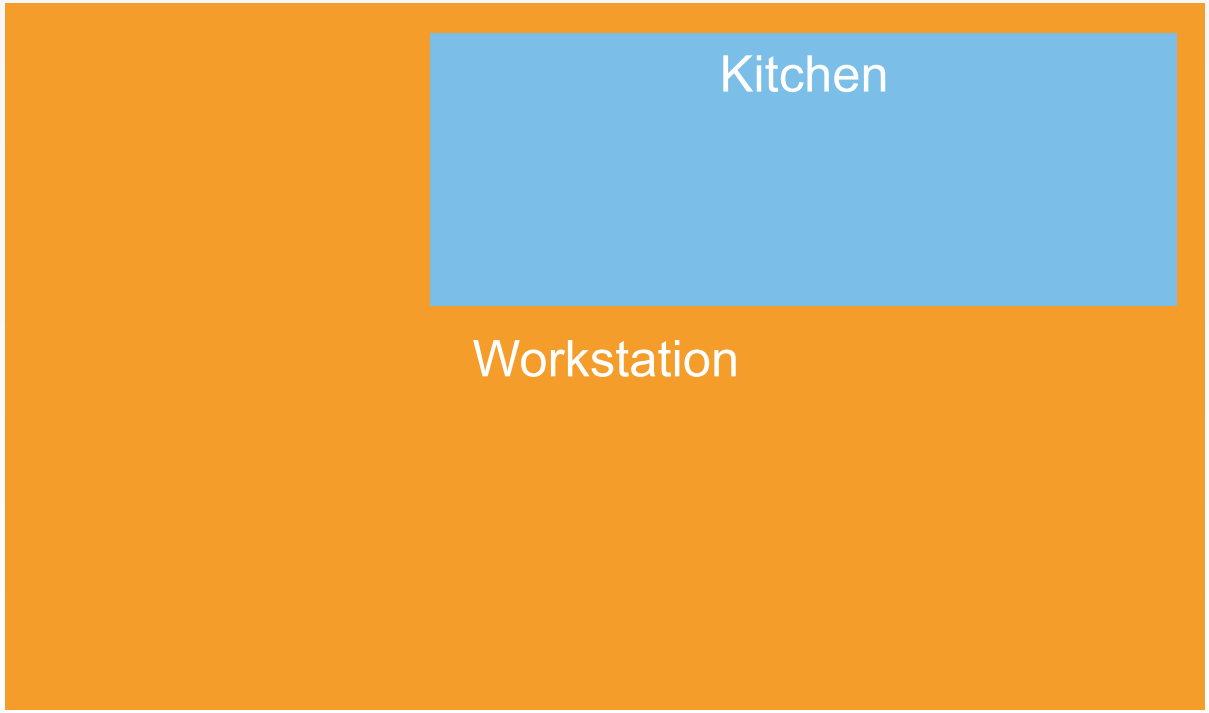
CHEF

Create the kitchen

```
$ kitchen create
```

```
-----> Starting Kitchen (v1.4.2)
-----> Creating <default-ubuntu-1404>...
       Sending build context to Docker daemon 103.4 kB
       Sending build context to Docker daemon
       Step 0 : FROM ubuntu:14.04
        ---> 91e54dfb1179
       Step 1 : RUN dpkg-divert --local --rename --add /sbin/initctl
        ---> Running in 896cb2b32162
       Leaving 'local diversion of /sbin/initctl to /sbin/initctl.distrib'
        ---> 7998c2114ff3
       Removing intermediate container 896cb2b32162
       Step 2 : RUN ln -sf /bin/true /sbin/initctl
        ---> Running in cc133734bf7d
```

Kitchen created



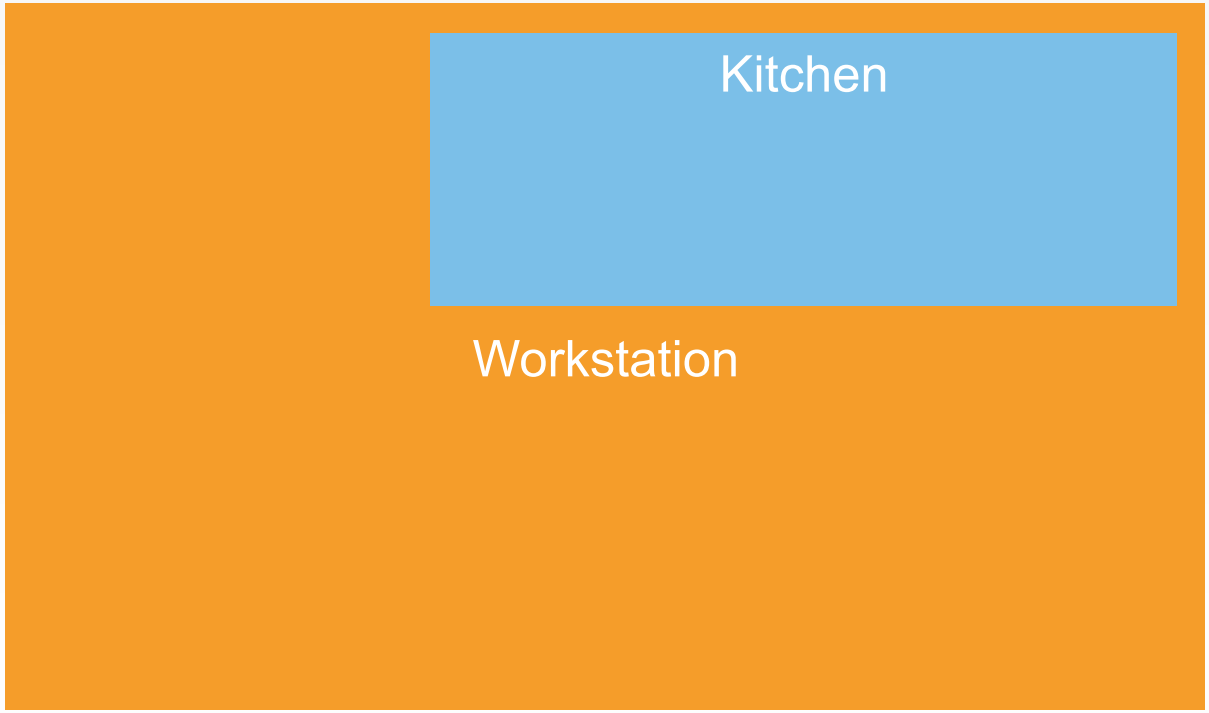
CHEF

Login to the kitchen

```
$ kitchen login
```

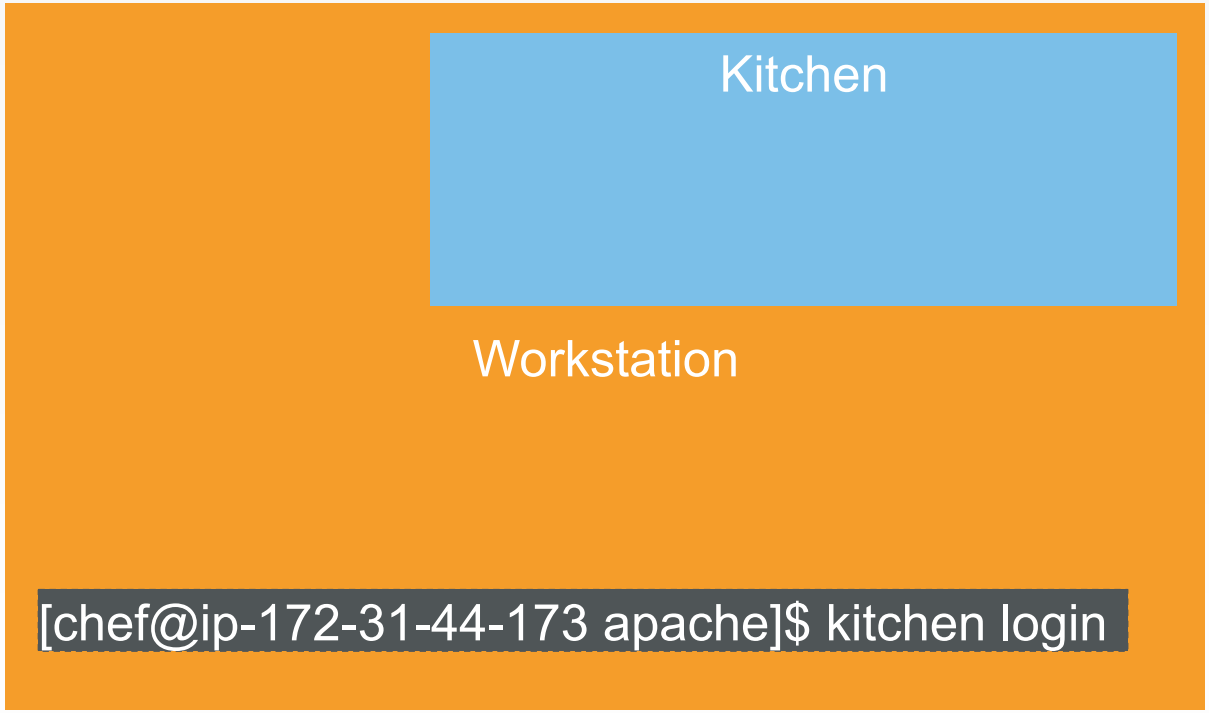
```
$$$$$$ Running legacy login for 'Docker' Driver
Last login: Wed Sep 23 09:48:21 2015 from 172.17.42.1
```

Kitchen login



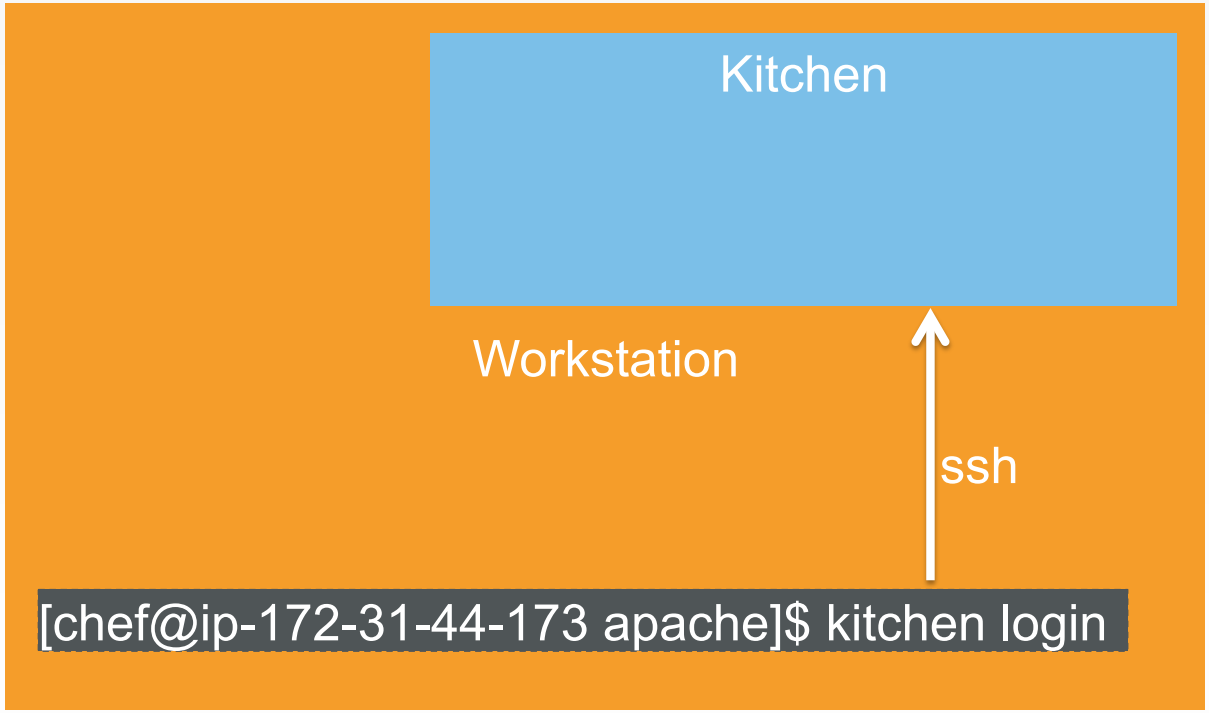
CHEF

Kitchen login



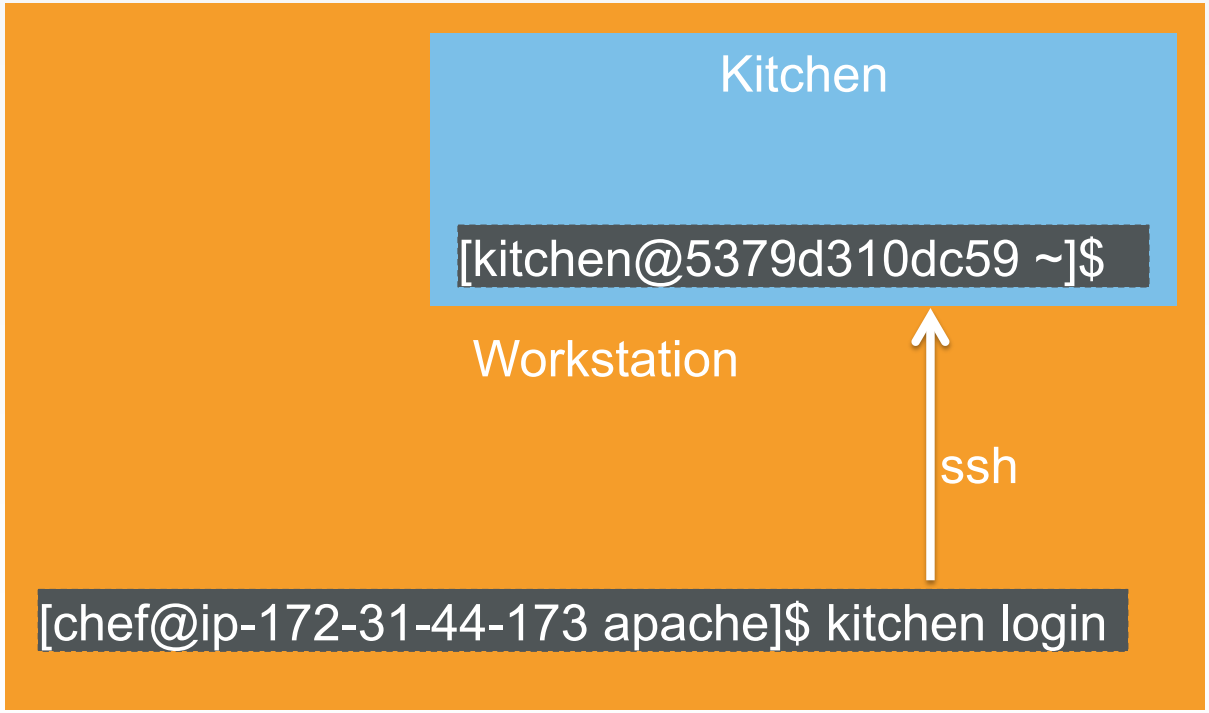
CHEF

Kitchen login



CHEF

Kitchen login



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

Lab – Apply our policy

- **Problem:** We have not applied our policy to the test environment.
- **Success Criteria:** The default apache recipe will be applied in the test environment

Leave the kitchen

```
$ exit
```

```
logout
Connection to localhost closed.
```

Go to the right place

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

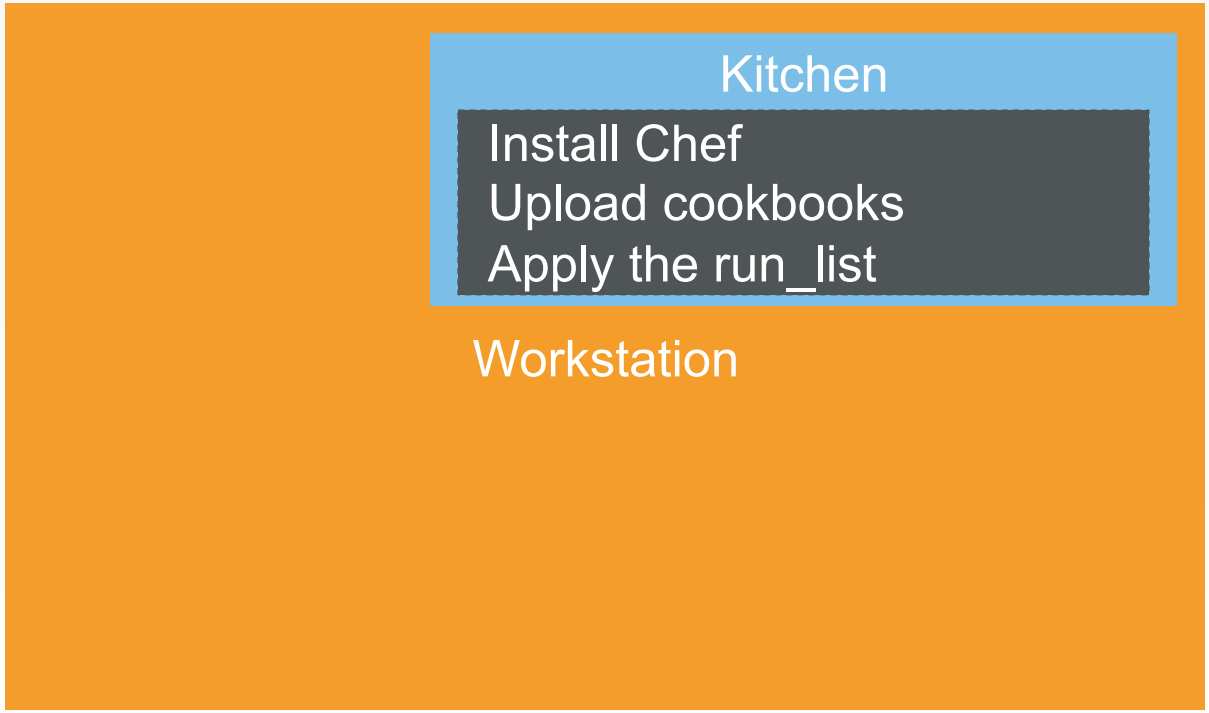
CHEF

Apply our policy

```
$ kitchen converge
```

```
-----> Starting Kitchen (v1.4.2)
-----> Converging <default-centos-71>...
$$$$$$ Running legacy converge for 'Docker' Driver
       Preparing files for transfer
       Preparing dna.json
       Resolving cookbook dependencies with Berkshelf 3.2.4...
       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...
       Download complete.
       Downloading Chef  for el...
       downloading https://www.chef.io/chef/metadata?v=&prerelease=false&nightlies=false&p=el&pv=7&m=x86_64
```

Kitchen converge



CHEF

Questions to ask when 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?

Lab 8 – Verify the node

- **Problem**: Successful converge does not necessarily mean the node is in the desired state.
- **Success Criteria**: Verify our home page is being served.

Manually Inspect with kitchen login

```
$ kitchen login
```

```
$$$$$$ Running legacy login for 'Docker' Driver
Last login: Wed Sep 23 10:44:37 2015 from 172.17.42.1
```

Manually Inspect with kitchen login

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

```
<h1>hello DC</h1>


<p>
  This is a ubuntu 14.04 server.
  with 2048484kB RAM.
</p>
```

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

Leave the Kitchen

```
$ exit
```

```
logout
Connection to localhost closed.
```

Move to the proper directory

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

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
```

Awesome Expectations

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

```
require 'spec_helper'

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

SAVE FILE!

CHEF

Run the serverspec test

```
$ kitchen verify
```

```
-----> Running serverspec test suite
       /opt/chef/embedded/bin/ruby -I/tmp/busser/suites/serverspec -I/tmp/
busser/gems/gems/rspec-support-3.1.2/lib:/tmp/busser/gems/gems/rspec-
core-3.1.7/lib /opt/chef/embedded/bin/rspec --pattern /tmp/busser/suites/
serverspec/\*\*/\*_spec.rb --color --format documentation --default-path /
tmp/busser/suites/serverspec

       apache::default
         is awesome

       Finished in 0.02823 seconds (files took 0.99875 seconds to load)
       1 example, 0 failures
       Finished verifying <default-centos-64> (0m5.03s).
```

How would you test our criteria?

- What would you test to make sure apache is running?

Verify package is installed

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

```
require 'spec_helper'

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

  it "is installed" do
    expect(package("apache2")).to be_installed
  end
end
```

SAVE FILE!

CHEF

Verify the kitchen

```
$ kitchen verify
```

```
-----> Setting up Busser
       Creating BUSSER_ROOT in /tmp/busser
       Creating busser binstub
       Plugin serverspec installed (version 0.2.6)
-----> Running postinstall for serverspec plugin
       Finished setting up <default-centos-64> (0m32.59s).
-----> Verifying <default-centos-64>...
       Suite path directory /tmp/busser/suites does not exist, skipping.
       Uploading /tmp/busser/suites/serverspec/default_spec.rb (mode=0664)
-----> Running serverspec test suite
       /opt/chef/embedded/bin/ruby -I/tmp/busser/suites/serverspec -S /opt/chef/embedded/bin/rspec /tmp/busser/suites/serverspec/default_spec.rb
--color --format documentation


       apache
         is installed

       Finished in 0.29547 seconds
       1 example, 0 failures
       Finished verifying <default-centos-64> (0m4.44s).
-----> Kitchen is finished. (1m25.74s)
```

CHEF

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).
```

Change the 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 81).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 (FAILED - 1)
        displays a custom home page

      Failures:

        1) apache is listening on port 80
           On host ``
           Failure/Error: expect(port 81).to be_listening
      netstat -tunl | grep -- :81\
      expected Port "81" to be listening
           # /tmp/busser/suites/serverspec/default_spec.rb:16:in `block (2 levels) in <top (required)>'

      Finished in 0.38488 seconds
      4 examples, 1 failure
```

Reset 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

Kitchen Workflow

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

- All at once with `kitchen test`

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?

Even Faster Feedback

ChefSpec

CC BY SA

CHEF

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

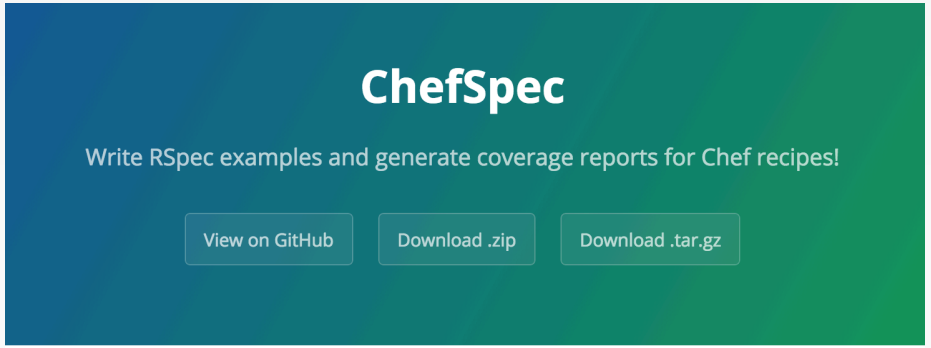
- We need a way to verify that the resources in our recipes are properly configured
- We want to get faster feedback

Lab – Verify the resources

- **Problem**: We should be able to catch errors before we need to converge a node
- **Success Criteria**: Catch a typo prior to converge

ChefSpec

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



ChefSpec

gem v4.2.0 build passing

ChefSpec is a unit testing framework for testing Chef cookbooks. ChefSpec makes it easy to write examples and get fast feedback on cookbook changes without the need for virtual machines or cloud servers.

ChefSpec runs your cookbook(s) locally with Chef Solo without actually converging a node. This has two primary benefits:

- It's really fast!
- Your tests can vary node attributes, operating systems, and search results to assert behavior under varying conditions.

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

CHEF

Change to the apache cookbook directory

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

Write a ChefSpec test

OPEN IN EDITOR: spec/unit/recipes/default_spec.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
      expect { chef_run }.to_not raise_error
    end

  end
end
```

SAVE FILE!

CHEF

Write a ChefSpec test

OPEN IN EDITOR: spec/unit/recipes/default_spec.rb

```
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
      expect { chef_run }.to_not raise_error
    end

    it 'installs apache' do
      expect(chef_run).to install_package 'apache2'
    end
  end
end
```

SAVE FILE!

CHEF

Run the ChefSpec tests

```
$ chef exec rspec spec -fd -c
```

```
apache::default
  When all attributes are default, on an unspecified platform
    converges successfully
    installs apache

Finished in 0.42858 seconds (files took 2.59 seconds to load)
2 examples, 0 failures
```

Break the cookbook

OPEN IN EDITOR: recipes/default.rb

```
package "apache"

service "httpd" do
  action :start
end
```

SAVE FILE!

CHEF

Run the ChefSpec tests

```
$ chef exec rspec spec -fd -c
```

```
apache::default
  When all attributes are default, on an unspecified platform
    converges successfully
    installs apache (FAILED - 1)

Failures:

  1) apache::default When all attributes are default, on an unspecified platform installs apache
     Failure/Error: expect(chef_run).to install_package 'apache2'
       expected "package[apache2]" with action :install to be in Chef run. Other package resources:

         package[apache]

     # ./spec/unit/recipes/default_spec.rb:21:in `block (3 levels) in <top (required)>'

Finished in 0.44712 seconds (files took 2.57 seconds to load)
2 examples, 1 failure
```

Fix the cookbook

OPEN IN EDITOR: recipes/default.rb

```
package "apache2"

service "apache2" do
  action :start
end
```

SAVE FILE!

CHEF

Run the ChefSpec tests

```
$ chef exec rspec spec -fd -c
```

```
apache::default
  When all attributes are default, on an unspecified platform
    converges successfully
    installs apache

Finished in 0.35496 seconds (files took 2.53 seconds to load)
2 examples, 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

CC BY SA

CHEF

Foodcritic

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

http://www.foodcritic.io/

CHEF

Change our recipe

OPEN IN EDITOR: recipes/default.rb

```
package_name = "apache2"

package "#{package_name}
```

```
service "apache2" do
  action :start
end
```

SAVE FILE!

CHEF

Run Foodcritic

```
$ foodcritic .
```

```
FC002: Avoid string interpolation
where not required: ./recipes/
default.rb:8
```

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?

Wrap Up

CC BY SA

CHEF

Course Objectives

- After completing this course you will be able to:
  - Automate common infrastructure tasks with Chef
  - Describe Chef's various tools
  - Apply Chef's primitives to solve your problems
  - Verify your automation code BEFORE it runs in production

Tool Survey

- chef-apply
- chef
- chef-client in local mode
- ohai
- Test Kitchen
- Docker
- Serverspec
- ChefSpec
- Foodcritic

Vocabulary

- Resources
- Recipes
- Cookbooks
- Run List
- Node
- Node Object

Resources

- Package
- Service
- File
- Template

But wait…

- …there's more, so much more!

- How much time do we have left? I could go on for days!

Further Resources

- learnchef.com
  - Guided tutorials
  - Chef Fundamental Series
- Upcoming Training
  - chef.io/blog/events/category/training-events/

Food Fight Show

- foodfightshow.org
- Podcast where DevOps Chefs Do Battle
- Best practices for working with Chef

CHEF


COMMUNITYSUMMIT2015
Seattle Summit
OCTOBER 14-15, 2015
London Summit
NOVEMBER 3-4, 2015
TEA

CHEF

Chef Community Summit – Seattle & London

Seattle – October 14 & 15
London – November 3 & 4

Why your participation matters

- Influence the path of the Chef roadmap
- Contribute to the formation of best practices and the avenues to best share them
- Share your experiences transforming your business
- Demonstrate your DevOps Kung Fu

Network with awesome engineers in the Community

- Engage with a community of people actively using Chef to automate their workflow
- Discuss “what keeps you up at night” with a passionate engaged audience
- Meet with CHEF engineers IRL

http://chef.io/summit

Nathen Harvey

- @nathenharvey
- nharvey@chef.io
- github.com/nathenharvey/surge_introduction_to_chef

- Occasional farmer – http://bit.ly/farmer-nathen
- Love Eggs – http://eggs.chef.io

CHEF

CYOA!

- Wrap-up
- Chef Delivery
- Chef + Vmware
- Remote execution with Chef (knife ssh)
- Audit Mode with Chef Analytics rules and integration to :slack:
- The audit-cis cookbook