Testing While Refactoring to Attributes

CHEF

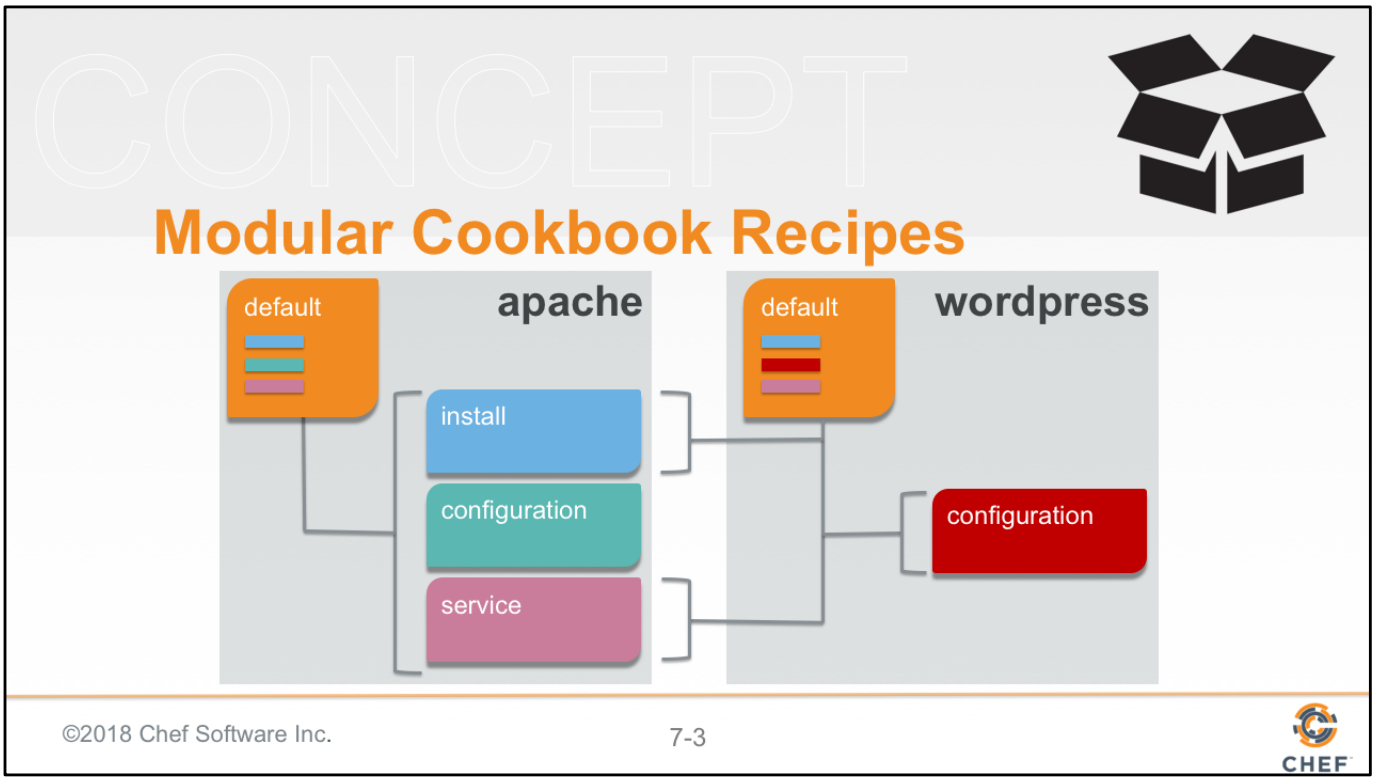
We now have the fastest feedback open source software can buy us! And right on time because it is time to refactor the cookbook again.

Objectives

After completing this module, you should be able to:

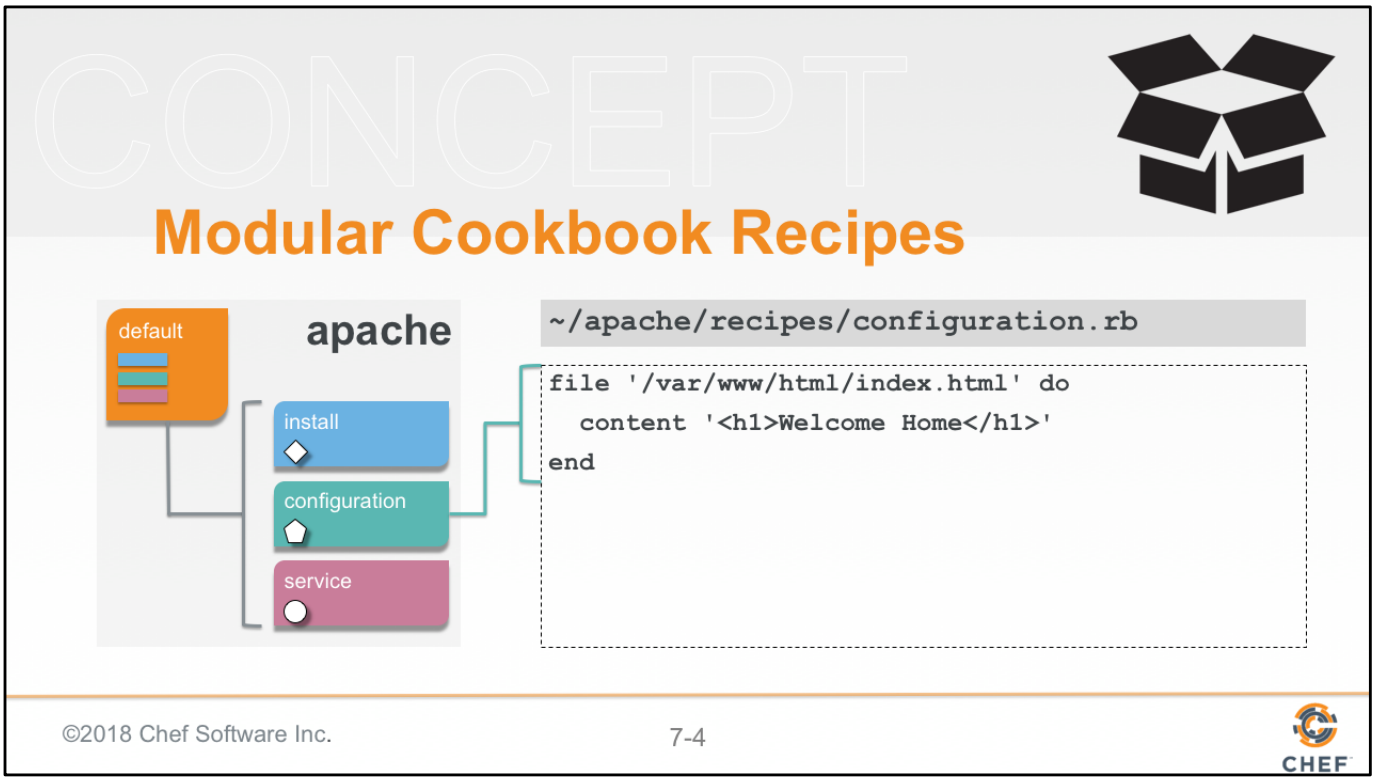
- Refactor resources to use attributes
- Use Pry to explore the current state of execution
- Make changes to your recipes with confidence

In this module you will learn how to refactor a cookbook to use node attributes, employ pry to set up break points in your code, and make changes with confidence.



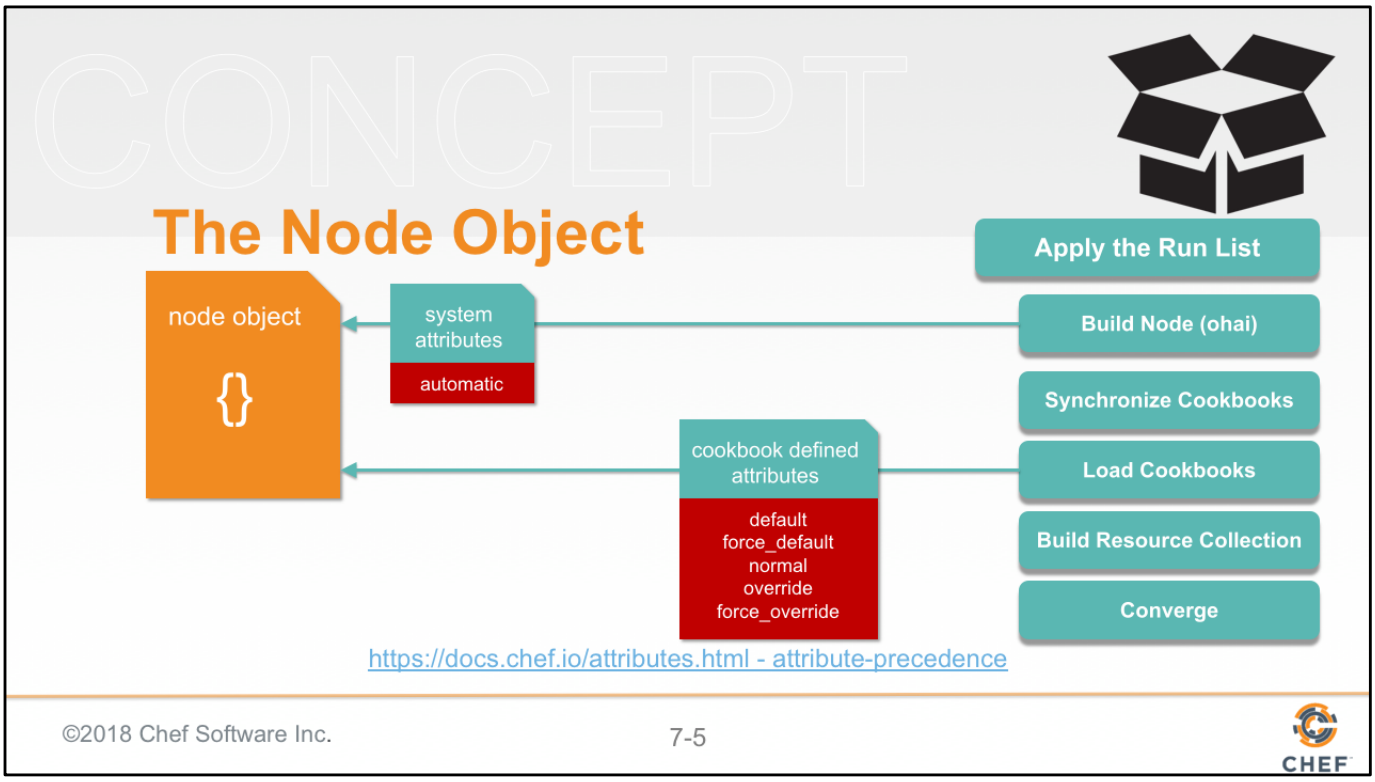
When we initially set out to create a cookbook that was more modular we broke the concerns of the webserver into three different recipes. This would allow an opportunity for cookbook authors within our organization re-use components of the cookbook by including only the recipes that they want.

Sometimes you do not want to re-define an entire new recipe and simply want to provide a different name or version for the package; a single file path for the configuration file.



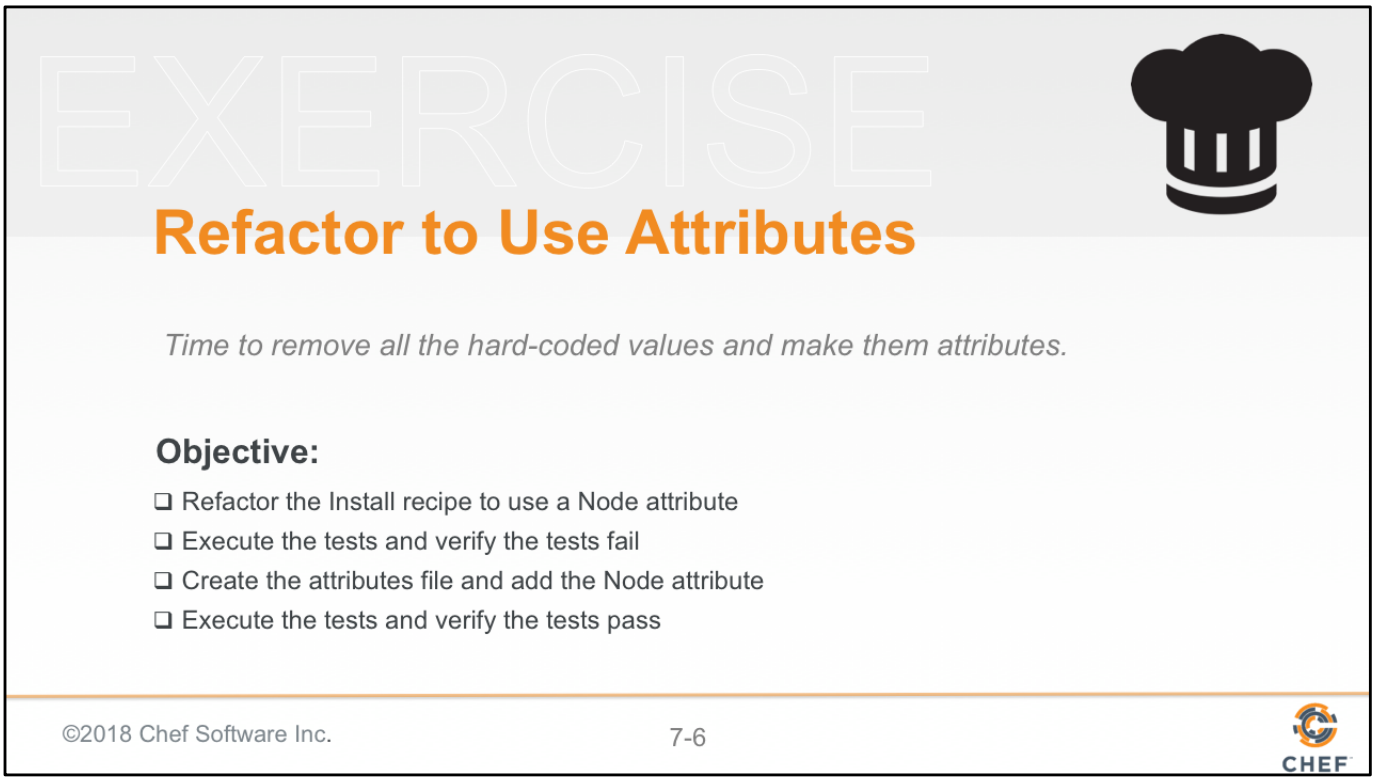
Within each recipe we defined the resources necessary to bring the webserver into the desired state. When we expressed these resources we did so with values that worked for this platform and version of the Operating System (OS).

The configuration recipe defined a file resource with a path to the location for the default HTML page. This path is hard-coded for this particular platform. If we had a situation where another cookbook or environment or role wanted to use this recipe but provide a custom value we could not do that unless we talk about making the file path a node attribute.



Cookbooks can define node attributes which are added to the node object after the initial discovery is done by Ohai. Ohai attributes are considered automatic and cannot be overwritten. However, the attributes defined in a cookbook can come in variety of levels. This allows for cookbooks to define a base value which another cookbook can replace when needed.

That is the kind of flexibility that we want to implement in our cookbook.



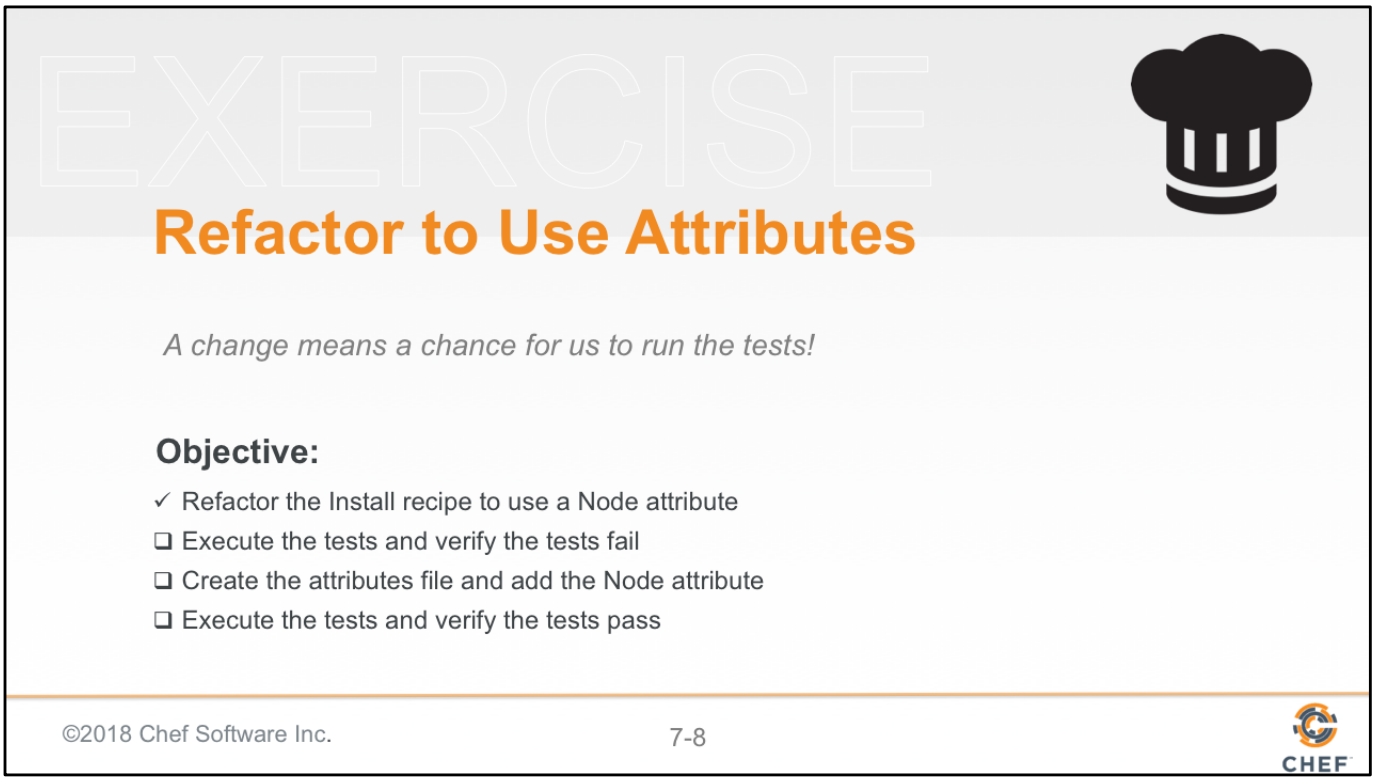
Together we will walk through refactoring the install recipe continuing to use our tests to prove that we have not caused a regression in recipes.

Replace the Value with a Node Attribute

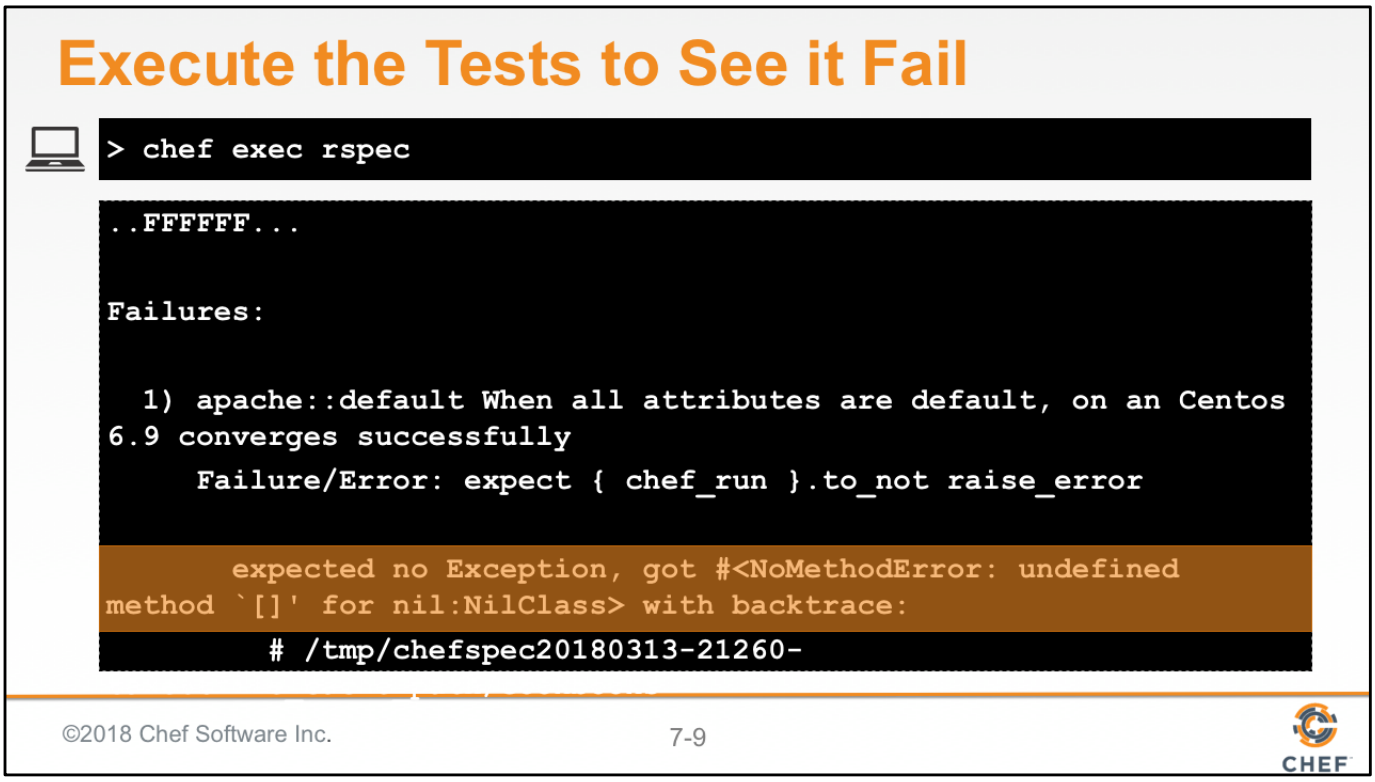
~/apache/recipes/install.rb

```
#
# Cookbook:: apache
# Recipe:: install
#
# Copyright:: 2018, The Authors, All Rights Reserved.
package node['apache']['package_name']
```

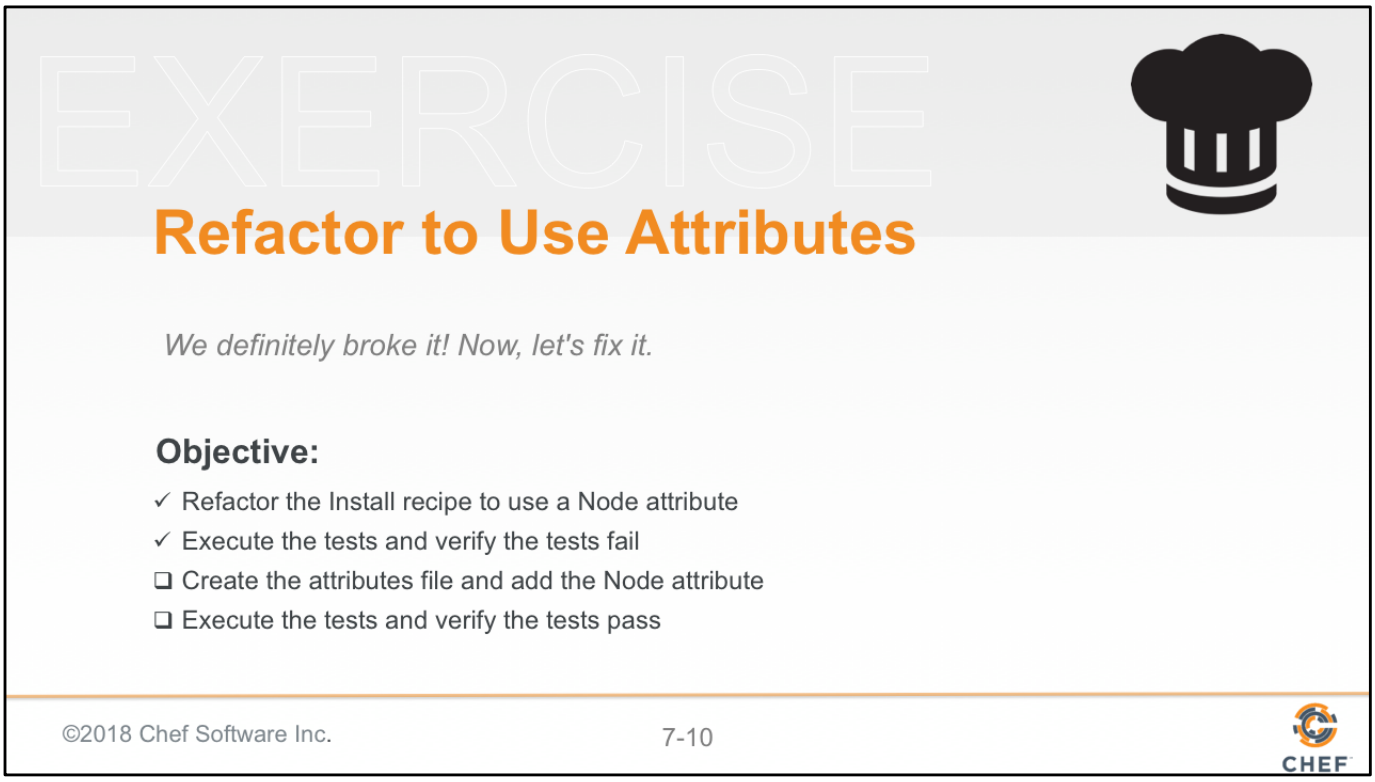

Because we have expectations in place we can start with a change to the install recipe. Here we are replacing the package name with a node attribute that we have yet to define in the attributes file.



We made a change. Before we define the node attribute we should run the tests.



When executing 'rspec' against all the examples that we have defined we see a large number of failures. The failure summary will show us that the chef run failed with an error. This error is informing us that we attempted to retrieve an attribute from the node object that does not exist. All of the failures should be the same.



Now it is time to create the attributes file and define the necessary attribute.

Ask Chef How to Generate an Attributes File

```
> chef generate attribute --help
```

```
Usage: chef generate attribute [path/to/cookbook] NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright hol...
    -m, --email EMAIL                Email address of the auth...
    -a, --generator-arg KEY=VALUE    Use to set arbitrary     ...
    -I, --license LICENSE            all_rights, apache2, mit,...
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PA...
        --generator-cookbook
```


The 'chef' tool is able to generate attributes. All it requires is the name of the file when you are inside the cookbook. We are currently inside the cookbook directory so now we need to give it a name.

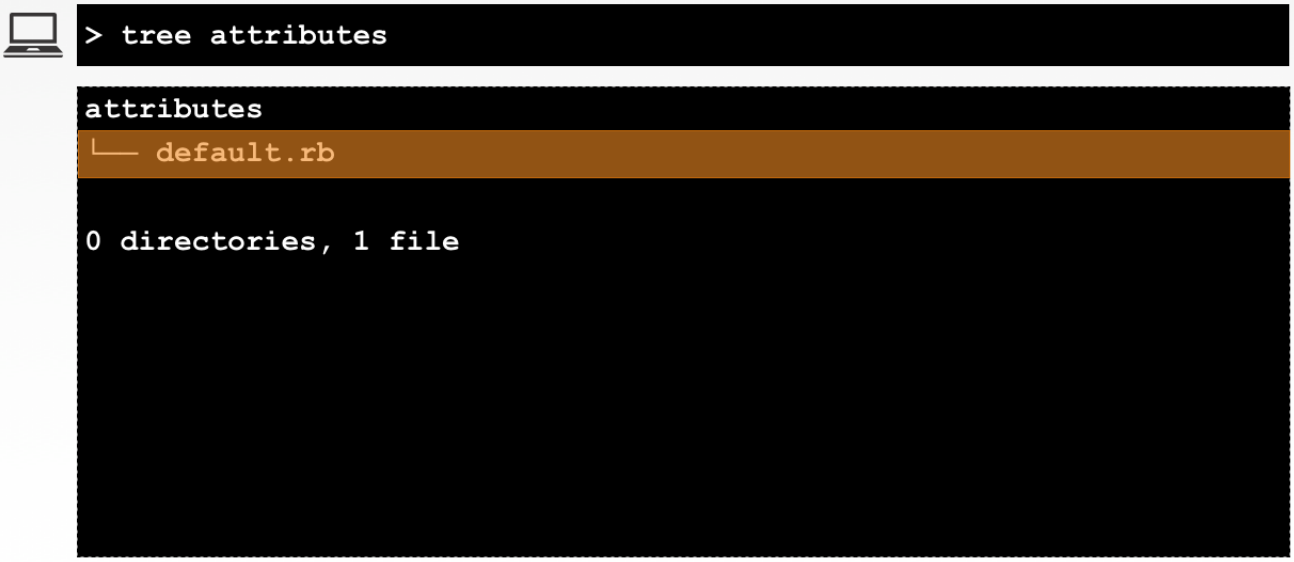
Use Chef to Generate a Default Attributes File

```
> chef generate attribute default
```

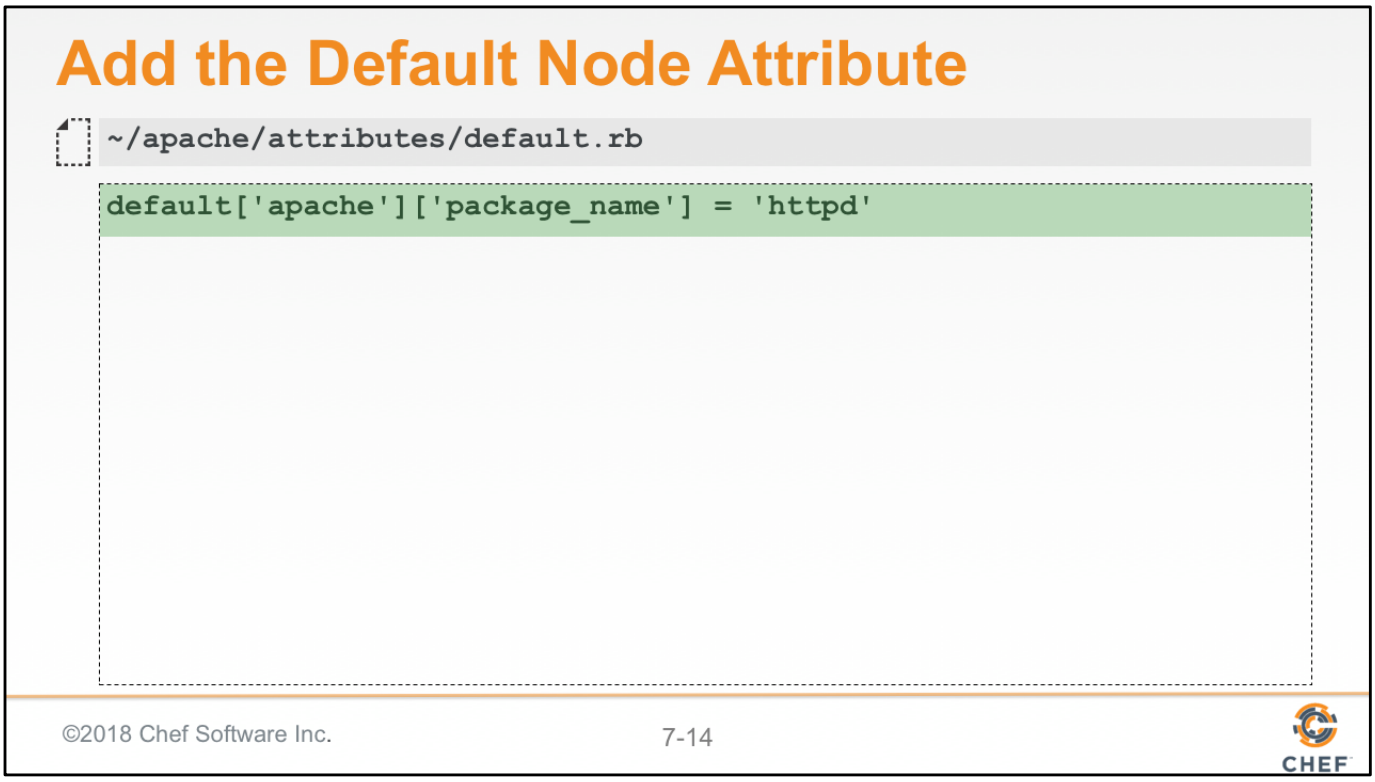
```
Compiling Cookbooks...
Recipe: code_generator::attribute
  * directory[/home/chef/apache/attributes] action create
    - create new directory /home/chef/apache/attributes
  * template[/home/chef/apache/attributes/default.rb] action
create
    - create new file /home/chef/apache/attributes/default.rb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


The standard name for the attributes file is 'default'. This command will generate an attributes file named 'default.rb' in the attributes directory.

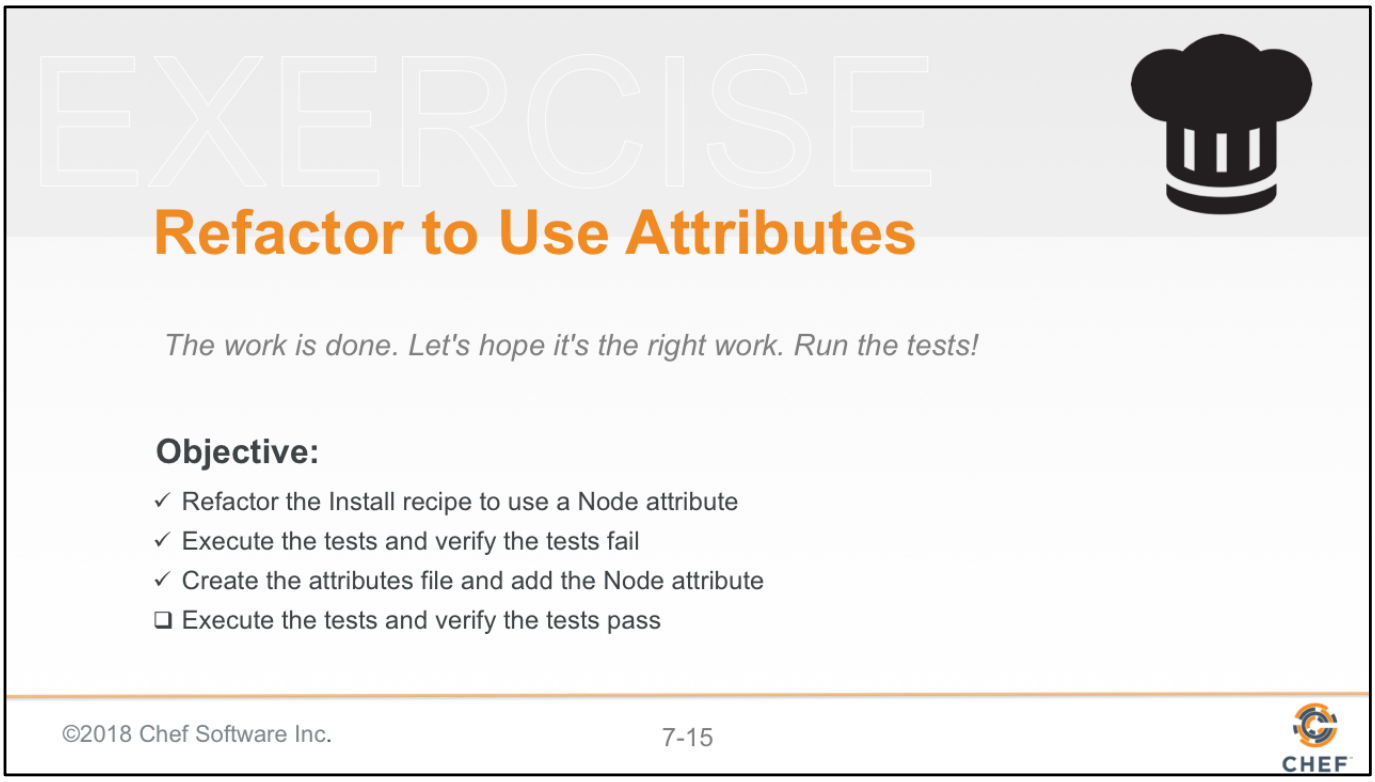
View the Attributes File Generated



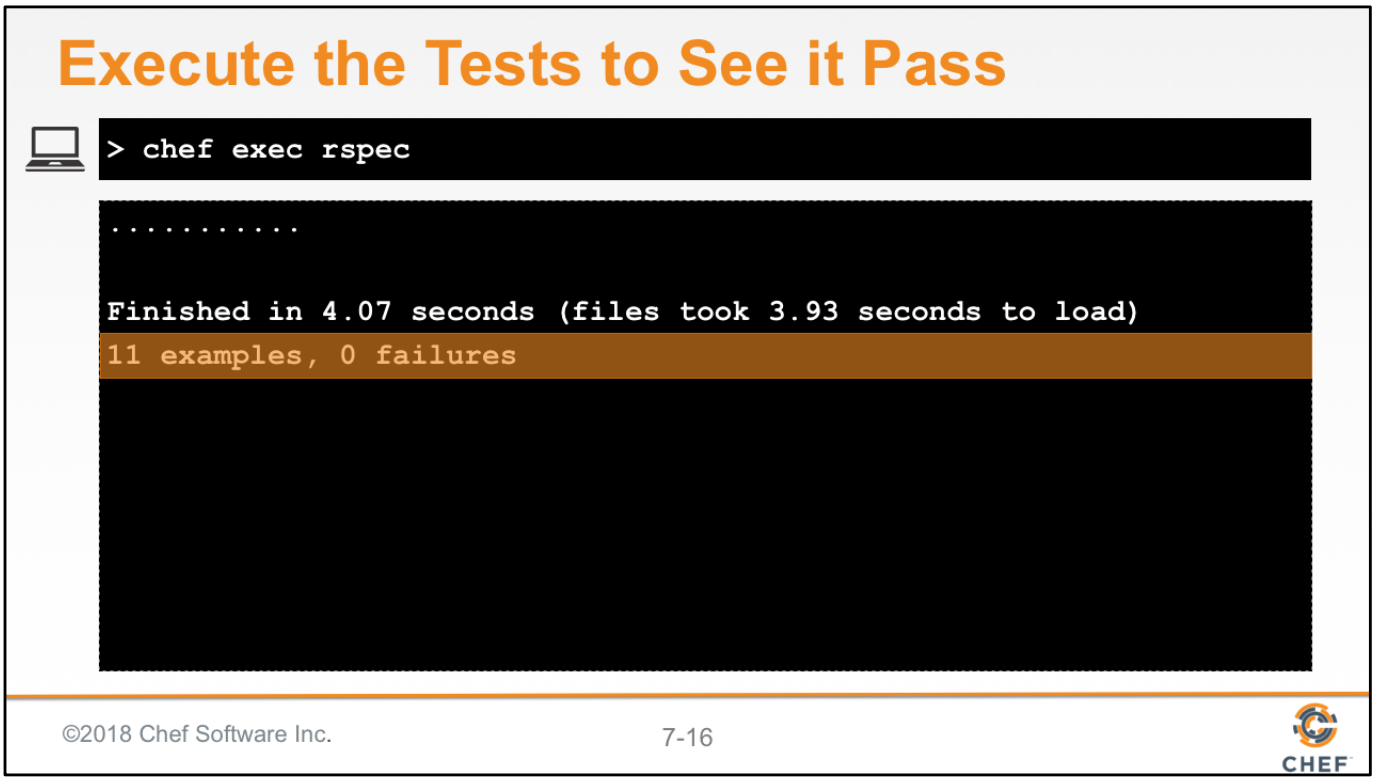
We can verify that by looking in the attributes directory to see the file has been generated.



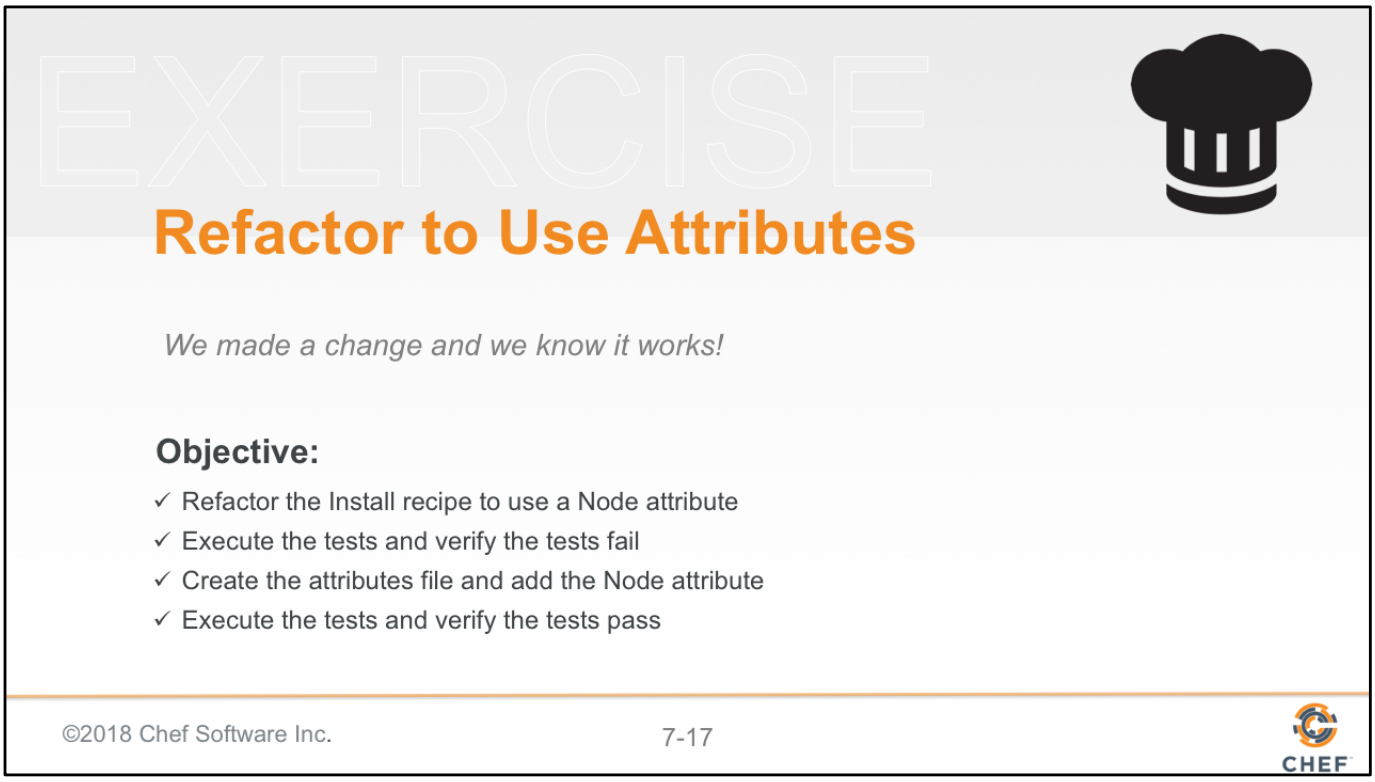
Now it is time to edit the attributes file and define the node attribute. Here we are defining the node attribute at the default level. Setting it to default will allow other cookbooks to override it if necessary.



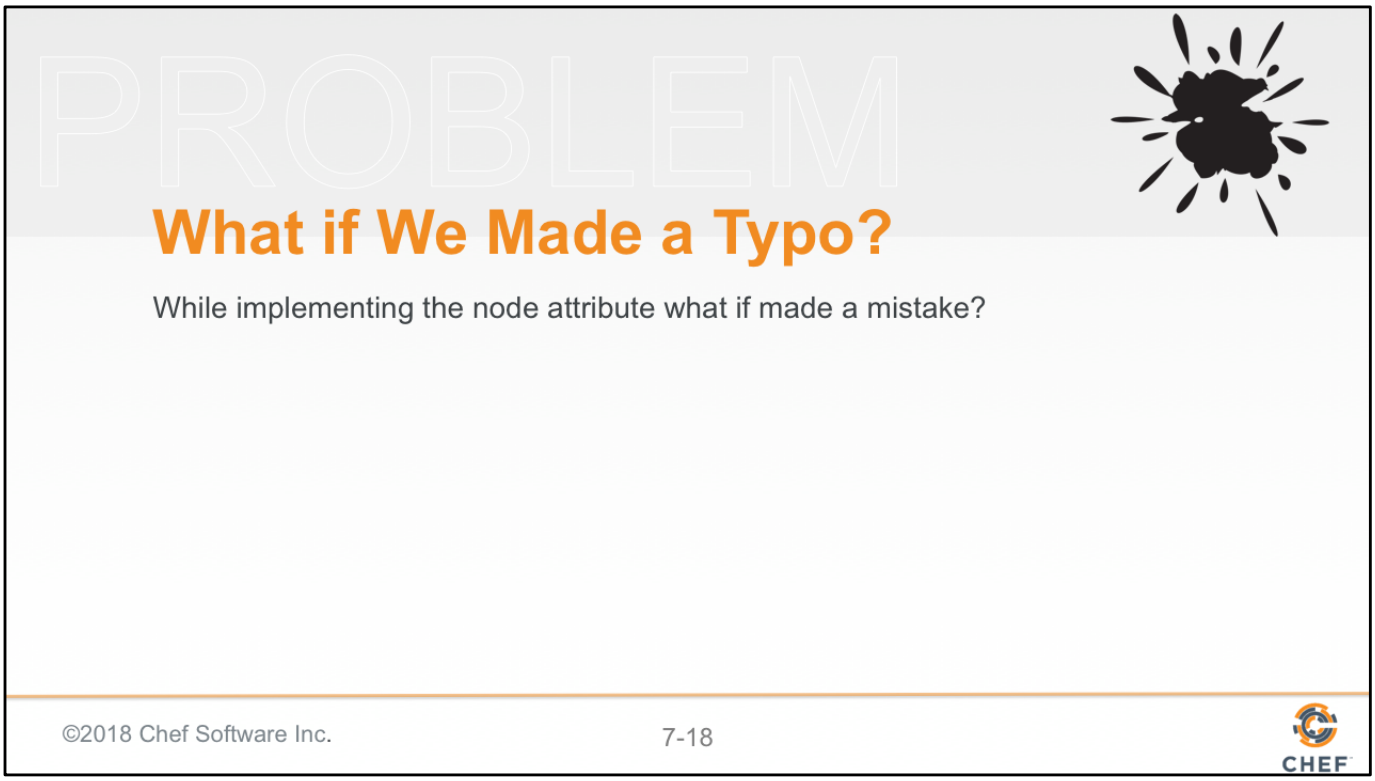
This change should fix all the examples that we broke when we used the node attribute without having defined it.



The results here show all the examples pass.

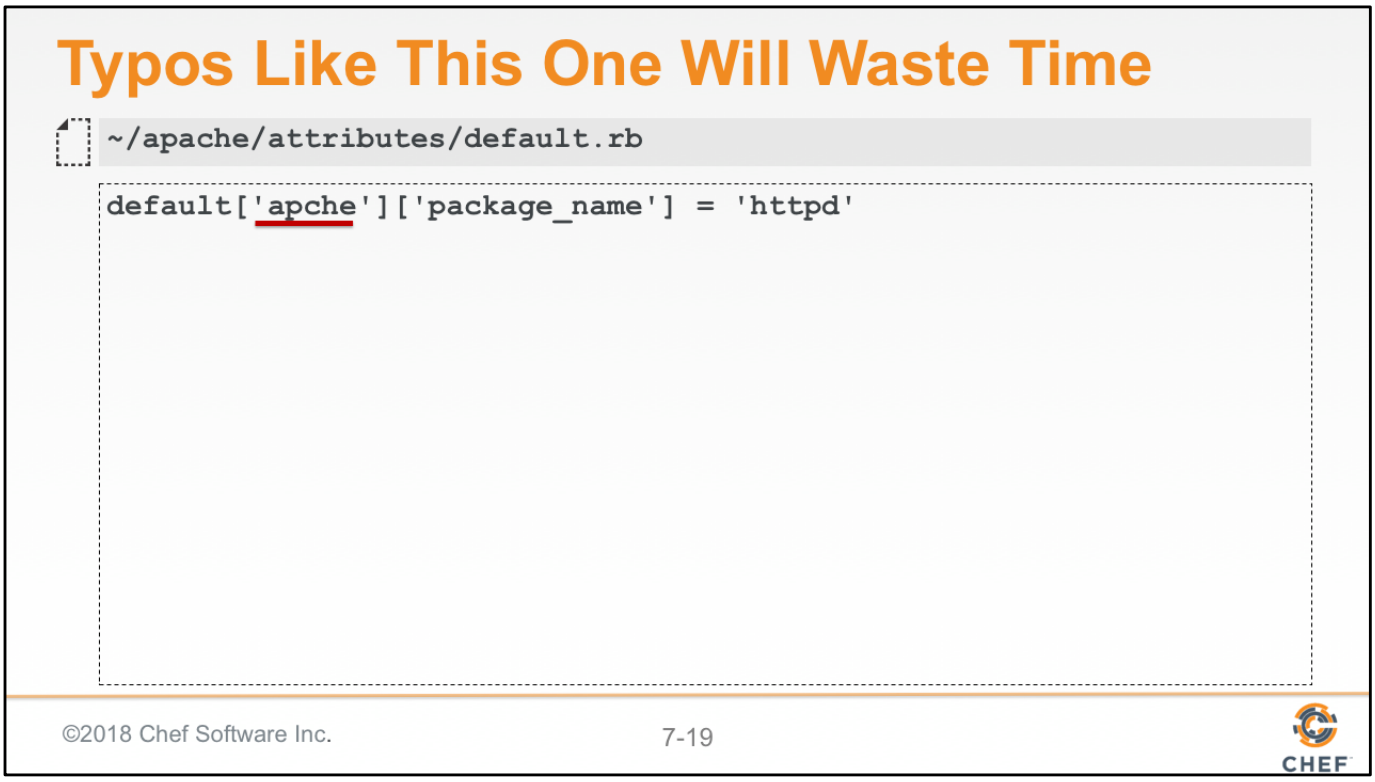


With all the expectations having been met we can confidently say that the cookbook has been refactored successfully.

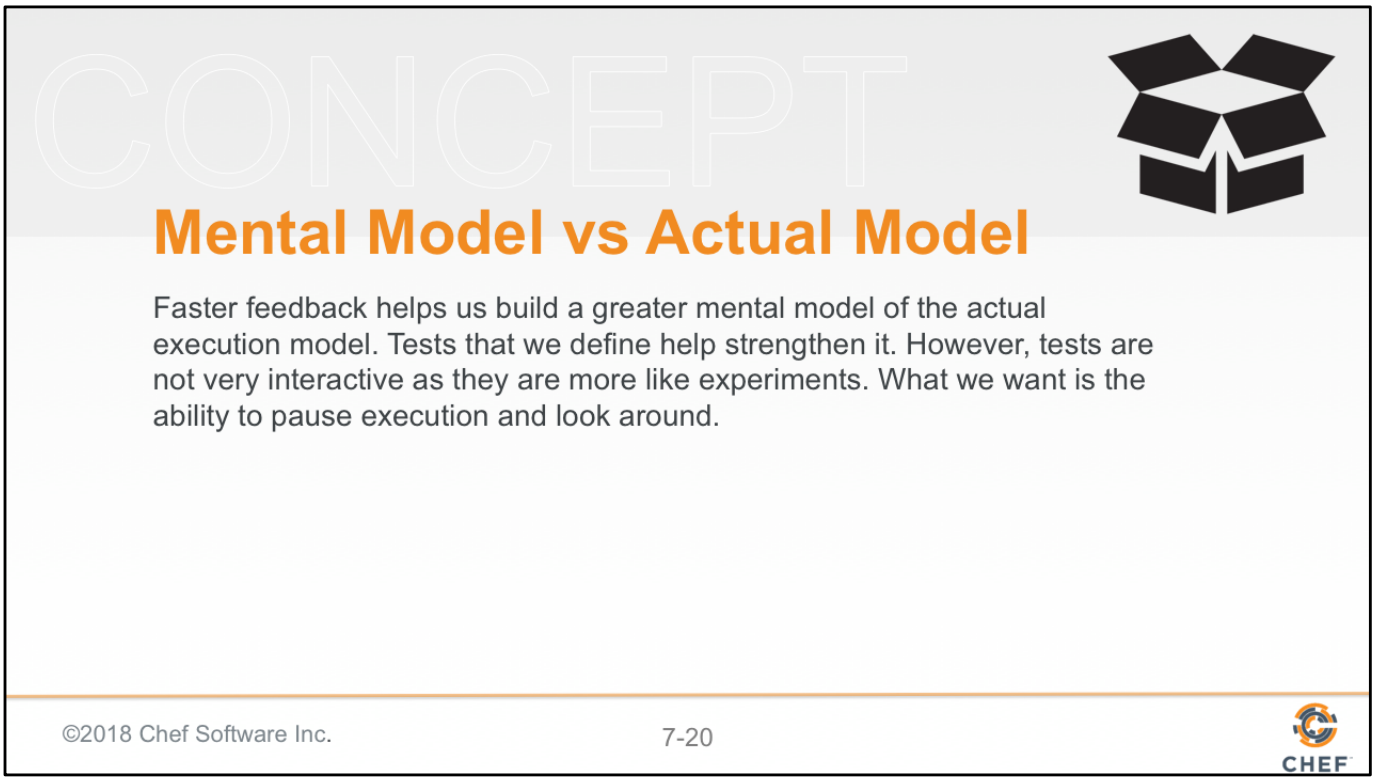


In the process of implementing the use of the node attribute in the recipe or in the attributes file we could have made a mistake. We proved that the examples would have caught the error.

What if an error occurred and we were unable to find it? Occasionally you will implement a change wrong and then find yourself staring at the failing expectations wondering what is wrong.

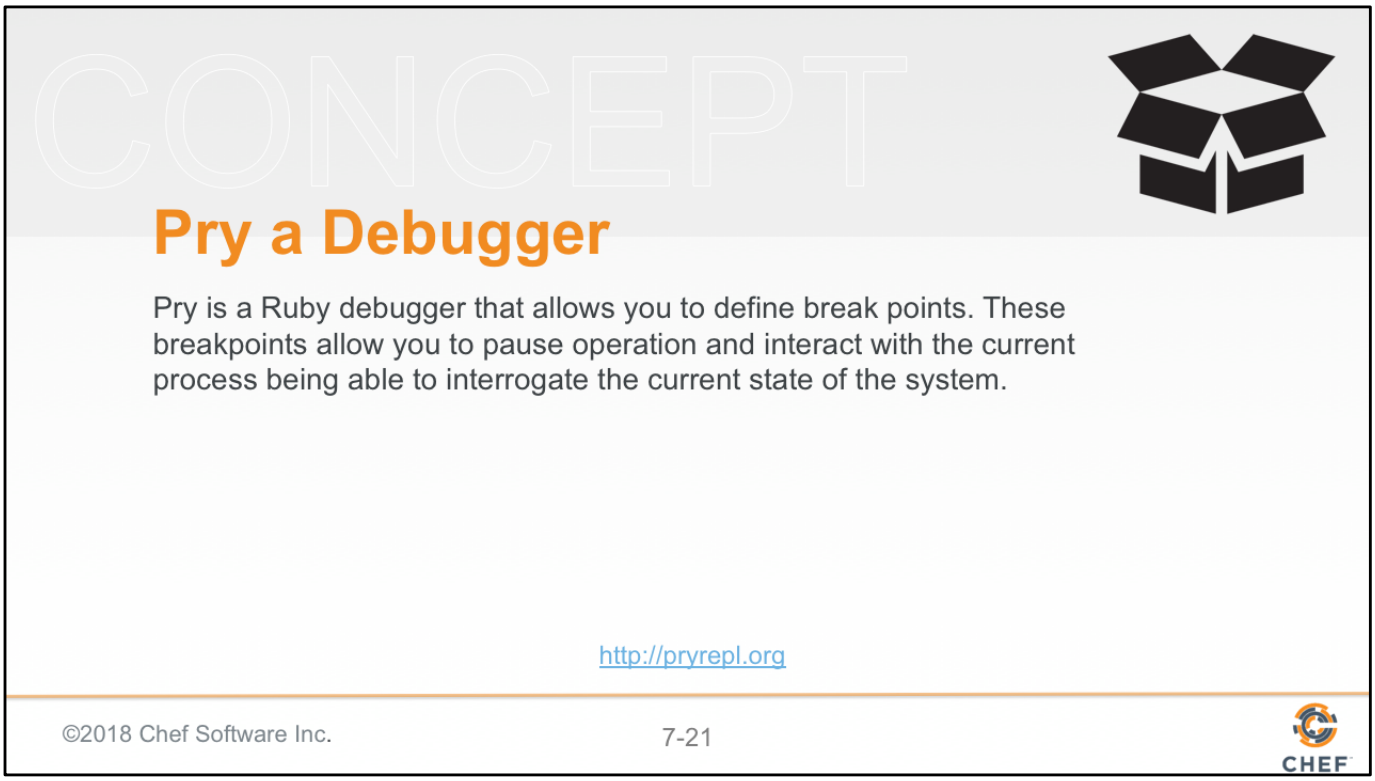


This is a simple typo that the examples would catch but when it comes time to find and fix the issue, our eyes may not immediately catch it. We may think the error lies somewhere in the recipe. If we cannot find it we keep running the tests and wondering what is going wrong.



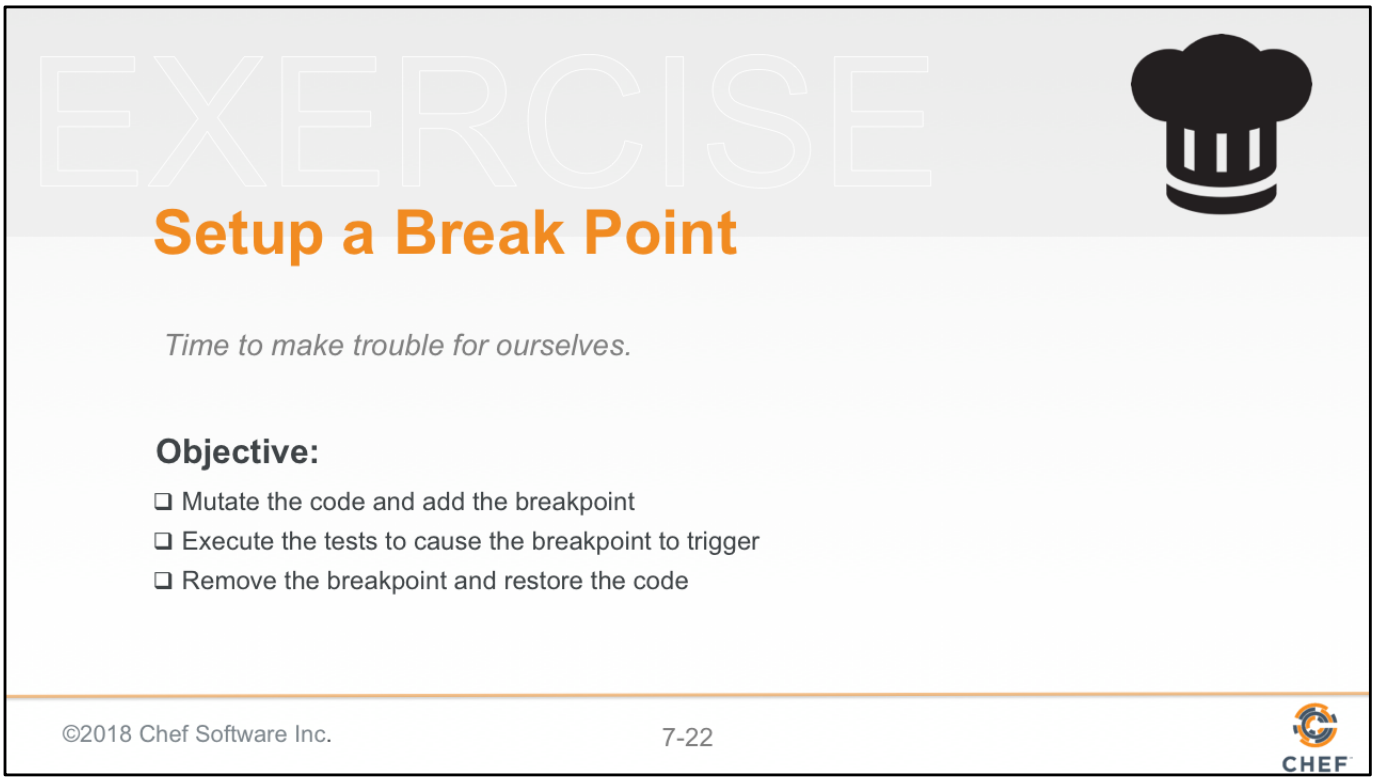
This is a situation where our mental model of the state of things is different than the actual model of execution. The benefit of tests is that it allows us to express the expectations about the model of how the execution should run. Testing is like a experiment: setup; execute; verify.

That feedback is not very interactive. There are moments where you want to be to stop the execution at a particular point and ask some questions.



This situation is one in which we want to use a tool called a debugger. Debuggers allow us to set up points where the execution flow will pause and allow us, the user, to interact with the system within the current context of where the execution paused.

Ruby has a well supported debugger project named 'Pry'. 'Pry' is a Ruby gem that is already installed in the Chef Development Kit (Chef DK).



To explore using Pry we need to create an issue for ourselves to troubleshoot. Doing so will allow us to see some of the power of Pry.

Create a Typo in the Defined Attribute

~/apache/attributes/default.rb

```
default['apche']['package_name'] = 'httpd'
```


This is a simple typo that the examples would catch but when it comes time to find and fix the issue, our eyes may not immediately catch it. We may think the error lies somewhere in the recipe. If we cannot find it we keep running the tests and wondering what is going wrong.

Add a Break Point in the Recipe

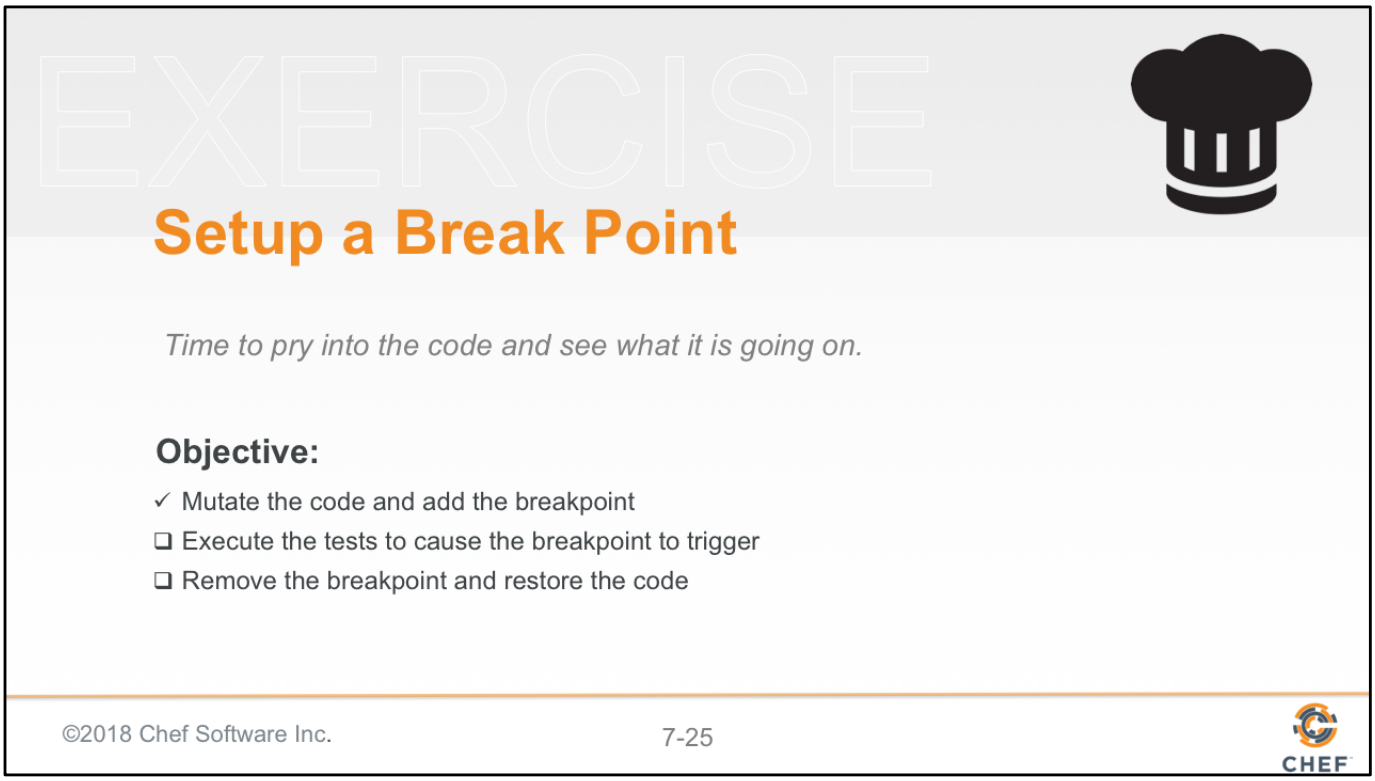
~/apache/recipes/install.rb

```
#
# Cookbook:: apache
# Recipe:: install
#
# Copyright:: 2018, The Authors, All Rights Reserved.
require 'pry'
binding.pry

package node['apache']['package_name']
```


To use Pry you first have to specify a require statement. The require here will look for a file name 'pry' give up on finding it locally and then look for the file inside all of the installed gems.

After the Pry code is loaded we access a method named 'binding' and then ask it to run 'pry'. 'binding' is a special method in Ruby that is like gaining access to the DNA of the current context. Pry, after it is loaded, will add the 'pry' method to the binding object to allow us the ability to setup a break point.

Wherever we want to set a breakpoint we can place these two lines.



The breakpoint cannot break itself. We need to execute the code to cause the execution to pause. The best way to do that is execute the tests that we have defined.

Execute the Test to Initiate Pry

```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
From: /tmp/chefspec20180313-23174-
grz9vbfile_cache_path/cookbooks/apache/recipes/install.rb @ line 9
Chef::Mixin::FromFile#from_file:

    4: #
    5: # Copyright:: 2018, The Authors, All Rights Reserved.
    6: require 'pry'
    7: binding.pry
    8:
 => 9: package node['apache']['package_name']

    # ... CONTINUES ON THE NEXT SLIDE ...
```


We can execute the tests for the install specification so that it will process that recipe. After a moment of normal execution the flow will pause and you will be shown where in the code the execution has paused. Along the top is the name of the file with the line number where it is paused. Below is a source code listing line-by-line before and after the breakpoint.

Pry Provides an Interactive Prompt

```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
    # ... CONTINUED FROM THE PREVIOUS SLIDE ...
    4: #
    5: # Copyright:: 2018, The Authors, All Rights Reserved.
    6: require 'pry'
    7: binding.pry
    8:
 => 9: package node['apache']['package_name']

[1] pry(#<Chef::Recipe>)>
```


Below the summary of the code around the breakpoint is a prompt. Pry launches a Read-Eval-Print-Loop (REPL). At this prompt we can type in a number of commands and any Ruby code.

Ask Pry for Help

```
[1] pry(#<Chef::Recipe>)> help
```

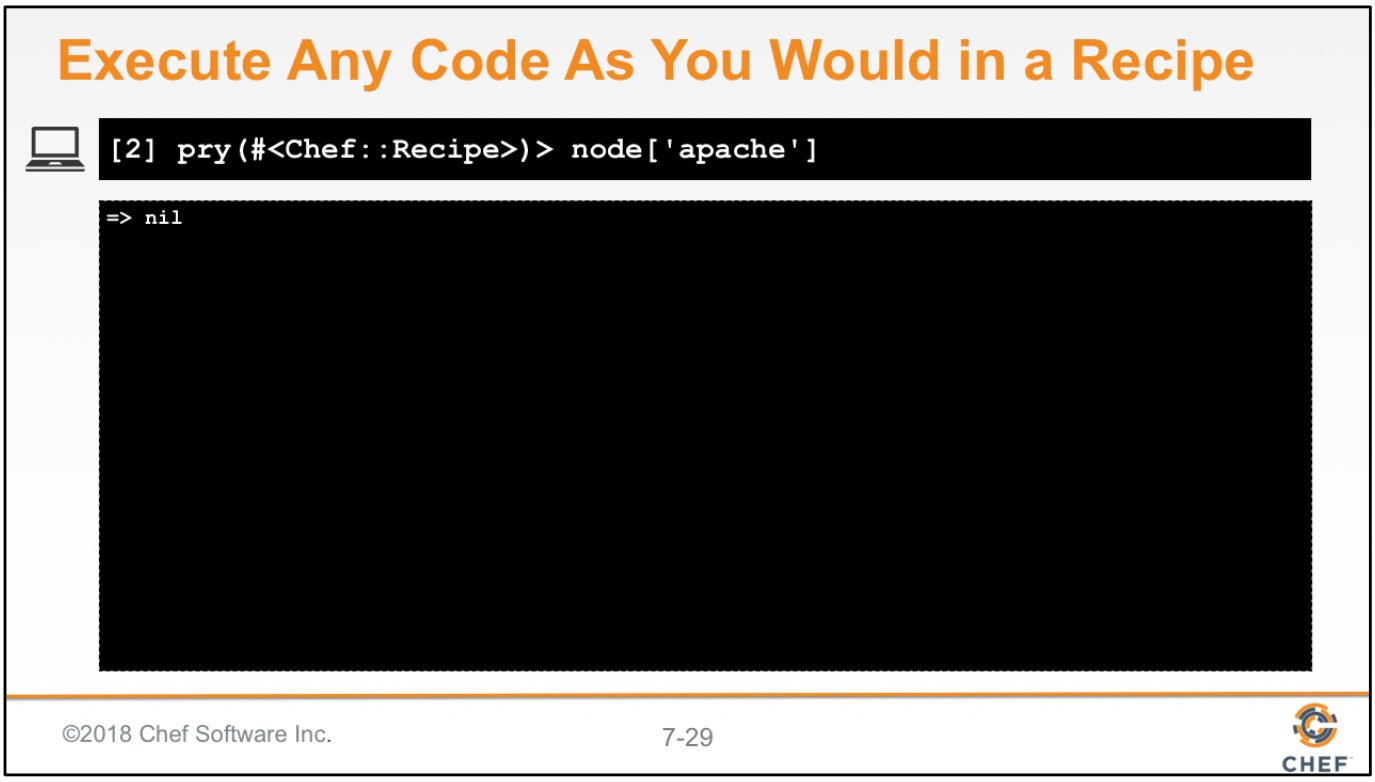
```
Help
  help              Show a list of commands or information about a specific command.

Context
  cd                Move into a new context (object or scope).
  find-method       Recursively search for a method within a class/module or the curr...
  ls                Show the list of vars and methods in the current scope.
  pry-backtrace     Show the backtrace for the pry session.
  raise-up          Raise an exception out of the current pry instance.
  reset             Reset the repl to a clean state.
```

To escape the help menu, type in q

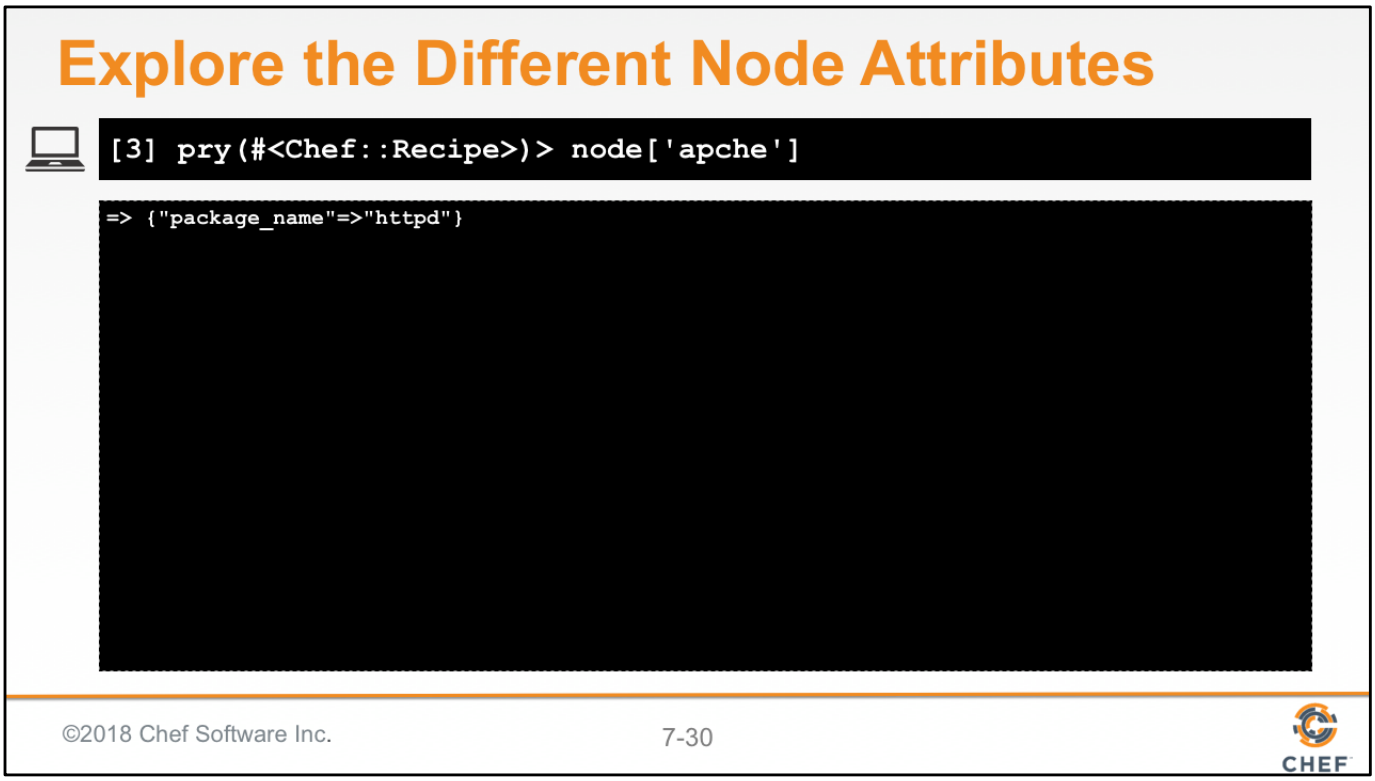
CHEF

The most important provided by Pry is probably the 'help' command. Within the results of this you will see all the commands available. The help will display in a scrolling page like a Linux man page. To escape out of the help output and return to being able to type in commands you will need to enter the keystrokes ':q'

Instructor Note: This content introduces Pry but will not go into explaining all of the different features.

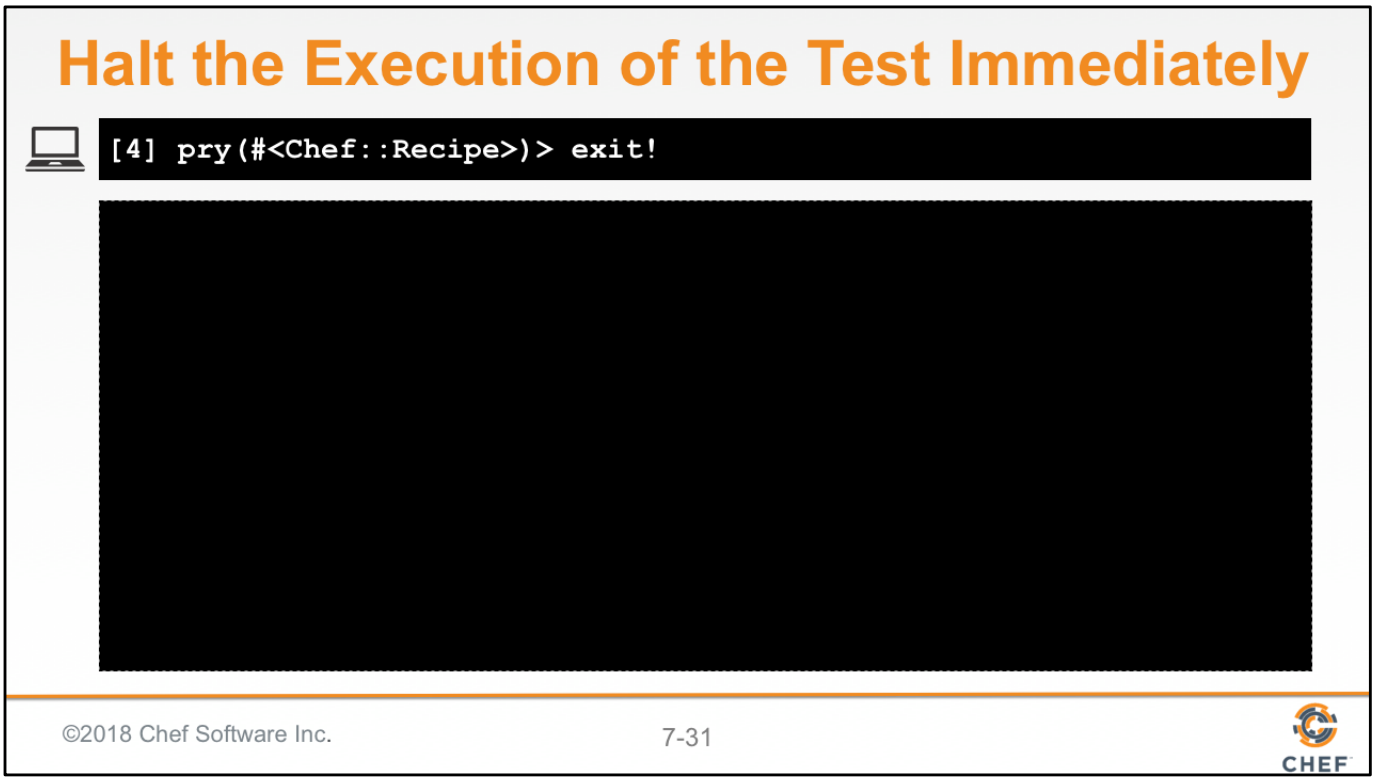


Back at the prompt you can enter in any code that you would normally write within the recipe. In this case we can start to examine the node object to see that the node object does not have the top-level attribute set as we expected.



This interactive session allows us to verify the actual state quickly. When it does not match our mental model we can try multiple hypothesizes quickly. Here we may return back to the attribute file and copy the text within the attribute and attempt this again and see what is actually going on.

We see in this example that

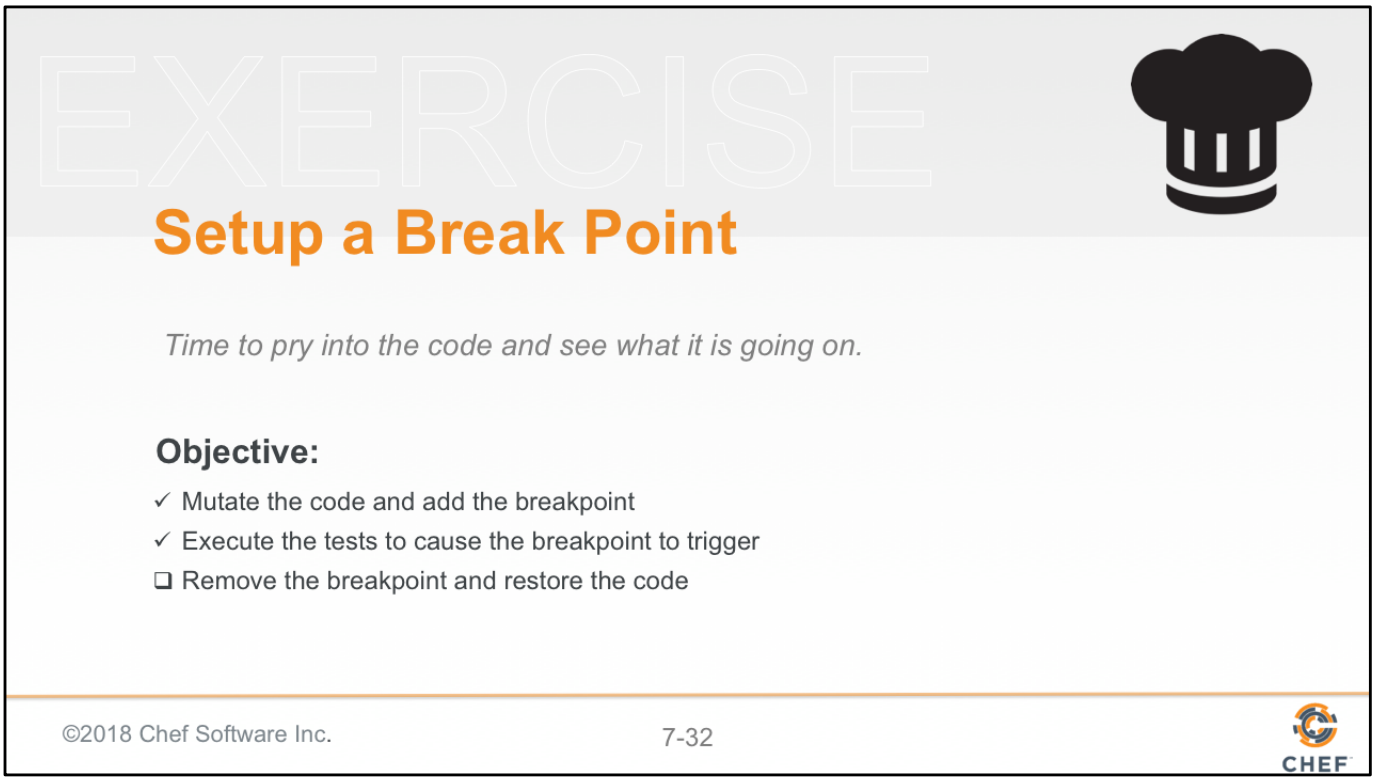


When you are satisfied with what you have discovered it is time to exit. Pry provides two versions of exit:

'exit' which will resume the execution and stop at any other breakpoints along the way.

'exit!' which halts the execution immediately and returns you to your shell.

In this situation we want to halt the execution immediately as we have discovered the issue.



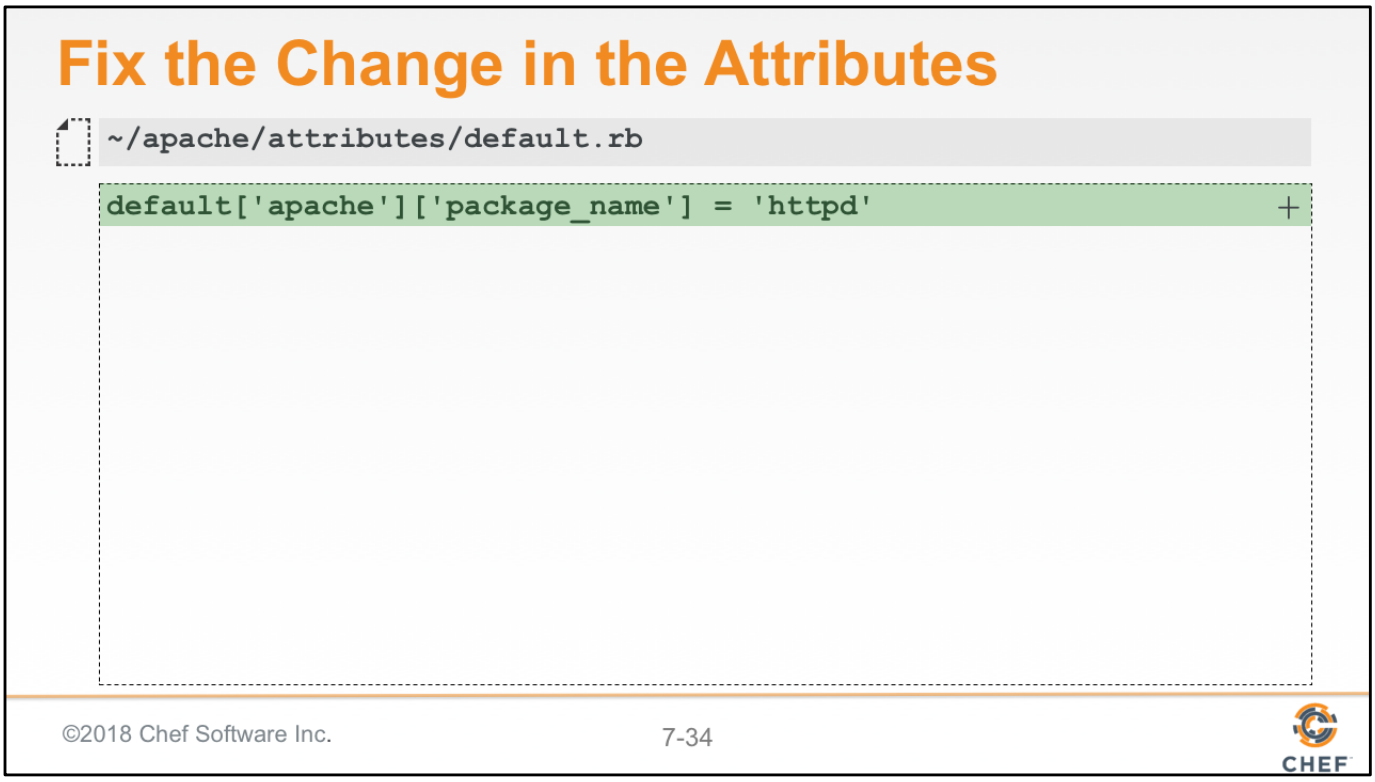
Now that we have discovered the issue in this scenario it is time to remove the breakpoint and restore the attributes code back to its correct state.

Remove the Break Point from the Recipe

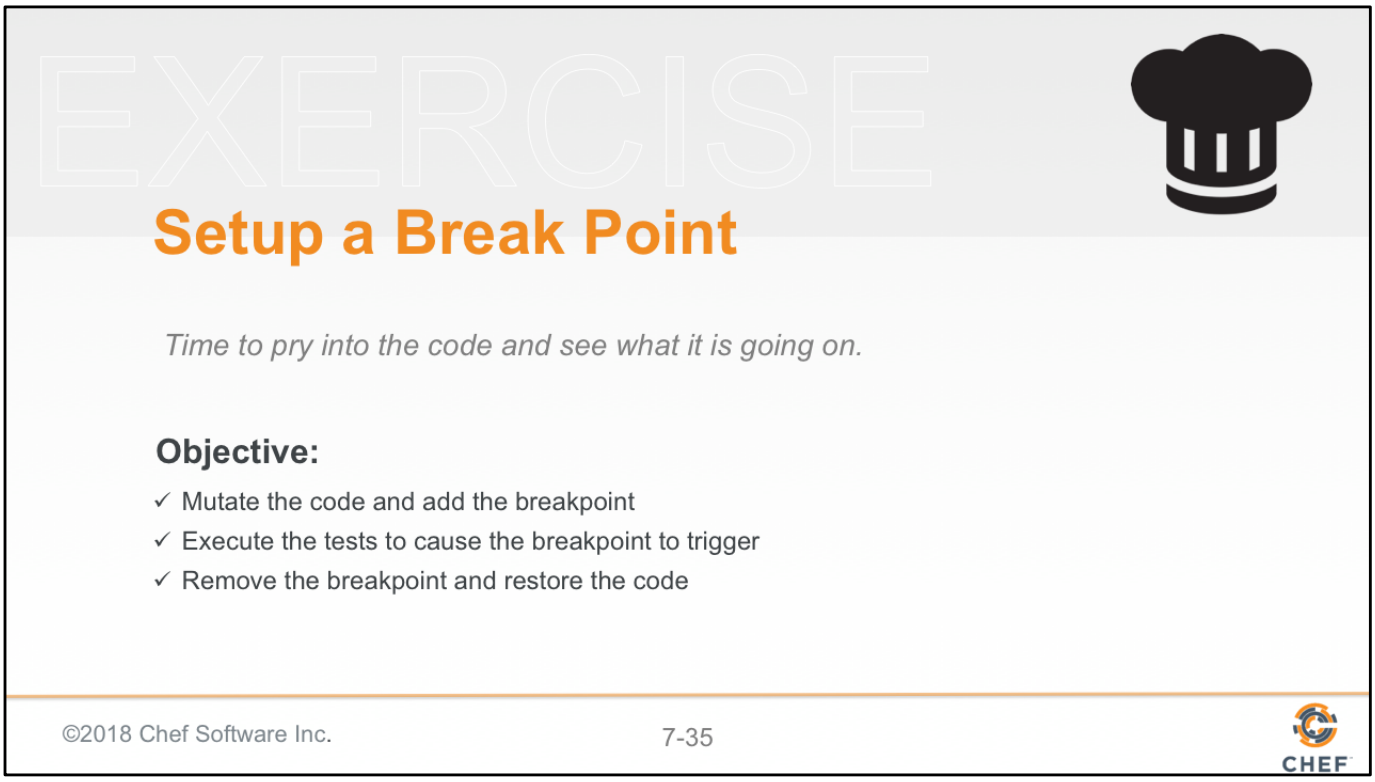
~/apache/recipes/install.rb

```
#
# Cookbook:: apache
# Recipe:: install
#
# Copyright:: 2018, The Authors, All Rights Reserved.
require 'pry'
binding.pry

package node['apache']['package_name']
```


CHEF



This is a simple typo that the examples would catch but when it comes time to find and fix the issue, our eyes may not immediately catch it. We may think the error lies somewhere in the recipe. If we cannot find it we keep running the tests and wondering what is going wrong.



This small exercise focused on a small subset of what is possible with Pry. It is a powerful tool that will aid you in understand the execution of the system much faster than tests alone.

LAB

Refactor Remaining Resources

- ❑ Refactor the resource to use a Node attribute
- ❑ Execute the tests and verify the tests fail
- ❑ Add the new Node attribute
- ❑ Execute the tests and verify the tests pass

BONUS: Use pry to verify that the attribute has been set.

❖ **Repeat this series of steps for the configuration recipe and service recipe**

CHEF

Now it is your turn. Two recipes remain that I want you to refactor to use attributes. Follow the same workflow you used here. As a bonus try using Pry again to reinforce setting it up and navigating through the execution flow with it.

Instructor Note: Allow 10 minutes to complete this exercise

Update the Recipe to use the Node Attribute

~/apache/recipes/service.rb

```
#
# Cookbook:: apache
# Recipe:: service
#
# Copyright:: 2018, The Authors, All Rights Reserved.
service node['apache']['service_name'] do
  action [:enable, :start]
end
```


Let's review the refactoring of the service resource. You returned first to the service resource in the service recipe and specify a node attribute that will give you the service name.

Execute the Tests to See it Fail

```
> chef exec rspec spec/unit/recipes/service_spec.rb

FFF

Failures:

  1) apache::service When all attributes are default, on an Centos 6.9
converges successfully
     Failure/Error: expect { chef_run }.to_not raise_error

       expected no Exception, got #<ArgumentError: You must supply a name when
declaring a service resource> with backtrace:
         # /tmp/chefspec20180313-17746-
14gwtwpfile_cache_path/cookbooks/apache/recipes/service.rb:6:in `from_file'
```


CHEF

You executed the tests against all the recipes or the specific service recipe. A large set of errors appear as we saw last time. The error is telling us to define the node attribute.

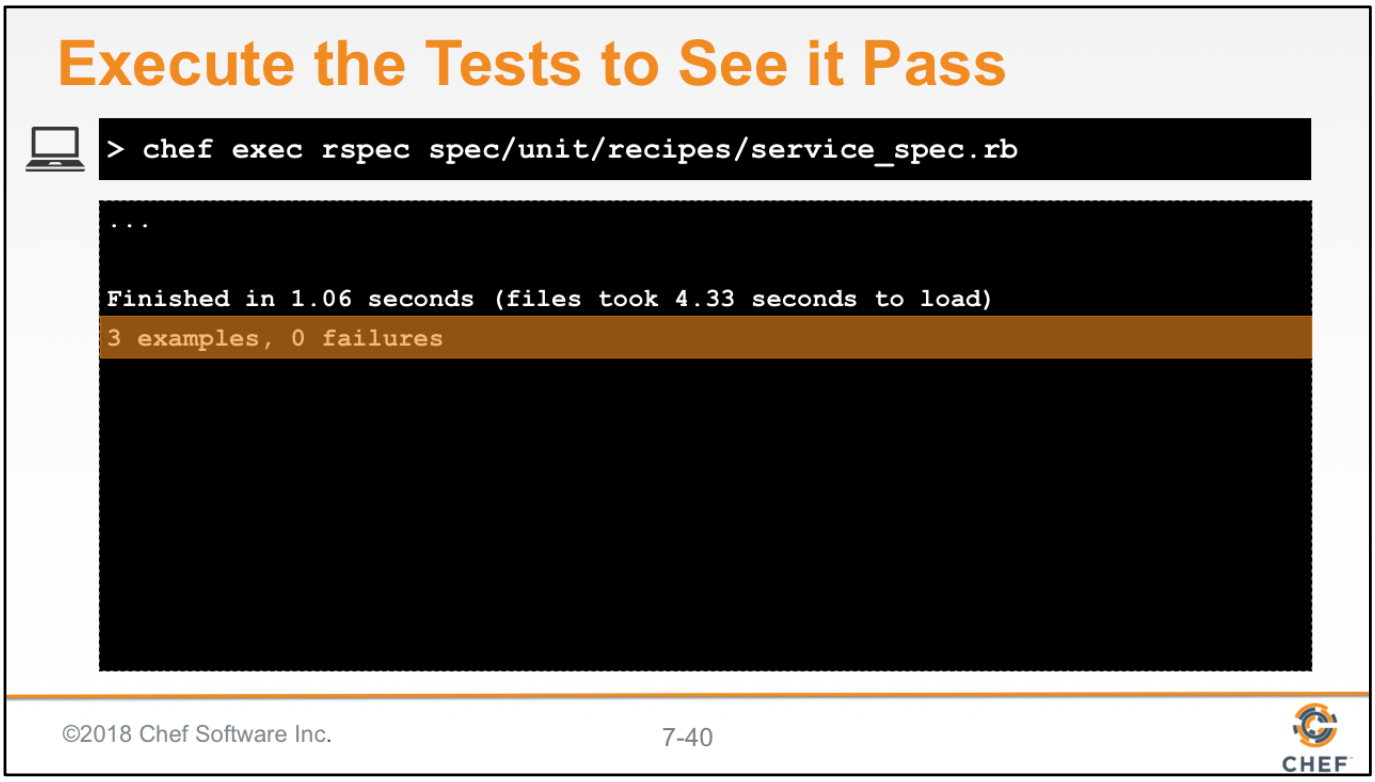
Add the Default Node Attribute

~/apache/attributes/default.rb

```
default['apache']['package_name'] = 'httpd'
default['apache']['service_name'] = 'httpd'
```


CHEF

You opened the default attributes file up and defined the new node attribute at the default level.



You executed the tests again and saw all the expectation have been met successfully.

Update the Recipe to use the Node Attribute

~/apache/recipes/configuration.rb

```
#
# Cookbook:: apache
# Recipe:: configuration
#
# Copyright:: 2018, The Authors, All Rights Reserved.
file node['apache']['default_index_html'] do
  content '<h1>Welcome Home!</h1>'
end
```


Let's review the refactoring of the service resource. You returned first to the service resource in the service recipe and specify a node attribute that will give you the service name.

Execute the Tests to See it Fail

```
> chef exec rspec spec/unit/recipes/configuration_spec.rb

FF

Failures:

  1) apache::configuration When all attributes are default, on an Centos 6.9
conve
     Failure/Error: expect { chef_run }.to_not raise_error

       expected no Exception, got #<ArgumentError: You must supply a name when
decurce> with backtrace:
```


CHEF

You executed the tests against all the recipes or the specific service recipe. A large set of errors appear as we saw last time. The error is telling us to define the node attribute.

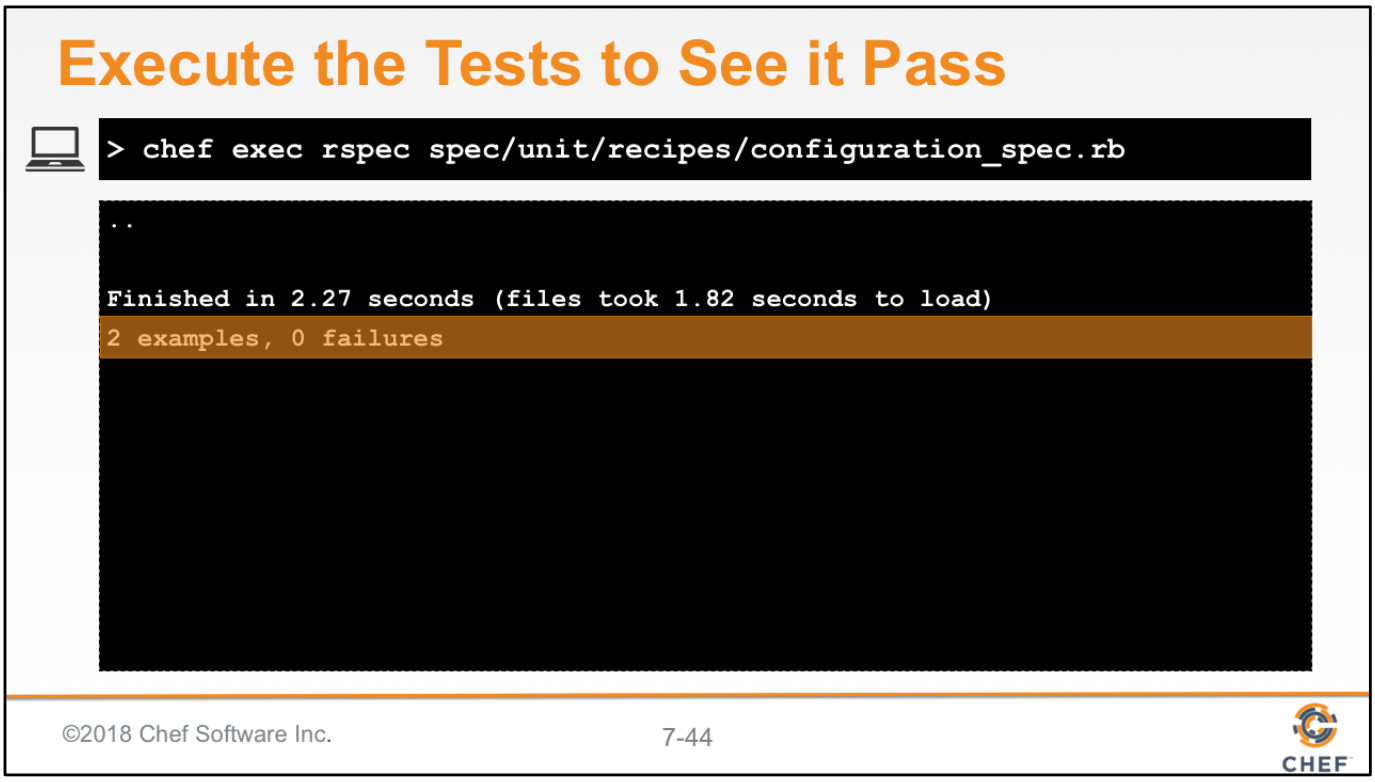
Add the Default Node Attribute

~/apache/attributes/default.rb

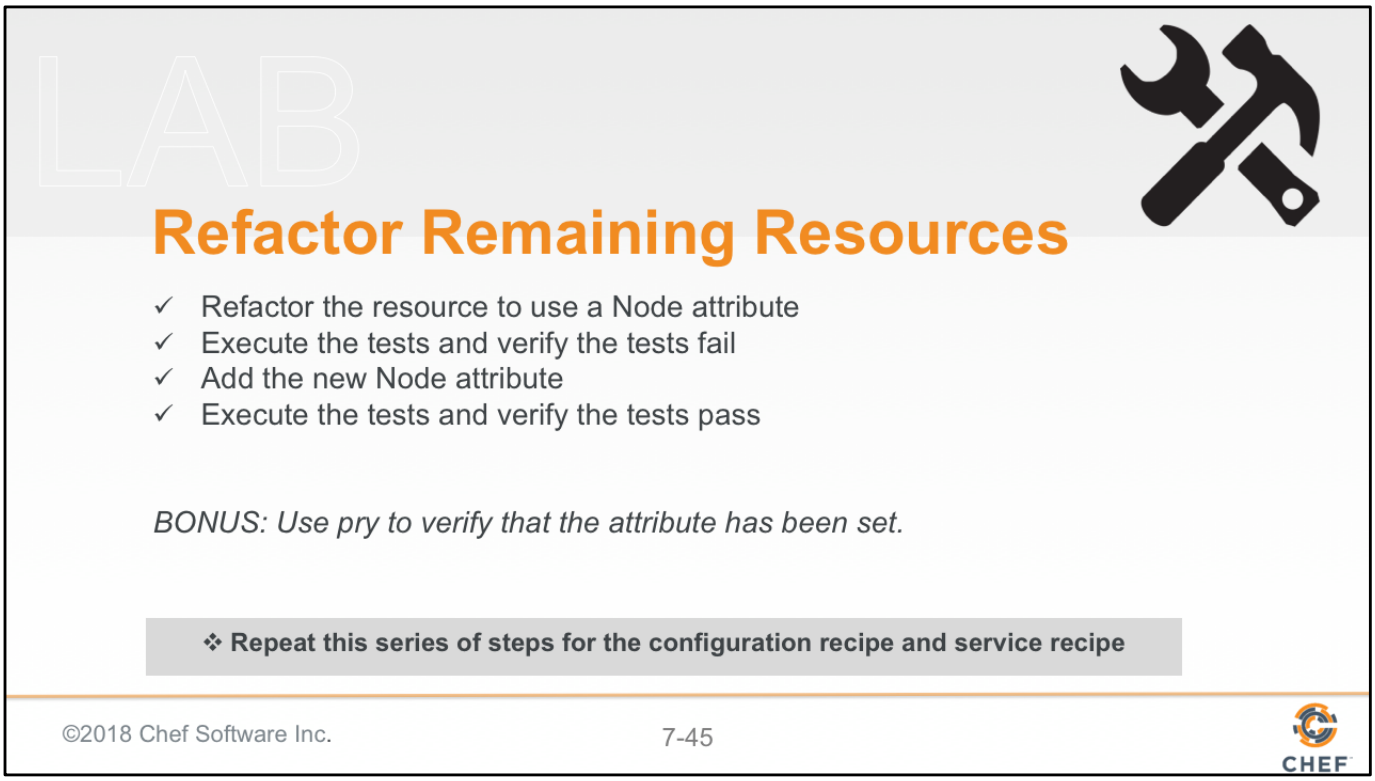
```
default['apache']['package_name'] = 'httpd'
default['apache']['service_name'] = 'httpd'
default['apache']['default_index_html'] = '/var/www/html/index.html'
```


CHEF

You opened the default attributes file up and defined the new node attribute at the default level.



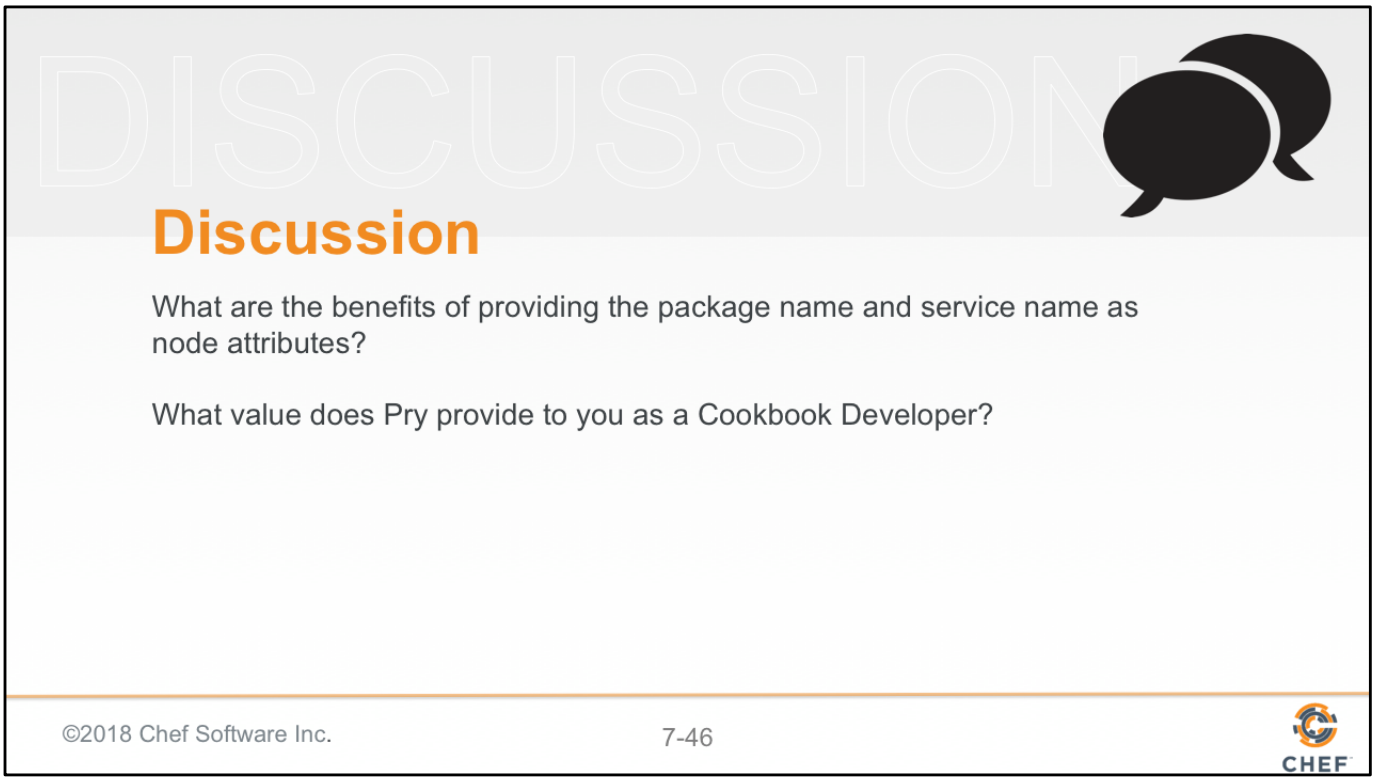
You executed the tests again and saw all the expectation have been met successfully.



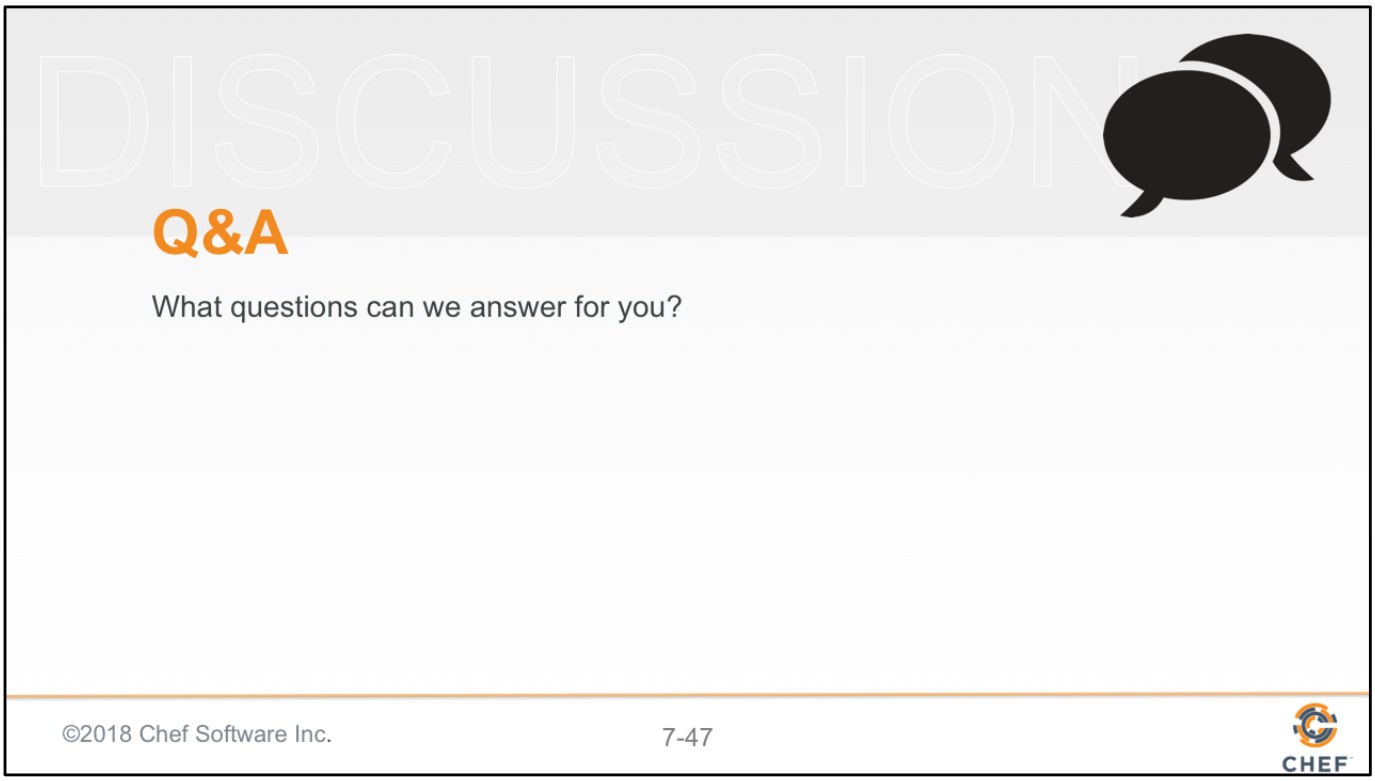
Congratulations. Now you have completely refactored the resources in the cookbook to use node attributes.

Let's have a discussion.

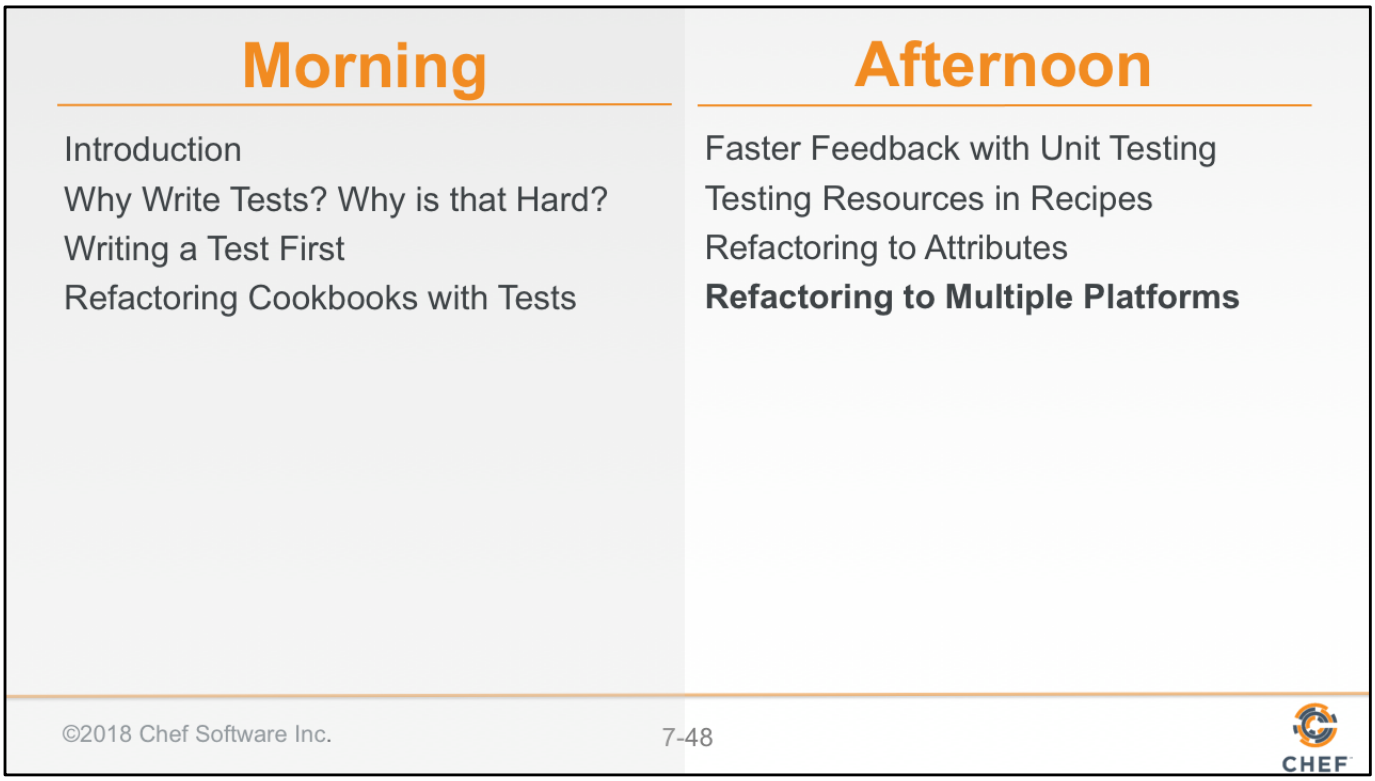
Instructor Note: We did not review the configuration recipe



Instructor Note: With large groups I often find it better to have individuals turn to the individuals around them, form groups of whatever size they feel comfortable, and have them take turns asking and answering the questions. When all the groups are done I then open the discussion up to the entire group allowing each group or individuals to share their answers.



Before we complete this section, let us pause for questions.



With the resources now using node attributes we are ready to explore the last section which will challenge us to expand the scope of this cookbook to support multiple platforms.


CHEF™