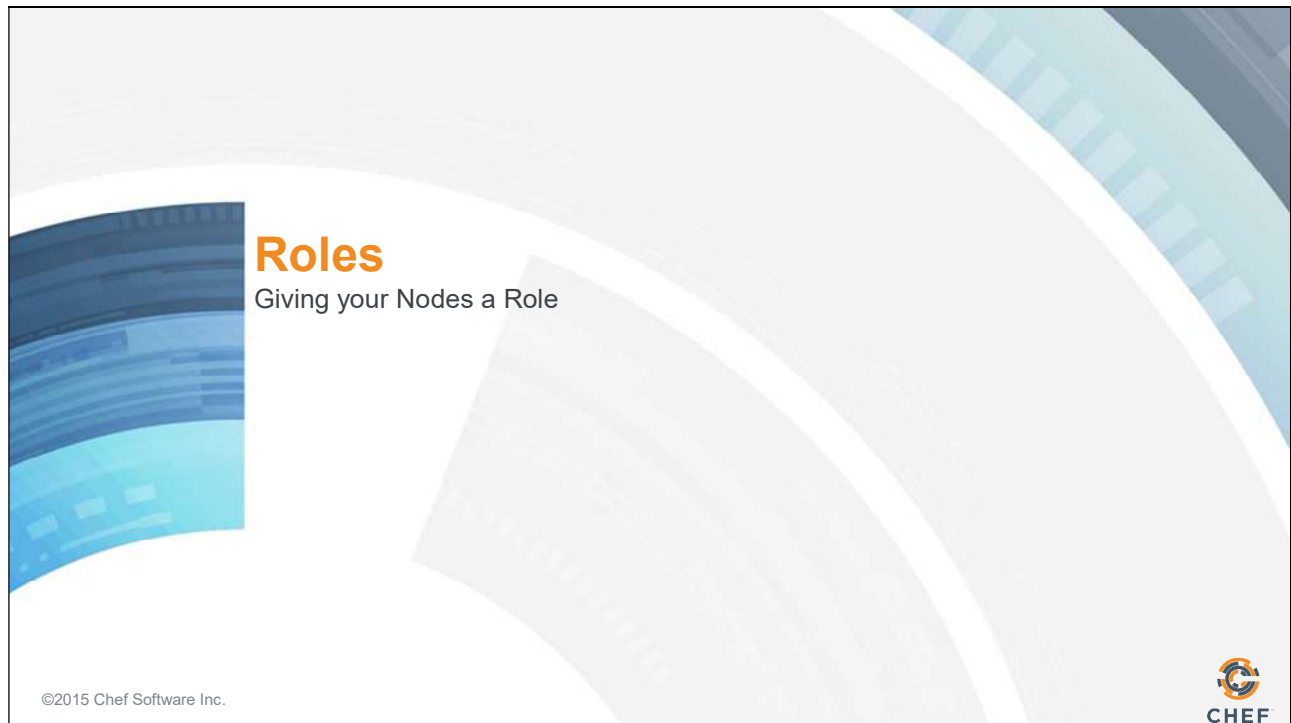


12: Roles



Objectives




After completing this module, you should be able to

- Assign roles to nodes so you can better describe them and configure them in a similar manner.

In this module you will give your nodes a role to better describe them so you can configure them in a similar manner.

CONCEPT

Roles




A role describes a run list of recipes that are executed on the node.

A role may also define new defaults or overrides for existing cookbook attribute values.

©2015 Chef Software Inc.

12-3




Up until this point it has been a mouthful to describe the nodes within our organization. We have two nodes, node1 and node3, that have the apache cookbook's default recipe in their run list. We have one node, node2, which has the myhaproxy cookbook's default recipe in its run list.

The Chef Server allows us to create and manage roles. A role describes a run list of recipes that are executed on the node. A role may also define new defaults or overrides for existing cookbook attribute values. Similar to what we accomplished with the wrapper cookbook.

A node may have zero or roles assigned to it.

CONCEPT

Roles




When you assign a role to a node you do so in its run list.

This allows you to configure many nodes in a similar fashion.

©2015 Chef Software Inc.

12-4



When you assign a role to a node you do so in its run list. This allows us to configure many nodes in a similar fashion because we no longer need to re-create a long run list for each node--we simply give it a role or all the roles it needs to accomplish its desired function.



GE: Roles for Everyone

We will give our nodes a role to better describe them and so we can configure them in a similar manner.

Objective:

- ❑ Give our load balancer node a "load_balancer" Role
- ❑ Give our web nodes a "web" Role

In this section you will create a `load_balancer` role and assign it to the run list of node2. You will also will create a `web` role and assign it to the run list of node1 and node3.

This is particularly powerful because we will no longer have to manage each of these identical nodes individually, instead we can make changes to the role that they share and all of the nodes that have this role will update accordingly.

GE: What Can 'knife role' Do?



```
$ cd ~/chef-repo
$ knife role --help

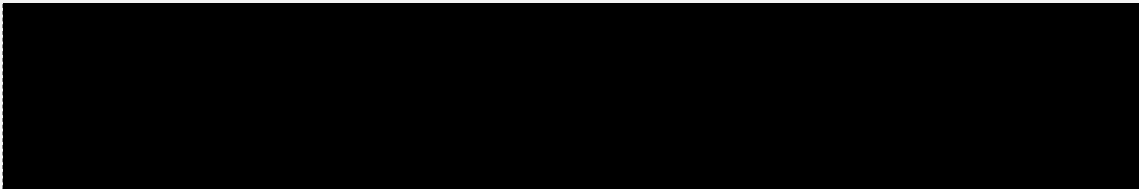
** ROLE COMMANDS **
knife role bulk delete REGEX (options)
knife role create ROLE (options)
knife role delete ROLE (options)
knife role edit ROLE (options)
knife role env_run_list add [ROLE] [ENVIRONMENT] [ENTRY[,ENTRY]] (options)
knife role env_run_list clear [ROLE] [ENVIRONMENT]
knife role env_run_list remove [ROLE] [ENVIRONMENT] [ENTRIES]
knife role env_run_list replace [ROLE] [ENVIRONMENT] [OLD_ENTRY] [NEW_ENTRY]
knife role env_run_list set [ROLE] [ENVIRONMENT] [ENTRIES]
knife role from file FILE [FILE..] (options)
```

Return to the base of your Chef repository and then run 'knife role --help' to see the available commands. Similar to other commands, you can see that 'knife role' supports the ability to list currently-defined roles.

GE: Run 'knife role list'



```
$ knife role list
```

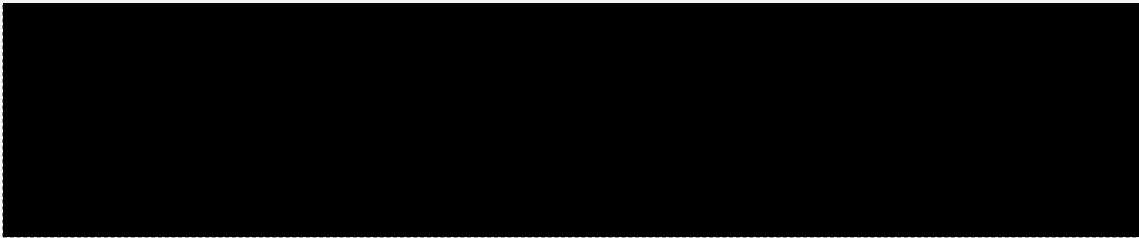


When you run 'knife role list' you can see from its lack of response that you have no roles defined.

GE: Create a Roles Directory



```
$ mkdir roles
```



Create a **roles** directory if necessary. If you are using the Chef Starter Kit this directory may already exist.

GE: Create the load_balancer.rb

```
~/chef-repo/roles/load_balancer.rb
```

```
name 'load_balancer'  
description 'Load Balancer'  
run_list 'recipe[myhaproxy]'
```

Create a file named `load_balancer.rb`. This is a ruby file that contains specific methods that allow you to express details about the role. You'll see that the role has a name, a description, and run list.

The name of the role as a practice will share the name of the ruby file unless it cannot for some reason. The name of the role should clearly describe what it attempts accomplish. The description of the role helps reinforce or clarify the intended purpose of the role. When selecting a role name that is not clear it is important that a helpful description is provided to help ensure everyone on the team understands its purpose. The run list defines the list of recipes that give the role its purpose. Currently the `load_balancer` role defines a single recipe - the `myhaproxy` cookbook's default recipe.

GE: Upload it to the Chef Server



```
$ knife role from file load_balancer.rb
```

```
Updated Role load_balancer!
```

Now you need to upload it to the Chef Server. This is done through the command 'knife role from file load_balancer.rb'.

The knife tool understands that you are uploading a role file and will look within the roles folder to find a file named knife role from file load_balancer.rb.

GE: Validate Chef Server Received It



```
$ knife role list
```

```
load_balancer
```

With the role uploaded, it is time to validate that the Chef Server received it correctly. We can do that by again asking the Chef Server for a list of all the roles on the system.

GE: View Details of the Role



```
$ knife role show load_balancer
```

```
chef_type:          role
default_attributes:
description:        Load Balancer
env_run_lists:
json_class:         Chef::Role
name:               load_balancer
override_attributes:
run_list:           recipe[myhaproxy]
```

You can ask for more details about a specific role using the above command. In this example we are requesting specific details about the role named `load_balancer`.

GE: Run 'knife node --help'



```
$ knife node --help
```

```
** NODE COMMANDS **  
knife node bulk delete REGEX (options)  
knife node create NODE (options)  
knife node delete NODE (options)  
knife node edit NODE (options)  
knife node environment set NODE ENVIRONMENT  
knife node from file FILE (options)  
knife node list (options)  
knife node run_list add [NODE] [ENTRY[,ENTRY]] (options)  
knife node run_list remove [NODE] [ENTRY[,ENTRY]] (options)  
knife node run_list set NODE ENTRIES (options)  
knife node show NODE (options)
```

Run 'knife node --help' to see its options.

GE: Set the load_balancer Role to node2



```
$ knife node run_list set node2 "role[load_balancer]"
```

```
node2:  
  run_list: role[load_balancer]
```

The last step is to redefine the run list for node2. We want the run list to contain only the `load_balancer` role.

Previously, we used the command `'knife node run_list add'` to append a new item to the existing run list. There is also a command that allows us to remove an item from the run list. There is a command that allows us to set the run list to a value provided. This will replace the existing run list with a new one that we provide.

GE: Verify the Run List



```
$ knife node show node2
```

```
Node Name:    node2
Environment:  _default
FQDN:         ip-172-31-0-128.ec2.internal
IP:           54.210.192.12
Run List:     role[load_balancer]
Roles:
Recipes:      myhaproxy, myhaproxy::default, haproxy::default, haproxy::install_package
Platform:     centos 6.6
Tags:
```

After you update the run list, you can verify that the node has the correctly-defined run list by running 'knife node show node2'.

GE: Converge All the Load Balancer Nodes



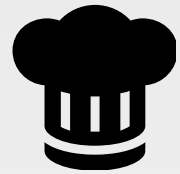
```
$ knife ssh "role:load_balancer" -x USER -P PWD "sudo chef-client"
```

```
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com   - build-essential
ec2-54-210-192-12.compute-1.amazonaws.com   - cpu
ec2-54-210-192-12.compute-1.amazonaws.com   - haproxy
ec2-54-210-192-12.compute-1.amazonaws.com   - myhaproxy
ec2-54-210-192-12.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-192-12.compute-1.amazonaws.com Converging 9 resources
ec2-54-210-192-12.compute-1.amazonaws.com Recipe: haproxy::install_package
ec2-54-210-192-12.compute-1.amazonaws.com   * yum_package[haproxy] action install
(up to date) ...
```

You can use 'knife ssh' to run 'sudo chef-client' on all the nodes again to ensure that nothing has changed.

In this instance we only interested in having node2 run the command so we can get a little more creative with the search criteria and find nodes with the role load_balancer. In this case there is only one result.

Within the results, nothing should change. Switching over to the role did not change the fundamental recipes that were applied to the node.



Roles for Everyone

*We will give our nodes a role to better describe them
and so we can configure them in a similar manner.*

Objective:

- ✓ Give our load balancer node a "load_balancer" Role
- Give our web nodes a "web" Role

Now if you want to setup a new node in the future to act as a load balancer, you can now simply set the new node's run list to be the `load_balancer` role and it will have identical functionality with all the other nodes that define this role.



Lab: Define a Web Role

- ☐ Create a role named 'web' that has the run list 'recipe[apache]'
- ☐ Set node1's run list to be "role[web]"
- ☐ Set node3's run list to be "role[web]"

In this lab, define a new role named 'web' that has the run list: including the apache cookbook's default recipe.

When you're done defining the role, upload it to the Chef Server, and then set the run list on node1 and node3 to the role that you have defined.

And for good measure, though nothing should have changed, run 'sudo chef-client' on both node1 and node3 to ensure that no functionality has been lost.

Lab: Create the web.rb File

```
~/chef-repo/roles/web.rb  
  
name 'web'  
description 'Web Server'  
run_list 'recipe[apache]'
```

First we create a file named web.rb in the roles directory.

The name of the role is web. The description should be Web Server. The run list you define should contain the apache cookbook's default recipe.

Lab: Upload the web.rb File



```
$ knife role from file web.rb
```

```
Updated Role web!
```

You need to share the role with the Chef Server so upload that file.

Use the command 'knife role from file web.rb'. 'knife' knows where to look for that role to upload it.

Lab: Verify the Role on the Chef Server



```
$ knife role list
```

```
load_balancer  
web
```

Verify that the role can be found on the Chef Server.

Lab: Verify Specific Information About the Role



```
$ knife role show web
```

```
chef_type:          role
default_attributes:
description:        Web Server
env_run_lists:
json_class:         Chef::Role
name:               web
override_attributes:
run_list:           recipe[apache]
```

Verify specific information about the role. Specifically, does it have the run list that we defined?

Lab: Set node1's Run List



```
$ knife node run_list set node1 "role[web]"
```

```
node1:  
  run_list: role[web]
```

Set node1's run list to be the web role.

Lab: Set node3's Run List



```
$ knife node run_list set node3 "role[web]"
```

```
node3:  
  run_list: role[web]
```

And we then set node3's run list to be the web role.

Lab: Converge All Web Nodes



```
$ knife ssh "role:web" -x USER -P PWD "sudo chef-client"
```

```
ec2-54-175-46-24.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list:
["apache"]
ec2-54-175-46-24.compute-1.amazonaws.com resolving cookbooks for run list:
["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
ec2-54-175-46-24.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com   - apache
```

To verify that everything is working the same as before, run 'knife ssh' for both of these nodes. In this instance the query syntax is going to find all nodes with the role set to web.



Roles for Everyone

*We will give our nodes a role to better describe them
and so we can configure them in a similar manner.*

Objective:

- ✓ Give our load balancer node a "load_balancer" Role
- ✓ Give our web nodes a "web" Role

With that we now have made it far easier to talk about our nodes. We can more casually describe a node as a 'web' server node or a 'load_balancer' node.


In the future if we needed to ensure that these types of nodes needed to run additional recipes, we could return to the role file, update its run list, and then upload it to the Chef Server again.

DISCUSSION

Discussion


What are the benefits of using roles? What are the drawbacks?

Roles can contain roles. How many of these nested roles would make sense?



©2015 Chef Software Inc.

12-27



Answer these questions.

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

DISCUSSION

Q&A

What questions can we help you answer?





CHEF™