# 4: chef-client



Slide 2



In this module you will learn how to use the 'chef-client' command to apply recipes, and include a recipe within another recipe.

Slide 3



'chef-apply' is valuable tool for exploring resources within recipes without having to wrestle with all the folders and files associated with cookbooks. For the remainder of the modules we will not return to using 'chef-apply'. In the future you will most likely be using 'chef-client'. You may return to 'chef-apply' in your adventures when you find yourself wanting to test out an idea for a new recipe on a new platform or platform version. The speed of the tool is valuable.

Slide 4



In the ChefDK, we package another tool that is called 'chef-client'.

'chef-client' is a command-line application that can be used to apply a recipe or multiple recipes. It also has the ability to communicate with a Chef server – a concept we will talk about in another section. For now think of the Chef Server as a central, artifact repository where we will later store our cookbooks.

Slide 5



Here is an example of using 'chef-client' to locally apply a run list of recipes. In this case we are applying one recipe and that is the setup recipe within our workstation cookbook.

Instructor Note: These commands if executed by a learner at this point will not work. These are being displayed solely as demonstration.

Slide 6



Here is an example of using 'chef-client' to locally apply the server recipe within our apache cookbook.

Slide 7



Here is an example of using 'chef-client' to locally apply two recipes -- the setup recipe from the workstation cookbook and the server recipe within our apache cookbook.

Instructor Note: The command given here includes the backslash '\'. That allows you to specify multiple lines within a terminal. Because of the character limitation of slides it is included to make the command more clear to the learner.

Instructor Note: It is important to note that when specifying a run list, recipes defined within it that are separated with a comma should NOT have a space after the comma or it will create an error.

Slide 8



Applying recipes with 'chef-client' is different than 'chef-apply' and that is because chef-client's default behavior is to communicate with a Chef server. So we use the '--local-mode' flag to ask 'chef-client' to look for the cookbooks locally.

Slide 9



When we apply a recipe with 'chef-client', we define a run list. This is an ordered list of recipes that we want to apply to the system. When you define a recipe from a cookbook on the run list, there is a particular convention:

"recipe[COOKBOOK::RECIPE]"

COOKBOOK means the name of the Cookbook.

RECIPE means the name of the Recipe without the Ruby file extension.

Slide 10



Before you start applying cookbooks through 'chef-client', make sure you are in your home directory.

Slide 11



Try applying our server recipe from the apache cookbook using `chef-client` in local mode.

Upon execution you unfortunately are presented with an error.

When executed we find that `chef-client` has an additional requirement. `chef-client` expects our cookbooks to be maintained in a directory named 'cookbooks'.

That seems simple enough to accommodate and a good way to start organizing the cookbooks that we are creating.

Instructor Note: This is supposed to fail. chef-client requires the cookbooks to be in a cookbooks directory. The second warning message tells the user of the application that it was unable to find a cookbooks directory.

Instructor Note: The other warning about 'No config file found or specified on command line, using command line options' is looking for a config file at a default location, which we have not created nor we will create one at this time. There is a flag '-c' that allows you to specify a configuration file as well. But again specifying the configuration file will be automatically when the instance is bootstrapped.

Slide 12



Make a directory named 'cookbooks'. Then move the workstation cookbook and apache cookbook into the cookbooks directory.

Slide 13



Let’s try that again--this time with all of our cookbooks in the cookbooks directory like `chef-client` expects.

Try applying the apache cookbook's recipe named server.

Instructor Note: The WARN messages were omitted from this output so you can see the converging resources.

Slide 14



Try applying the workstation cookbook's recipe named 'setup'.

Slide 15



Try applying both recipes from both cookbooks again at one time.

Instructor Note: It is important to note that when specifying a run list, recipes defined within it that are separated with a comma should NOT have a space after the comma or it will create an error.

Slide 16



Actually, we didn't tell you everything about specifying the run list for the `chef-client` command.

When defining a recipe in the run list you may omit the name of the recipe, and only use the cookbook name, when that recipe's name is 'default'.

Similar to how resources have default actions and default attributes Chef uses the concept of providing sane defaults. This makes our faster when we understand the concepts.

A cookbook doesn't have to have a default recipe but most every cookbook has one. It's called default because when you think of a cookbook it is the recipe that defines the most common configuration policy.

When you think about the two cookbooks that we created -- the apache cookbook with the server recipe and the workstation cookbook with the setup recipe -- it seems like those recipes would be good default recipes for their respective cookbooks.

Slide 17



A simple solution would be to rename the setup recipe to the default recipe. However, a better practice would instead leave our recipes as they are and have the default recipe include the recipes with a method called `include\_recipe`

This allows us to maintain all the current policies within its own recipe file and that way we can more easily switch our cookbooks default behavior, which can be useful when new requirements surface.

Slide 18



In this example we are including the 'workstation' cookbook's 'setup' recipe.

Slide 19



In this example, we are including the 'apache' cookbook's 'server' recipe.

Slide 20



We are interested in having the default recipe for our workstation cookbook run the contents of the setup recipe.

Within the default recipe, define the `include\_recipe` method and provide one parameter, which is the name of our recipe as it appears within a run list: cookbook\_name::recipe\_name.

Slide 21



Use 'chef-client' to locally apply the cookbook named workstation. This will load your workstation cookbook's default recipe, which in turn loads the workstation cookbook's setup recipe.

Slide 22



With everything working it is time to commit the latest changes.

Slide 23



In this lab you will update the apache cookbook's default recipe to include the apache cookbook's recipe named server.

Instructor Note: Allow 5 minutes to complete this exercise.

Slide 24



We are interested in having the default recipe for our apache cookbook run the contents of the server recipe.

Within the default recipe, define the `include\_recipe` method and provide one parameter, which is the name of our recipe as it appears within a run list: cookbook\_name::recipe\_name.

Slide 25



Use 'chef-client' to locally apply the cookbook named apache. This will load your apache cookbook's default recipe, which in turn loads the apache cookbook's server recipe.

Slide 26



With everything working it is time to commit the latest changes.

Slide 27



Answer these questions.

With your answers, turn to another person and alternate asking each other asking these questions and sharing your answers.

Slide 28



What questions can we help you answer?

Generally or specifically about chef-client, local mode, run lists, and include\_recipe.

Slide 29

