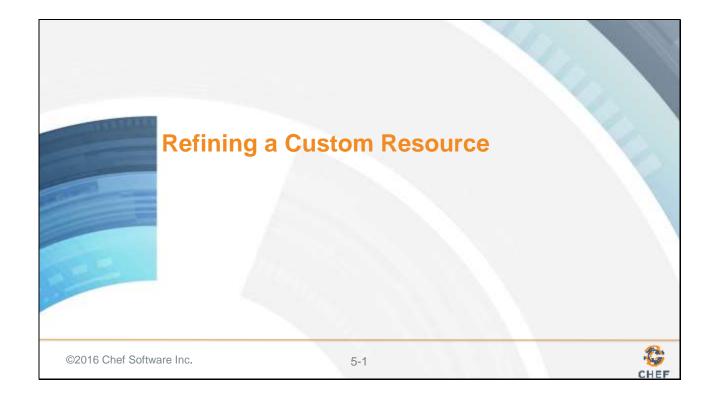
5: Refining a Custom Resource



Objectives

After completing this module, you should be able to:

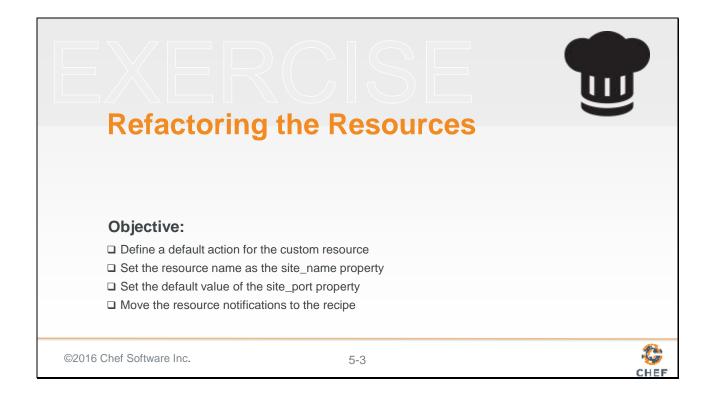
- > Set a custom resource's name to a property
- > Set a default value for a custom resource property
- > Define notifications correctly when creating custom resources

©2016 Chef Software Inc.

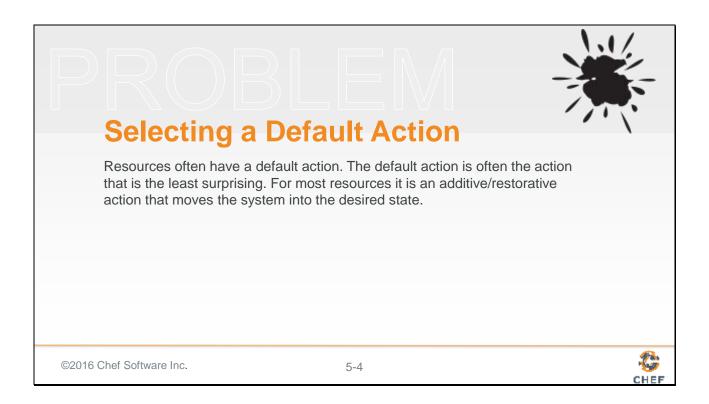
5-2



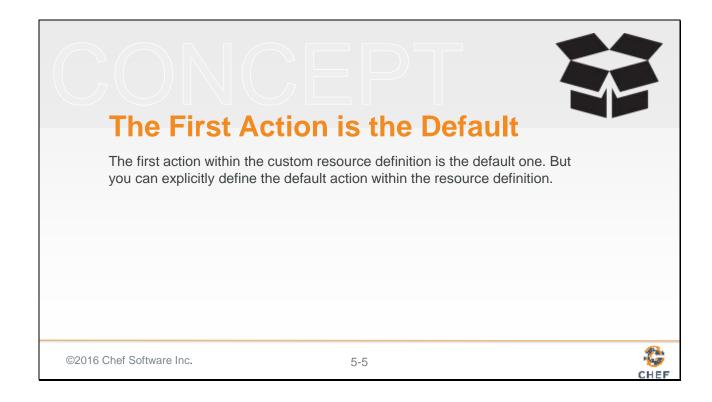
After completing this module you should be able to set a custom resource's name to a property, set a default value if the property is not provided, and define notifications correctly within the custom resource.



Every resource that you have used within Chef has had a default action.



Our custom resource should be no different. Having a default action that performs a nonsurprising operation is important. Of the two actions that we have defined the create action seems like the current correct default action.



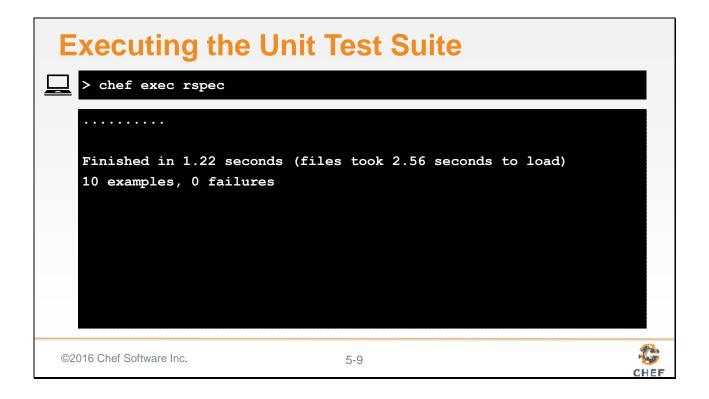
If you defined the create action as the first action within a resource definition file it automatically is the default action. The first action is the default action and that probably makes sense to those reading the custom resource definition.

You may also explicitly declare the default action.

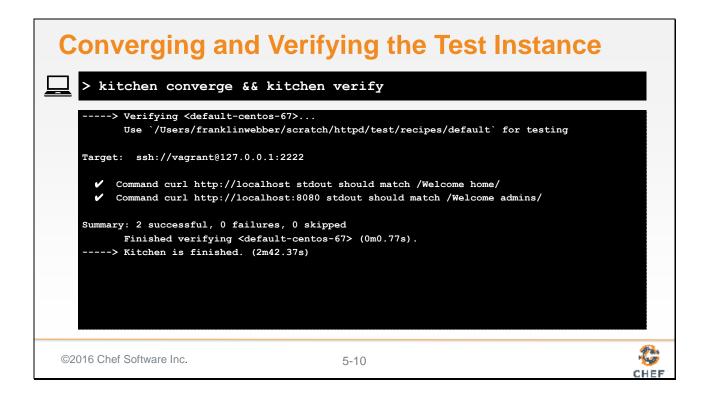
If the first action is the create action then you do not need to explicitly define the default action. However, you may decide that it makes it clearer for you or those you are collaborating with to specify this within the resource definition.

With the default action defined all uses of the custom resource which explicitly defined it may have those lines removed.

With the default action defined all uses of the custom resource which explicitly defined it may have those lines removed.



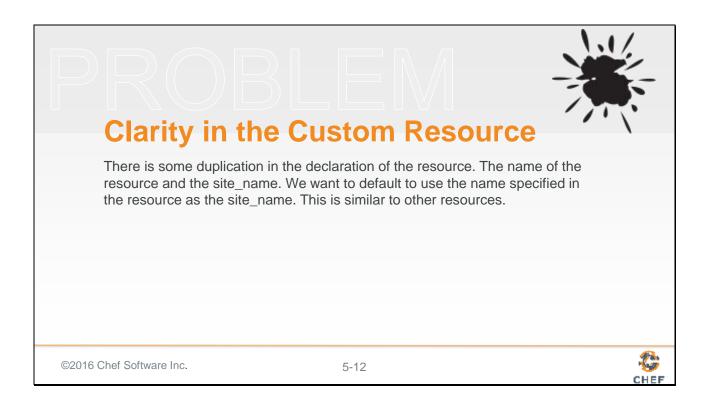
All the unit tests should pass.



All the integration test should pass.



Now we will look at tying the name provided to the custom resource to the site_name property.

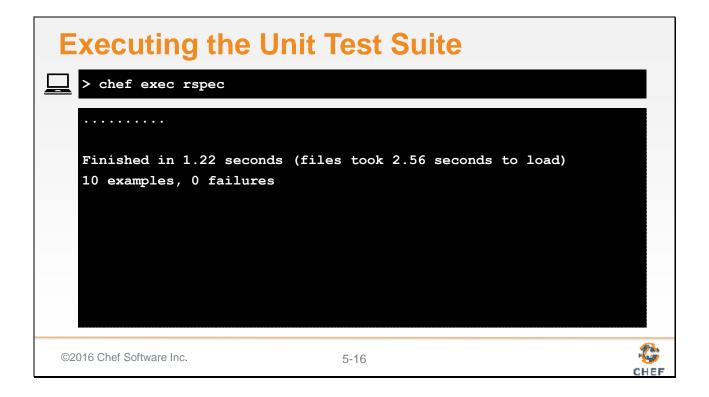


We want to ensure our custom resource is clear and concise. At the moment when you define the resource within the recipe you specify a value as the name of the custom resource and then a property that matches that same name. This seems like a redundancy that we want to remove.

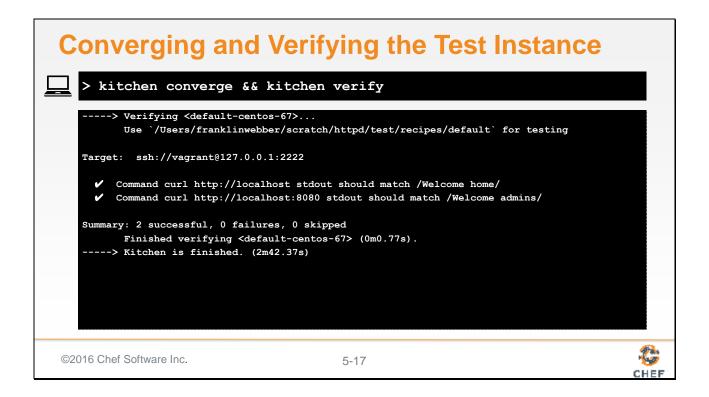
One property may have the 'name_attribute' option set to true. This property will be automatically populated from the name of the resource. Using the name of the resource will allow us to remove the need to specify that property which is repeating a value within the use of the custom resource.

Now that the site_name property is set as the name attribute we can remove the site name property from each use of the custom resource.

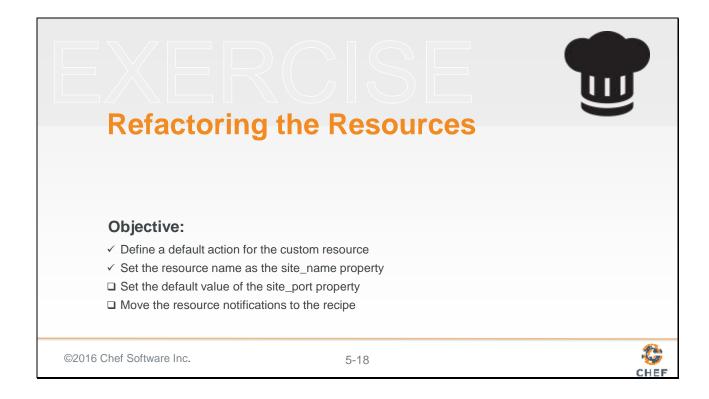
Now that the site_name property is set as the name attribute we can remove the site name property from each use of the custom resource.



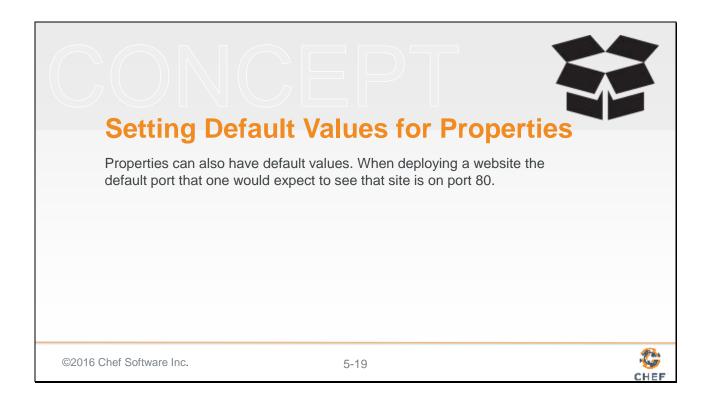
All the unit tests should pass.



All the integration test should pass.



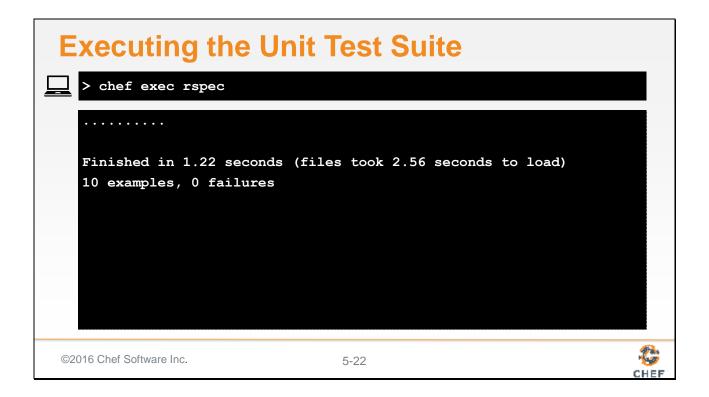
Resource properties can also have default values setup for them. Lets explore setting a default value for the site_port property.



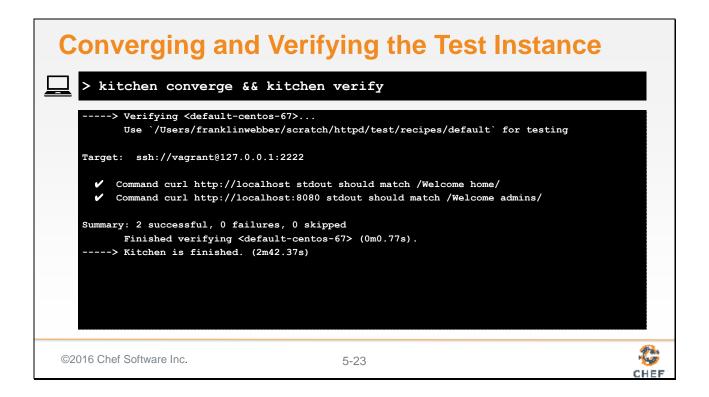
Setting a default value for a property allows you to define the resource and if you omit setting the property then the default value is used. We can use this behavior for our site port, choosing to say that if you do not specify a port we want the port to be 80.

All properties may have a default value. To add one requires that you simply add the option 'default' with its corresponding default value. Here we are setting the 'site_port' value to 80.

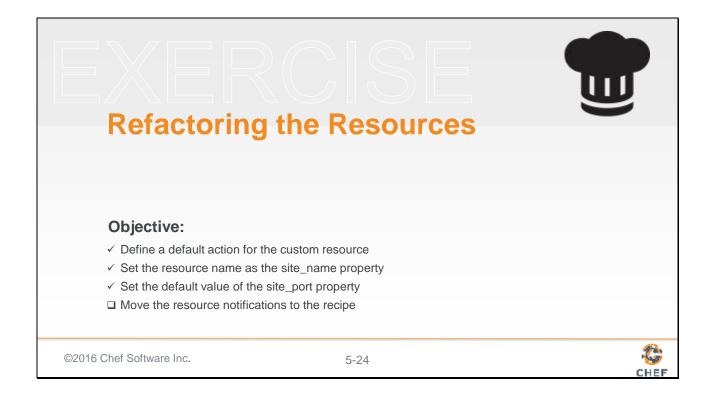
This allows us to remove the value from a single resource.



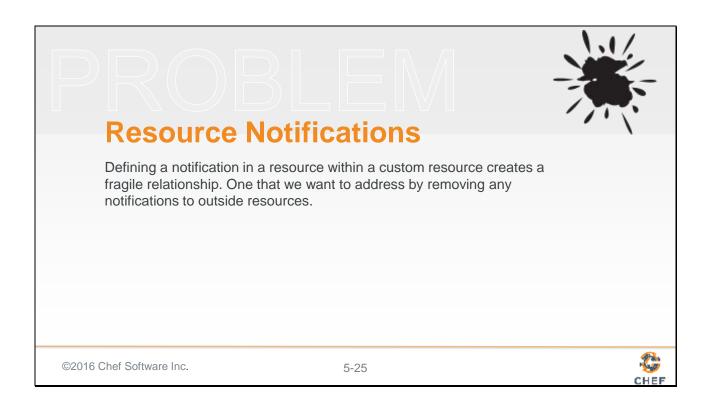
All the unit tests should pass.



All the integration test should pass.



Finally we want to properly address how the custom resource handles notifications.



When defining notifications within a resource action if you reference a resource outside of the action implementation there is a chance that your code may break if that resource's name were to change or simply not be implemented at all.

If any resource were to take action within the custom resource then the custom resource considers itself as taking action. We often say that any changed resource events are sent the parent custom resource and from that custom resource you can define your notifications.

First we remove the dependency on a resource not present within the action implementation of the custom resource.

```
Adding the Notification to the Resources

-/apache/recipes/default.rb

package 'apache'

apache_vhost 'welcome' do

notifies :restart, 'service[httpd]'

action :remove
end

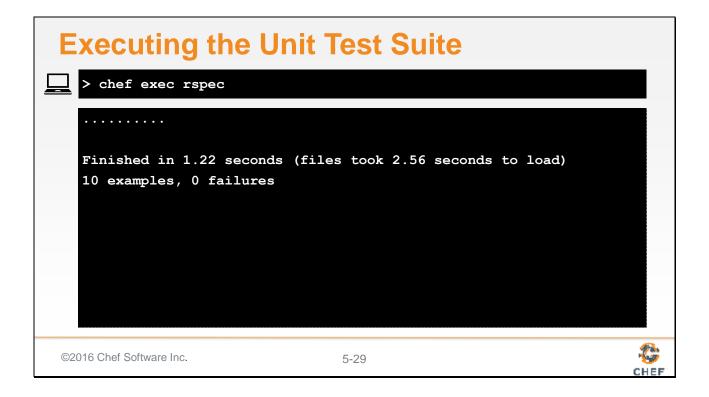
apache_vhost 'users' do

notifies :restart, 'service[httpd]'
end

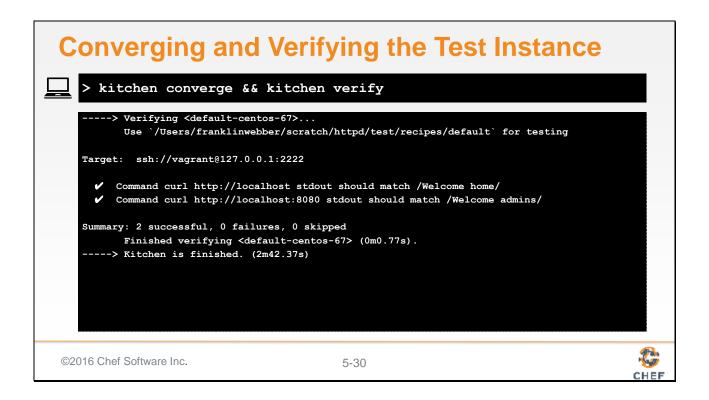
@2016 Chef Software Inc.
```

We then add the notifications to the custom resource implementations within the default recipe.

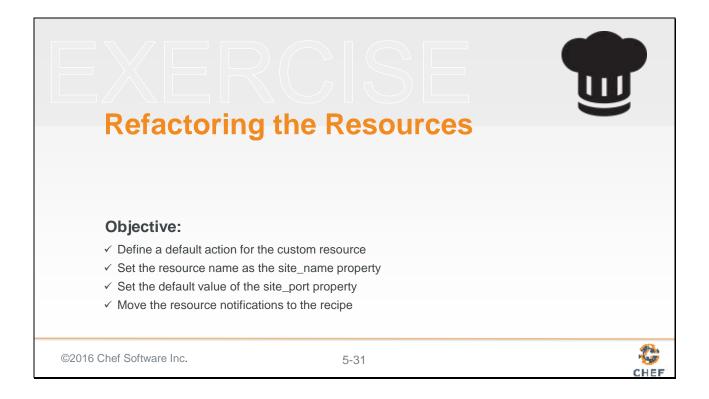
We then add the notifications to the custom resource implementations within the default recipe.



All the unit tests should pass.



All the integration test should pass.



We now have refactored the custom resource to have it behave more like other resources we are familiar within Chef.

Slide 32



What questions can we answer for you?

Slide 33

