Introduction to Chef

Testing your automation code

Nathen Harvey - @nathenharvey

Wifi: surge2014 login: surge2014 password: surge2014

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)

- http://bit.ly/1msreNZ

CHEF

Introductions

Instructor

- Nathen Harvey
- Community Director, Chef
- Co-host of the Food Fight Show
- Co-organizer of DevOpsDC
- 5th year at Surge

CHEF

Lab Assistants

- Mark Harrison
- Peter Burkholder
- Karloff Rosario
- YOU ☺

Hello!

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

Hello!

- Experience with configuration management?
- Experience with Chef?

Course Objectives & Style

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

Learning Chef

- You bring the domain expertise about your business and problems
- Chef provides a framework for solving those problems
- Our job is to work together to help you express solutions to your problems with Chef

Chef is a Language

- Learning Chef is like learning the basics of a language
  - 80% fluency reached quickly
  - 20% just takes practice
- The best way to ***LEARN*** Chef is to ***USE*** Chef

Training is a discussion

- Lots of hands on labs
- Lots of typing
- Ask questions when they come to you
- Ask for help when you need it
- Help each other
- We will troubleshoot and fix bugs on the spot

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

- You are getting an early access preview of this course!
- We're going to try out some new things. There may be some rough edges
- Follow along with the class on GitHub –
- Github.com/nathenharvey/surge_introduction_to_chef

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

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.getchef.com/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

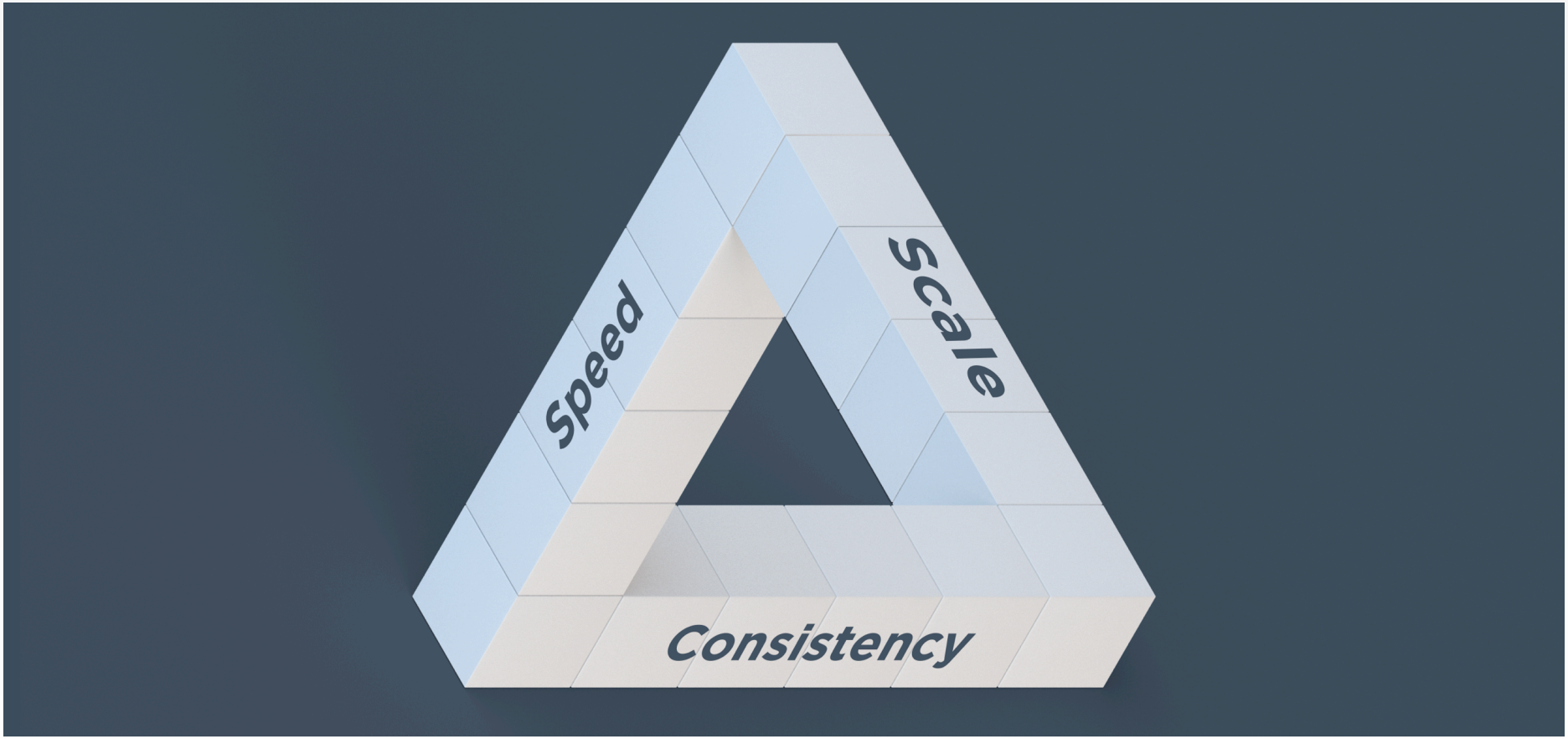
Overview of Chef

Policy-based Infrastructure as Code

v0.1.0

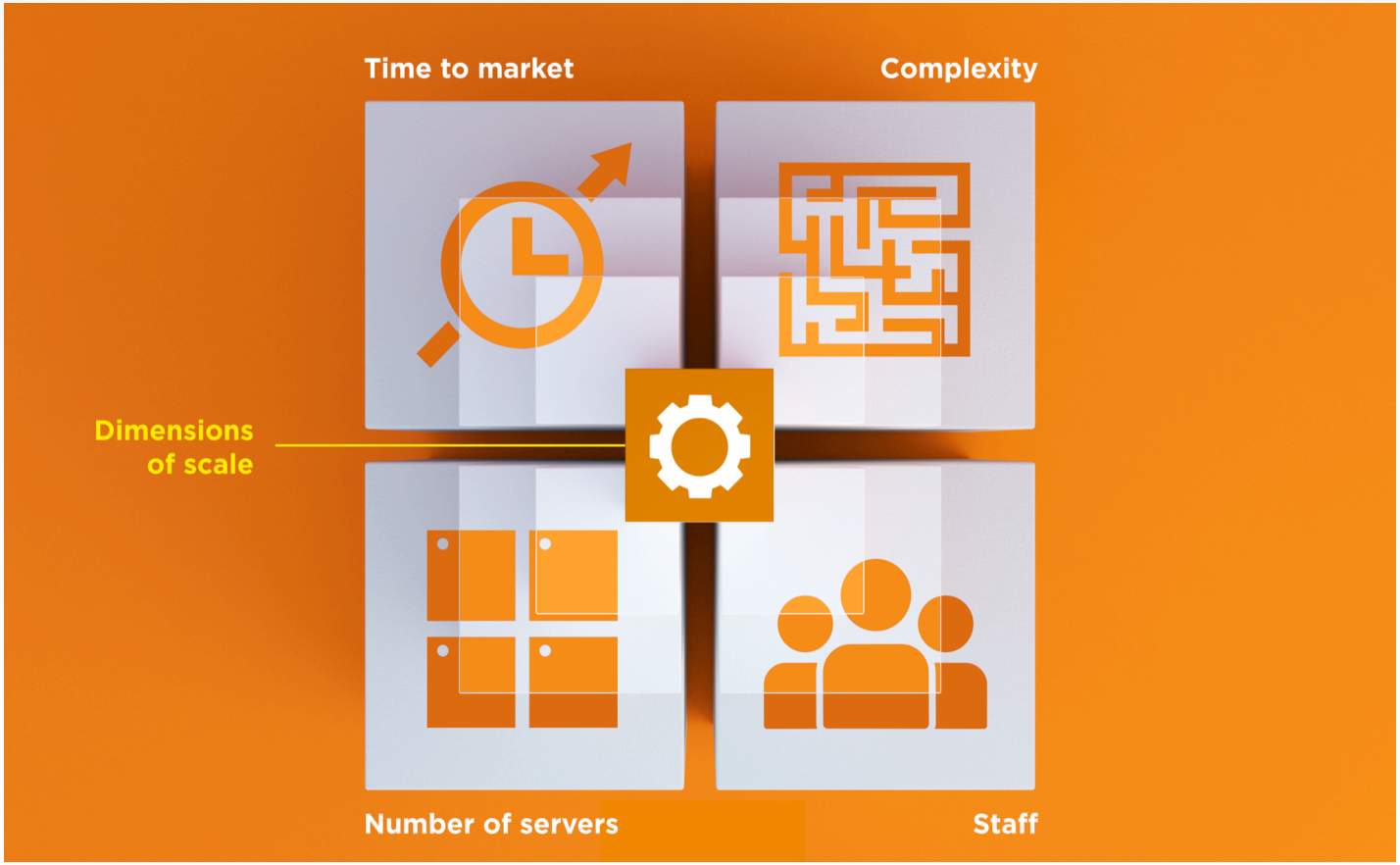
CHEF

Benefits of Automation



CHEF

Dimensions of Scale

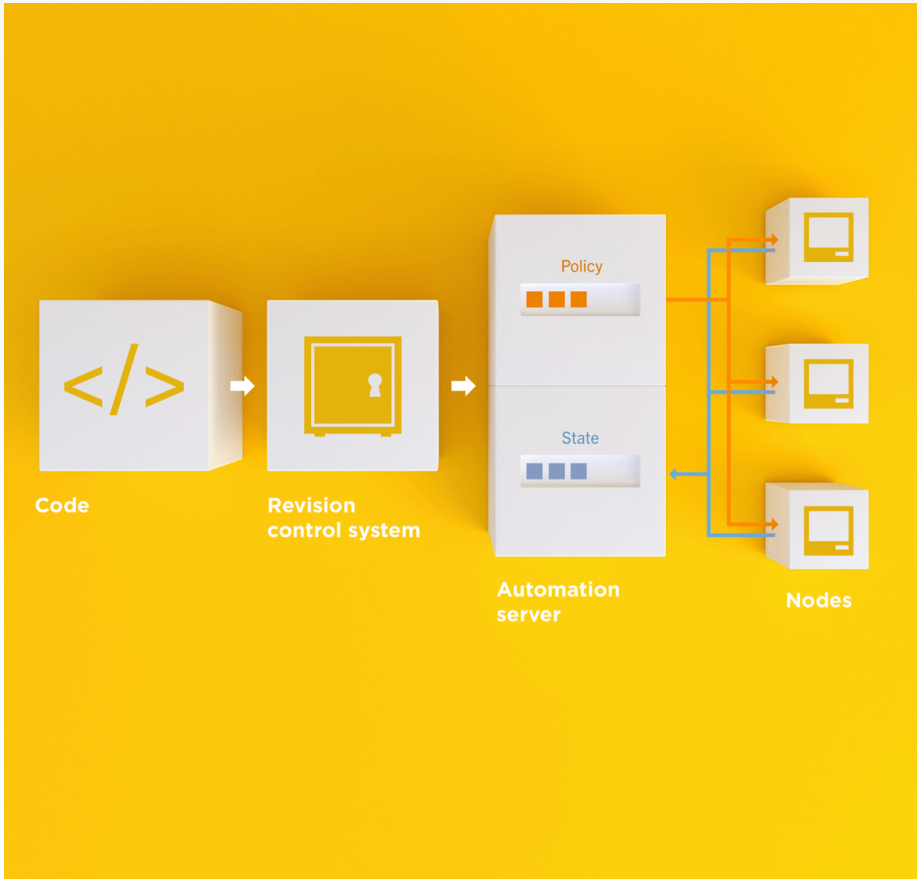


CHEF

Automation Platform

- Creates a dependable view of your entire network's state.
- Can handle complex dependencies among the nodes of your network.
- Is fault tolerant.
- Is secure.
- Can handle multiple platforms
- Can manage cloud resources
- Provides a foundation for innovation

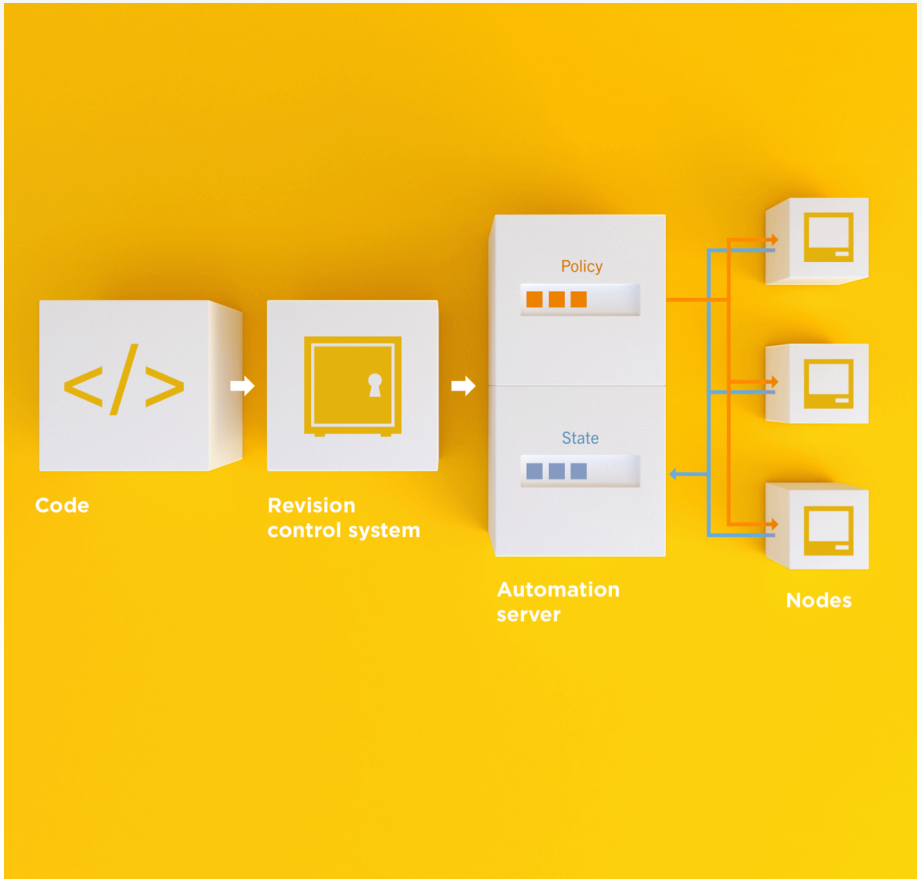
Infrastructure as Code



- Programmatically provision and configure components

CHEF

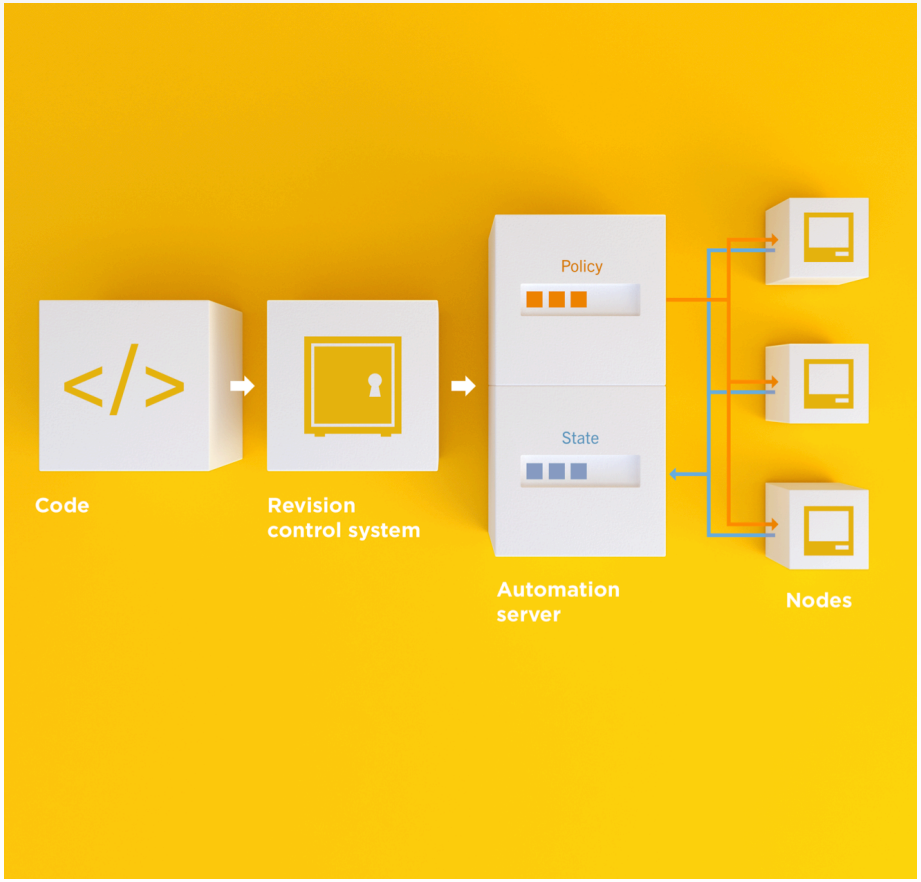
Infrastructure as Code



- Treat like any other code base

CHEF

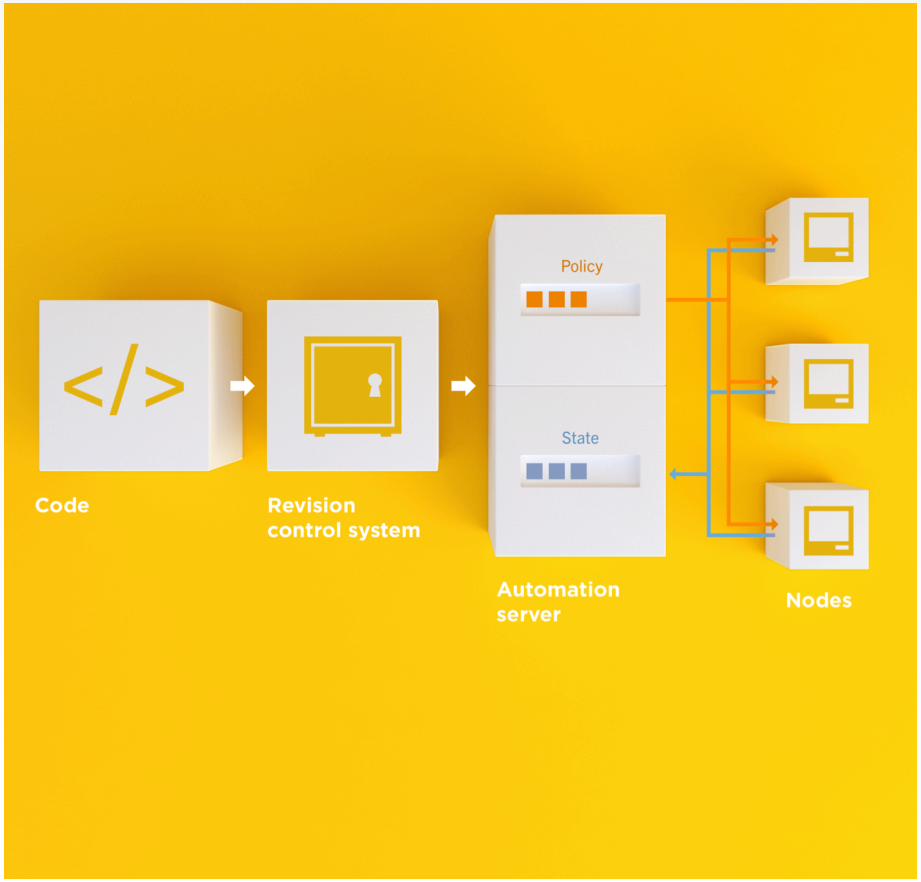
Infrastructure as Code



- Reconstruct business from code repository, data backup, and compute resources

CHEF

Infrastructure as Code



- Programmatically provision and configure components
- Treat like any other code base
- Reconstruct business from code repository, data backup, and compute resources

CHEF

Policy-based

- You capture the policy for your infrastructure in code
- Chef ensures each node in your infrastructure complies with the policy

Policy-based

- Chef provides a domain-specific language (DSL) that allows you to specify policy for your infrastructure
- Policy describes the desired state
- Policies can be statically or dynamically defined

Resources

Fundamental building blocks

CHEF

Resources

- Piece of the system and its desired state
  - Package that should be installed
  - Service that should be running
  - File that should be generated
  - Cron job that should be configured
  - User that should be managed
  - And more
- docs.getchef.com/chef/resources.html

Lab 1 – Install a text editor

- **Problem**: Our workstation does not have $EDITOR installed
- **Success Criteria**: You can edit files with $EDITOR

- $EDITOR is your favorite command line text editor: vim, emacs, or nano

What's up with the card?

- http://bit.ly/1msreNZ

- Login: chef
- Password: sxypnerg

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

Is $EDITOR installed?

```
$ which vim
```

```
/usr/bin/which: no vim in (/opt/
chefdk/bin:/home/chef/.chefdk/gem/
ruby/2.1.0/bin:/opt/chefdk/embedded/
bin:/usr/local/bin:/bin:/usr/bin:/
usr/local/sbin:/usr/sbin:/sbin:/
home/chef/bin)
```

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

What does chef-apply do?

```
$ chef-apply --help
```

```
Usage: chef-apply [RECIPE_FILE] [-e RECIPE_TEXT] [-s]
        --[no-]color                 Use colored output,
defaults to enabled
    -e, --execute RECIPE_TEXT        Execute resources
supplied in a string
    -l, --log_level LEVEL            Set the log level
(debug, info, warn, error, fatal)
    -s, --stdin                      Execute resources
read from STDIN
    -v, --version                    Show chef version
    -W, --why-run                    Enable whyrun mode
    -h, --help                       Show this message
```

CHEF

Install vim

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[vim] action install
    - install version 7.2.411-1.8.el6 of package vim-enhanced
```

Install emacs

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[emacs] action install
    - install version 23.1-25.el6 of package emacs
```

Install nano

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

```
Recipe: (chef-apply cookbook)::(chef-apply
recipe)
  * package[nano] action install
    - install version 2.0.9-7.el6 of package nano
```

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 $EDITOR again with chef-apply

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

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

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)

Recipes

Collection of resources

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 "httpd"
```

SAVE FILE!

CHEF

Apply the apache recipe

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[httpd] action install
    - install version 2.2.15-31.el6.centos of package httpd
```

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
```

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

Write a recipe for apache

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

```
package "httpd"

service "httpd"
```

SAVE FILE!

CHEF

Apply the apache recipe

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[httpd] action install (up to date)
  * service[httpd] action nothing (up to date)
```

Is it working?

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

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

Apply the apache recipe

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[httpd] action install (up to date)
  * service[httpd] action nothing (up to date)
```

Apply the apache recipe

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[httpd] action install (up to date)
  * service[httpd] action nothing (up to date)
```

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?

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

Write a recipe for apache

OPEN IN EDITOR: ~/learnchef/apache.rb

```
package "httpd"

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

SAVE FILE!

CHEF

Apply the apache recipe

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

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

Is it working?

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

```
 <title>Apache HTTP Server Test Page powered by CentOS</title>
                <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
                <style type="text/css">
                        body {
                                background-color: #fff;
                                color: #000;
                                font-size: 0.9em;
                                font-family: sans-serif,helvetica;
                                margin: 0;
                                padding: 0;
                        }
```

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 "httpd"

service "httpd" 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
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[httpd] action install (up to date)
  * service[httpd] action start (up to date)
  * file[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 372e08 to 43191a
    --- /var/www/html/index.html        2014-09-23 06:05:01.239625369 +0000
    +++ /tmp/.index.html20140923-13241-piz164   2014-09-23
06:06:39.971626393 +0000
    @@ -1,2 +1,2 @@
+<h1>Hello, world!</h1>
    - restore selinux security context
```

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)
  * package[httpd] action install (up to date)
  * service[httpd] action start (up to date)
  * file[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 372e08 to 43191a
    --- /var/www/html/index.html        2014-09-23 06:05:01.239625369 +0000
    +++ /tmp/.index.html20140923-13241-piz164   2014-09-23
06:06:39.971626393 +0000
    @@ -1,2 +1,2 @@
    -<h1>Hello, You!</h1>
    +<h1>Hello, world!</h1>
    - restore selinux security context
```

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

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

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-repo

- Managing infrastructure as code means storing that code in a version control system
- Any version control system will do but…
  - Chef community prefers and recommends git
  - Many tools support git by default

How many git repos?

- Once you have more than one cookbook, you may ask yourself this question
- The answer is easy:

How many git repos?

- Once you have more than one cookbook, you may ask yourself this questions
- The answer is easy:
  - It depends!

How many git repos?

- Once you have more than one cookbook, you may ask yourself this questions
- The answer is easy:
  - It depends!

- Two options are common:
  - Monolithic Repository
  - Independent Software Projects

Monolithic Repository

- All of your Chef related source code tracked in one source code repository
- External dependencies are made with built-in vendor branches

Independent Software Projects

- All Chef cookbooks are treated as independent software projects
- External dependencies are
  - fetched as needed
  - treated as artifacts

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. Install git?
2. Create a chef-repo
3. 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
```

A few questions…

- Will you store this repo in git?
  - How can you install git?
- Which style of repo will you use?
  - Monolithic repository?
  - Independent software projects?

Create a directory

```
$ mkdir -p ~/learnchef/lab3
```

CHEF

Install git

OPEN IN EDITOR: `~/learnchef/lab3/git.rb`

SAVE FILE!

CHEF

Install git

OPEN IN EDITOR: ~/learnchef/lab3/git.rb

```
package 'git'
```

SAVE FILE!

CHEF

Install git

OPEN IN EDITOR: ~/learnchef/lab3/git.rb

```
package 'git'

file '/home/chef/.gitconfig' do
  content "[user]\n  name=John Doe\n  email=jdoe@example\n"
  user 'chef'
  group 'chef'
end
```

SAVE FILE!

CHEF

Install git

```
$ sudo chef-apply ~/learnchef/lab3/git.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * package[git] action install
    - install version 1.7.1-3.el6_4.1 of package git
  * file[/home/chef/.gitconfig] action create
    - create new file /home/chef/.gitconfig
    - update content in file /home/chef/.gitconfig from none to 259950
    --- /home/chef/.gitconfig   2014-09-24 00:24:13.558127555 +0000
    +++ /tmp/..gitconfig20140924-10180-1ij68vq  2014-09-24 00:24:13.559127555 +0000
    @@ -1 +1,4 @@
    +[user]
    +  name=John Doe
    +  email=jdoe@example.com
    - change owner from '' to 'chef'
    - change group from '' to 'chef'
    - restore selinux security context
```

Lab 3 – Manage the homepage content separately

- ✓ Install git?
2. Create a chef-repo
3. 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

```
$ 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

- ✓ Install git?
- ✓ Create a chef-repo
3. 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 "httpd"

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

- ✓ Install git?
- ✓ Create a chef-repo
- ✓ Create an apache cookbook

Verify the changes

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

```
[2014-09-17T11:21:52+00:00] WARN: No config file found or specified on command line, using
command line options.
Starting Chef Client, version 11.16.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[httpd] action install (up to date)
  * service[httpd] 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 5.637631958 seconds
```

CHEF

Verify the changes

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

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

Ensuring desired state

chef-client and run lists

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)
- Typically run with elevated privileges
  - root, sudo, or Administrator
- Can run as non-root user

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

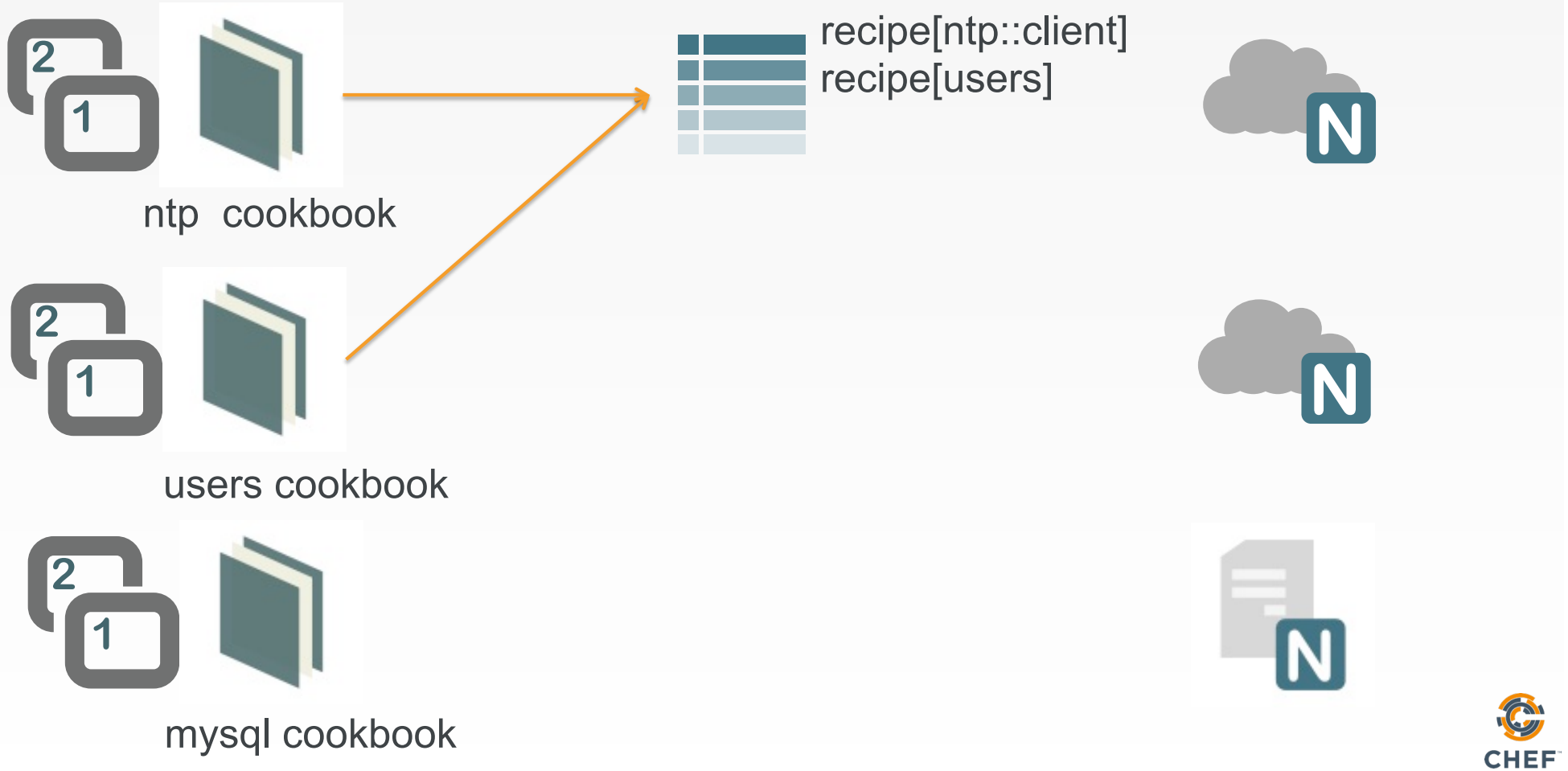
ntp cookbook

users cookbook

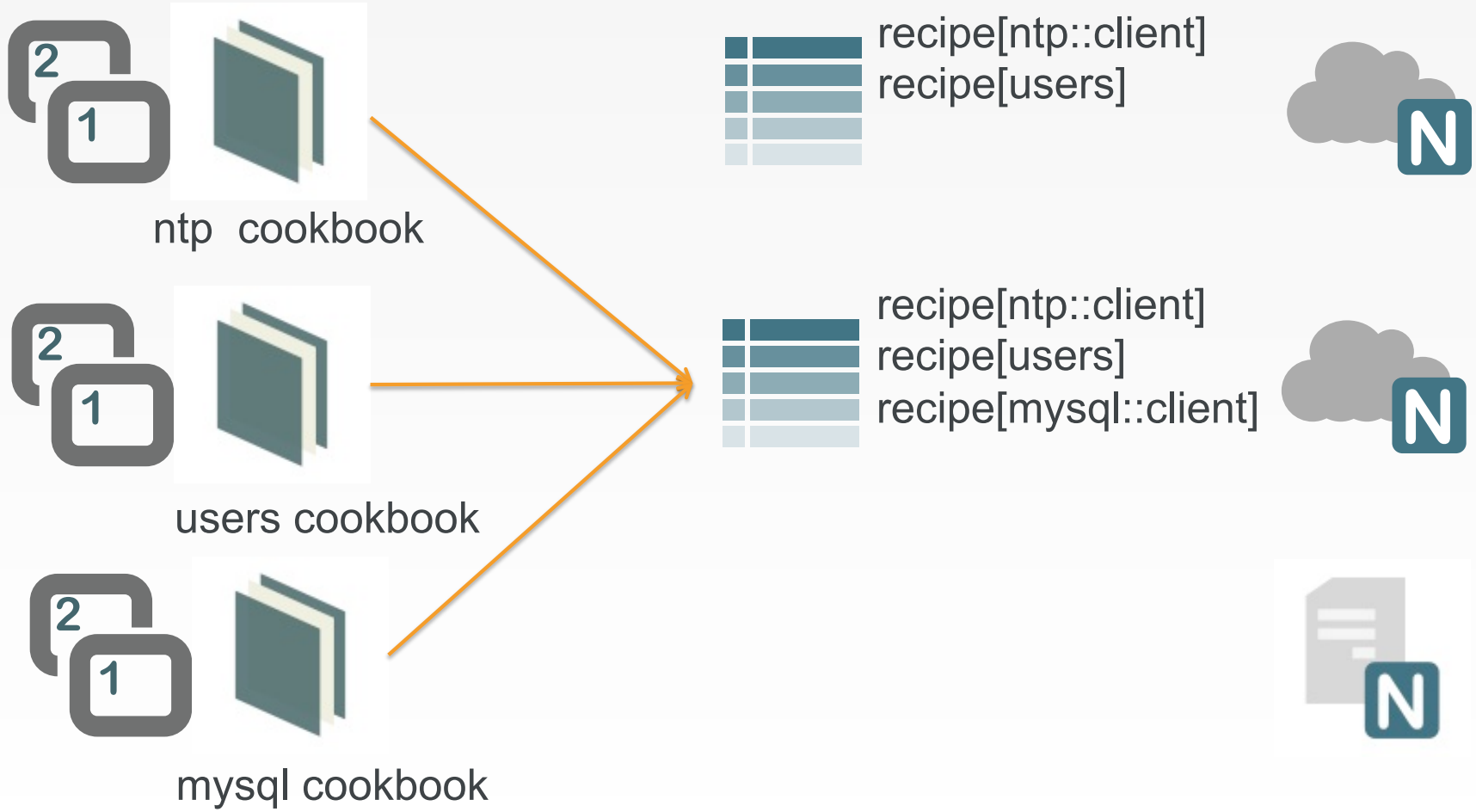
mysql cookbook

CHEF

Run List

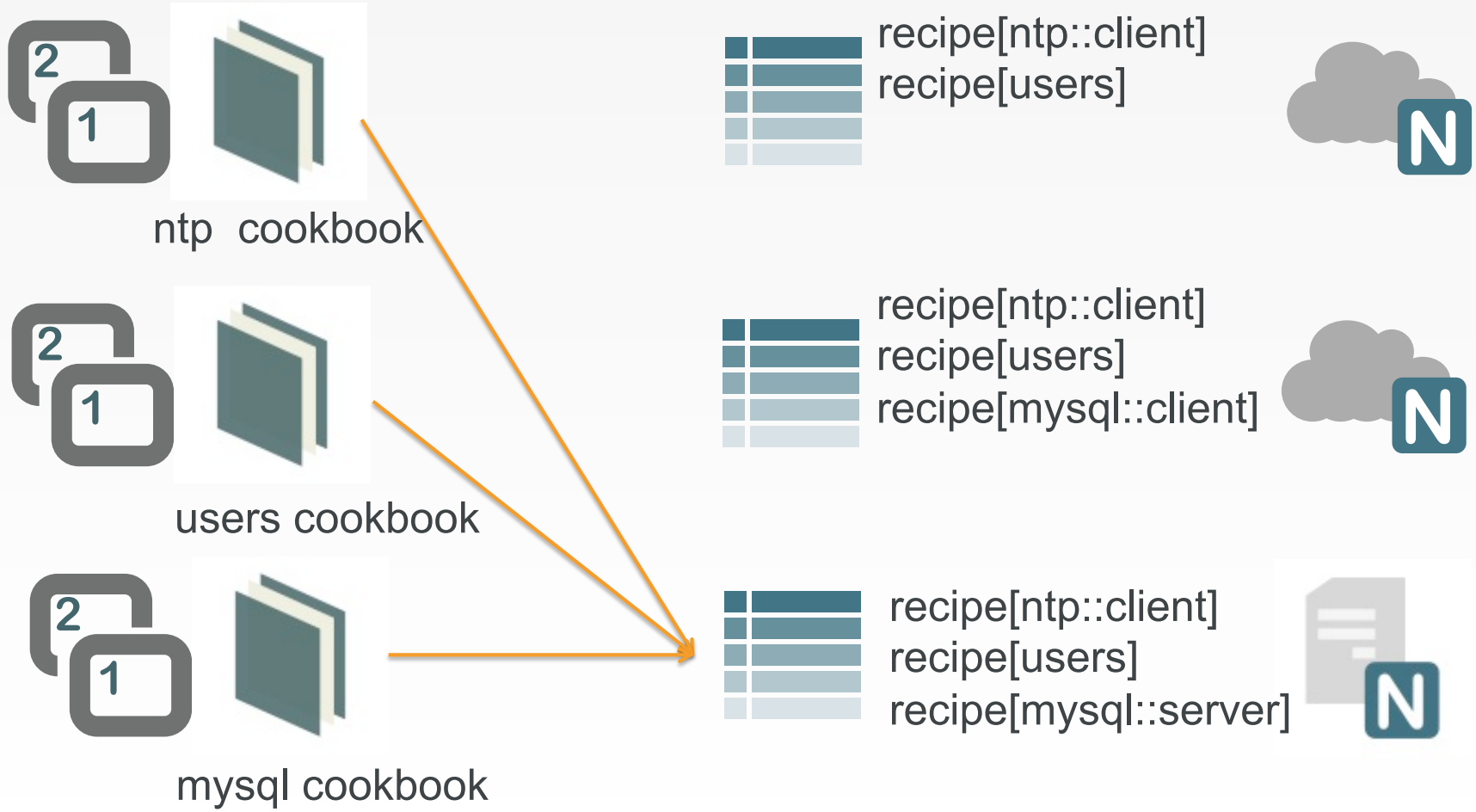

recipe[ntp::client]
recipe[users]
ntp cookbook
users cookbook
mysql cookbook

Run List



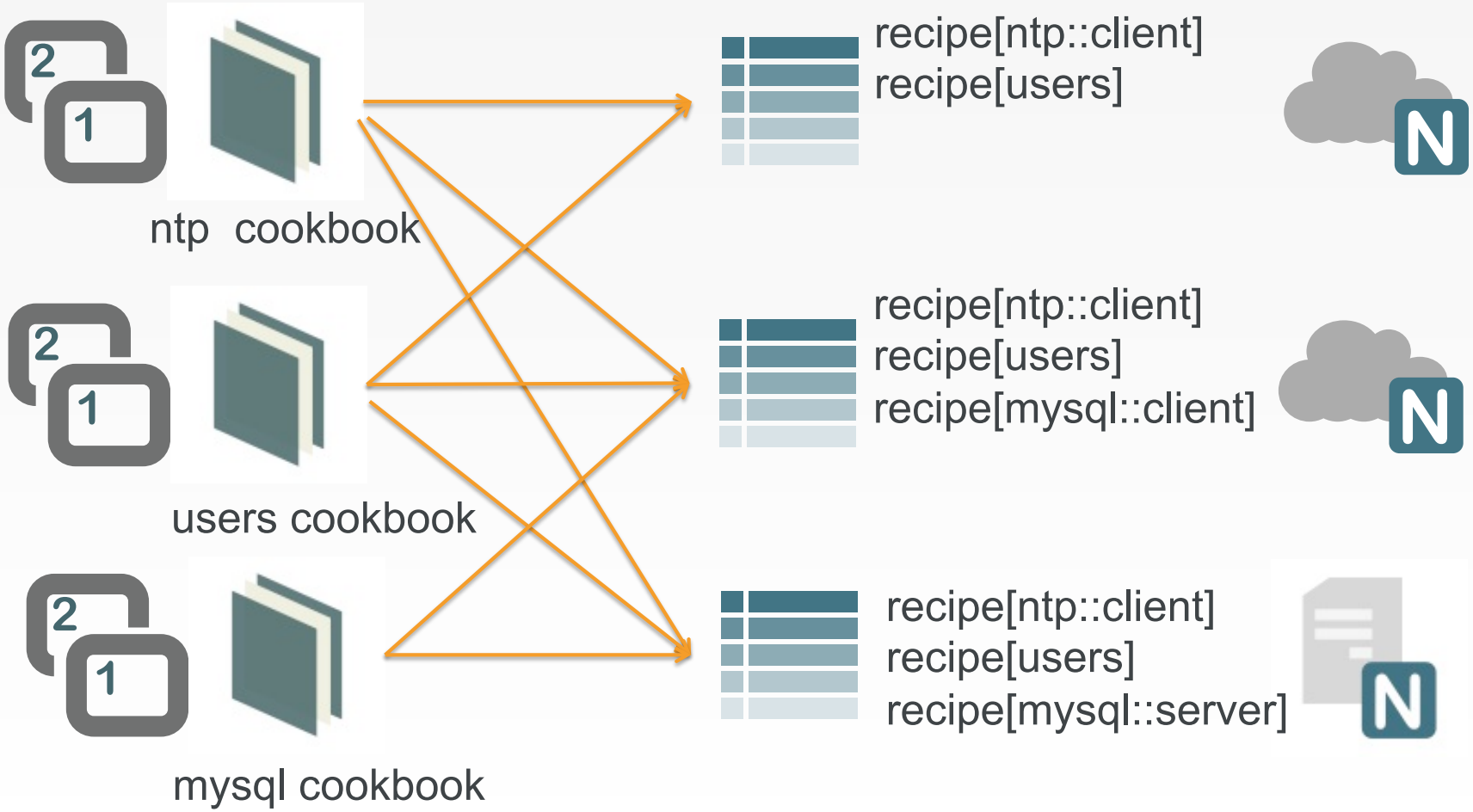
CHEF

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]"
```

```
Starting Chef Client, version 11.16.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[httpd] action install (up to date)
  * service[httpd] action start (up to date)
  * file[/var/www/html/index.html] action create (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/3 resources updated in 14.375566874 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
```

```
[
  "centos"
]
```

Find the platform in the node object

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

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

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]"
```

```
Starting Chef Client, version 11.16.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[httpd] action install (up to date)
  * service[httpd] 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 4ff592
    --- /var/www/html/index.html        2014-09-23 13:28:23.912127556 +0000
    +++ /tmp/chef-rendered-template20140924-11927-m0wkgp        2014-09-24 02:07:43.954127556 +0000
    @@ -1,2 +1,8 @@
     <h1>hello world</h1>
    +
    +<p>
    +  This is a centos 6.5 server.
    +  with 1695028kB RAM.
    +</p>
    +
    - restore selinux security context
```

CHEF

Verify the change

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

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


<p>
  This is a centos 6.5 server.
  with 1695028kB 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
    - restore selinux security context
  * 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)
    - restore selinux security context
```

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

Verify the change

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

```
Starting Chef Client, version 11.16.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[httpd] action install (up to date)
  * service[httpd] action start (up to date)
  * template[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 4ff592 to 470462
    --- /var/www/html/index.html        2014-09-24 02:07:43.954127556 +0000
    +++ /tmp/chef-rendered-template20140924-12185-11ypb0t       2014-09-24 02:25:25.833127556 +0000
    @@ -1,4 +1,4 @@
    -<h1>hello world</h1>
    +<h1>hello DC</h1>

     <p>
       This is a centos 6.5 server.
    - restore selinux security context

Running handlers:
```

CHEF

Verify the change

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

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


<p>
  This is a centos 6.5 server.
  with 1695028kB RAM.
</p>
```

Faster Feedback

Automate your testing

v0.1.0

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

The pedantries of testing

- Unit testing
- Integration testing
- Acceptance testing
- Functional testing
- Regression testing
- Smoke testing
- Load testing

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

.kitchen.yml

- The configuration file for your Test Kitchen
- driver – virtualization or cloud provider
- provisioner – application to configure the node
- platforms – target operating systems
- suites – target configurations

Update .kitchen.yml

OPEN IN EDITOR: cookbooks/apache/.kitchen.yml

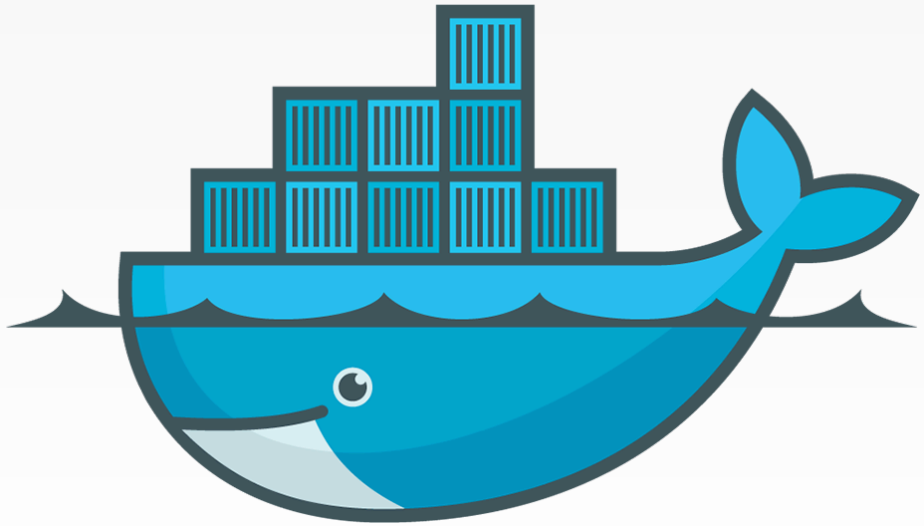
```
---
driver:
  name: docker

provisioner:
  name: chef_zero

platforms:
  - name: centos-6.4

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

SAVE FILE!

CHEF

Docker

- Portable, lightweight application runtime
- Linux containers
- Installed on the workstation



https://d3oypxn00j2a10.cloudfront.net/0.10.3/img/homepage/docker-whale-home-logo-@2x.png?cf34b4b2b839

CHEF

Verify docker

```
$ sudo docker images
```

```
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
centos              centos5             5a1ebaa356ff        2 weeks ago         484 MB
centos              centos7             70214e5d0a90        2 weeks ago         224 MB
centos              latest              70214e5d0a90        2 weeks ago         224 MB
centos              centos6             68eb857ffb51        2 weeks ago         212.7 MB
```

kitchen-docker gem

- A driver that allows Test Kitchen to work with Docker
- Installed on the workstation

- ChefDK includes kitchen-vagrant

Verify kitchen-docker is installed

```
$ gem list kitchen
```

```
*** LOCAL GEMS ***

kitchen-docker (1.5.0)
kitchen-vagrant (0.15.0)
test-kitchen (1.2.1)
```

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
```

List the Test Kitchens

```
$ kitchen list
```

```
Instance             Driver   Provisioner   Last Action
default-centos-64    Docker   ChefZero      <Not Created>
```

Create the kitchen

```
$ kitchen create
```

```
-----> Starting Kitchen (v1.2.1)
-----> 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
```

Converge: run chef in the kitchen

```
$ kitchen converge
```

```
-----> Starting Kitchen (v1.2.1)
-----> Converging <default-centos-64>...
       Preparing files for transfer
       Resolving cookbook dependencies with Berkshelf 3.1.5...
       Removing non-cookbook files before transfer
-----> Installing Chef Omnibus (true)
       downloading https://www.getchef.com/chef/install.sh
         to file /tmp/install.sh
       trying curl...
```

Break the cookbook

OPEN IN EDITOR: recipes/default.rb

```
package "apache"

service "httpd" do
  action :start
end

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

SAVE FILE!

CHEF

Re-converge

```
$ kitchen converge
```

```
…
 Chef Client failed. 0 resources updated in 41.075753164 seconds
       [2014-09-24T04:29:31+01:00] ERROR: package[apache] (apache::default line 6) had an
error: Chef::Exceptions::Package: No version specified, and no candidate version available
for apache
       [2014-09-24T04:29:31+01:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run
process exited unsuccessfully (exit code 1)
>>>>>> Converge failed on instance <default-centos-64>.
>>>>>> Please see .kitchen/logs/default-centos-64.log for more details
>>>>>> ------Exception-------
>>>>>> Class: Kitchen::ActionFailed
>>>>>> Message: SSH exited (1) for command: [sudo -E chef-client -z --config /tmp/kitchen/
client.rb --log_level info --json-attributes /tmp/kitchen/dna.json]
>>>>>> ----------------------
```

Fix the cookbook

OPEN IN EDITOR: recipes/default.rb

```
package "httpd"

service "httpd" do
  action :start
end

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

SAVE FILE!

CHEF

Converge: run chef in the kitchen

```
$ kitchen converge
```

```
 [2014-09-24T04:30:45+01:00] INFO: Chef Run complete in
4.110425475 seconds

       Running handlers:
       [2014-09-24T04:30:45+01:00] INFO: Running report handlers
       Running handlers complete
       [2014-09-24T04:30:45+01:00] INFO: Report handlers complete
       Chef Client finished, 0/3 resources updated in
11.323451399 seconds
       Finished converging <default-centos-64> (0m18.84s).
-----> Kitchen is finished. (0m20.09s)
```

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?

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
```

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

Manually Inspect with kitchen login

```
$ kitchen login
```

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

Manually Inspect with kitchen login

```
$ kitchen login
```

```
kitchen@localhost's password: kitchen
Last login: Wed Sep 24 04:30:29 2014 from 172.17.42.1
```

Manually Inspect with kitchen login

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

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


<p>
  This is a centos 6.5 server.
  with 1695028kB RAM.
</p>
```

Serverspec

- Write RSpec 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

Create directory for serverspec tests

```
$ mkdir -p test/integration/default/serverspec
```

CHEF

Write a Serverspec test

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

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

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

SAVE FILE!

CHEF

Destroy the existing kitchen

```
$ kitchen destroy
```

```
-----> Starting Kitchen (v1.2.1)
-----> Destroying <default-centos-64>...
       [{
           "Args": [
        "-D",
        "-o",
        "UseDNS=no",
        "-o",
        "UsePAM=no",
        "-o",
        "UsePrivilegeSeparation=no",
        "-o",
```

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 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 (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 Testing

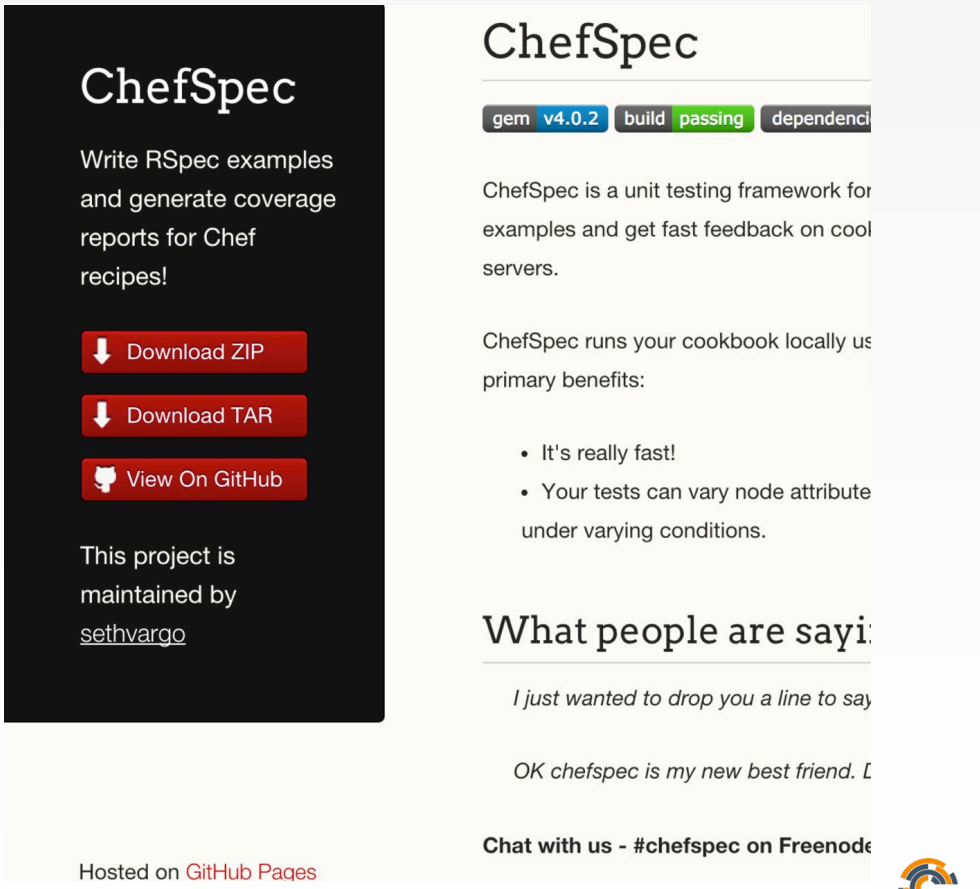
- ✓ Did chef-client complete successfully?
- ✓ Did the recipe put the node in the desired state?
- Are the resources properly defined?
- Does the code following our style guide?

Lab 9 – 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



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

CHEF

Make a directory for our ChefSpec tests

```
$ mkdir -p spec/unit
```

Write a ChefSpec test

OPEN IN EDITOR: spec/unit/default.rb

```
require 'chefspec'

describe 'apache::default' do
  let(:chef_run) do
    ChefSpec::Runner.new.converge(described_recipe)
  end

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

SAVE FILE!

CHEF

Run the ChefSpec tests

```
$ rspec spec/unit/*.rb
```

```
.

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

Break the cookbook

OPEN IN EDITOR: recipes/default.rb

```
package "http"

service "httpd" do
  action :start
end

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

SAVE FILE!

CHEF

Run the ChefSpec tests

```
$ rspec spec/unit/*.rb
```

```
F

Failures:

  1) apache::default installs apache
     Failure/Error: expect(chef_run).to install_package('httpd')
       expected "package[httpd]" with action :install to be in Chef run. Other package resources:

         package[http]

     # ./spec/unit/default_spec.rb:9:in `block (2 levels) in <top (required)>'

Finished in 0.00847 seconds (files took 4.85 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./spec/unit/default_spec.rb:8 # apache::default installs apache
```

Fix the cookbook

OPEN IN EDITOR: recipes/default.rb

```
package "httpd"

service "httpd" do
  action :start
end

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

SAVE FILE!

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?

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 = "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?

Wrap Up

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

Repository Layout

- Monolithic Repository
- Independent Software Projects

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
  - getchef.com/blog/events/category/training-events/

Training in DC

- SURGE14 will save 20%
- Chef Fundamentals – October 27-28
- Chef Intermediate Topics – October 29-30

Chef Fundamentals Q & A Forum

- Chef Fundamentals Google Group Q&A Forum
- http://bit.ly/ChefFundamentalsForum
- Join the group and post questions

A list of URLs

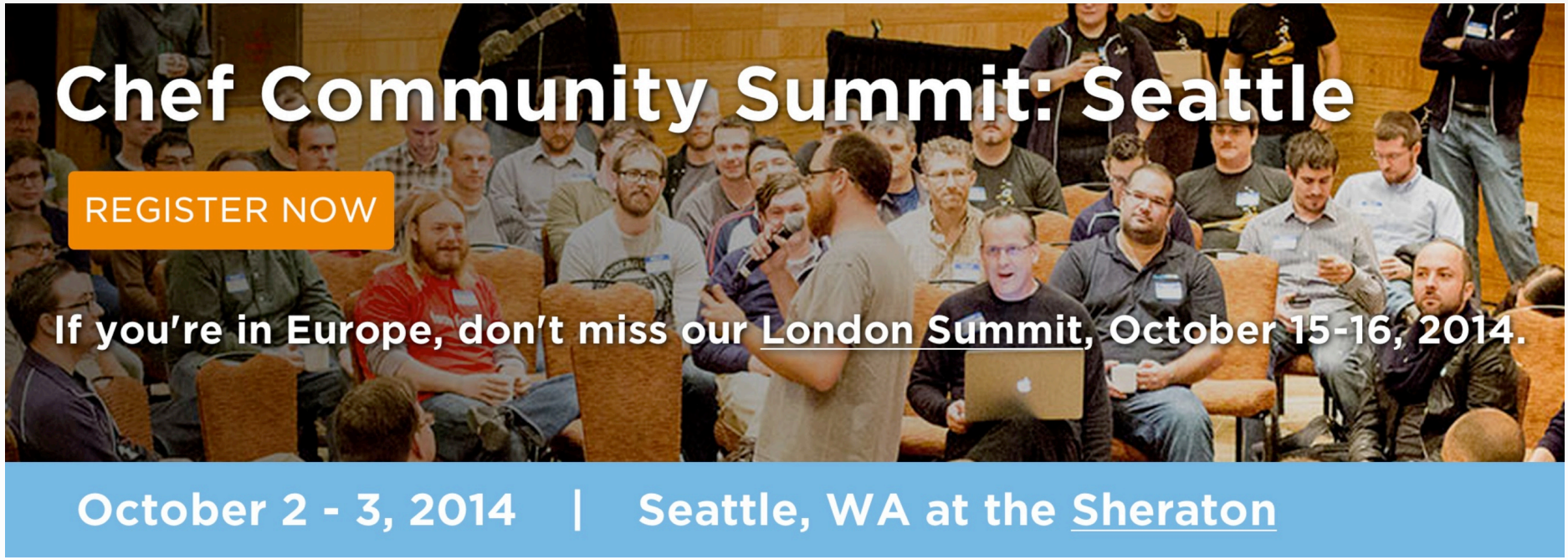
- http://getchef.com
- http://docs.getchef.com
- http://supermarket.getchef.com
- http://youtube.com/getchef
- http://lists.opscode.com
- irc.freenode.net: #chef, #chef-hacking
- Twitter: @chef #getchef, @learnchef #learnchef

Food Fight Show

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

CHEF

getchef.com/summit



CHEF

Local Meetup Groups

- DevOpsDC
- DC Configuration Management Group
- Charm City Linux
- AWS Washington DC Meetup
- LOPSA DC
- Amazon Web Services – Virginia
- DevOpsCV – Central Virginia (Charlottesville)
- DC Continuous Delivery Meetup
- Scale DC
- Cloud DC
- Washington DevOps in Government Meetup
- DC Area Enterprise DevOps and Agile Forum

What questions do you have?

- Chef Server
- Roles
- Environments
- Data Bags
- Bootstrapping new nodes
- Open source projects
- Working with IaaS providers
- chef-metal
- Search
- Suspenders?!

- Thank You!
- @nathenharvey