Cookbooks

Organizing Recipes

CHEF

Objectives

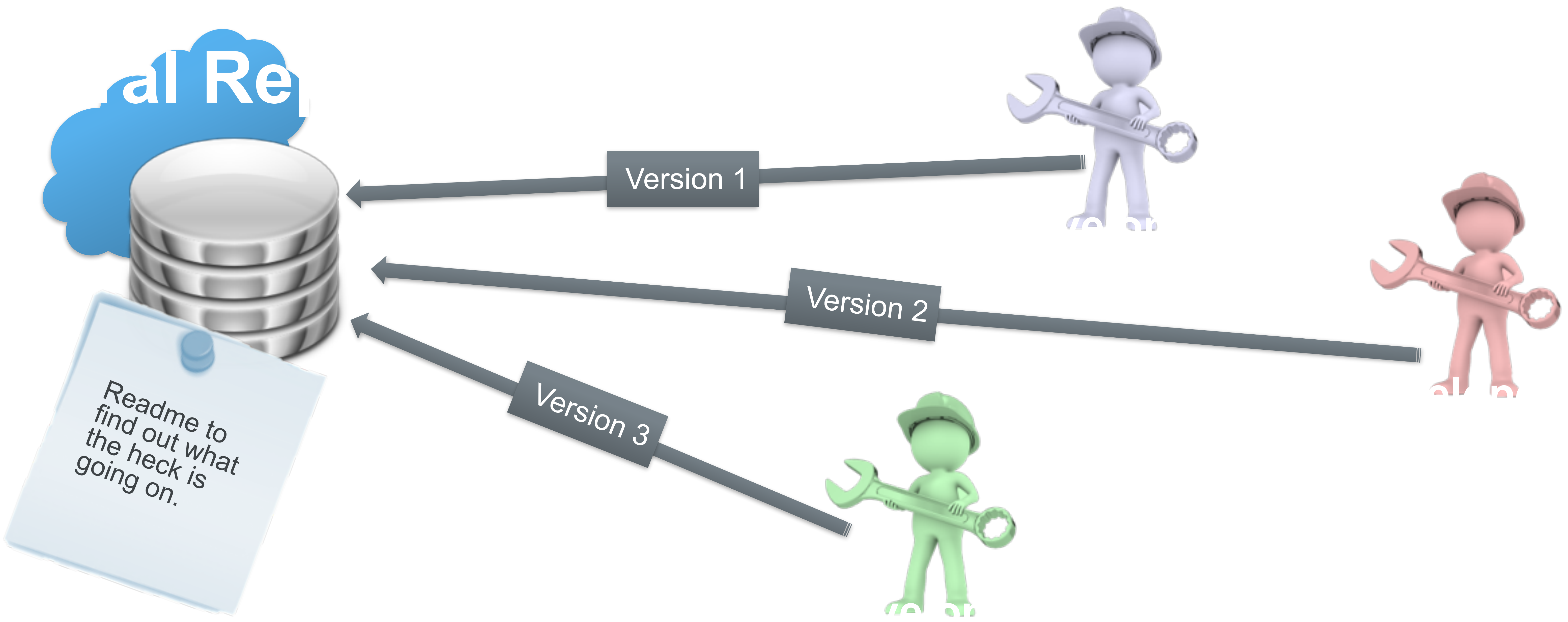
After completing this module, you should be able to:

- ➢ Modify a recipe
- ➢ Use version control
- ➢ Generate a Chef cookbook
- ➢ Define a Chef recipe that sets up a web server

Questions You May Have

1. Thinking about the workstation recipe, could we do something like that for a web server?
2. Is there a way to package up recipes you create with a version number (and maybe a README)?
3. I think `chef` is able to generate something called a cookbook. Shouldn't we start thinking about some version control so we don't lose all our hard work?

Collaboration and Version Control



Versioning Pros and Cons

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

Saving a copy of the original file as another filename.

Git Version Control

git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.

We will be using **git** throughout the rest of this course.

Lab: Install git

- ❑ Add the additional policy to the "setup.rb":

  The package named 'git' is installed.

- ❑ Then apply this recipe with chef-client.

Lab: Adding the git Package

~/setup.rb

```
package 'cowsay' do
  action :install
end

package 'tree' do
  action :install
end

package 'git' do
  action :install
end

file '/etc/motd' do
  content 'Property of ...'
end
```

Lab: Re-apply the Setup Recipe

```
$ sudo chef-client --local-mode setup.rb
```

```
Converging 4 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[cowsay] action install (up to date)
  * yum_package[tree] action install (up to date)
  * yum_package[git] action install
    - install version 1.7.1-3.el6_4.1 of package git
  * file[/etc/motd] action create (up to date)
```

Lab: Install git

- ✓ Add the additional policy to the "setup.rb":

  The package named 'git' is installed.

- ✓ Then apply this recipe with chef-client.

GL: Create a Cookbook

How are we going to manage this file? Does it need a README?

Objective:

- ❑ Use chef to generate a cookbook
- ❑ Move the setup recipe into the new cookbook
- ❑ Add the new cookbook to version control

Cookbooks

A Chef cookbook is the fundamental unit of configuration and policy distribution.

Each cookbook defines a scenario, such as everything needed to install and configure MySQL, and then it contains all of the components that are required to support that scenario.

Read the first three paragraphs here: http://docs.chef.io/cookbooks.html

GL: Create a Cookbooks Directory

```
$ mkdir cookbooks
```

What is 'chef'?

An executable program that allows you generate cookbooks and cookbook components.

What can 'chef' do?

```
$ chef --help
```

```
UsaGL:
    chef -h/--help
    chef -v/--version
    chef command [arguments...] [options...]

Available Commands:
    exec         Runs the command in context of the embedded ruby
    gem          Runs the `gem` command in context of the embedded ruby
    generate     Generate a new app, cookbook, or component
    shell-init   Initialize your shell to use ChefDK as your primary ruby
    install      Install cookbooks from a Policyfile and generate a locked cookboo...
    update       Updates a Policyfile.lock.json with latest run_list and cookbooks
```

What Can 'chef generate' Do?

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

```
UsaGL: 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
  policyfile   Generate a Policyfile for use with the install/push commands
```

GL: Let's Create a Cookbook

```
$ chef generate cookbook cookbooks/workstation
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
* directory[/home/chef/workstation] action create
    - create new directory /home/chef/workstation
  * template[/home/chef/workstation/metadata.rb] action create_if_missing
    - create new file /home/chef/workstation/metadata.rb
    - update content in file /home/chef/workstation/metadata.rb from none to bd85d3
    (diff output suppressed by config)
  * template[/home/chef/workstation/README.md] action create_if_missing
    - create new file /home/chef/workstation/README.md
    - update content in file /home/chef/workstation/README.md from none to 44d165
    (diff output suppressed by config)
  * cookbook_file[/home/chef/workstation/chefignore] action create
```

GL: The Cookbook Has a README

```
$ tree cookbooks/workstation
```

```
workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
10 directories, 9 files
```

CONCEPT

README.md

The description of the cookbook's features written in Markdown.

http://daringfireball.net/projects/markdown/syntax

GL: The Cookbook Has Some Metadata

```
$ tree cookbooks/workstation
```

```
workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
10 directories, 9 files
```

metadata.rb

Every cookbook requires a small amount of metadata. Metadata is stored in a file called metadata.rb that lives at the top of each cookbook's directory.

http://docs.chef.io/config_rb_metadata.html

GL: Let's Take a Look at the Metadata

```
$ cat cookbooks/workstation/metadata.rb
```

```
name             'workstation'
maintainer       'The Authors'
maintainer_email 'you@example.com'
license          'all_rights'
description      'Installs/Configures workstation'
long_description 'Installs/Configures workstation'
version          '0.1.0'
```

GL: The Cookbook Has a Folder for Recipes

```
$ tree cookbooks/workstation
```

```
workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
10 directories, 9 files
```

GL: The Cookbook Has a Default Recipe

```
$ cat cookbooks/workstation/recipes/default.rb
```

```
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2016 The Authors, All Rights Reserved.
```

GL: Create a Cookbook

How are we going to manage this file? Does it need a README?

Objective:

- ✓ Use chef to generate a cookbook
- ❑ Move the setup recipe into the new cookbook
- ❑ Add the new cookbook to version control

GL: Copy the Recipe into the Cookbook

```
$ mv setup.rb cookbooks/workstation/recipes
```

GL: Verify the Cookbook has the Recipe

```
$ tree cookbooks/workstation
```

```
cookbooks/workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   ├── default.rb
│   └── setup.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
```

Group Exercise: Version Control

This is a probably a good point to capture the initial state of our cookbook.

Objective:

- ✓ Use chef to generate a cookbook
- ✓ Move the setup recipe into the new cookbook
- ❑ Add the new cookbook to version control

GL: Move into the Cookbook Directory

```
$ cd cookbooks/workstation
```

GL: Initialize the Directory as a git Repository

```
$ git init
```

```
Reinitialized existing Git repository in /home/chef/cookbooks/workstation/.git/
```

GL: Use 'git add' to Stage Files to be Committed

```
$ git add .
```

Staging Area

The staging area has a file, generally contained in your Git directory, that stores information about what will go into your next commit.

It's sometimes referred to as the "index", but it's also common to refer to it as the staging area.

http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

GL: Use 'git status' to View the Staged Files

```
$ git status
```

```
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   .kitchen.yml
        new file:   Berksfile
        new file:   README.md
        new file:   chefignore
        new file:   metadata.rb
```

GL: Use 'git commit' to Save the Staged Changes

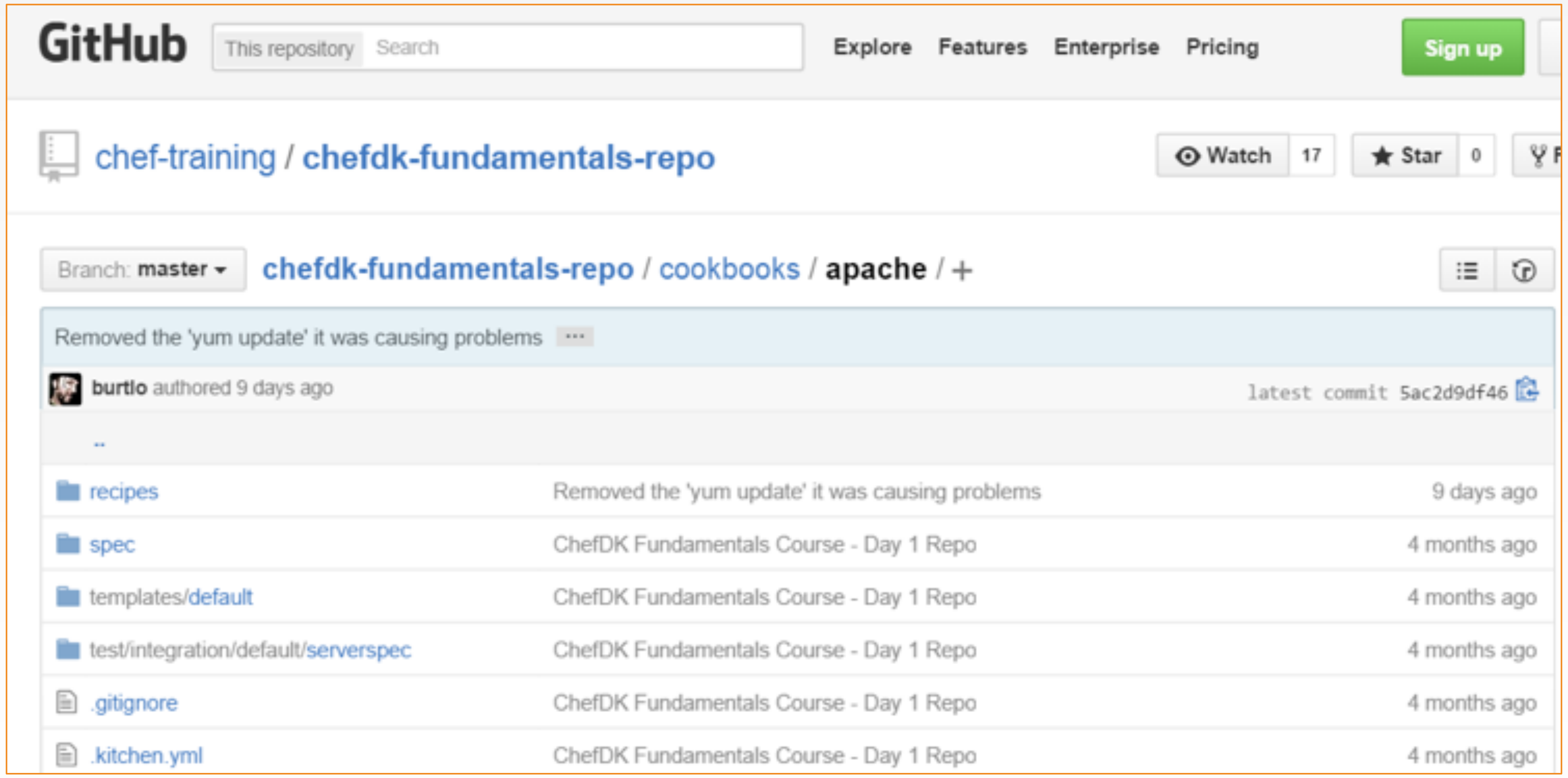
```
$ git commit -m "Initial commit"
```

```
master (root-commit) 9998472] Initial workstation cookbook
 Committer: ChefDK User <chef@ip-172-31-59-191.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author
```

Git Version Control

If you use git versioning you should ultimately push the local git repository to a shared remote git repository.

In this way others could collaborate with you from a centralized location.



chef-training / chefdk-fundamentals-repo

Branch: master

chefdk-fundamentals-repo / cookbooks / apache / +

Removed the 'yum update' it was causing problems

burtlo authored 9 days ago — latest commit 5ac2d9df46

..

recipes	Removed the 'yum update' it was causing problems	9 days ago
spec	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
templates/default	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
test/integration/default/serverspec	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
.gitignore	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
.kitchen.yml	ChefDK Fundamentals Course - Day 1 Repo	4 months ago

GL: Return to the Home Directory

```
$ cd ~
```

Lab: Setting up a Web Server

- ❑ Use `chef generate` to create a cookbook named "apache".
- ❑ Write and apply a recipe named **`"server.rb"`** with the policy:

  The package named 'httpd' is installed.

  The file named '/var/www/html/index.html' is created with the content '<h1>Hello, world!</h1>'

  The service named 'httpd' is started and enabled.
- ❑ Apply the recipe with chef-client
- ❑ Verify the site is available by running **`curl localhost`**

Lab: Create a Cookbook

```
$ chef generate cookbook cookbooks/apache
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[/home/chef/cookbooks/apache] action create
    - create new directory /home/chef/cookbooks/apache
  * template[/home/chef/cookbooks/apache/metadata.rb] action create_if_missing
    - create new file /home/chef/cookbooks/apache/metadata.rb
    - update content in file /home/chef/cookbooks/apache/metadata.rb from none to
37ed5f
    (diff output suppressed by config)
  * template[/home/chef/cookbooks/apache/README.md] action create_if_missing
    - create new file /home/chef/cookbooks/apache/README.md
    - update content in file /home/chef/cookbooks/apache/README.md from none to
5c3d3a
    (diff output suppressed by config)
```

Lab: Create a Cookbook

```
$ chef generate recipe cookbooks/apache server
```

```
Compiling Cookbooks...
Recipe: code_generator::recipe
  * directory[cookbooks/apache/spec/unit/recipes] action create (up to date)
  * cookbook_file[cookbooks/apache/spec/spec_helper.rb] action create_if_missing (up
to date)
  * template[cookbooks/apache/spec/unit/recipes/server_spec.rb] action
create_if_missing
    - create new file cookbooks/apache/spec/unit/recipes/server_spec.rb
    - update content in file cookbooks/apache/spec/unit/recipes/server_spec.rb from
none to a43970
    (diff output suppressed by config)
  * template[cookbooks/apache/recipes/server.rb] action create
    - create new file cookbooks/apache/recipes/server.rb
    - update content in file cookbooks/apache/recipes/server.rb from none to 3d6b92
    (diff output suppressed by config)
```

Lab: Create the Server Recipe

```
~/cookbooks/apache/recipes/server.rb
```

```
package 'httpd'

file '/var/www/html/index.html' do
  content '<h1>Hello, world!</h1>'
end

service 'httpd' do
  action [ :enable, :start ]
end
```

Lab: Apply the Server Recipe

```
$ sudo chef-client -z cookbooks/apache/recipes/server.rb
```

```
Converging 3 resources
Recipe: @recipe_files::/home/chef/cookbooks/apache/recipes/server.rb
  * yum_package[httpd] action install
    - install version 2.2.15-47.el6.centos.3 of package httpd
  * file[/var/www/html/index.html] action create
    - create new file /var/www/html/index.html
    - update content in file /var/www/html/index.html from none to 17d291
    --- /var/www/html/index.html    2016-02-24 21:41:45.494844958 +0000
    +++ /var/www/html/.index.html20160224-10036-6y8on7    2016-02-24
21:41:45.493844958 +0000
    @@ -1 +1,2 @@
    +<h1>Hello, world!</h1>
  * service[httpd] action enable
    - enable service service[httpd]
```

Lab: Verify That the Website is Available

```
$ curl localhost
```

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

Lab: Setting up a Web Server

- ✓ Use `chef generate` to create a cookbook named "apache".
- ✓ Write and apply a recipe named **`"server.rb"`** with the policy:

  The package named 'httpd' is installed.

  The file named '/var/www/html/index.html' is created with the content '<h1>Hello, world!</h1>'

  The service named 'httpd' is started and enabled.

- ✓ Apply the recipe with chef-client
- ✓ Verify the site is available by running **`curl localhost`**

GL: Commit Your Work

```
$ cd cookbooks/apache
$ git init
$ git add .
$ git commit -m "Initial commit"
```

Discussion

What file would you read first when examining a cookbook?

What other recipes might you include in the apache or workstation cookbook?

Can resources accept multiple actions?

How often would you commit changes with version control?

CHEF™