Cookbooks

Organizing Recipes

CHEF

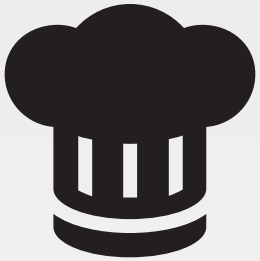
Objectives

After completing this module, you should be able to:

- Set up a repo as a working folder
- Modify a recipe
- Discuss version control
- Generate a Chef cookbook
- Define a Chef recipe that sets up a web server

Repository

A repository is the working folder on your Workstation where you keep all your Chef files.



Lab: Create a Repository

Lets create our workspace

Objective:

- Use chef to generate a repo that will act as our workspace on the workstation

CONCEPT

What is 'chef' command?

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

What can 'chef' do?

```
$ chef --help
```

```
Usage:
    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
```

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

What Can 'chef generate repo' Do?

```
$ 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'
    -a, --generator-arg KEY=VALUE    Use to set arbitrary attribute KEY to VALUE in the
code_generator cookbook
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 - defaults to
all_rights
    -P, --policy                     Use policyfiles instead of Berkshelf
    -p, --policy-only                Create a repository for policy only, not cookbooks
    -r, --roles                      Create roles and environments directories instead of
using policyfiles
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the code_generator
cookbook
        --generator-cookbook
```

Lab: Create your repo

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

```
...
    - create new file /home/chef/chef-repo/chef-repo/cookbooks/README.md
    - update content in file /home/chef/chef-repo/chef-
repo/cookbooks/README.md from none to 86e9ef
    (diff output suppressed by config)
  * execute[initialize-git] action run
    - execute git init .
  * template[/home/chef/chef-repo/chef-repo/.gitignore] action
create_if_missing
    - create new file /home/chef/chef-repo/chef-repo/.gitignore
    - update content in file /home/chef/chef-repo/chef-repo/.gitignore from
none to 3523c4
    (diff output suppressed by config)
```

Lab: Navigate to the chef-repo directory

```
$ cd chef-repo
```

Navigate to the chef-repo directory. You will perform most the tasks in this class from this directory

Lab: View repo structure

```
$ tree
├── chefignore
├── cookbooks
│   ├── example
│   │   ├── attributes
│   │   │   └── default.rb
│   │   ├── metadata.rb
│   │   ├── README.md
│   │   └── recipes
│   │       └── default.rb
│   └── README.md
├── data_bags
│   ├── example
...
```

Lab: View repo structure

```
$ tree
├── chefignore
├── cookbooks
│   ├── example
│   │   ├── attributes
│   │   │   └── default.rb
│   │   ├── metadata.rb
│   │   ├── README.md
│   │   └── recipes
│   │       └── default.rb
│   └── README.md
├── data_bags
│   ├── example
...
```

Cookbooks reside in the 'cookbooks' directory

CHEF

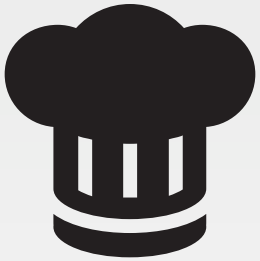
Cookbooks

A cookbook is a container for recipes, and supporting files

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.

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



Lab: Create a Cookbook

How are we going to manage this 'setup.rb' file? Does it need a README?

Objective:

- Use chef to generate a cookbook to store our setup recipe.

What Can 'chef generate' Do?

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

```
UsaLab: 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
```

Lab: Let's Create a Cookbook

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

```
...
- create new file /home/chef/chef-
repo/cookbooks/workstation/recipes/default.rb
    - update content in file /home/chef/chef-
repo/cookbooks/workstation/recipes/default.rb from none to 0d310d
    (diff output suppressed by config)
  * cookbook_file[/home/chef/chef-repo/cookbooks/workstation/.gitignore]
action create
    - create new file /home/chef/chef-repo/cookbooks/workstation/.gitignore
    - update content in file /home/chef/chef-
repo/cookbooks/workstation/.gitignore from none to dd37b2
    (diff output suppressed by config)
```

The Cookbook Has a README

```
$ tree cookbooks/workstation
```

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

CONCEPT

README.md

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

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

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
```

DOCS

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

Lab: 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'
```


CHEF

Lab: The Cookbook Has a Folder for Recipes

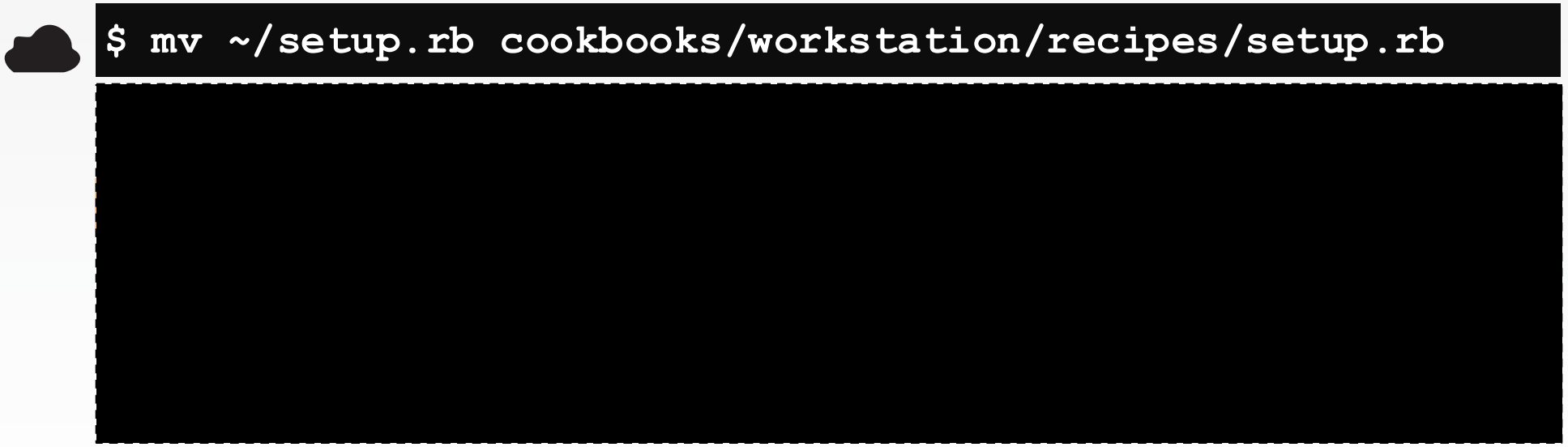
```
$ tree cookbooks/workstation
```

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

Lab: The Cookbook Has a Default Recipe

```
$ cat cookbooks/workstation/recipes/default.rb
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.
```

Lab: Copy the Recipe into the Cookbook



CONCEPT

chef-client

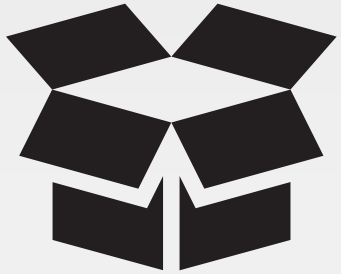
Up until now we have ran chef-client on individual recipe .rb files at the command line.

But what if we need to run multiple recipes – we cannot add multiple paths/to/.rb/files

https://docs.chef.io/chef_client.html

CONCEPT

-r "recipe[COOKBOOK::RECIPE]"



In local mode, we need to provide a list of recipes to apply to the system. This is called a **run list**. A run list is an **ordered** collection of recipes to execute

Each recipe in the run list must be addressed with the format **recipe[COOKBOOK::RECIPE]**

To use this format the cookbooks must be under a directory called 'cookbooks'

Using 'chef-client' to Apply Recipes

```
$ sudo chef-client --local-mode -r "recipe[cookbook::recipe]"
```

Use chef-client with '-r' flag to specify the recipe run as 'cookbook::recipe' instead of path/to/recipe.rb

Lab: Using 'chef-client' to Locally Apply Recipes

```
$ sudo chef-client --local-mode -r "recipe[workstation::setup]"
```

Applying the following recipes locally:

- The 'setup' recipe from the 'workstation' cookbook

Using 'chef-client' to Locally Apply Recipes

```
$ sudo chef-client --local-mode \
  -r "recipe[cookbook1::default],recipe[cookbook2::default]"
```

Applying multiple recipes:

- Run the ’default' recipe from the cookbook ‘cookbook1'
- Then if it completes successfully
- Run the ’default' recipe from the cookbook ‘cookbook2'

Lab: Return Home First

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

Lab: Apply the ’workstation::setup' Recipe

```
$ sudo chef-client -z -r "recipe[workstation::setup]"
```

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-04-07T14:34:55+00:00] WARN: Node ip-172-31-11-163.ec2.internal has an
empty run list.
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
```

Lab: Apply the 'workstation::setup' Recipe

```
$ sudo chef-client -z -r "recipe[workstation::setup]"
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-04-07T14:34:55+00:00] WARN: Node ip-172-31-11-163.ec2.internal has an
empty run list.
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
```

Note the flag '-z' can be used instead of '--local-mode'

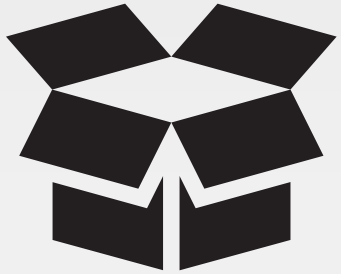
Lab: Apply two recipes

```
$ sudo chef-client -z -r
"recipe[workstation::default],recipe[workstation::setup]"
```

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: ["workstation::default", "workstation::setup"]
Synchronizing Cookbooks:
  - workstation (0.0.0)
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/2 resources updated in 08 seconds
```

CONCEPT



-r "recipe[COOKBOOK(::default)]"

When you are referencing the default recipe within a cookbook you may optionally specify only the name of the cookbook.

chef-client understands that you mean to apply the default recipe from within that cookbook.

Using 'chef-client' to apply default recipe

```
$ sudo chef-client --local-mode -r "recipe[cookbook]"
```

Is equivalent to

```
$ sudo chef-client --local-mode -r "recipe[cookbook::default]"
```

When the recipe is specified as "recipe[cookbook]", chef-client assumes the default recipe, default.rb

DOCS

include_recipe

A recipe can include one (or more) recipes located in cookbooks by using the **include_recipe** method. When a recipe is included, the resources found in that recipe will be inserted (in the same exact order) at the point where the include_recipe keyword is located.

https://docs.chef.io/recipes.html#include-recipes

Demo: Including a Recipe

```
include_recipe 'workstation::setup'
```

Include the 'setup' recipe from the 'workstation' cookbook in this recipe

Lab: The Default Recipe Includes the Setup Recipe

cookbooks/workstation/recipes/default.rb

```
#
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

include_recipe 'workstation::setup'
```

Lab: Apply the Cookbook's Default Recipe

```
$ sudo chef-client -zr "recipe[workstation]"
```

```
 WARN: No config file found or specified on command line, using command line
options.
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
Converging 0 resources

Running handlers:
Running handlers complete
Chef Client finished, 0/0 resources updated in 3.300489827 seconds
```

Note the flags '-z –r' can be combined into '-zr'!

CHEF

Discussion

Why would you want to apply more than one recipe at a time?

What file would you read first when examining a cookbook?

What are the benefits and drawbacks of using "include_recipe" within a recipe?

Do default values make it easier or harder to learn?

Q&A

What questions can we help you answer?

- chef-client
- local mode
- cookbooks
- run list
- include_recipe

CHEF™