

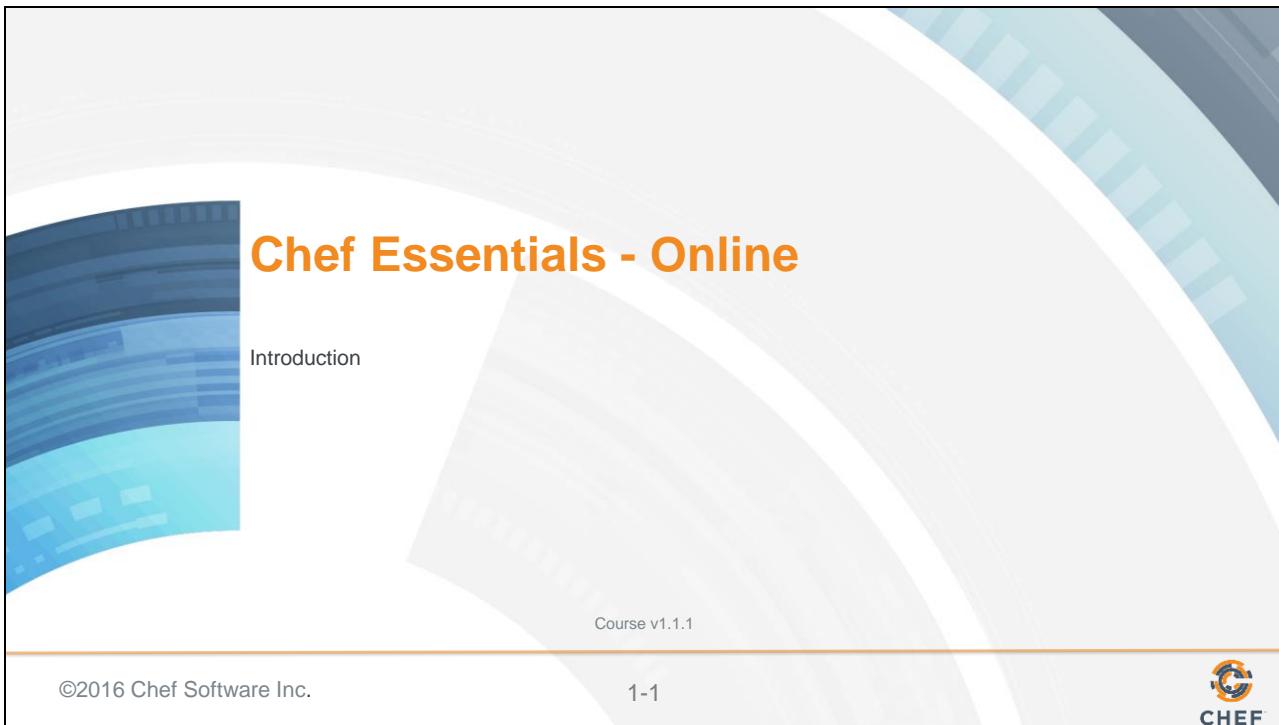


**CHEF**<sup>TM</sup>

Chef Training Services

# Chef Essentials for Linux (CentOS) Online

Participant Guide

**1: Introduction**

The slide features a blue and white abstract background with curved lines. In the center, the text "Chef Essentials - Online" is displayed in a large, bold, orange font. Below it, the word "Introduction" is written in a smaller, gray font. At the bottom of the slide, there is a thin orange horizontal bar. On the left side of this bar, the text "©2016 Chef Software Inc." is visible. In the center, the text "1-1" is displayed. On the right side, the Chef logo (a stylized orange 'C' with a circular arrow) is followed by the word "CHEF".

**Chef Essentials - Online**

Introduction

Course v1.1.1

©2016 Chef Software Inc.

1-1

 CHEF

This Chef Essentials course provides a basic understanding of Chef's core components, basic architecture, commonly used tools, and basic troubleshooting methods.

This should provide you with enough knowledge to start using Chef to automate common infrastructure tasks and express solutions to common infrastructure problems.

Slide 2

# EXERCISE

## Introductions



*Let's get to know each other and the training.*

**Objective:**

- Introduce ourselves
- Introduce this training experience

Before we get started with this training let's take a moment to get acquainted with each other and with the content that we are going to be exploring.

Slide 3

## Introduce Ourselves

Name

Current job role

Previous job roles/background

Experience with Chef and/or config management

Favorite Text Editor

Slide 4

# EXERCISE

## Introductions



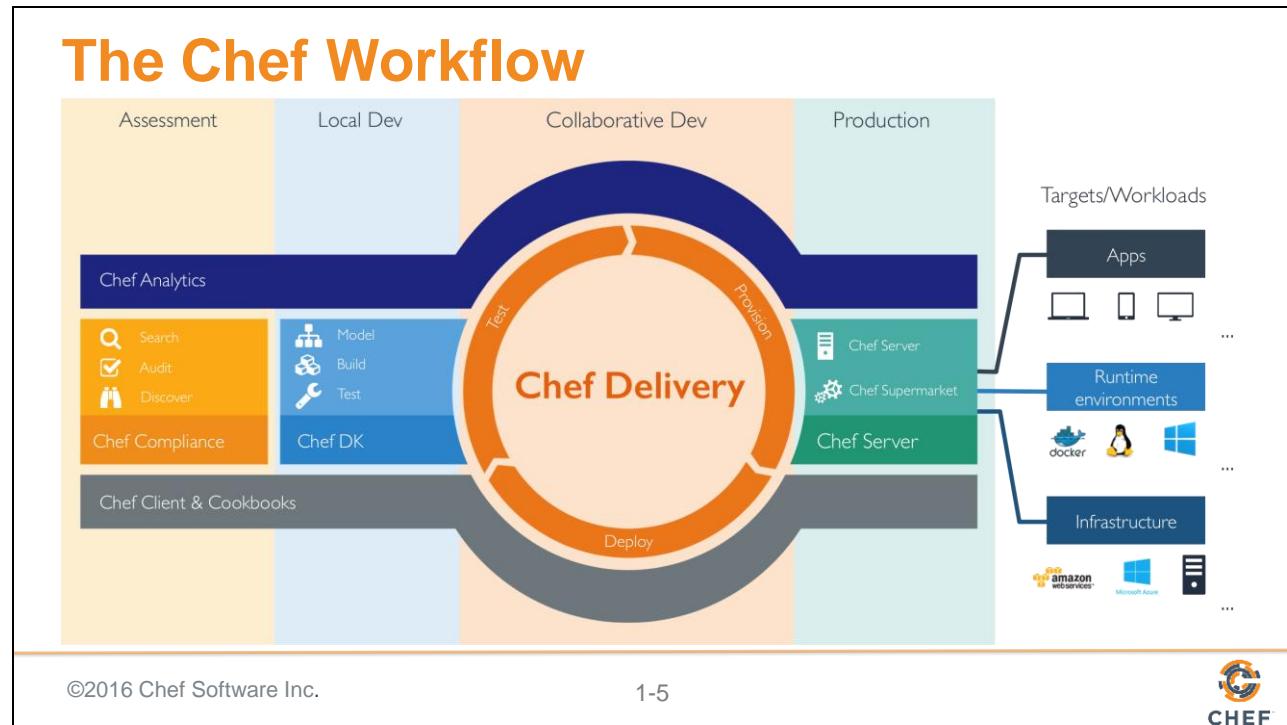
*Now that we know who we are let's learn a little why we are all here.*

**Objective:**

- ✓ Introduce ourselves
- Introduce this training experience

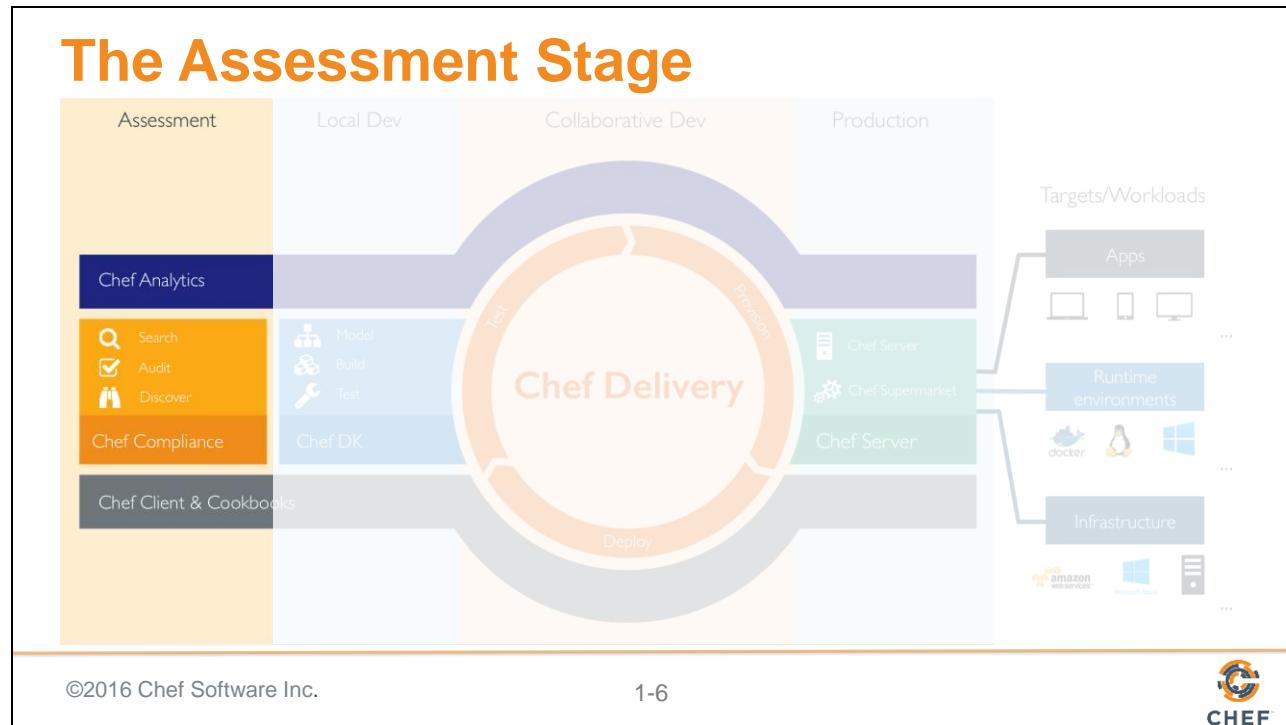
Now that we have introduced ourselves to each other let me introduce you to this training content so you understand where it fits within the bigger picture.

## Slide 5



This is the Chef workflow. Chef enables you to build your applications and infrastructure, maintain them, test them, and expand them as your organization grows. Think of each of these sections as stages. Let's look at each stage individually and talk about what happens there.

## Slide 6

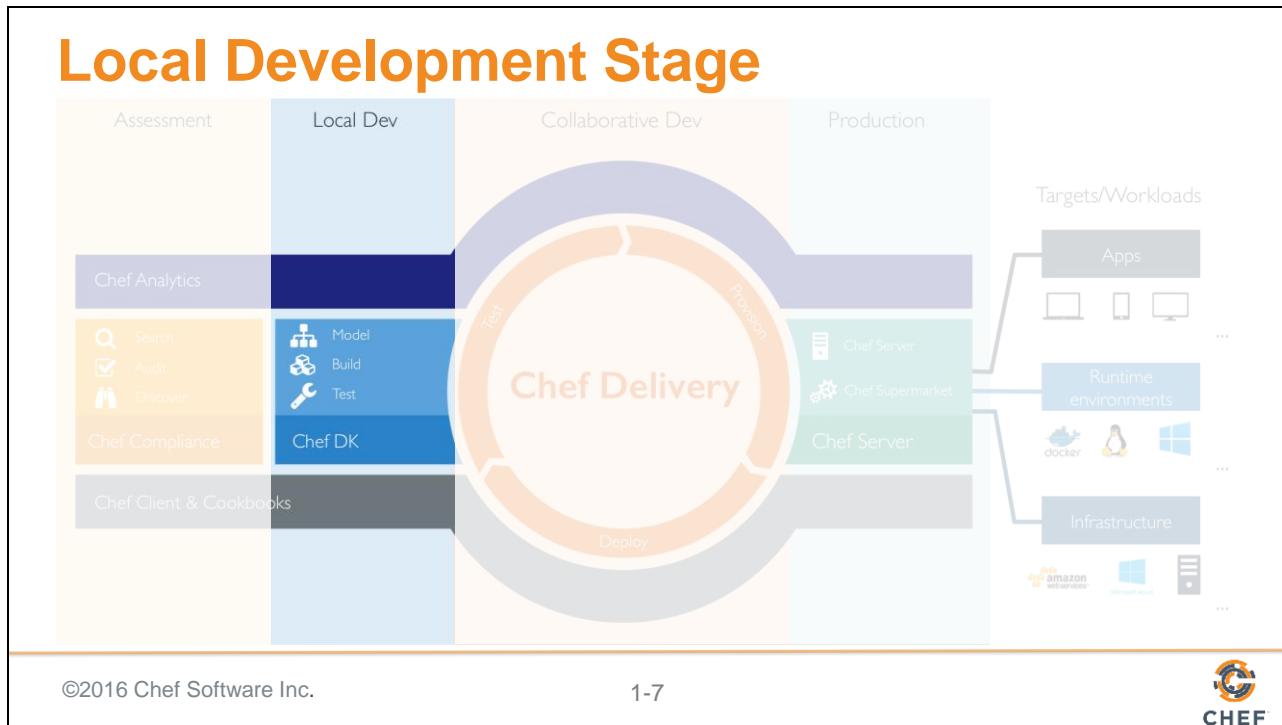


During the assessment stage you evaluate your system to ensure everything meets your requirements. Is your infrastructure in its desired state? If the work you do is overseen by another organization, do you meet all the compliance criteria of that organization? Are you doing everything that you need to do, or are there holes in your infrastructure.

While it is true that you don't need Chef to assess your infrastructure, one tool, Chef Compliance, can help make the assessment process easier. Chef Compliance enables you to assess your infrastructure's adherence to compliance requirements, and to monitor that infrastructure on an ongoing basis.

Chef Compliance scans for risks and compliance issues and generates easy-to-understand, customizable reports and visualization.

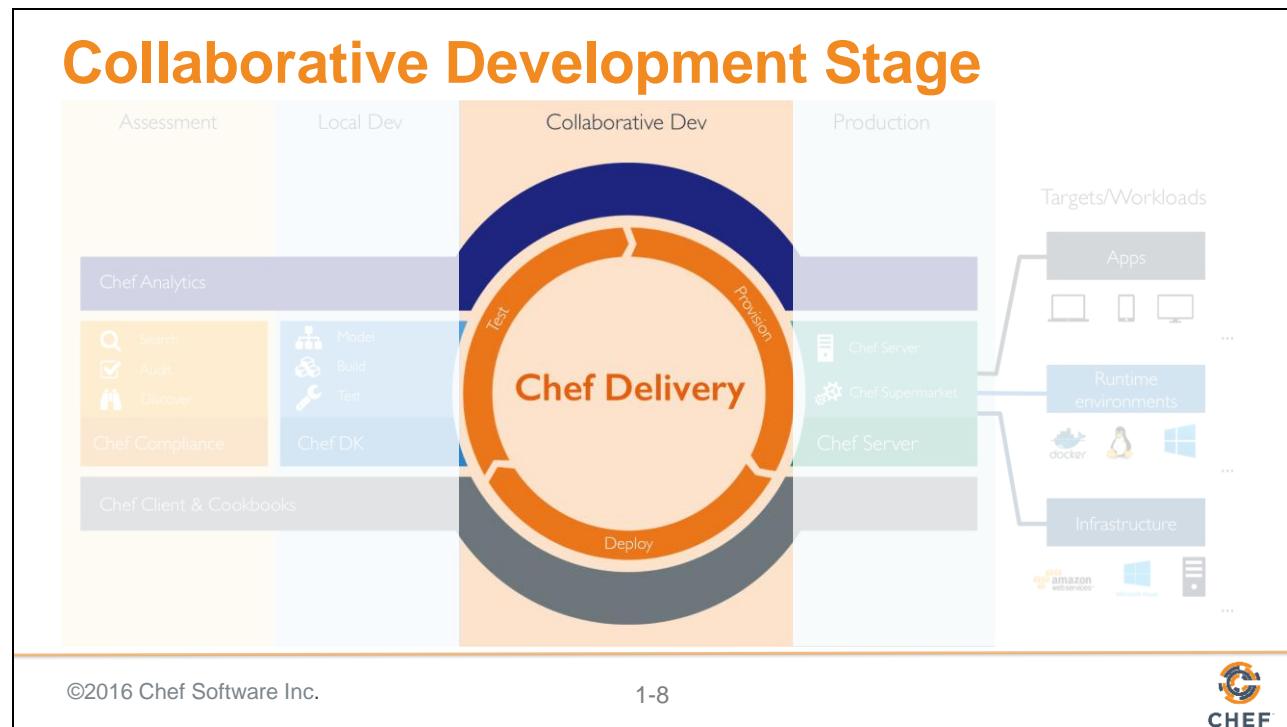
## Slide 7



During the assessment state you find that your system is not in your desired state or does not meet your requirements, you can use the Chef Development Kit (Chef DK) to fix it.

The Chef DK enables you to model and build your infrastructure on your local machine. This enables you to catch any problems before they show up in shared environments. You may have heard the phrase “Infrastructure as Code”. Because you build your infrastructure as code, you can test your solutions and incorporate version control to track your development history.

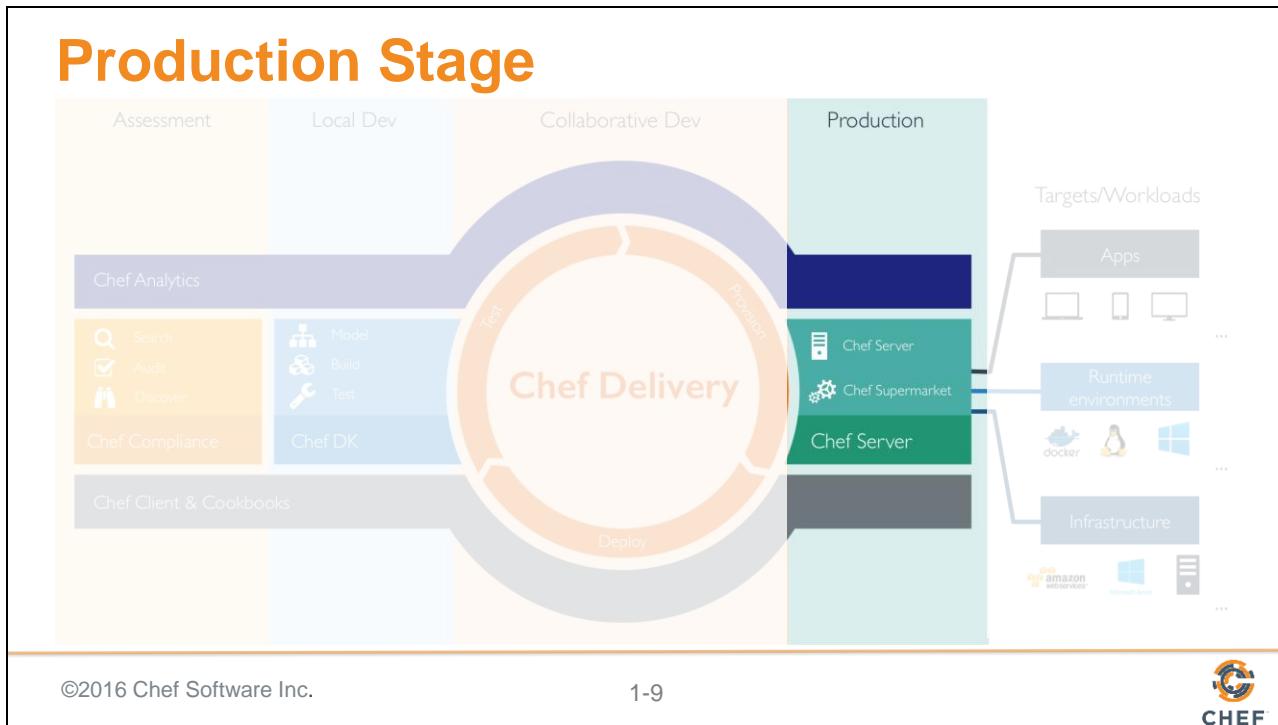
## Slide 8



After local development, you typically enter a stage of collaborative development. This is when all your teams work together to test, provision, and deploy your infrastructure. You can streamline these stages by leveraging Chef Delivery, a single platform that promotes collaboration.

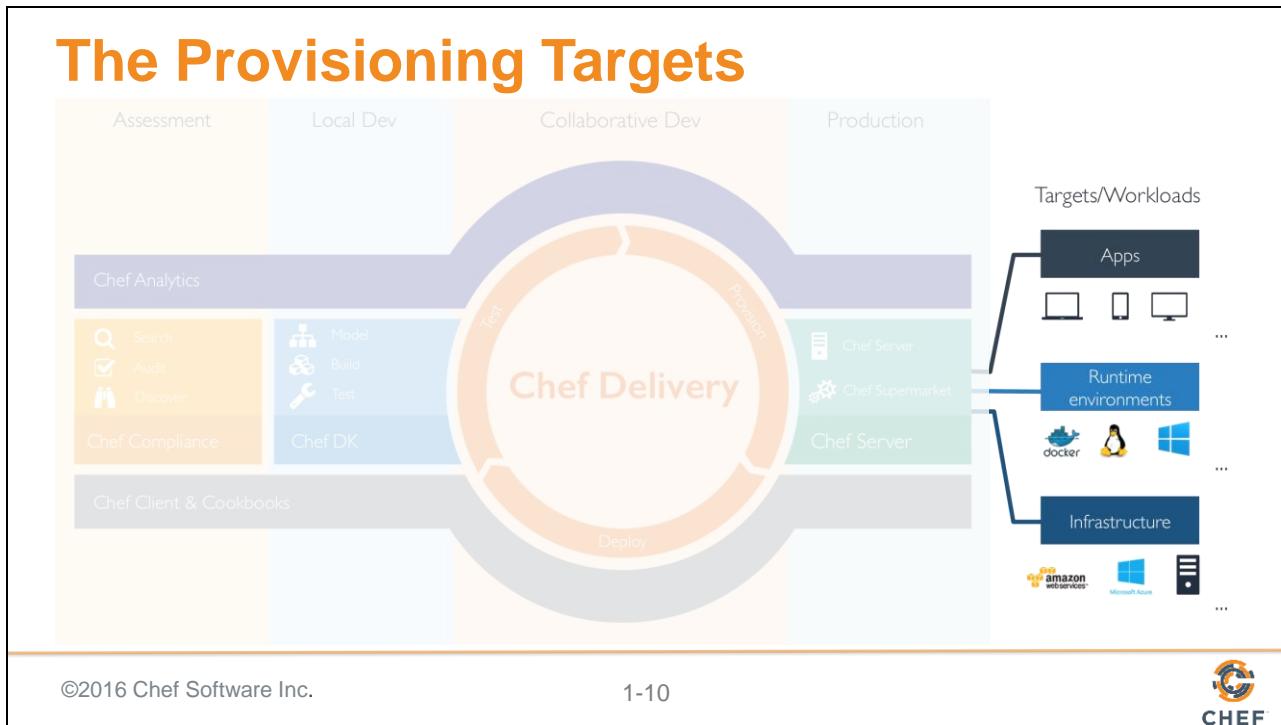
Chef Delivery accelerates the adoption of continuous delivery and encourages DevOps collaboration by providing a proven, reproducible workflow for managing changes as they flow from your local workstation through a pipeline into production.

## Slide 9



Once your team tests, provisions, and deploys your infrastructure using Chef Delivery, it's ready to be uploaded to the Chef Server.

Slide 10



You provision your target environments/infrastructure using the code that funneled through the pipeline.

Slide 11

## Agenda: Day 1



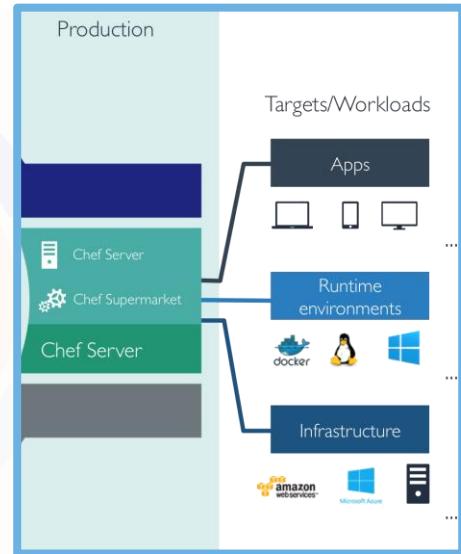
- Using Chef resources
- Building Chef cookbooks
- Collecting details about the system
- Managing data with templates
- Workstation setup

This training first focuses on local development. We will use the Chef Development Kit to model infrastructure, building solutions with Chef resources within recipes that we store in cookbooks, and then I will demonstrate how to test those cookbooks.

Slide 12

## Agenda: Day 2

- Connecting to Chef Server
- Community cookbooks
- Managing multiple nodes
- Roles
- Searching the Chef Server
- Environments



## Slide 13

## Course Objectives

You will leave this class with a basic understanding of Chef's core components, architecture, and commonly used tools. After completing this course, you should be able to:

- Write Chef recipes with Chef Resources that model the desired state of a system
- Manage these recipes in cookbooks that you are able to apply to a system
- Add multiple nodes to be managed by a Chef Server
- Manage the deployment of cookbooks to nodes with Roles and Environments

Slide 14

# EXERCISE

## Introductions



*Now that we have gotten the introductions out of the way we can get to work.*

**Objective:**

- ✓ Introduce ourselves
- ✓ Introduce this training experience

Slide 15

# EXERCISE



## Pre-built Workstation

*We will provide for you a workstation with all the tools installed.*

**Objective:**

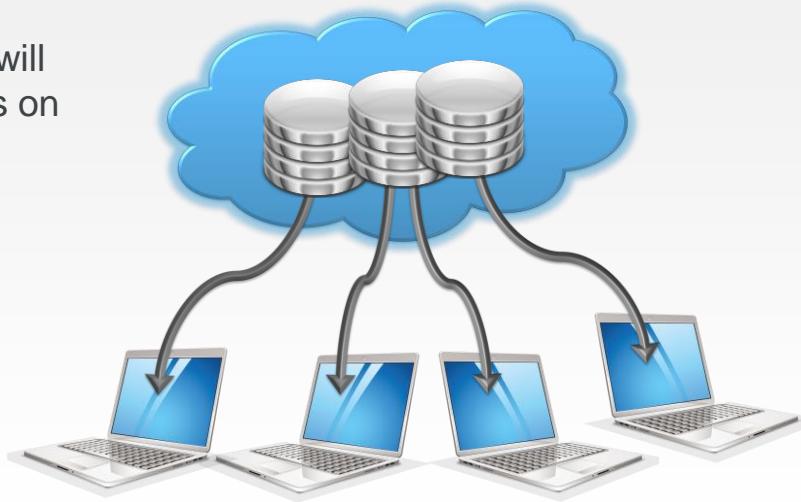
- Login to the Remote Workstation
- Select a Text Editor

As I mentioned there is a lot work planned for the day. To ensure we focus on the concepts and not on troubleshooting systems we are providing you a workstation with the necessary tools installed to get started right away.

Slide 16

## Setting up Your Workstation

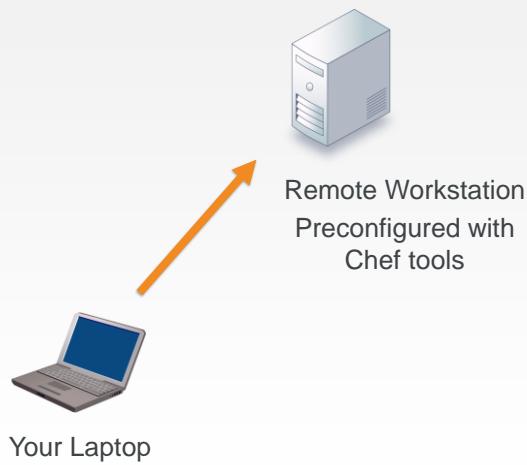
At the end of the day we will install the necessary tools on your workstation.



At the end of the day we have set aside time to install the necessary tools on your local computer. If you already have the tools we will ensure that they are working correctly and troubleshoot any issues to ensure you have a smooth experience for when you leave this training.

Slide 17

## Chef Lab System Architecture



For now you are going to use your local computer to connect to a remote workstation. This workstation has all the necessary tools installed.

## Logging in to the Workstation



```
> ssh IPADDRESS -l USERNAME
```

```
The authenticity of host '54.209.164.144 (54.209.164.144)' can't
be established. RSA key fingerprint is
SHA256:tKoTsPbn6ER9BLThZqntXTxIYem3zV/iTQWvhLrBIBQ. Are you sure
you want to continue connecting (yes/no)? yes
chef@54.209.164.144's password: PASSWORD
chef@ip-172-31-15-97 ~]$
```

We will provide you with the address, username and password of the workstation. With that information you will need to use the SSH tool that you have installed to connect that workstation. On Windows you should use an SSH client like PuTTY to connect to the remote workstation that we assign to you. You'll need to ssh into your assigned workstation in order to issue Chef commands.

This demonstrates how you might connect to the remote machine using your terminal or command-prompt if you have access to the application ssh. This may be different based on your operating system.

Slide 19

# EXERCISE



## Pre-built Workstation

*We will provide for you a workstation with all the tools installed.*

**Objective:**

- ✓ Login to the Remote Workstation
- ❑ Select a Text Editor

Now that you are connected to that workstation we have taken care of all the necessary work to get started with the training.

## Choose an Editor

You'll need to choose an editor to edit files:

*Tips for using these editors can be found below in your participant guide.*

**emacs**

**nano**

**vi / vim**

During this course we are going to use the text-based editors installed on these virtual workstations. There are at least three command-line editors that we can choose from on the Linux workstation: Emacs, Nano, or Vim.

**Emacs:** (Emacs is fairly straightforward for editing files.)

```
OPEN FILE    $ emacs FILENAME  
WRITE FILE   ctrl+x, ctrl+w  
EXIT        ctrl+x, ctrl+c
```

**Nano:** (Nano is usually touted as the easiest editor to get started with editing through the command-line.)

```
OPEN FILE    $ nano FILENAME  
WRITE (When exiting) ctrl+x, y, ENTER  
EXIT        ctrl+x
```

**VIM:** (Vim, like vi, is more complex because of its different modes. )

```
OPEN FILE    $ vim FILENAME  
START EDITING    i  
WRITE FILE    ESC, :w  
EXIT        ESC, :q  
EXIT (don't write)    ESC, :q!
```

Slide 21

# EXERCISE



## Pre-built Workstation

*We will provide for you a workstation with all the tools installed.*

**Objective:**

- ✓ Login to the Remote Workstation
- ✓ Select a Text Editor

Now that you are logged into the machine and have an editor we are ready to get started.

Slide 22



## 2: Chef Resources

# Chef Resources

Chef's Fundamental Building Blocks

## Slide 2

# Objectives

After completing this module, you should be able to:

- Use Chef to install packages on your virtual workstation
- Use the chef-client command
- Create a basic Chef recipe file
- Define Chef Resources

In this module you will learn how to install packages on a virtual workstation, use the 'chef-client' command, create a basic Chef recipe file and define Chef Resources.

Slide 3

# EXERCISE



## Time for Some Fun!

*The workstation needs a little personal touch; something that makes it a little more fun.*

**Objective:**

- Write a recipe that installs the 'cowsay' package
- Apply the recipe to the workstation
- Use 'cowsay' to say something

I have given you a workstation with a number of tools installed but it is missing something delightful to make the system fun. Together let's walk through using Chef to install the 'cowsay' package.

## Slide 4

## Learning Chef

One of the best ways to learn a technology is to apply the technology in every situation that it can be applied.

A number of chef tools are installed on the system so lets put them to use.

For those comfortable with Linux distributions it seems rather straight forward to installing packages through the distribution's specific package manager. This is a perfect opportunity to experiment with how to solve configuration problems with Chef. For those not familiar with Linux distributions do not worry, Chef will take care of figuring out those details for us when it comes time to do the installation of the package.

One of the best ways to learn a technology is to apply the technology in every situation that it can be applied. A number of chef tools are installed on the system so lets put them to use.

## Slide 5

# REFERENCE

## Resources



A resource is a statement of configuration policy.

It describes the desired state of an element of your infrastructure and the steps needed to bring that item to the desired state.

<https://docs.chef.io/resources.html>

First, let's look at Chef's documentation about resources. Visit the docs page on resources and read the first three paragraphs.

Afterwards, let us look at a few examples of resources.

Slide 6

## Example: Package

```
package 'httpd' do
  action :install
end
```

The package named 'httpd' is installed.

[https://docs.chef.io/resource\\_package.html](https://docs.chef.io/resource_package.html)

Here is an example of the package resource. The package named 'httpd' is installed.

Slide 7

## Example: Service

```
service 'ntp' do
  action [ :enable, :start ]
end
```

The service named 'ntp' is enabled (start on reboot) and started.

[https://docs.chef.io/resource\\_service.html](https://docs.chef.io/resource_service.html)

In this example, the service named 'ntp' is enabled and started.

## Slide 8

## Example: File

```
file '/etc/motd' do
  content 'This computer is the property ...'
end
```

The file name '/etc/motd' is created with content 'This computer is the property ...'

[https://docs.chef.io/resource\\_file.html](https://docs.chef.io/resource_file.html)

In this example, the file named '/etc/motd' is created with content 'This company is the property...'

Slide 9

## Example: File

```
file '/etc/php.ini.default' do
  action :delete
end
```

The file name '/etc/php.ini.default' is deleted.

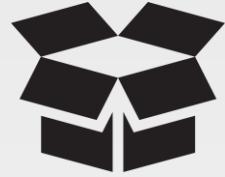
[https://docs.chef.io/resource\\_file.html](https://docs.chef.io/resource_file.html)

In this example, the file named '/etc/php.ini.default' is deleted.

Slide 10

# CONCEPT

## Resource Definition

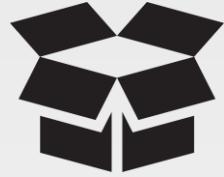


```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

Let's take a moment and talk about the structure of a resource definition. We'll break down the resource that we defined in our recipe file.

# CONCEPT



## Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

The first element of the resource definition is the resource type. In this instance the type is 'file'. Earlier we used 'package'. We showed you an example of 'service'.

# CONCEPT



## Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

The second element is the name of the resource. This is also the first parameter being passed to the resource.

In this instance the resource name is also the relative file path to the file we want created. We could have specified a fully-qualified file path to ensure the file was written to the exact same location and not dependent on our current working directory.

# CONCEPT



## Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

The `do` and `end` keywords here define the beginning of a ruby block. The ruby block and all the contents of it are the second parameters to our resource.

The contents of this block contains properties (and other things) that help describe the state of the resource. In this instance, the content attribute here specifies the contents of the file.

Properties are laid out with the name of the property followed by a space and then the value for the attribute.

# CONCEPT



## Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```



The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

The interesting part is that there is no action defined. And if you think back to the previous examples that we showed you, not all of the resources have defined actions.

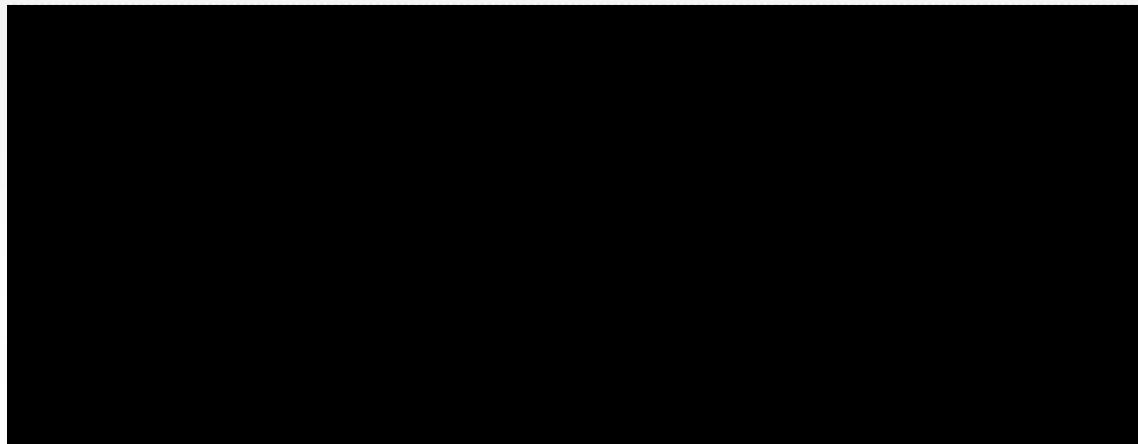
When an action is not specified a default action is chosen. The default action is often times will not surprise you in most cases and perform an action that is creative or additive to the system. In this instance the default action for the file resource is to create the file if it does not exist.

Slide 15

## Opening a Recipe File with an Editor



```
> nano moo.rb
```



Now that we have seen a few examples of resources let's get to work installing that package. Using your editor of choice open up a file named 'moo.rb'.

Slide 16

## Adding a Resource to the Recipe



~/moo.rb

```
package 'cowsay' do
  action :install
end
```

In the file add the following resource to install the 'cowsay' package.

Slide 17

# EXERCISE



## Time for Some Fun!

*The workstation needs a little personal touch; something that makes it a little more fun.*

**Objective:**

- Write a recipe that installs the 'cowsay' package
- Apply the recipe to the workstation
- Use 'cowsay' to say something

Save the file and return back to the shell. It is now time to apply the recipe to the workstation.

## Slide 18

# CONCEPT

## chef-client



chef-client is an agent that runs locally on every node that is under management by Chef.

When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state.

[https://docs.chef.io/chef\\_client.html](https://docs.chef.io/chef_client.html)

In the Chef Development Kit (ChefDK), we package a tool that is called 'chef-client'.

'chef-client' is a command-line application that can be used to apply a recipe file. 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 19

# CONCEPT



## --local-mode (or -z)

chef-client's default mode attempts to contact a Chef Server and ask it for the recipes to run for the given node.

We are overriding that behavior to have it work in a local mode.

'chef-client' has the default default behavior to communicate with a Chef server. So we use the '--local-mode' flag to ask 'chef-client' to look for the recipe file locally.

Slide 20

## Applying the Recipe



```
> sudo chef-client --local-mode moo.rb
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-02-19T13:08:13+00:00] WARN: Node ip-172-31-12-176.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files:::/home/chef/moo.rb
  * yum_package[nano] action install
    - install version 3.03-8.el6 of package cowsay
Running handlers:
Running handlers complete
Chef Client finished, 1/1 resources updated in 38 seconds
```

Execute the following command to have chef-client apply the recipe file. Because we are installing a package the prefix 'sudo' is necessary. This ensures that we have elevated our permissions to the appropriate level to install the package.

In the output you should see Chef installing the appropriate package.

Slide 21

# EXERCISE



## Time for Some Fun!

*The workstation needs a little personal touch; something that makes it a little more fun.*

**Objective:**

- ✓ Write a recipe that installs the 'cowsay' package
- ✓ Apply the recipe to the workstation
- ❑ Use 'cowsay' to say something

With the package installed it is time to use it.

Slide 22

## Running cowsay with a Message



```
> cowsay will moo for food
```

```
_____  
< will moo for food >  
-----  
      \  ^__^  
       \  (oo)\_____  
         (__)\       )\/\|  
           ||----w |  
           ||     ||
```

Run cowsay and give it a parameter or a few parameters. Enjoy your new bovine friend that will parrot back what you type into the shell.

Slide 23

# EXERCISE



## Time for Some Fun!

*The workstation needs a little personal touch; something that makes it a little more fun.*

**Objective:**

- ✓ Write a recipe that installs the 'cowsay' package
- ✓ Apply the recipe to the workstation
- ✓ Use 'cowsay' to say something

In this exercise we wrote a resource in a recipe file and applied that recipe file to the workstation. More importantly we brought a little more fun to our workstation.

Slide 24

# DISCUSSION



## Discussion

What would happen if you applied the recipe again?

What would happen if the package were to become uninstalled?

What would happen if you applied the recipe again? Before you execute the command to apply the recipe think about what will happen. Think about what you would want to happen. Look at the output from the previous execution. Then take a guess. Write it down or type out what you think will happen. Then execute the command again.

What would happen if the package were to become uninstalled? What would the output be if you applied the recipe again? Was there a situation where the package was already uninstalled and we applied this recipe?

Slide 25

# CONCEPT



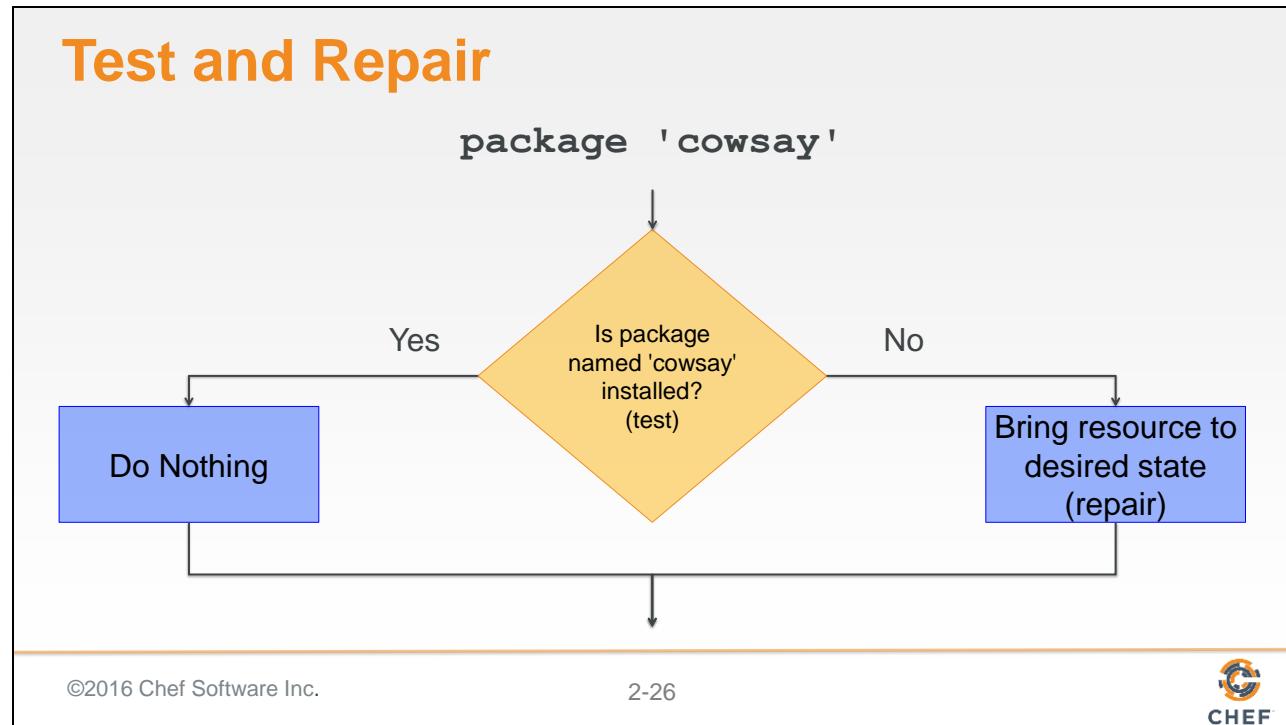
## Test and Repair

`chef-client` takes action only when it needs to. Think of it as test and repair.

Chef looks at the current state of each resource and takes action only when that resource is out of policy.

Hopefully it is clear from running the `chef-client` command a few times that the resource we defined only takes action when it needs to take action.

We call this test and repair. Test and repair means the resource first tested the system before it takes action.



If the package is already installed, then the resource does not need to take action.

If the package is not installed, then the resource NEEDS to take action to install that package.

Slide 27

# EXERCISE

## Hello, World?



*I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application.*

**Objective:**

- Create a recipe that writes out a file with the contents "Hello, world!"
- Apply that recipe to the workstation
- Verify the contents of the file

Great! You installed a package with `chef-client` but we missed a very important step.

Chef is written in Ruby. Ruby is a programming language and it is required that the first program you write in a programming language is 'Hello World'.

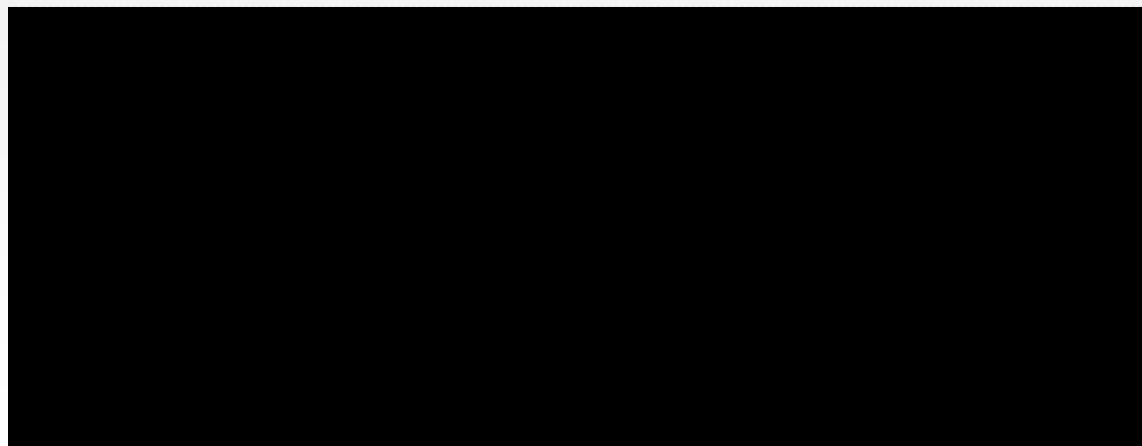
So let's walk through creating a recipe file that creates a file named 'hello.txt' with the contents 'Hello world!'.

Slide 28

## Opening a Recipe File with an Editor



```
> nano hello.rb
```



Using your editor open the file named 'hello.rb'. 'hello.rb' is a recipe file. It has the extension '.rb' because it is a ruby file.

Slide 29

## Adding a Resource to the Recipe



~/hello.rb

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

Add the resource definition displayed above. We are defining a resource with the type called 'file' and named 'hello.txt'. We also are stating what the contents of that file should contain 'Hello, world!'.

Slide 30

# EXERCISE

## Hello, World?



*I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application.*

**Objective:**

- ✓ Create a recipe that writes out a file with the contents "Hello, world!"
- ❑ Apply that recipe to the workstation
- ❑ Verify the contents of the file

Now that we have created the recipe file it is time to apply it.

## Slide 31

## Applying the Recipe



```
> sudo chef-client --local-mode hello.rb
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-02-19T13:08:13+00:00] WARN: Node ip-172-31-12-176.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files:::/home/chef/hello.rb
* file[hello.txt] action create
  - create new file hello.txt
  - update content in file hello.txt from non to 315f5b
    +++ ./hello.txt20160224-8559-19kqial
      2016-02-24 16:51:04.400844959 +0000
    @@ -1 +1,2 @@
    +Hello, world!
```

Using `chef-client` with the local mode flag we specify the new recipe file and apply it to the system. In this instance we are creating a file locally within the current directory and do not actually need to use the 'sudo' prefix.

Slide 32

# EXERCISE

## Hello, World?



*I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application.*

**Objective:**

- ✓ Create a recipe that writes out a file with the contents "Hello, world!"
- ✓ Apply that recipe to the workstation
- Verify the contents of the file

In the output it looks like the recipe we applied to the system created a hello.txt file. Now it is time to examine the contents.

Slide 33

## Looking at the Contents of a File



```
> cat /hello.txt
```

```
Hello, world!
```

Let's look at the contents of the 'hello.txt' file in the root directory to prove that it was created and the contents of file is what we wrote in the recipe. The result of the command should show you the contents 'Hello, world!'.

Slide 34

# EXERCISE

## Hello, World?



*I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application.*

**Objective:**

- ✓ Create a recipe that writes out a file with the contents "Hello, world!"
- ✓ Apply that recipe to the workstation
- ✓ Verify the contents of the file

Great. Again we created a recipe file with a resource and applied it to the system. This time it was a file and not a package but we can start to see that with Chef there are many different resources that we can use to express the desired state of the system.

Slide 35

# DISCUSSION



## Discussion

What is a resource?

What are some other possible examples of resources?

How did the example resources we wrote describe the desired state of an element of our infrastructure?

What does it mean for a resource to be a statement of configuration policy?

Let's finish this Resources module with a discussion. Answer these four questions. Remember that the answer "I don't know! That's why I'm here!" is a great answer.

Slide 36

# DISCUSSION



## Q&A

What questions can we answer for you?

- Resources
- chef-client
- Test and Repair

What questions can we answer for you?

About anything or specifically about:

- chef-client
- resources
- Test and Repair

Slide 37



### 3: Cookbooks

## Cookbooks

Organizing Recipes

## Slide 2

# Objectives

After completing this module, you should be able to:

- Generate a Chef cookbook
- Generate a Chef recipe
- Applying a run-list of recipes to a system

In this module you will learn how to modify a recipe, use version control, generate a Chef cookbook and define a Chef recipe that sets up a web server.

Slide 3

# EXERCISE

## Setting up the Workstation



*Time to create a recipe that sets up the workstation that we can share with others.*

**Objective:**

- Create a cookbook
- Create a setup recipe within the cookbook
- Apply the recipe to the workstation
- Verify the workstation has been setup

Now that you've practiced creating a recipe file that installs an application with a package resource and a file resource it is time that we put together a recipe that can setup the workstation and future workstations in the exact same way. But instead of creating any more single recipes we are going to create a cookbook to store this recipe. When you are done defining the policy apply the policy to the system.

## Slide 4

# REFERENCE

## Cookbooks



A Chef cookbook is the fundamental unit of configuration and policy distribution. Contains a README, version number, dependency information, recipes and other files that support that common scenario.

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

A cookbook is a structure that contains recipes. It also contains a number of other things -- right now we are most interested in finding a home for our recipes, giving them a version, and providing a README to help describe them.

## Slide 5

# CONCEPT

## Cookbook



Each cookbook defines a scenario, such as everything needed to install and configure an application, and then it contains all of the components that are required to support that scenario.

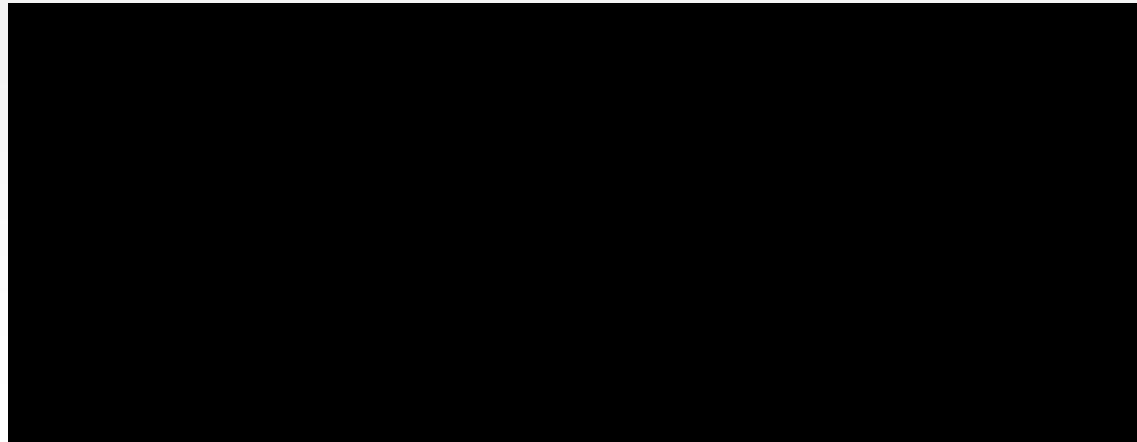
A cookbook usually maps 1:1 to an application or to a scenario. When we define a cookbook we usually have a goal in mind that this cookbook will accomplish. In our case we are interested in a cookbook that configures a workstation. So within that cookbook we will define all the recipes to accomplish this goal.

Slide 6

## Creating a Directory for Cookbooks



```
> mkdir cookbooks
```



Storing our recipes within a cookbook sounds like a better idea than storing them in a scripts directory. To prepare for the cookbook we are about to create and the other cookbooks we will also be creating today it would be a good idea to store them in shared directory.

Create a directory named cookbooks.

## Slide 7

# CONCEPT



## What is 'chef'?

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

Cookbooks are nothing more than directories with specific files. We could create a cookbook by hand by finding an example and copying that pattern or we could use a tool to help us generate a cookbook.

The Chef Development Kit (Chef DK) comes with a tool named 'chef'. This command-line tool has a number of features.

## Slide 8

## Executing chef with the help flag



```
> 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
```

'chef' is a command-line application that does quite a few things. The most important thing to us right now is its ability to generate cookbooks and components.

Slide 9

## Executing chef generate with the help flag



```
> 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

Let's examine the 'chef generate' command. We can see that the command is capable of generating a large number of different things for us. It looks like if we want to generate a cookbook we're going to need to use 'chef generate cookbook'.

Slide 10

## Generating a cookbook



```
> chef generate cookbook cookbooks/workstation
```

```
Compiling Cookbooks...
```

```
Recipe: code_generator::cookbook
```

```
* directory[/home/chef/cookbooks/workstation] action create
  - create new directory /home/chef/cookbooks/workstation
* template[/home/chef/cookbooks/workstation/metadata.rb] action
  create_if_missing
  - create new file /home/chef/cookbooks/workstation/metadata.rb
  - update content in file
/home/chef/cookbooks/workstation/metadata.rb from none to 0c09e4
  (diff output suppressed by config)
* template[/home/chef/cookbooks/workstation/README.md] action
```

We have you covered. Call the cookbook *workstation*. When you specify the command we provide the name of the cookbook with the proceeding cookbooks path. This is because we are in the home directory and we want the cookbook to be generated in the cookbooks directory we recently created.

Slide 11

# EXERCISE

## Setting up the Workstation



*Time to create a recipe that sets up the workstation that we can share with others.*

**Objective:**

- ✓ Create a workstation cookbook
- Create a setup recipe within the cookbook
- Apply the recipe to the workstation
- Verify the workstation has been setup

## Slide 12

## Executing chef generate with the help flag



```
> 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

Let's examine the 'chef generate' command. We can see that the command is capable of generating a large number of different things for us. It looks like if we want to generate a cookbook we're going to need to use 'chef generate cookbook'.

## Executing chef generate with the help flag



```
> chef generate recipe --help
```

```
Usage: chef generate recipe [path/to/cookbook] 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
      -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH
      for the code_generator cookbook
      --generator-cookbook
```

Let's examine the 'chef generate' command. We can see that the command is capable of generating a large number of different things for us. It looks like if we want to generate a cookbook we're going to need to use 'chef generate cookbook'.

## Generating a recipe within the cookbook



```
> chef generate recipe cookbooks/workstation setup
```

```
Compiling Cookbooks...
Recipe: code_generator::recipe
  * directory[cookbooks/workstation/spec/unit/recipes] action
    create (up to date)
      * cookbook_file[cookbooks/workstation/spec/spec_helper.rb]
        action create_if_missing (up to date)
          *
          template[cookbooks/workstation/spec/unit/recipes/setup_spec.rb]
            action create_if_missing
              - create new file
                cookbooks/workstation/spec/unit/recipes/setup_spec.rb
                  - update_content_in_file cookbooks/workstation/spec/unit/
```

## Defining the Setup Recipe

```
~/cookbooks/workstation/recipes/setup.rb

#
# Cookbook Name:: workstation
# Recipe:: setup
#
# Copyright (c) 2016 The Authors, All Rights Reserved.

package 'tree'

file '/etc/motd' do
  content 'Property of ...'
end
```

Here is a version of the recipe file that installs tree, and creates the message-of -the-day file.

Slide 16

# EXERCISE

## Setting up the Workstation



*Time to create a recipe that sets up the workstation that we can share with others.*

**Objective:**

- ✓ Create a workstation cookbook
- ✓ Create a setup recipe within the cookbook
- ❑ Apply the recipe to the workstation
- ❑ Verify the workstation has been setup

## Slide 17

# CONCEPT

## chef-client



chef-client is an agent that runs locally on every node that is under management by Chef.

When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state.

[https://docs.chef.io/chef\\_client.html](https://docs.chef.io/chef_client.html)

In the Chef Development Kit (ChefDK), we package a tool that is called 'chef-client'.

'chef-client' is a command-line application that can be used to apply a recipe file. 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 18

# CONCEPT



## --runlist "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]**.

Another flag we can use is '--run-list' or '-r' to specify a list of recipes we want to apply to the system. We call this list of recipes a run list.

This ordered list specifies the recipes in a different way. We are no longer interested in the filepath to the particular recipe file. We instead specify that we want a recipe and then within the square brackets we specify the name of the cookbook and then finally the name of the recipe.

"**recipe[COOKBOOK::RECIPE]**"

COOKBOOK means the name of the Cookbook.

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

## Applying the workstation's setup recipe



```
> sudo chef-client --local-mode --runlist "recipe[workstation::setup]"
```

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: ["workstation::setup"]
Synchronizing Cookbooks:
  - workstation (0.1.0)
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install
    - install version 1.5.3-3.el6 of package tree
  * file[/etc/motd] action create
```

Slide 20

# EXERCISE

## Setting up the Workstation



*Time to create a recipe that sets up the workstation that we can share with others.*

**Objective:**

- ✓ Create a workstation cookbook
- ✓ Create a setup recipe within the cookbook
- ✓ Apply the recipe to the workstation
- Verify the workstation has been setup

Slide 21

## Using the tree application



```
> tree cookbooks/workstation
```

```
|   ├── Berksfile  
|   ├── chefignore  
|   ├── metadata.rb  
|   ├── README.md  
|   ├── recipes  
|   |   ├── default.rb  
|   |   └── setup.rb  
|   ├── spec  
|   |   ├── spec_helper.rb  
10 directories, 11 files
```

Slide 22

## Viewing the Message of the Day file



```
> cat /etc/motd
```

```
Property of ...
```

Slide 23

# EXERCISE

## Setting up the Workstation



*Time to create a recipe that sets up the workstation that we can share with others.*

**Objective:**

- ✓ Create a workstation cookbook
- ✓ Create a setup recipe within the cookbook
- ✓ Apply the recipe to the workstation
- ✓ Verify the workstation has been setup

Slide 24

# DISCUSSION



## Discussion

What file would you read first when examining a cookbook? second?

What other recipes might you include workstation cookbook?

Answer these questions.

Slide 25

# DISCUSSION



## Q&A

What questions can we answer for you?

- Cookbooks
- Recipes
- Run-lists

What questions can we help you answer?

General questions or more specifically about cookbooks, versioning and version control.

Slide 26



## 4: Ohai

# Ohai

Finding and Displaying Information About Our System

## Slide 2

# Objectives

After completing this module, you should be able to

- Capture details about a system
- Use the node object within a recipe
- Use Ruby's string interpolation
- Update the version of a cookbook

In this module you will learn how to capture details about a system, use the node object within a recipe, use Ruby's string interpolation, and update the version of a cookbook.

## Slide 3

# MOTIVATION



## Managing a Large Number of Servers

Have you ever had to manage a large number of servers that were almost identical?

How about a large number of identical servers except that each one had to have host-specific information in a configuration file?

Have you ever had to manage a large number of servers that were almost identical? How about a large number of identical servers except that each one had to have host-specific information in a configuration file?

The file needed to have the hostname or the IP address of the system. Maybe you needed to allocate two-thirds of available system memory into HugePages for a database. Perhaps you needed to set your thread max to number of CPUs minus one. The uniqueness of each system required you to define custom configuration files. Custom configurations that you need to manage by hand.

Slide 4

# MOTIVATION



## Some Useful System Data

- IP Address
- hostname
- memory
- CPU - MHz

Thinking about some of the scenarios it would be useful to capture: The IP address, hostname, memory, and CPU megahertz of our current system.

## Slide 5

# EXERCISE



## Details About the Node

*Displaying system details in the MOTD definitely sounds useful.*

**Objective:**

- Discover attributes about the system with Ohai
- Update the MOTD file contents, in the "workstation" cookbook, to include node details
- Update the cookbook's version number
- Apply the updated recipe and verify the results

Let's first explore the Ohai tool and show how it can find out details about the system. Then we will update our MOTD file so that it displays that information whenever we log into the system. This change to the cookbook is a new feature so we should talk about the process of updating the version number. Finally we should apply the updated recipe to the system to ensure that everything works correctly.

## Slide 6

# CONCEPT

## Ohai!



Ohai is a tool that is used to detect attributes on a node, and then provide these attributes to the chef-client at the start of every chef-client run. Ohai is required by the chef-client and must be present on a node. (Ohai is installed on a node as part of the chef-client install process.)

<http://docs.chef.io/ohai.html>

Ohai is a tool that detects and captures attributes about our system. Attributes like the ones we want to capture and display.

## Slide 7

# CONCEPT

## All About The System



Ohai queries the operating system with a number of commands, similar to the ones demonstrated.

The data is presented in JSON (JavaScript Object Notation).

Ohai, the command-line application, will output all the system details represented in JavaScript Object Notation (JSON).

Slide 8

## Running Ohai to Show All Attributes



&gt; ohai

```
{  
  "kernel": {  
    "name": "Linux",  
    "release": "2.6.32-431.1.2.0.1.el6.x86_64",  
    "version": "#1 SMP Fri Dec 13 13:06:13 UTC 2013",  
    "machine": "x86_64",  
    "os": "GNU/Linux",  
    "modules": {  
      "veth": {  
        "size": "5040",  
        "refcount": "0"  
      },  
      "ipt_addrtype": {  
        "size": "5040",  
        "refcount": "0"  
      }  
    }  
  }  
}
```

Ohai is also a command-line application that is part of the ChefDK. When you run it you will see the entire JSON representation of the system.

Slide 9

## Running Ohai to Show the IP Address



```
> ohai ipaddress
```

```
[  
  "172.31.57.153"  
]
```

You can also run ohai with a parameter. In this case when we want only the ipaddress from the entire body of information we can provide it as a parameter.

Slide 10

## Running Ohai to Show the Hostname



```
> ohai hostname
```

```
[  
  "ip-172-31-57-153"  
]
```

Similar, we can specify the hostname to return only the hostname of the system.

Slide 11

## Running Ohai to Show the Memory



&gt; ohai memory

```
{  
  "swap": {  
    "cached": "0kB",  
    "total": "0kB",  
    "free": "0kB"  
  },  
  "total": "604308kB",  
  "free": "297940kB",  
  "buffers": "24824kB",  
  "cached": "198264kB",
```

When we ask for the memory of the system we receive a hash that contains a number of keys and values.

Slide 12

## Running Ohai to Show the Total Memory



```
> ohai memory/total
```

```
[  
  "604308kB"  
]
```

We can grab a single value, like the total memory, by specifying a slash between the top-level key and the next level key underneath it. This command will return the total memory of the system.

Slide 13

## Running Ohai to Show the CPU



&gt; ohai cpu

```
{  
  "0": {  
    "vendor_id": "GenuineIntel",  
    "family": "6",  
    "model": "45",  
    "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",  
    "stepping": "7",  
    "mhz": "1795.673",  
    "cache_size": "20480 KB",  
    "physical_id": "34"
```

We can return all the details about the cpu. We see that there is one cpu, named '0', that contains more information.

Slide 14

## Running Ohai to Show the First CPU



&gt; ohai cpu/0

```
{  
  "vendor_id": "GenuineIntel",  
  "family": "6",  
  "model": "45",  
  "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",  
  "stepping": "7",  
  "mhz": "1795.673",  
  "cache_size": "20480 KB",  
  "physical_id": "34",  
  "core_id": "0",  
  "cores": "1"
```

Here we are asking for all the details about the cpu named '0'.

Slide 15

## Running Ohai to Show the First CPU Mhz



```
> ohai cpu/0/mhz
```

```
[  
  "1795.673"  
]
```

And finally if we wanted to display the Megahertz of that specific cpu we can append an additional key to the parameter.

Slide 16

# EXERCISE



## Details About the Node

*Now it is time to explore how we can use Ohai data in the recipe.*

**Objective:**

- ✓ Discover attributes about the system with Ohai
- Update the MOTD file contents, in the "workstation" cookbook, to include node details
- Update the cookbook's version number
- Apply the updated recipe and verify the results

Ohai finds and provides these values to us on the command line. But we want to use these values within our recipe. Let's walk through using the Ohai data in the recipe.

Slide 17

# CONCEPT

**ohai + chef-client = <3**



chef-client automatically executes ohai and stores the data about the node in an object we can use within the recipes named node.

<http://docs.chef.io/ohai.html>

These values are available in our recipes because `chef-client` automatically execute Ohai. This information is stored within a variable we call 'the node object'

Slide 18

# CONCEPT



## The Node Object

The node object is a representation of our system. It stores all the attributes found about the system.

<http://docs.chef.io/nodes.html#attributes>

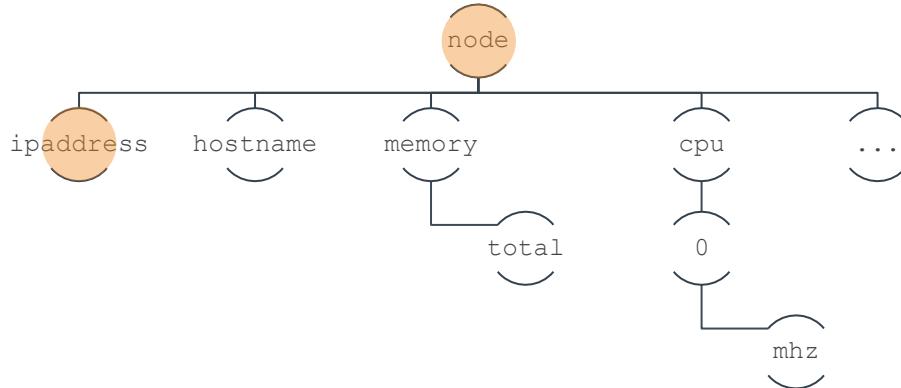
The node object is a representation of our system. It stores all these attributes found about the system. It is available within all the recipes that we write to assist us with solving the similar problems we outlined at the start.

An attribute is a specific detail about a node, such as an IP address, a host name, a list of loaded kernel modules, the version(s) of available programming languages that are available, and so on.

Let's look at using the node object to retrieve the ipaddress, hostname, total memory, and cpu megahertz.

Slide 19

## The Node



```
IPADDRESS: 104.236.192.102
```

```
"IPADDRESS: #{node['ipaddress']} "
```

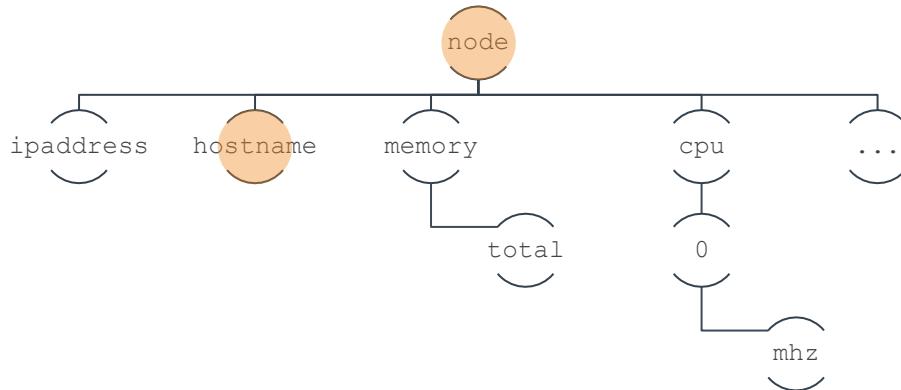
This is the visualization of the node attributes as a tree. That is done here to illustrate that the node maintains a tree of attributes that we can request from it.

The shaded text near the bottom of this slide is the hard-coded value we currently have in the file resource's content attribute.

At the very bottom is an example of how we could use the node's dynamic value within a string instead of the hard-coded one.

Slide 20

## The Node



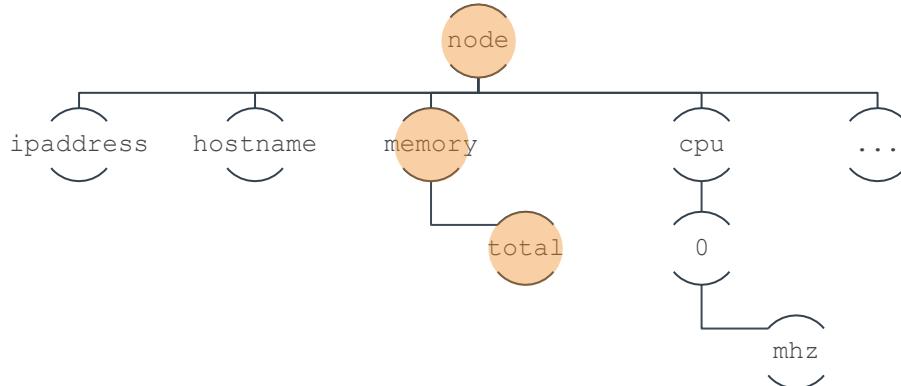
**HOSTNAME: banana-stand**

```
"HOSTNAME: #{node['hostname']}
```

The node maintains a `hostname` attribute. This is how we retrieve and display it.

Slide 21

## The Node



**MEMORY: 502272kB**

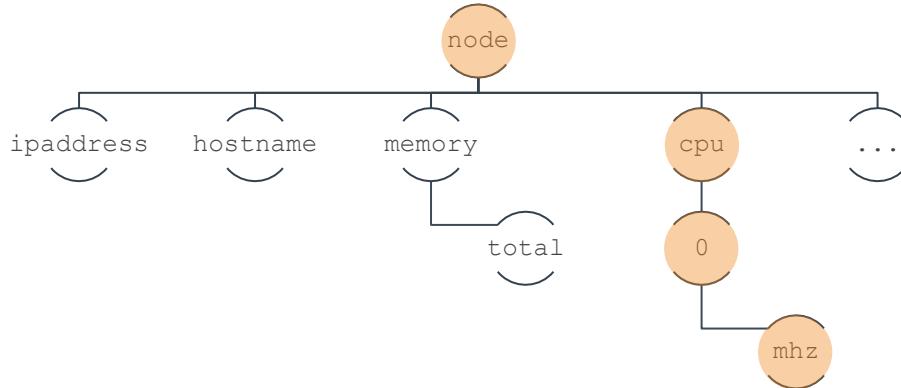
```
"MEMORY: #{node['memory']['total']}
```

The node contains a top-level value `memory` which has a number of child elements. One of those child elements is the total amount of system memory.

Accessing the node information is different. We retrieve the first value `'memory'`, returning a subset of keys and values at that level, and then immediately select to return the total value.

Slide 22

## The Node



CPU: 2399.998 MHz

```
"CPU: #{node['cpu']['0']['mhz']} MHz"
```

And finally, here we return the megahertz of the first CPU.

Slide 23

# CONCEPT

## String Interpolation



```
I have 4 apples
```

```
apple_count = 4  
puts "I have #{apple_count} apples"
```

[http://en.wikipedia.org/wiki/String\\_interpolation#Ruby](http://en.wikipedia.org/wiki/String_interpolation#Ruby)

In all of the previous examples, at the bottom of the slide, we demonstrated retrieving the values and displaying them within a string using a ruby language convention called string interpolation.

Slide 24

# CONCEPT

## String Interpolation



```
I have 4 apples
```

```
apple_count = 4  
puts "I have #{apple_count} apples"
```



String interpolation is only possible with strings that start and end with double-quotes.

Slide 25

# CONCEPT

## String Interpolation



```
I have 4 apples
```

```
apple_count = 4  
puts "I have #{apple_count} apples"
```



To escape out to display a ruby variable or ruby code you use the following sequence: number sign, left curly brace, the ruby variables or ruby code, and then a right curly brace.

## Updating the content Property

```
~/cookbooks/workstation/recipes/setup.rb

# ... PACKAGE RESOURCES ...
file '/etc/motd' do
  content "Property of ...

  IPADDRESS: #{node['ipaddress']}
  HOSTNAME : #{node['hostname']}
  MEMORY    : #{node['memory']['total']}
  CPU       : #{node['cpu'][0]['mhz']}
"
end
```

Let's update the content property first to use double quotes. Then we want to use String Interpolation and insert the value of the various node object values for ipaddress, hostname, memory, and cpu.

Slide 27

# EXERCISE



## Details About the Node

*Now that we have added a new feature it is time to update the cookbook's version number.*

**Objective:**

- ✓ Discover attributes about the system with Ohai
- ✓ Update the MOTD file contents, in the "workstation" cookbook, to include node details
- Update the cookbook's version number
- Apply the updated recipe and verify the results

Now that the MOTD file has been update it is time to update the version of our cookbook. As a cookbook author you are responsible to versioning the cookbook.

Slide 28

# CONCEPT

## Cookbook Versions



A cookbook version represents a set of functionality that is different from the cookbook on which it is based.

[https://docs.chef.io/cookbook\\_versions.html](https://docs.chef.io/cookbook_versions.html)

A version may exist for many reasons, such as ensuring the correct use of a third-party component, updating a bug fix, or adding an improvement.

The first version of the cookbook displayed a simple property message in the /etc/motd. The changes that we finished are new features of the cookbook.

Slide 29

# CONCEPT

## Semantic Versions



Given a version number **MAJOR.MINOR.PATCH**, increment the:

- **MAJOR** version when you make incompatible API changes
- **MINOR** version when you add functionality in a backwards-compatible manner
- **PATCH** version when you make backwards-compatible bug fixes

<http://semver.org>

Cookbooks use semantic version. The version number helps represent the state or feature set of the cookbook. Semantic versioning allows us three fields to describe our changes: major; minor; and patch.

Major versions are often large rewrites or large changes that have the potential to not be backwards compatible with previous versions. This might mean adding support for a new platform or a fundamental change to what the cookbook accomplishes. Minor versions represent smaller changes that are still compatible with previous versions. This could be new features that extend the existing functionality without breaking any of the existing features. And finally Patch versions describe changes like bug fixes or minor adjustments to the existing documentation.

Slide 30

# DISCUSSION



## Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

So what kind of changes did you make to the cookbook? How could we best represent that in an updated version?

Slide 31

## Updating the Cookbook Version

~/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        '0.2.0'
```

Changing the contents of an existing resource--by adding the attributes of the node doesn't seem like a bug fix and it doesn't seem like a major rewrite. It is like a new set of features while remaining backwards compatible.

Edit the cookbook's metadata file and update the version's minor number to 0.2.0.

Slide 32

# EXERCISE



## Details About the Node

*Now let's apply this updated policy to the state of the system.*

**Objective:**

- ✓ Discover attributes about the system with Ohai
- ✓ Update the MOTD file contents, in the "workstation" cookbook, to include node details
- ✓ Update the cookbook's version number
- ❑ Apply the updated recipe and verify the results

The cookbook is ready to be applied to the system.

Slide 33

## Applying the workstation Cookbook



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

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: ["workstation::setup"]
Synchronizing Cookbooks:
  - workstation (0.2.0)
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create
    - update content in file /etc/motd from d100eb t
```

When we apply the cookbook we should see the updated version number and the file contents of the MOTD file has been changed.

Slide 34

## Verifying that the MOTD has been Updated



```
$ cat /etc/motd
```

```
Property of ...
```

```
IPADDRESS: 172.31.57.153
HOSTNAME : ip-172-31-57-153
MEMORY    : 604308kB
CPU       : 1795.673
```

We can logout and login to the remote workstation to see the new MOTD or we can look at the contents of the MOTD file with the following command.

Slide 35

# EXERCISE



## Details About the Node

*The workstation will now report back with information about it.*

**Objective:**

- ✓ Discover attributes about the system with Ohai
- ✓ Update the MOTD file contents, in the "workstation" cookbook, to include node details
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results

We learned about Ohai and its ability to find and return attributes about the system. We saw how we could use this information with our recipes through the node object. We made a change to the cookbook that provided a new, minor feature to the cookbook so the version number was updated. And we finally applied the updated policy to the system to see those changes.

Slide 36

# DISCUSSION



## Discussion

What is the major difference between a single-quoted string and a double-quoted string?

How are the details about the system available within a recipe?

How does the version number help convey information about the state of the cookbook?

Answer these questions.

Slide 37

# DISCUSSION



## Q&A

What questions can we help you answer?

- Ohai
- Node Object
- Node Attributes
- String Interpolation
- Semantic Versions

With that we have added all of the requested features.

What questions can we help you answer?

In general or about specifically about ohai, the node object, node attributes, string interpolation, or semantic versioning.

Slide 38



## 5: Using Templates

# Using Templates

Extracting the Content for Clarity

## Slide 2

# Objectives

After completing this module, you should be able to

- Explain when to use a template resource
- Create a template file
- Use ERB tags to display node data in a template
- Define a template resource

In this module you will learn how to understand when to use a template resource, create a template file, use ERB tags to display node data in a template, define a template resource.

Slide 3

# MOTIVATION

## Cleaner Recipes



In the last section we updated our cookbook to display information about our node.

We expanded the text within the file resource's content property.

The content that we defined in our recipe allowed us to display dynamic content about the node but this caused our recipe to lose some of the clarity that it once possessed.

## Slide 4

## Viewing the workstation's setup recipe

```
~/cookbooks/workstation/recipes/setup.rb

package 'tree'

file '/etc/motd' do
  content "Property of ...

IPADDRESS: #{node['ipaddress']}
HOSTNAME : #{node['hostname']}
MEMORY    : #{node['memory']['total']}
CPU       : #{node['cpu'][0]['mhz']}
"
end
```

What if we wanted to store more content within that content property? Right now we have a small subset of information that we may find helpful. This content would need to be arranged and carefully laid out within the property. Every change we would need to take special care as we edited or copied new content into the property. Let's talk about some of the possible pitfalls and see how using a Template resource will help us better manage this content.

Slide 5

# PROBLEM



## Double Quotes close Double Quotes

Double quoted strings are terminated by double quotes.

```
"<h1 style="color: red;">Hello, World!</h1>"
```



There are some things that you need to be careful of when working with double-quoted strings in Ruby:

Double-quoted strings are terminated by double-quotes so if any of the text that we paste into this content field has double quotes it is going to have to be escaped.

## Slide 6

# CONCEPT

## Backslash



We can use double-quotes as long as we prefix them with a backslash.

```
"<h1 style=\"color: red;\">"Hello, World!</h1>"
```



With Ruby strings you can use the backslash character as an escape character. In this case, if you wanted to have a double-quote inside a double-quoted string, you would need to place a backslash before the double-quote.

## Slide 7

# PROBLEM

## Backslash



Backslashes are reserved characters. So to use them you need to use a backslash.

"Root Path: \\



That also brings up an issue with continually-pasting text. You will also need to keep an eye out for backslash characters because backslash characters are now the escape character.

If you want to literally represent a backslash you'll need to use two-backslashes.

## Slide 8

# CONCEPT

## Backslash



Backslashes are reserved characters. So to use them you need to use a backslash.

```
"Root Path: \\"
```

So every time text is pasted into the string value of the content attribute, you will need to find and replace all backslashes with double-backslashes and then replace all double-quotes with backslash double-quotes.

## Slide 9

## This content property generates unexpected formatting

```
file '/etc/motd' do
  content 'This is the first line of the file.
           This is the second line. If I try and line it up...
'
end
```

Indented; better for humans to read

Placing the terminating quote on a new line  
makes it easier to find

```
/etc/motd
```

```
This is the first line of the file.
      This is the second line. If I try and line it up...
```

Unexpected indentation in the resulting file

Unexpected new line in resulting file

It is important to note that the file content may have some important formatting that might be easily overlooked when working with the content in a recipe file. For some configuration files that may require particular formatting (e.g. tabs; spaces; certain indentation) this could cause your configuration files to be read incorrectly causing your applications depending on that configuration to fail.

Besides that, if the size of the string value of the content field grows, it will consume the recipe--making it difficult to understand what is desired state and what is data.

Slide 10

# PROBLEM



## Copy Paste

This process is definitely error prone. Especially because a human has to edit the file again before it is deployed.

This could sound like a bug waiting to happen.

Any process that requires you to manually copy and paste values and then remember to escape out characters in a particular order, is likely going to lead to issues later when you deploy this recipe to production.

Slide 11

# CONCEPT



## What We Need

We need the ability to store the data in another file, which is in the native format of the file we are writing out but that still allows us to insert ruby code...

...specifically, the node attributes we have defined.

It is better to store this data in another file. The file would be native to whatever format is required so it you wouldn't need to escape any common characters.

But you still need a way to insert node attributes. So you really need a native file format that allows us to escape out to ruby.

Slide 12

# EXERCISE

## Cleaner Setup Recipe



*Adding all the information into the recipe did make it hard to read.*

**Objective:**

- Create a template with chef generate
- Define the contents of the ERB template
- Change the file resource to the template resource
- Update the cookbook's version number
- Apply the updated recipe and verify the results

So our objective is clear. We need to use a template resource and create a template and then link them together.

Let's start by creating the actual template file and then we will update the recipe.

Slide 13

# REFERENCE

## Template



A cookbook template is an Embedded Ruby (ERB) template that is used to generate files ... Templates may contain Ruby expressions and statements and are a great way to...

Use the template resource to add cookbook templates to recipes; place the corresponding Embedded Ruby (ERB) template in a cookbook's /templates directory.

[https://docs.chef.io/resource\\_template.html](https://docs.chef.io/resource_template.html)

Let's explore templates.

Reviewing the documentation, it seems as though it shares some similarities to the `cookbook_file` resource.

Slide 14

# REFERENCE

## Template



To use a template, two things must happen:

1. A template resource must be added to a recipe
2. An Embedded Ruby (ERB) template must be added to a cookbook

[https://docs.chef.io/resource\\_template.html#using-templates](https://docs.chef.io/resource_template.html#using-templates)

And if we look at the bottom section about "Using Templates", we'll see more information about what is required and how we can use them to escape out to execute ruby code.

Slide 15

## Showing chef generate's Help



```
> 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

Remember that application Chef--the one that generated our cookbooks. Well it is able to generate cookbook components as well.

Templates and files (for cookbook\_files) are a few of the other things it can generate for us.

Let's use help to review the command again. And let's ask for help about the 'generate' subcommand.

## Showing chef generate template's Help



```
> chef generate template --help
```

```
Usage: chef generate template [path/to/cookbook] NAME [options]
      -C, --copyright COPYRIGHT           Name of the copyright holder
      - defaults to 'The Authors'
      -m, --email EMAIL                  Email address of the author -
      defaults to ...
      -a, --generator-arg KEY=VALUE     Use to set arbitrary
      attribute KEY to VALUE in the
      -I, --license LICENSE             all_rights, apache2, mit,
      gplv2, gplv3 - defaults to
      -s, --source SOURCE_FILE          Copy content from SOURCE_FILE
      -g GENERATOR_COOKBOOK_PATH,       Use GENERATOR_COOKBOOK_PATH
      for the code generator
```

Finally let's ask for help for generating templates.

The command requires two parameters--the path to where the cookbook is located and the name of the template to generate. There are some other additional options but these two seem like the most important.

## Generating a motd Template



```
> chef generate template cookbooks/workstation motd
```

```
Compiling Cookbooks...
Recipe: code_generator::template
  * directory[cookbooks/workstation/templates/default] action
    create
      - create new directory cookbooks/workstation/templates/default
  * template[cookbooks/workstation/templates/default/motd.erb]
    action create
      - create new file
        cookbooks/workstation/templates/default/motd.erb
      - update content in file
        cookbooks/workstation/templates/default/motd.erb from none to
          e3b0c4
```

Use '**chef generate template**' to create a template in the workstation cookbook found in the cookbooks/apache directory and the file we want to create is named index.html.

Slide 18

## Examining the templates Directory



```
> tree cookbooks/workstation/templates
```

```
cookbooks/workstation/templates/
└── default
    └── motd.erb
```

```
1 directory, 1 file
```

That is the first step. Now that the template exists, we are ready to define the content within the template file.

Slide 19

# EXERCISE

## Cleaner Recipes



*Now it is time to populate the template file*

**Objective:**

- ✓ Create a template with chef generate
- Define the contents of the ERB template
- Change the file resource to the template resource
- Update the cookbook's version number
- Apply the updated recipe and verify the results

Now we need to understand what ERB means and how we can define our content in the template with ERB.

Slide 20

# CONCEPT

## ERB



An Embedded Ruby (ERB) template allows Ruby code to be embedded inside a text file within specially formatted tags.

Ruby code can be embedded using expressions and statements.

<https://docs.chef.io/templates.html#variables>

ERB template files are special files because they are the native file format we want to deploy but we are allowed to include special tags to execute ruby code to insert values or logically build the contents.

Slide 21

## Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Each ERB tag has a beginning tag and a matched ending tag.

Here is an example of a text file that has several ERB tags defined in it.

Slide 22

## Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Each ERB tag has a beginning tag and a matched ending tag.

Each ERB tag has a beginning tag and an ending tag.

Slide 23

## Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Each ERB tag has a beginning tag and a matched ending tag.

The beginning tag is a less-than sign followed by a percent sign. The closing tag is a percent sign followed by a greater-than sign.

## Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Executes the ruby code within the brackets and do not display the result.

These tags are used to execute ruby but the results are not displayed.

## Text Within an ERB Template

```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

Executes the ruby code within the brackets and display the results.

ERB supports additional tags, one of those is one that allows you to output some variable or some ruby code. Here the example is going to display that 50 plus 50 equals the result of ruby calculating 50 plus 50 and then displaying the result.

Slide 26

# CONCEPT

## The Angry Squid



<%=

The starting tag is different. It has an equals sign. This means show the value stored in a variable or the result of some calculation.

We often refer to this opening tag that outputs the content as the Angry Squid. The less-than is its head, the percent sign as its eyes, and the equals sign its tentacles shooting away after blasting some ink.

## Copying the Existing Content into the Template



~/cookbooks/workstation/templates/default/motd.erb

Property of ...

```
IPADDRESS: #{node['ipaddress']}
HOSTNAME : #{node['hostname']}
MEMORY   : #{node['memory']['total']}
CPU       : #{node['cpu'][0]['mhz']}
```

We can start by copying and pasting the existing content for the Message of the Day file into the template file.

## Changing String Interpolation to ERB Tags



~/cookbooks/workstation/templates/default/motd.erb

Property of ...

```
IPADDRESS: <%= node['ipaddress'] %>
HOSTNAME : <%= node['hostname'] %>
MEMORY    : <%= node['memory']['total'] %>
CPU       : <%= node['cpu']['0']['mhz'] %>
```

Replace all the string interpolation with ERB tags.

Slide 29

# EXERCISE

## Cleaner Recipes



*The template is created and defined. It now needs to be used within the recipe.*

**Objective:**

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- Change the file resource to the template resource
- Update the cookbook's version number
- Apply the updated recipe and verify the results

The template is created and the contents are correctly defined. It is time to update the recipe.

## Removing the file Resource

```
~/cookbooks/workstation/recipes/setup.rb

# ... PACKAGE RESOURCES ...

file '/etc/motd' do
  content "Property of ...
  IPADDRESS: #{node['ipaddress']}
  HOSTNAME : #{node['hostname']}
  MEMORY    : #{node['memory']['total']}
  CPU       : #{node['cpu']['0']['mhz']}
"
end
```

Remove the file resource from the setup recipe.

Slide 31

## Changing from file to template Resource

~/cookbooks/workstation/recipes/setup.rb

```
# ... PACKAGE RESOURCES ...

template '/etc/motd' do
  source 'motd.erb'
end
```

...and replace it with the Template resource. The source attribute specifies the file path 'motd.erb' - the new template file that was created.

Slide 32

# EXERCISE

## Cleaner Recipes



*This is a change to the cookbook so it is time to update the version again.*

**Objective:**

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ❑ Update the cookbook's version number
- ❑ Apply the updated recipe and verify the results

We hopefully haven't changed the original goal of our recipe but we have made some changes.

## Updating the Cookbook's Version Number



~/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 '0.2.1'
```

The overall functionality of the implementation has not changed at all but the underlying structure has so this seems like a minor change so we update the version number to reflect that change.

Slide 34

# EXERCISE

## Cleaner Recipes



*This is a change to the cookbook so it is time to update the version again.*

**Objective:**

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ✓ Update the cookbook's version number
- Apply the updated recipe and verify the results

Now that we have the change defined and the version number updated to reflect that change it is time to apply the updated cookbook to the state of the system.

## Applying the Updated Cookbook



```
> sudo chef-client --local-mode --runlist "recipe[workstation::setup]"  
  
- workstation (0.2.1)  
Compiling Cookbooks...  
Converging 2 resources  
Recipe: workstation::setup  
  * yum_package[tree] action install (up to date)  
  * template[/etc/motd] action create (up to date)  
  
Running handlers:  
Running handlers complete  
Chef Client finished, 0/2 resources updated in 12 seconds
```

We apply the cookbook's recipe with the chef-client command and we should see that the content of the template does not change at all or if it does change it is simply some whitespace. It all depends on how different the contents of the template was from the previously defined source property.

Slide 36

## Verifying the Contents of the MOTD File



```
> cat /etc/motd
```

```
Property of ...
```

```
IPADDRESS: 172.31.57.153  
HOSTNAME : ip-172-31-57-153  
MEMORY   : 604308kB  
CPU       : 1795.673
```

We can log out of the workstation and log back in to see the MOTD contents or we can look directly at the file itself and see the contents.

Slide 37

# EXERCISE

## Cleaner Recipes



*This is a change to the cookbook so it is time to update the version again.*

**Objective:**

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results

We have successfully updated the recipe so that it is clearer, while accomplishing the same job as it did before.

Slide 38

# DISCUSSION



## Discussion

What is the benefit of using a template over defining the content within a recipe? What are the drawbacks?

What are the two types of ERB tags we talked about?

What do each of the ERB tags accomplish?

Answer these questions.

Slide 39

# DISCUSSION



## Q&A

What questions can we help you answer?

- template resource
- Files in the template directory
- ERB

What questions can we help you answer?

Generally or specifically about the template resource, files in the template directory and ERB.

Slide 40



**6: Lab**

## Lab: Building a Webserver Cookbook

©2016 Chef Software Inc.

6-1



This module is going to present a challenge to you to exercise all the things that you have learned over the last few modules.

## Slide 2

# LAB

## Setting up a Web Server



- Create a cookbook named '**apache**' with a recipe named '**server**'
- The '**server**' recipe defines the following policy:
  - The package named 'httpd' is installed.
  - The template named '/var/www/html/index.html' is created with the source 'index.html.erb'
  - The service named 'httpd' is started and enabled.
- Create a template named '**index.html.erb**' and populate it with a welcome message, the node's ipaddress, and the node's hostname.
- Use chef-client to apply the apache cookbook's server recipe
- Verify the site is available by running **curl localhost**

Your challenge is to deploy the Apache webserver. To do that you will need to create a cookbook, create a recipe, define the following policy within that recipe, create a template with specific content, apply the policy defined in the recipe, and verify that the policy applied successfully.

The lab's activity here is to challenge you to employ all that you have done already but in a new context. When everyone has completed the activity we will review the commands and code in the following slides.

Slide 3

## Creating the apache Cookbook



```
> chef generate cookbook cookbooks/apache
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[/home/chef/cookbooks/apache] action create
    - create new directory /home/chef/cookbooks/apache
  * template[/home/chef/cookbooks/apache/metadata.rb] action
create_if_missing
    - create new file /home/chef/cookbooks/apache/metadata.rb
    - update content in file
/home/chef/cookbooks/apache/metadata.rb from none to 37ed5f
  (diff output suppressed by config)
  * template[/home/chef/cookbooks/apache/README.md] action
```

## Slide 4

## Creating the server Recipe



```
> chef generate recipe cookbooks/apache server
```

```
Compiling Cookbooks...
Recipe: code_generator::recipe
  * directory[cookbooks/apache/spec/unit/recipes] action create
    (up to date)
  * cookbook_file[cookbooks/apache/spec/spec_helper.rb] action
    create_if_missing (up to date)
  * template[cookbooks/apache/spec/unit/recipes/server_spec.rb]
    action create_if_missing
      - create new file
  cookbooks/apache/spec/unit/recipes/server_spec.rb
    - update content in file
  cookbooks/apache/spec/unit/recipes/server_spec.rb from none to
```

## Defining the Policy in the server Recipe

~/cookbooks/apache/recipes/server.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: server  
#  
# Copyright (c) 2016 The Authors, All Rights Reserved.  
package 'httpd'  
  
template '/var/www/html/index.html' do  
  source 'index.html.erb'  
end  
  
service 'httpd' do  
  action [:start, :enable]  
end
```

## Slide 6

## Creating the html Template



```
> chef generate template cookbooks/workstation index.html
```

```
Compiling Cookbooks...
Recipe: code_generator::template
  * directory[cookbooks/apache/templates/default] action create
    - create new directory cookbooks/apache/templates/default
  * template[cookbooks/apache/templates/default/index.html.erb]
action create
  - create new file
cookbooks/apache/templates/default/index.html.erb
  - update content in file
cookbooks/apache/templates/default/index.html.erb from none to
e3b0c4
```

## Slide 7

## Defining the index.html Template

~/cookbooks/apache/templates/default/index.html.erb

```
<html>
  <body>
    <h1>Welcome Home!</h1>
    <h2>IPADDRESS: <%= node['ipaddress'] %></h2>
    <h2>HOSTNAME: <%= node['hostname'] %></h2>
  </body>
</html>
```

## Slide 8

## Applying the apache Cookbook's server Recipe



```
> sudo chef-client --local-mode --runlist "recipe[apache::server]"
```

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: ["apache::server"]
Synchronizing Cookbooks:
  - apache (0.1.0)
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::server
  * yum_package[httpd] action install
    - install version 2.2.15-47.el6.centos.4 of package httpd
  * template[/var/www/html/index.html] action create
    - create new file /var/www/html/index.html
```

## Verifying the Default Website is Available



```
> curl localhost
```

```
<html>
  <body>
    <h1>Welcome Home!</h1>
    <h2>IPADDRESS: 172.31.57.153</h2>
    <h2>HOSTNAME: ip-172-31-57-153</h2>
  </body>
</html>
```

Slide 10

# DISCUSSION

## Q&A



What questions can we help you answer?

What questions can we help you answer?

Slide 11



## 7: Workstation Installation

# Workstation Installation

Configuring Your Laptop as a Workstation

## Slide 2

## Objectives

After completing this module, you should be able to

- Ensure that ChefDK is installed on your laptop
- Execute a series of commands to ensure everything is installed

We have been doing a lot of great work with Chef on this remote workstation that we have provided for you.

In this section we will walk through the installation of the necessary tools and the commands to verify your installation.

Slide 3

# EXERCISE

## Installing the ChefDK



*Installing the tools on your system*

**Objective:**

- Install the ChefDK
- Open a Terminal / Command Prompt
- Execute a series of commands to ensure everything is installed
- Download a repository of cookbooks
- Install a text editor (optional)

To become a successful Chef developer, you will want to install Chef tools on your local workstation. These tools are available in the Chef Development Kit (ChefDK). After the installation is complete, we will verify that the various tools are working.

Then we will download a copy of the cookbooks that we created together.

After that you can optionally download a number of other tools that will help you in your journey using Chef. The first is git and the second is a text editor.

Let's get started.

## Slide 4

# CONCEPT



## Chef Development Kit (ChefDK)

The ChefDK contains tools like chef, and chef-client.

The omnibus installer is used to set up the Chef development kit on a workstation, including an embedded version of Ruby, RubyGems, OpenSSL, key-value stores, parsers, libraries, command line utilities, and community tools such as Kitchen, Berkshelf, and ChefSpec.

<https://downloads.chef.io/chef-dk/>

Throughout this course we have been using a number of tools found within the ChefDK. The ChefDK contains tools like 'chef' and 'chef-client'. It also has a number of other great tools that we will use when connecting to a Chef Server, managing cookbook dependencies, or ensuring the quality of the cookbooks that we write.

You can download the ChefDK at <https://downloads.chef.io/chef-dk>.

## Slide 5

# CONCEPT



## Download the ChefDK

ChefDK is a tool chain built on top of the Ruby programming language.

The ChefDK installer does not install any particular graphical-user-interface—installs CLI instead

<https://downloads.chef.io/chef-dk/>

The ChefDK is a tool chain built on top of the Ruby programming language. To assist with making the tools more portable to all platforms we package Ruby and all these tools together in a single platform specific installation package.

The installer does not install any particular graphical user interface, GUI, but instead installs the command-line tools we have been using thus far.

You may have already downloaded the ChefDK previously in this course.

Slide 6

## Installing ChefDK



©2016 Chef Software Inc.

7-6



Follow the ChefDK installation wizard's instructions. It could take over 10 minutes to install ChefDK.

ChefDk will be installed into an opscode folder on your laptop.

Slide 7

# EXERCISE

## Installing the ChefDK



*Installing the tools on your system*

**Objective:**

- ✓ Install the ChefDK
- Execute a series of commands to ensure everything is installed
- Download a repository of cookbooks
- Install a text editor (optional)

To become a successful Chef developer, you will want to install Chef tools on your local workstation. These tools are available in the Chef Development Kit (ChefDK). After the installation is complete, we will verify that the various tools are working.

Then we will download a copy of the cookbooks that we created together.

After that you can optionally download a number of other tools that will help you in your journey using Chef. The first is git and the second is a text editor.

Let's get started.

## Slide 8

## Run All These Commands



```
> chef --version  
> chef-client --version  
> knife --version  
> ohai --version  
> berks --version  
> kitchen --version  
> foodcritic --version  
> rubocop --version
```

Open a local command prompt or something like Windows Power Shell if you prefer and then run these commands.

Some of these commands, like 'chef', 'chef-client', 'ohai', and 'kitchen', are the ones that we have used on our remote workstation. Some of these commands you have not seen yet. Later in this course, we'll explore the commands 'knife' and 'berks'. Some of the remaining commands, like 'foodcritic' and 'rubocop', verify the quality of our cookbook code but will not be discussed in the next sections.

All of these commands have the ability to report their versions. This ensures that all the commands are installed properly on your execution path. If any of these commands fail to run this is the time to stop and troubleshoot them.

Slide 9

# EXERCISE

## Installing the ChefDK



*Installing the tools on your system*

**Objective:**

- ✓ Install the ChefDK
- ✓ Execute a series of commands to ensure everything is installed
- Download a repository of cookbooks
- Install a text editor (optional)

Now we need the work that we have created up until this point on our local workstations. This is because we want to able to deploy these cookbooks to a Chef Server and eventually to the nodes we are going to setup.

Slide 10

## Cookbooks on GitHub

A repository containing a similar copy of the work you did previously in this course can be downloaded from here:

<https://github.com/chef-training/chef-essentials-repo>

The cookbooks that were created during the first modules can be found here. These are not the exact cookbooks that you created but ones that have been completed with additional comments and details added.

Slide 11

## Download the Cookbooks Repository

The cookbooks created after the first day of completing the Chef DK Fundamentals

Branch: master [New pull request](#)

6 commits 4 branches 0 releases 1 contributor

**burtlo** Fix the README - essentials less fundamentals ... Latest commit 021944e on Jan 13

.chef For this to be a Chef Repository it needs a .chef directory 5 months ago

cookbooks Removed the 'yum update' it was causing problems 5 months ago

README.md Fix the README - essentials less fundamentals a month ago

README.md

**Chef Essentials Repository - CentOS**

©2016 Chef Software Inc. 7-11

 CHEF

You may clone the repository or download the zip file.

Slide 12

# EXERCISE

## Installing the ChefDK



*Installing the tools on your system*

**Objective:**

- ✓ Install the ChefDK
- ✓ Execute a series of commands to ensure everything is installed
- ✓ Download a repository of cookbooks
- ❑ Install a text editor (optional)

Now that we have the content copied down locally it is important that you have a text editor installed on your local workstation.

Slide 13

# CONCEPT

## Text Editors



When working with Chef you spend a large time editing files.

Whatever editor you use should optimize for this workflow.

As you have experienced during this introduction to working with Chef, a lot of what you are doing is writing source code in an editor. To work with Chef, you spend a large amount of time editing files, saving your work, and then opening more files. Whatever editor you use should optimize for this workflow.

A large number of basic editors that come standard on your operating system are capable of working with chef: notepad; textedit; kedit; etc. However, they are not always optimized for this workflow.

Slide 14

# CONCEPT

## ATOM Editor



Atom is modern, approachable, and hackable to the core. We can't wait to see what you build with it.

<https://atom.io>

©2016 Chef Software Inc.

7-14



The Atom editor tool can be customized to do anything, but can also be used productively on the first day without ever touching a config file. Atom is modern, approachable, and hackable to the core. We can't wait to see what you build with it.

You can download Atom at this time, if you don't already have it. You could also use Sublime Text if you already have it.

Slide 15

# EXERCISE

## Installing the ChefDK



*Installing the tools on your system*

**Objective:**

- ✓ Install the ChefDK
- ✓ Execute a series of commands to ensure everything is installed
- ✓ Download a repository of cookbooks
- ✓ Install a text editor (optional)

Now that we have the content copied down locally it is important that you have a text editor installed on your local workstation.

Slide 16

# DISCUSSION



## Q&A

What questions can we answer for you?

Slide 17



## 8: Chef Server



# The Chef Server

A Hub for Configuration Data

©2016 Chef Software Inc. 9-1 

You accomplished a lot so far. You created two cookbooks; one to setup workstations with your tools and a second cookbook that set up a web server that delivered a "Hello, world!" message with some pertinent information about your system.

## Slide 2

# Objectives

After completing this module, you should be able to

- Connect your local workstation (laptop) to a Chef Server

In this module you will learn how to connect your local workstation to a Chef Server, upload cookbooks to a Chef Server, bootstrap a node, manage a node via a Chef Server.

Slide 3

# EXERCISE

## Hosted Chef



*A Hosted Chef Account will allow us to manage our cookbooks across multiple nodes.*

**Objective:**

- Create a Hosted Chef Account

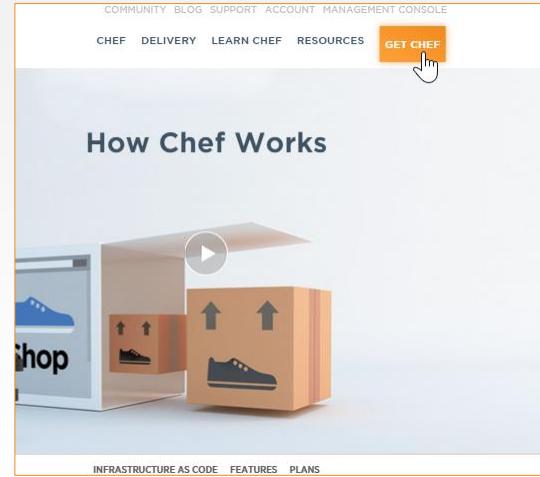
In the interest of getting things done with a relatively small node count, it seems like the Hosted Chef Server option is best.

Slide 4

## Signing Up for a Hosted Chef Account

### Steps

1. Navigate to <https://www.chef.io>
2. From the resulting window, click **Get Chef**.
3. From the resulting window, click the Hosted Chef Sign Up button.
4. Fill out the form as indicated in this image using your name and a valid email address and then click **Get Started**.



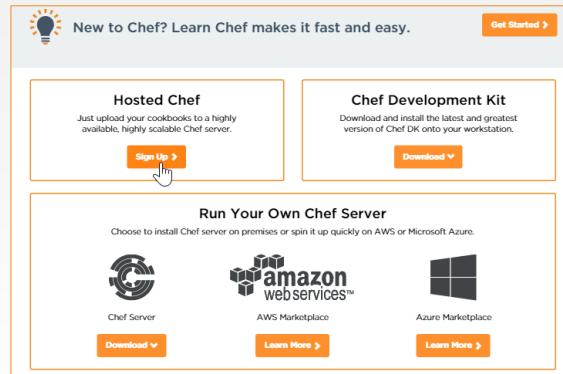
To get started with Hosted Chef Server, visit the Chef website and sign up for a Hosted Chef Account.

## Slide 5

# Signing Up for a Hosted Chef Account

## Steps

1. Navigate to <https://www.chef.io>
2. From the resulting window, click Get Chef.
3. From the resulting window, click the Hosted Chef **Sign Up** button.
4. Fill out the form as indicated in this image using your name and a valid email address and then click **Get Started**.



## Slide 6

# Signing Up for a Hosted Chef Account

## Steps

1. Navigate to <https://www.chef.io>
2. From the resulting window, click Get Chef.
3. From the resulting window, click the Hosted Chef Sign Up button.
4. Fill out the form as indicated in this image using your name and a valid email address and then click **Get Started**.

**Note:** You should write down your new user name and remember your password.

### Start your free trial of hosted Chef

You're one step away from access to all the power and flexibility of Chef. Get ready to automate your infrastructure, accelerate your time to market, manage scale and complexity, and safeguard your systems. Just complete the form to get started.

Full Name	Jane Doe
Email	Jane@chef.io
Username	janedoe
Password	.....
Company	Chef

I agree to the [Terms of Service](#) and the Master License and Services Agreement.

**Get Started**

Already  
Click here  
Looking  
Start with  
and chec  
Join the  
Join our

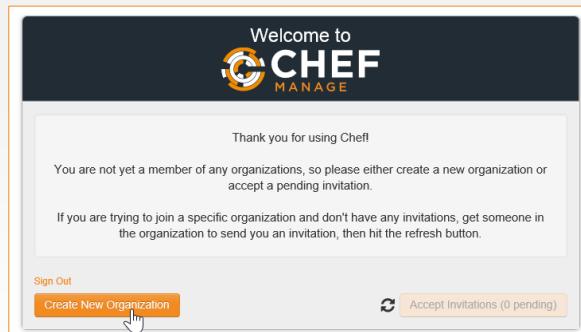


Slide 7

# Signing Up for a Hosted Chef Account

## Steps

6. From the resulting page, click the **Create New Organization** button.
7. Fill out the resulting Create Organization form and then click Create Organization.
8. From the resulting page, click your new organization to highlight it and then click Starter Kit.
9. From the resulting window, click the Download Starter Kit button.

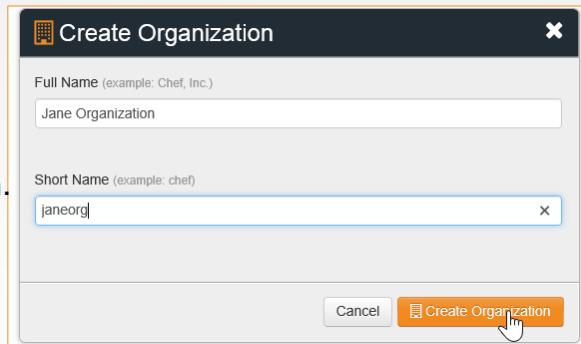


Slide 8

# Signing Up for a Hosted Chef Account

## Steps

6. From the resulting page, click the Create New Organization button.
7. Fill out the resulting Create Organization form and then click **Create Organization**.
8. From the resulting page, click your new organization to highlight it and then click Starter Kit.
9. From the resulting window, click the Download Starter Kit button.



An organization is a structure within managed Chef that allows multiple companies or entities to exist on the same Chef Server without your paths ever crossing. You might think of it as like setting up a unique username for your organization.

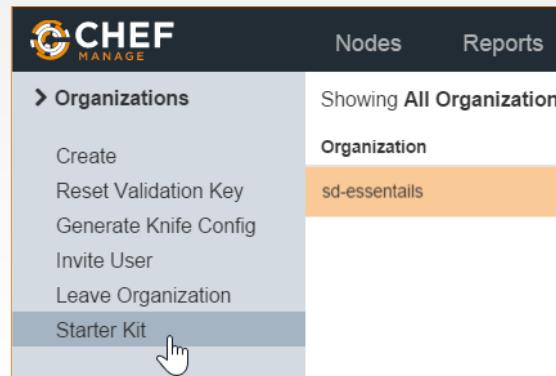
All of the cookbooks, instances and other configuration details that you manage with Chef will be stored on the Chef Server for this particular organization. No other organization will have access to it.

Slide 9

# Signing Up for a Hosted Chef Account

## Steps

6. From the resulting page, click the Create New Organization button.
7. Fill out the resulting Create Organization form and then click Create Organization.
8. From the resulting page, click your **new organization** to highlight it and then click **Starter Kit**.
9. From the resulting window, click the Download Starter Kit button.

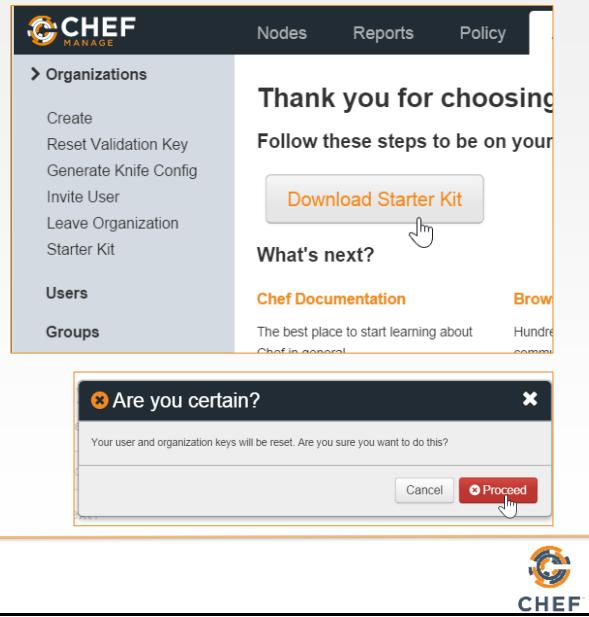


Slide 10

## Signing Up for a Hosted Chef Account

### Steps

6. From the resulting page, click the Create New Organization button.
7. Fill out the resulting Create Organization form and then click Create Organization.
8. From the resulting page, click your **new organization** to highlight it and then click **Starter Kit**.
9. From the resulting window, click the **Download Starter Kit** button.



The starter kit will warn that it will reset your organization key and personal key. If this is a new account and new organization this reset is totally fine. If you already have an account or this is an existing organization please understand that you are destroying the existing keys that already exist on a workstation.

# Signing Up for a Hosted Chef Account

## Steps

10. Open the download zip file and copy the **chef-repo** folder that's contained in the zip file.
11. Paste the **chef-repo** folder to a location on your laptop, such as your home directory

Name
 <u>chef-repo</u>

**Note:** Ensure that the path to the chef-repo does not have a space in it. Examples:

Mac: /home/username/chef-repo

Windows: C:\Users\username\chef-repo

Slide 12



## 9: Chef Server

### The Chef Server

A Hub for Configuration Data

You accomplished a lot so far. You created two cookbooks; one to setup workstations with your tools and a second cookbook that set up a web server that delivered a "Hello, world!" message with some pertinent information about your system.

## Slide 2

## Objectives

After completing this module, you should be able to

- Connect your local workstation (laptop) to a Chef Server
- Upload cookbooks to a Chef Server
- Bootstrap a node
- Manage a node via a Chef Server

In this module you will learn how to connect your local workstation to a Chef Server, upload cookbooks to a Chef Server, bootstrap a node, manage a node via a Chef Server.

## Slide 3

## Managing an Additional System

To manage another system, you would need to:

1. Provision a new node within your company or appropriate cloud provider with the appropriate access to login to administrate the system.
2. Install the Chef tools.
3. Transfer the apache cookbook.
4. Run chef-client on the new node to apply the apache cookbook's default recipe.

As an exercise, roughly estimate the time it would take to accomplish this series of steps of preparing another node.

- A new system would require us to provision a new node within your company or appropriate cloud provider with the appropriate access to login to administrate the system.
- Install the Chef tools.
- Transfer the apache cookbook.
- Run chef-client locally to apply the apache cookbook's default recipe.

## Slide 4

## Managing Additional Systems

Installing the Chef tools, transferring the apache cookbook, and applying the run list is not terribly expensive.

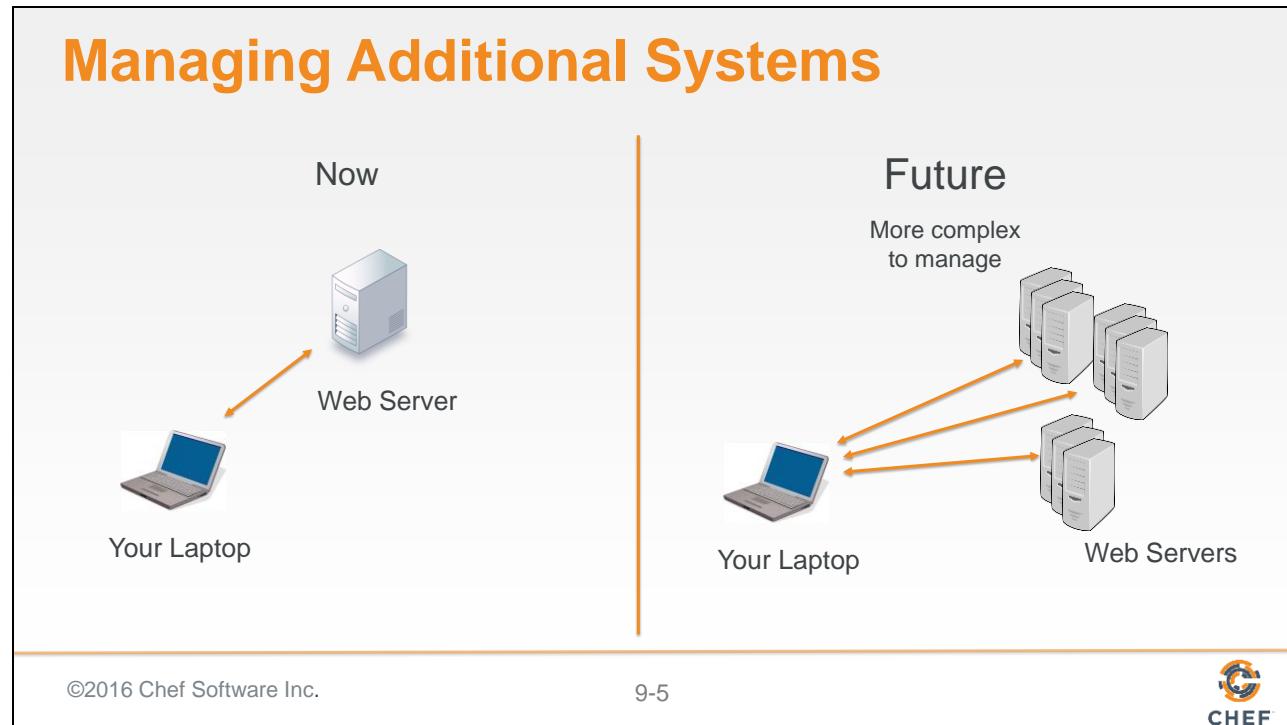
- Chef provides a one-line curl install.
- You could use **git** to clone the repository from a common **git** repository.
- Applying the run list.

The cost of installing the Chef tools, transferring the apache cookbook, and applying the run list is not terribly expensive.

Chef provides a one-line curl install for the Chef Development Kit (ChefDK).

You could use git to clone the repository from a common git repository. Another option is to archive the cookbook and then using SCP to copy over the contents. A third might be to mount a file share. There are a myriad ways to transfer the cookbooks to the new instance.

Then applying the run list requires the execution of a command on that system.



So the overall time required to setup a new instance is not a massive time investment. This manual process will definitely take its toll when requirements demand you manage more than a few additional nodes.

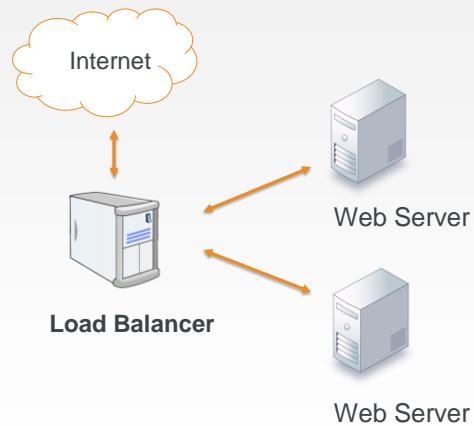
Some may think 10 minutes is not so bad. But what if there were 10 new nodes? 20 new nodes?

As the popularity of your site grows, one server will not be able to keep with all of the web requests. You will need to provision additional machines as demand increases.

Slide 6

## Managing User Traffic

A load balancer can forward incoming user web requests to other nodes.



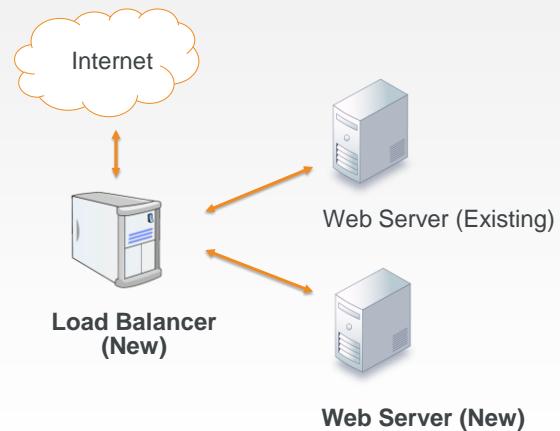
Let's change topics for a moment to managing user web traffic.

In addition to the complexities of configuring and managing multi-server infrastructure, such as web servers, you also need to develop a way to route incoming traffic to each of those web servers and other nodes. There are many ways that you can route the traffic from one node to a group of similar nodes. This can be done with services by some of the major cloud providers or it can be done with another instance running as a load balancer. A load balancer allows us to receive incoming requests and forward those requests to other nodes. A load balancer allows us to receive incoming requests and forward those requests to other nodes.

Slide 7

## Managing User Traffic

Today you will set up a new load balancer that will direct web requests to similarly-configured nodes.



Today you are going to set up a load balancer that will direct web requests to similar configured nodes. Those nodes will be running your default web page that you deploy with the apache cookbook's default recipe.

You have one system already configured as a web server. You will need to set up another web server.

You will also need to set up a node to act as the load balancer to both of these web servers.

## Slide 8

## Steps to Set up Load Balancer and Web Servers

### Web Server

1. Provision the instance
2. Install Chef
3. Copy the Web Server cookbook
4. Apply the cookbook

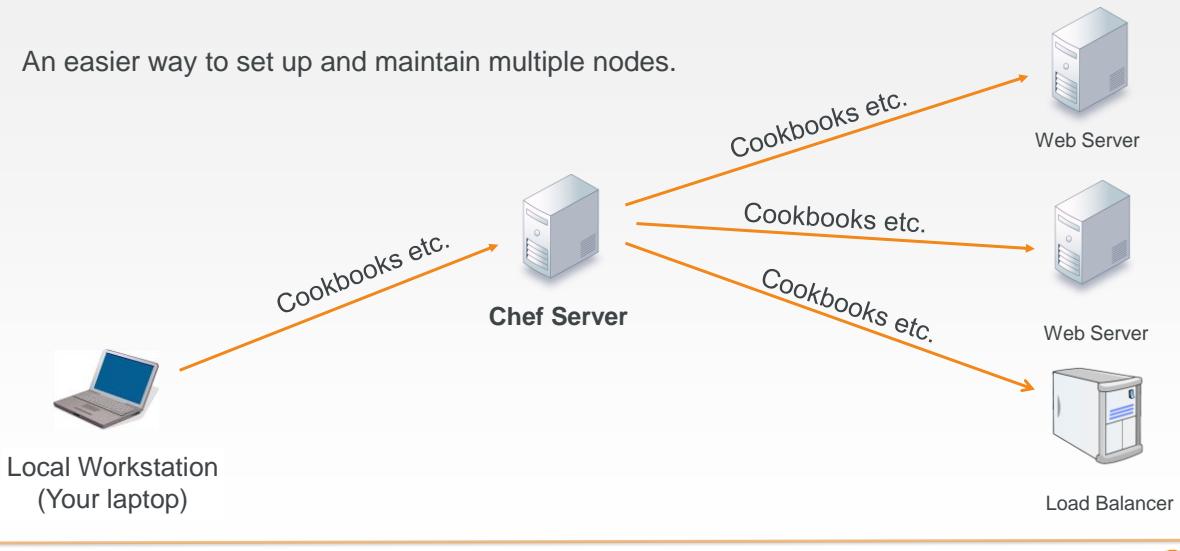
### Load Balancer

1. Create the haproxy (load balancer) cookbook
2. Provision the instance
3. Install Chef
4. Copy the haproxy cookbook
5. Apply the cookbook

Whether you tackle installing, configuring, or running a load balancer or recreate a second instance running the apache cookbook's default recipe, you will need to solve the problem of how you can manage multiple systems. Each system would need to have Chef installed, the cookbooks copied onto each system, and a run list of the recipes to apply to each system.

## The Chef Server

An easier way to set up and maintain multiple nodes.



One way to solve that problem is with a Chef Server.

The Chef Server is designed to help us manage multiple nodes in this situation. The Chef Server acts as a hub for configuration data. The Chef server stores cookbooks, the policies that are applied to nodes, and metadata that describes each registered node that is being managed by 'chef-client'. Nodes, such as web servers, load balancers, etc., use 'chef-client' to ask the Chef server for configuration details, such as recipes, templates, and file distributions. The chef-client then does as much of the configuration work as possible on the nodes themselves (and not on the Chef server). In a production environment, the 'chef-client' runs in an automated mode—it polls the Chef Server for updates at set intervals and then applies any configuration changes. This scalable approach distributes the configuration effort throughout the organization.

Slide 10

# EXERCISE



## Managing Nodes with Hosted Chef

*It will be easier to distribute the cookbooks we write to multiple nodes if we store them in a central repository.*

**Objective:**

- Download and copy the required cookbooks
- Upload your cookbooks to the Hosted Chef Server
- Add your old workstation as a managed node

In the interest of getting things done with a relatively small node count, it seems like the Hosted Chef Server option is best.

Slide 11

# CONCEPT

## Code Repository



A repository containing a similar copy of the work you did previously in this sections can be downloaded from here:

<https://github.com/chef-training/chef-essentials-repo>

The cookbooks that were created during the first modules can be found here. These are not the exact cookbooks that you created but ones that have been completed with additional comments and details added.

## Slide 12

# Downloading the Cookbook Repository

The cookbooks created after the first day of completing the Chef DK Fundamentals

Branch: master [New pull request](#)

6 commits 4 branches 0 releases 1 contributor

[.chef](#) Fix the README - essentials less fundamentals ... Latest commit 021944e on Jan 13

[cookbooks](#) For this to be a Chef Repository it needs a .chef directory 5 months ago

[README.md](#) Removed the 'yum update' it was causing problems 5 months ago

[README.md](#) Fix the README - essentials less fundamentals a month ago

**Chef Essentials Repository - CentOS**

©2016 Chef Software Inc. 9-12

CHEF

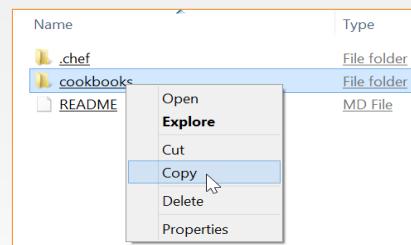
You may clone the repository or download the zip file. Both of those links can be found in the bottom right.

# Overwriting the Cookbooks Folder

## Steps

1. Open the downloaded **chef-essentials-master.zip** file and then copy **only** the **cookbooks** folder.
2. Replace the **cookbooks** folder that's in your chef-repo folder with the copied cookbooks folder.

chef-essentials-master



chef-repo



After you download and open the chef-essentials-master archive, copy the included cookbooks folder and paste it into your chef-repo that you unzipped from the Start Kit. Let the new cookbooks folder (that you got from the chefdk-fundamentals-repo) overwrite the existing cookbooks folder that was in your chef-repo folder.

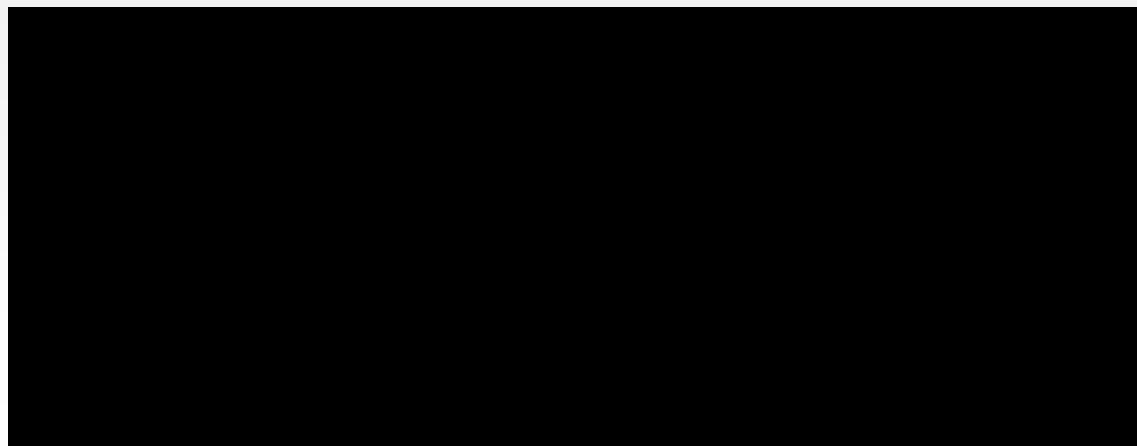
Important: If you had an existing chef-repo prior to class that you want to preserve, save a copy of your old cookbooks folder before pasting the new one into your chef-repo.

Slide 14

## Moving to the Chef Repository Directory



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



©2016 Chef Software Inc.

9-14



The starter kit contains the configuration to reach the Chef Server and your credentials to validate the communication between your workstation and the Chef Server.

To verify the connection with the Chef Server you will need to run commands within the repository you downloaded.

Open a terminal or command prompt and navigate to the chef-repo directory.

Slide 15

# CONCEPT

## knife



knife is a command-line tool that provides an interface between a local chef-repo and the Chef Server.

knife is a command-line tool that allows us to request and send information to the Chef Server.

knife helps users manage: nodes; cookbooks; roles; environments; and more. knife does this through a series of sub-commands.

## Viewing the Help of the Knife Command



```
$ knife --help
```

```
Available subcommands: (for details, knife SUB-COMMAND --help)
```

```
** BOOTSTRAP COMMANDS **
```

```
knife bootstrap FQDN (options)
knife bootstrap windows ssh FQDN (options)
knife bootstrap windows winrm FQDN (options)
```

```
** CLIENT COMMANDS **
```

```
knife client bulk delete REGEX (options)
knife client create CLIENT (options)
knife client delete CLIENT (options)
```

You can look at all the commands with 'knife --help'.

This will display all the sub-commands available. In your case you want to verify that the client list contains a single entry so you need to look for help for the specific command 'knife client --help'.

## Viewing the Help of the Client Subcommand



```
$ knife client --help
```

```
Available client subcommands: (for details, knife SUB-COMMAND --help)

** CLIENT COMMANDS **

knife client bulk delete REGEX (options)
knife client create CLIENT (options)
knife client delete CLIENT (options)
knife client edit CLIENT (options)
knife client list (options) knife client list (options)
knife client reregister CLIENT (options)
knife client show CLIENT (options)
```

This will give us an even smaller subset of the commands related specifically to asking the Chef Server about client information. A general command is the list command which will output all the clients that the Chef Server currently maintains.

Slide 18

## Viewing the Current List of Clients



```
$ knife client list
```

```
ORGNAME-validator
```

For your Chef Server account there should be a single client that is the organization name: validator. This is a special key that has access to the Chef Server. The important thing is that the result does not contain an error with the configuration or authenticating with the Chef Server.

If you receive an error ensure that you: typed the command correctly; executed the command within the chef repository directory; are connected to the internet and not blocking ssl connections from your own system's proxy servers or virtual private networks; and have a .chef directory, within the chef repository, which contains the knife configuration file (knife.rb), personal key, and organizational key.

Slide 19

# EXERCISE



## Managing Nodes with Hosted Chef

*It will be easier to distribute the cookbooks we write to multiple nodes if we store them in a central repository.*

**Objective:**

- ✓ Download and copy the required cookbooks
- ❑ Upload your cookbooks to the Hosted Chef Server
- ❑ Add your old workstation as a managed node

With all that complete, you are now able to communicate with the Chef Server. At this point we will refer to the system in front of you, with the chef repository, the configuration, and the keys installed as your workstation.

When working with Chef with a Chef Server, the workstation is the location where you will compose your cookbook code. When that code is complete, you will then upload it to the Chef Server.

## Viewing the Help for the Cookbook Subcommand



```
$ knife cookbook --help
```

```
** COOKBOOK COMMANDS **  
knife cookbook bulk delete REGEX (options)  
knife cookbook create COOKBOOK (options)  
knife cookbook delete COOKBOOK VERSION (options)  
knife cookbook download COOKBOOK [VERSION] (options)  
knife cookbook list (options)  
knife cookbook metadata COOKBOOK (options)  
knife cookbook metadata from FILE (options)  
knife cookbook show COOKBOOK [VERSION] [PART] [FILENAME] (options)  
knife cookbook test [COOKBOOKS...] (options)  
knife cookbook upload [COOKBOOKS...] (options)
```

Similar to asking the Chef Server about the list of available clients, you can also ask for information about cookbooks. You can find all the commands related to the cookbooks subcommand by running `knife cookbook --help`.

Similar to the list of clients, you can examine a list of cookbooks.

Slide 21

## Viewing the Current List of Cookbooks



```
$ knife cookbook list
```



©2016 Chef Software Inc.

9-21



Running this command will return the cookbooks currently uploaded to the Chef Server. The empty response should come as no surprise.

You want to change that. So you are going to upload each of your cookbooks to the Chef Server.

Slide 22

## Changing to the Directory of the Apache Cookbook



```
$ cd cookbooks/apache
```



To upload a cookbook to the Chef Server you need to be within the directory of the cookbook. Let us start with the apache cookbook. Change directory into the apache cookbook directory which is within the cookbooks directory.

Slide 23

# CONCEPT

## Berkshelf



Berkshelf is a cookbook management tool that allows us to upload your cookbooks and all of its dependencies to the Chef Server.

<http://berkshelf.com>

To upload the cookbook you will need to use another tool called Berkshelf.

Berkshelf is a cookbook management tool that allows us to upload your cookbooks and all of its dependencies to the Chef Server. In this instance, your current cookbooks have no dependencies, but in the future when they do, Berkshelf will assist you in ensuring those are all uploaded.

## Viewing the Help of the Berks Command



```
$ berks --help
```

```
Commands:
berks apply ENVIRONMENT      # Apply version locks from Berksfile.lock to a Chef
environment
berks contingent COOKBOOK    # List all cookbooks that depend on the given cookbook in
your
berks cookbook NAME [PATH]    # Create a skeleton for a new cookbook
berks help [COMMAND]          # Describe available commands or one specific command
berks info [COOKBOOK]          # Display name, author, copyright, and dependency information
berks init [PATH]              # Initialize Berkshelf in the given directory
berks install                  # Install the cookbooks specified in the Berksfile
berks list                     # List cookbooks and their dependencies specified by your
berks outdated [COOKBOOKS]     # List dependencies that have new versions available that
berks package [PATH]            # Vendor and archive the dependencies of a Berksfile
berks search NAME              # Search the remote source for cookbooks matching the partial
```

Berkshelf is a command-line tool that you can ask to see available the commands.

## Installing Any Cookbook Dependencies



```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'apache' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using apache (0.2.1) from source at .
```

Berkshelf is used on a per-cookbook basis. As dependencies are often per cookbook you'll need to change into the directory of the cookbook.

You should install any dependencies that your cookbook might have. Again, in this instance there are no dependencies external to this cookbook but Berkshelf ensures that this is the case when it runs the 'berks install' command.

You'll see that it finds the current cookbook within your current directory, it contacts the Supermarket for any external dependencies, and then ...

## Viewing the Cookbook's Main Directory



```
$ ls -al (or ls -Force if using Powershell)
```

```
drwxr-xr-x 7 chef chef 4096 Aug 27 18:44 .
drwxr-xr-x 4 chef chef 4096 Aug 27 16:17 ..
drwxr-xr-x 8 chef chef 4096 Aug 27 16:07 .git
-rw-r--r-- 1 chef chef 126 Aug 27 15:46 .gitignore
drwxr-xr-x 3 chef chef 4096 Aug 27 18:45 .kitchen
-rw-r--r-- 1 chef chef 183 Aug 27 18:44 .kitchen.yml
-rw-r--r-- 1 chef chef 47 Aug 27 15:46 Berksfile
-rw----- 1 chef chef 77 Aug 27 18:45 Berksfile.lock
-rw-r--r-- 1 chef chef 54 Aug 27 15:46 README.md
-rw-r--r-- 1 chef chef 974 Aug 27 15:46 chefignore
-rw-r--r-- 1 chef chef 198 Aug 27 15:46 metadata.rb
drwxr-xr-x 2 chef chef 4096 Aug 27 16:34 recipes
```

...it completes by writing a Berksfile.lock to the file system.

The Berksfile.lock is a receipt of all the cookbooks and dependencies found at the exact moment that you ran 'berks install'.

## Viewing the Contents of the Berksfile.lock



```
$ cat Berksfile.lock
```

```
DEPENDENCIES
apache
  path: .
  metadata: true

GRAPH
  apache (0.2.1)
```

This lock file is useful to ensure that in the future you use the same dependencies when working with the cookbook.

Slide 28

## Uploading a Cookbook to the Chef Server



```
$ berks upload
```

```
Uploaded apache (0.2.1) to:  
'https://api.opscode.com:443/organizations/steveessentials2'
```

With the dependencies accounted for, it is time to upload the to the Chef Server. This is another sub-command that Berkshelf provides called 'upload'. Run the command to upload the apache cookbook to the Chef Server.

Slide 29

## Displaying the Current List of Cookbooks



```
$ knife cookbook list
```

```
apache      0.2.1
```

When that is complete you can return to the cookbook command that allows you to display the cookbooks within your organization by running this command. This will show you that the Chef Server has the apache cookbook that you have uploaded.

Slide 30

# EXERCISE



## Managing Nodes with Hosted Chef

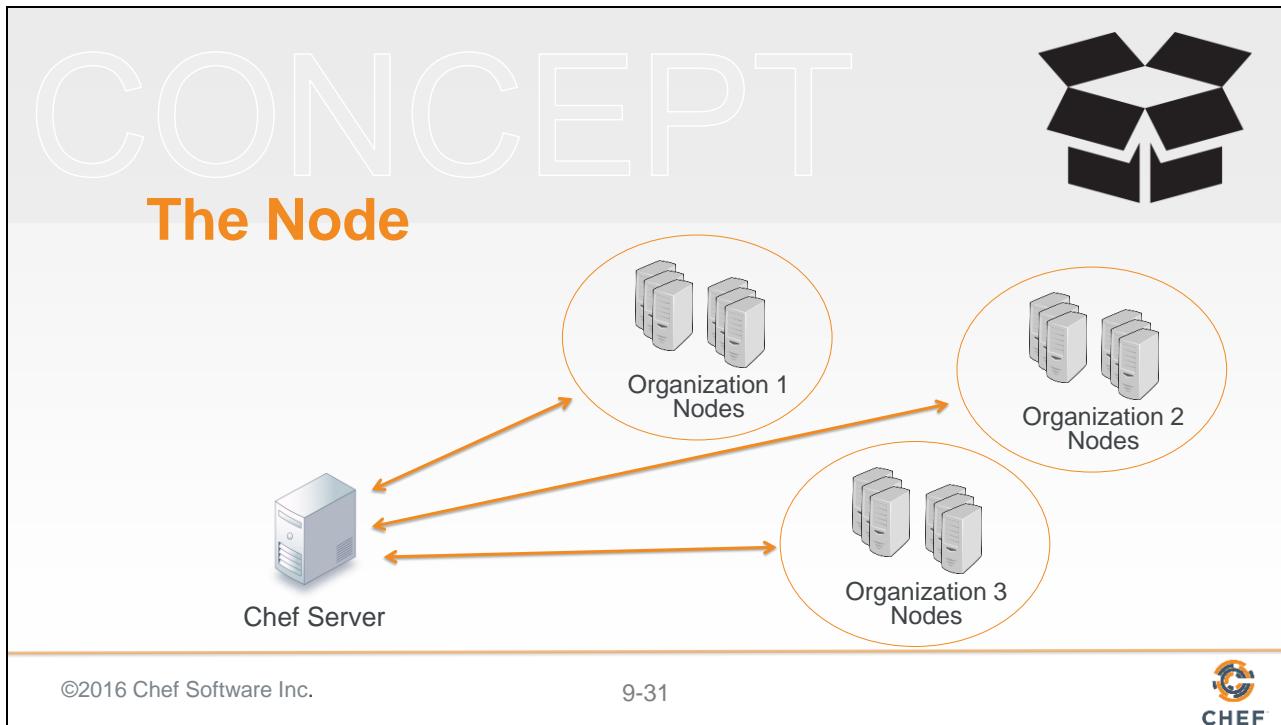
*It will be easier to distribute the cookbooks we write to multiple nodes if we store them in a central repository.*

**Objective:**

- ✓ Download and copy the required cookbooks
- ✓ Upload your cookbooks to the Hosted Chef Server
- ❑ Add your old workstation as a managed node

You have one remaining objective and that is to add an instance as a node within your organization.

Slide 31



As you know by now, a node is a server that Chef is managing. A node could be a web server, an application server, a database server, a load balancer, and so on.

A node can only join one organization. To be a node means that it has Chef installed, has configuration files in place, and when you run the chef-client application with no parameters it will successfully contact the Chef Server and ask it for the run list that it should apply and the cookbooks required to execute that run list.

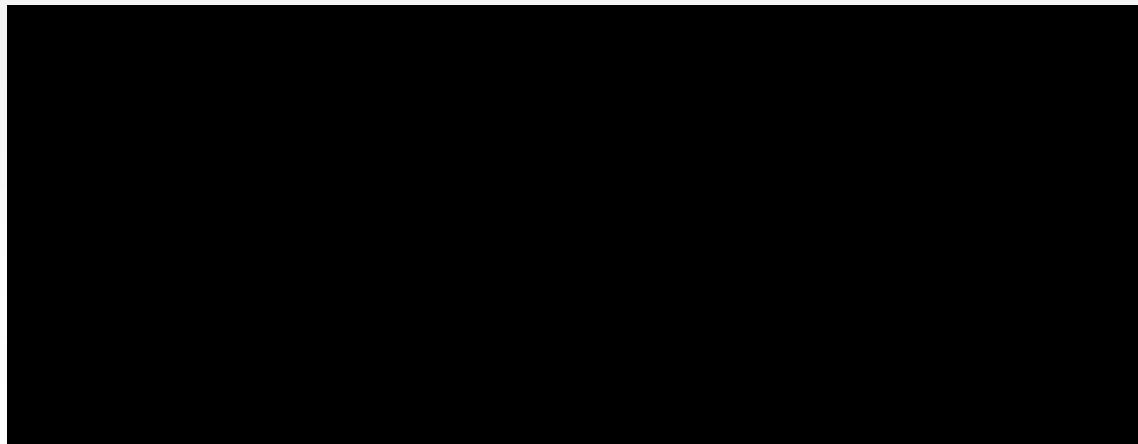
When a node is part of the organization you manage that information on the Chef Server as well. A Chef Server can manage multiple organizations. Managing that information in a Chef Server allows us to use for inventory, querying and searching.

Slide 32

## Changing to the chef-repo Directory



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



Let's add the instance we used previously as a workstation now as a managed node.  
Return to the root of the chef repository.

## Viewing the Help for the Node Subcommand



```
$ 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)
```

Verify that you have no existing nodes within your organization. You can use the 'knife node --help' command to see that you can ask for the list of all nodes within your organization with the list command.

Slide 34

## Viewing the Current List of Nodes



```
$ knife node list
```



Run "knife node list" to see that you have no nodes currently registered with your Chef Server. At this point the results should be blank.

Slide 35

# CONCEPT



## Bootstrapping a Node

The node may not have Chef installed. It may also not have details of where the Chef Server is located or the credentials to securely talk to that Server.

To add those credentials we can **bootstrap** that node to install all those components.

<https://learn.chef.io/skills/beyond-essentials-1>

We want to add the new instance that we have as a node within our organization. This instance may or may not have Chef installed on it. It also probably does not know where the Chef Server is or have the credentials to even talk to it securely. We could manually configure a node but there is an easier way of doing that through a process called 'bootstrapping'.

Bootstrapping will install Chef if necessary and then configure the node to talk securely to a specified Chef Server.

## Reviewing the Help for Bootstrapping



```
$ knife bootstrap --help
```

```
knife bootstrap FQDN (options)
  --bootstrap-curl-options OPTIONS
    Add options to curl when install chef-client
  --bootstrap-install-command COMMANDS
    Custom command to install chef-client
  --bootstrap-no-proxy [NO_PROXY_URL|NO_PROXY_IP]
    Do not proxy locations for the node being
bootstrapped; this option is used internally by Opscode
  --bootstrap-proxy PROXY_URL  The proxy server for the node being
bootstrapped
  -t TEMPLATE,
    Bootstrap Chef using a built-in or custom
template. Set to the full path of an erb
template or use one of the built-in templates.
```

Knife provides a bootstrap subcommand that takes a number of options.

When you bootstrap an instance it is performing the following: Installing chef tools if they are not already installed; Configuring Chef to communicate with the Chef Server; Running chef-client to apply a default run list.

## Bootstrapping a New Node



```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node1
```

```
Creating new client for node1
Creating new node for node1
C   Fully Qualified Domain      6-24.compute-1.amazonaws.com
e   Name                         .a
ec2-54-175-46-24.compute-1.a user name St password Che
ec2-54-175-46-24.compute-1.amazonaws.com resolving cookbooks for run list: []
ec2-54-175-46-24.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-175-46-24.compute-1.amazonaws.com [2016-09-16T16:51:21+00:00] WARN: Node
node1 has an empty run list.
ec2-54-175-46-24.compute-1.amazonaws.com Converging 0 resources
ec2-54-175-46-24.compute-1.amazonaws.com
ec2-54-175-46-24.compute-1.amazonaws.com Running handlers:
```

To communicate with the remote instance you need to provide it the credentials to connect to the system. Use the user name with the '-x' flag and the password '-P' flag. Include the '--sudo' flag because you are installing software and writing configuration to directories traditionally owned by the root user. Name the node with the '-N' flag. This is optional but makes it easier for us to communicate. When we ask you to look at the details of node 1 or login to node 1, it will be easier to remember than the fully-qualified domain name.

When executing the command, the output will tell us what it installed and ran.

Slide 38

## Viewing the Updated List of Nodes



```
$ knife node list
```

```
node1
```

When bootstrapping is done, you can see that your organization knows about the new node by again running the command "knife node list". You now see that you have a new node, node1, uploaded to the Chef Server.

Slide 39

## Viewing More Information About Your Node



```
$ knife node show node1
```

```
Node Name:    node1
Environment:  _default
FQDN:        ip-172-31-8-68.ec2.internal
IP:          54.175.46.24
Run List:
Roles:
Recipes:
Platform:    centos 6.7
Tags:
```

You can see more information about a particular node with the command 'knife node show node1'. This will display a summary of the node information that the Chef Server stores.

Slide 40

## Adding a Recipe to a Run List



```
$ knife node run_list add node1 "recipe[apache]"
```

```
node1:
  run_list: recipe[apache]
```

node1 does not have a list of recipes that it applies to the system by default. You can make Chef Server tell node1 to apply a specific run-list the next time node 1 runs 'chef-client'.

You can do that through the 'knife node run\_list add' command. In this example, you are adding to node1's run-list the apache cookbook's default recipe.

Slide 41

# EXERCISE



## Managing Nodes with Hosted Chef

*It will be easier to distribute the cookbooks we write to multiple nodes if we store them in a central repository.*

**Objective:**

- ✓ Download and copy the required cookbooks
- ✓ Upload your cookbooks to the Hosted Chef Server
- ✓ Add your old workstation as a managed node

Slide 42

# DISCUSSION



## Discussion

What is the benefit of storing cookbooks in a central repository?

What is the primary tool for communicating with the Chef Server?

How did you add a node to your organization?

Answer these questions.

Slide 43

# DISCUSSION



## Q&A

What questions can you help you answer?

- Chef Server
- Managed Chef
- Berkshelf
- Bootstrapping Nodes

With all of the objectives complete you are finished with this section. What question can you answer for you?

Slide 44



## 10: Community Cookbooks

# Community Cookbooks

Find, Explore and View Chef Cookbooks

©2016 Chef Software Inc.

10-1



## Slide 2

# Objectives

After completing this module, you should be able to

- Find cookbooks on the Chef Super Market
- Create a wrapper cookbook
- Replace the existing default values
- Upload a cookbook to Chef Server
- Bootstrap a new node that runs the cookbook

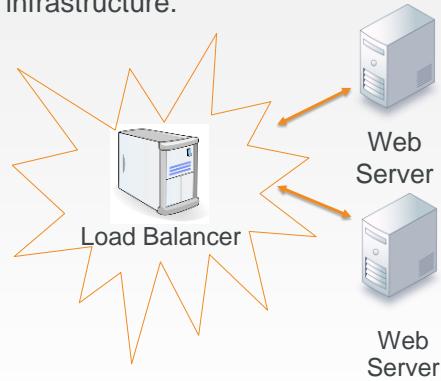
In this module you will learn how to find cookbooks on the Chef Super Market, create a wrapper cookbook, replace the existing default values, upload a cookbook to Chef Server, and bootstrap a new node that runs the cookbook.

Slide 3

## Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Receives requests and relays them to other systems.



With a single web server running with our organization, it's now time to talk about the next goal to tackle. We need to setup a load balancer.

A load balancer is able to receive requests and relay them to other systems. In our case, we specifically want to use the load balancer to balance the entire traffic load between one or more systems.

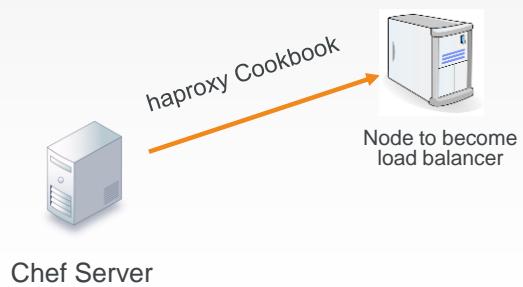
This means we will need to establish a new node within our organization, install the necessary software to make the node a load balancer, and configure it so that it will relay requests to our existing node running apache and to future nodes.

## Slide 4

## Load Balancer

Work that needs to be accomplished to setup a load balancer within our infrastructure:

- Write a haproxy (load balancer) cookbook.
- We will need to establish a new node within our organization to which we apply that cookbook.



Similar to how we installed and configured apache on our first node, we could do the same thing here with a load balancer. We could learn the package name for the application 'haproxy', learn which file manages the configuration, learn how to compose the configuration with custom values, and then manage the service.

Package, Template and Service are the core of configuration management. Nearly all the recipes you write for an application will center on using these three resources. We could spend some time focused on composing the cookbook recipe and testing it on our platform with our custom configuration.

Slide 5

# CONCEPT

## Community Cookbooks



Someone already wrote that cookbook?

Available through the community site called Supermarket/

<https://supermarket.chef.io>

©2016 Chef Software Inc.

10-5



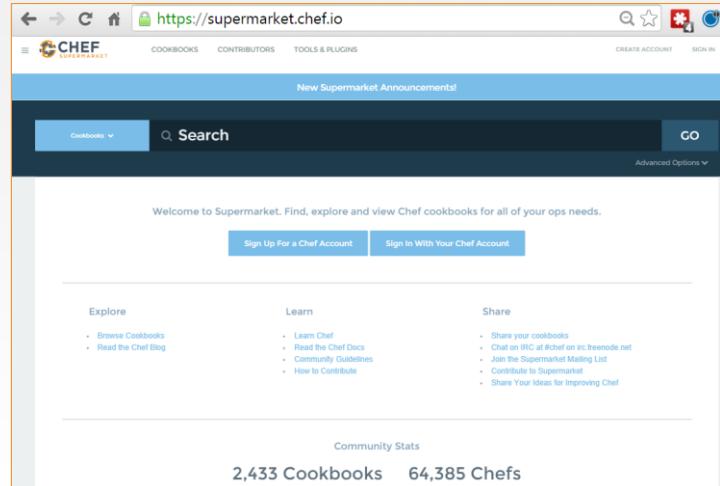
But what if we told you someone already wrote that cookbook?

Someone already has and that cookbook is available through the community site called Supermarket. Supermarket is a public repository of all the cookbooks shared by other developers, teams, and companies who want to share their knowledge and hard work with you to save you time.

## Slide 6

## Community Cookbooks

- Community cookbooks are managed by individuals.
- Chef does not verify or approve cookbooks in the Supermarket.
- Cookbooks may not work for various reasons.
- Still, there are real benefits to community cookbooks.



©2016 Chef Software Inc.

10-6



An important thing to remember is that on the community site are cookbooks managed by individuals. Chef does not verify or approve the cookbooks found in the Supermarket. These cookbooks solved problems for the original authors and then they decided to share them. This means that the cookbooks you find in the Supermarket may not be built or designed for your platform. It may not take into special consideration your needs and requirements. It may no longer be actively maintained.

Even if the cookbook does not work as a whole, there is still value in reading and understand the source code and extracting the pieces you need when creating your own. With all that said, there is a real benefit to the community site. When you find a cookbook that helps you deliver value quickly, it can be a tremendous boon to your productivity. This is what we are going to take advantage of with the haproxy cookbook.

Slide 7

# EXERCISE

## Load Balancer



*Adding a load balancer will allow us to better grow our infrastructure.*

**Objective:**

- Find or Create a Cookbook to Manage a load balancer
- Configure the load balancer to send traffic to the new node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy (load balancer) cookbook

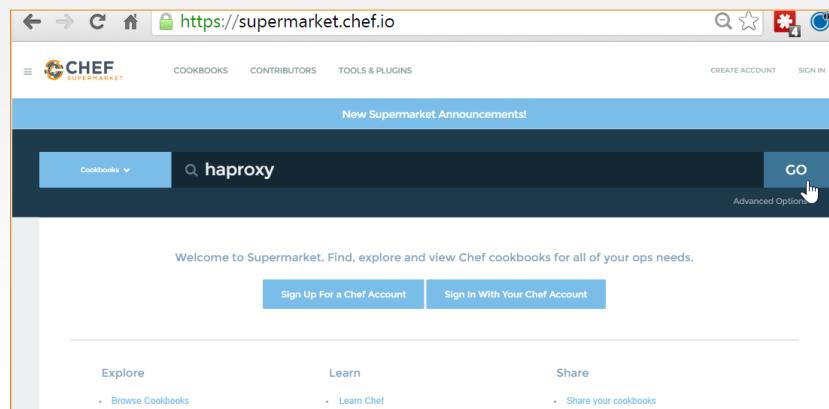
Let's find the haproxy (load balancer) cookbook within the community site to learn more about it.

## Slide 8

# Searching in the Supermarket

## STEPS

1. Visit [supermarket.chef.io](https://supermarket.chef.io)
2. Select the search field and type in [haproxy](#) in the search field. Then click the **GO** button.
3. Click the resulting [haproxy](#) link.



From the Supermarket main page type the search term "haproxy" and the click the **GO** button.

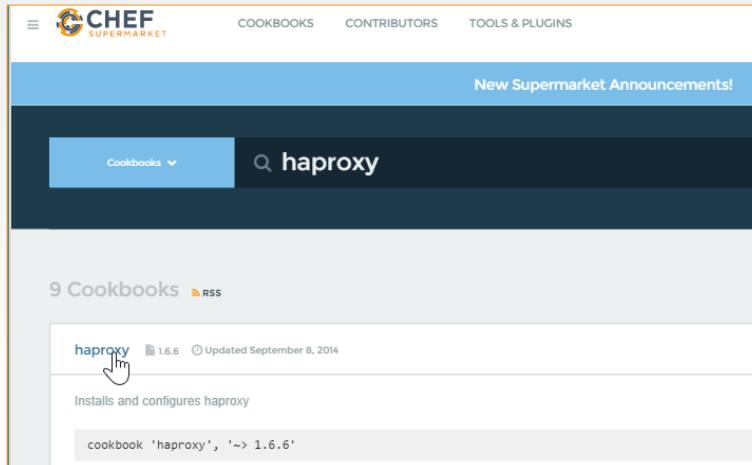
Below the search term will show us all the matching cookbooks. The haproxy cookbook is in that result set.

## Slide 9

# Searching in the Supermarket

## STEPS

1. Visit [supermarket.chef.io](http://supermarket.chef.io)
2. Select the search field and type in **haproxy** in the search field. Then click the **GO** button.
3. Click the resulting **haproxy** link.



Cookbooks usually map one-to-one to a piece of software and usually are named after the piece of software that they manage. Select the cookbook named haproxy from the search results.

## Slide 10

## Supermarket Cookbooks

On the right-hand side we can see the individuals that maintain the cookbook...

On the left, we are presented with the various ways we can install the cookbook...

The screenshot shows the Chef Supermarket interface for the 'haproxy' cookbook. On the left, there's a summary box with the title 'haproxy', version '1.6.6', and a description: 'Installs and configures haproxy'. It includes links for 'Berkshelf', 'Librarian', and 'Knife', and a dependency line: 'cookbook \'haproxy\', \'~> 1.6.6\''. Below this is a 'README' section with a link to 'Requirements'. On the right, there's a 'heavywater Heavy Water Software' profile with a list of maintainers, a 'View Source' button, and sections for 'UPDATED SEPTEMBER 8, 2014', 'PLATFORMS', and 'LICENSE'.

©2016 Chef Software Inc. 10-10 

At this point you are presented with information that describes the cookbook. Starting on the right-hand side we see the individuals that maintain the cookbook, a link to view the source details, last updated date, supported platforms, licensing, and a link to download the cookbook.

On the left, we are presented with the various ways we can install the cookbook, the README that describes information about the cookbook, any cookbooks that this cookbook may depend on, a history of the changes, and its food critic rating--which is a code evaluator for best practices.

## Slide 11

# Supermarket Cookbooks

The area to focus most of your attention from the beginning is the README.

Reading and understanding the README at a glance is difficult. It is a skill that comes with time.

The screenshot shows the README page for the 'haproxy Cookbook' on the Chef Supermarket. The page has a header with tabs for 'README', 'Dependencies', 'Changelog', and 'Foodcritic'. Below the header, the title 'haproxy Cookbook' is displayed, followed by a brief description: 'Installs haproxy and prepares the configuration location.' A 'Requirements' section lists supported platforms: Ubuntu (10.04+ due to config option change), Redhat (6.0+), and Debian (6.0+). The 'Attributes' section contains a list of attributes and their descriptions, including 'node['haproxy'][‘incoming\_address’]' which sets the address to bind the haproxy process on 0.0.0.0 (all addresses) by default, and 'node['haproxy'][‘incoming\_port’]' which sets the port on which haproxy listens. There is also a note about 'node['haproxy'][‘members’]' being used by the default recipe to specify member systems to add. Default. At the bottom of the attributes section, there is a code block containing the line: `[{"hostname" => "localhost"},`

The area to focus most of your attention from the beginning is the README. The README describes the various attributes that are defined within the cookbook and the purpose of the recipe. This is the same README file found in the cookbooks we currently have within our organization. This one, however, has had far more details added to give new users like us the ability to understand more quickly what the cookbook does and how it does it.

Reading and understanding the README at a glance is difficult. It is a skill that comes with time. For the haproxy cookbook, there is an attribute that establishes the members that receive the proxy requests from the load balancer. This is available in a node attribute available through `node['haproxy']['members']`.

## Slide 12

# Supermarket Cookbooks

These node attributes are different than the automatic ones defined by Ohai.

Attributes defined in a cookbook are not considered automatic.

## Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4000,
  "ssl_port" => 4000
}, {
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4001,
  "ssl_port" => 4001
}]
```

- `node['haproxy']['member_port']` - the port that member systems will be listening on if not otherwise

<https://docs.chef.io/attributes.html>

Prior to this point we have seen how node attributes are defined by Ohai but cookbooks also have this ability to define node attributes. These node attributes are different than the ones defined by Ohai as well. Ohai attributes are considered automatic attributes and generally inalienable characteristics about the node.

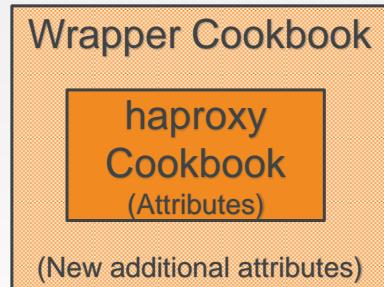
Attributes defined in a cookbook are not considered automatic. They are simply default values that we may change. There are many ways that we provide new default values for these. One way that we will learn is defining a wrapper cookbook.

Slide 13

## Supermarket Cookbooks

A wrapper cookbook is a new cookbook that encapsulates the functionality of the original cookbook.

It defines new default values for the recipes.



<https://docs.chef.io/supermarket.html#wrapper-cookbooks>

<https://www.chef.io/blog/2013/12/03/doing-wrapper-cookbooks-right/>

A wrapper cookbook is a new cookbook that encapsulates the functionality of the original cookbook but allows us to define new default values for the recipes.

This is a common method for overriding cookbooks because it allows us to leave the original cookbook untouched. We simply provide new default values that we want and then include the recipes that we want to run.

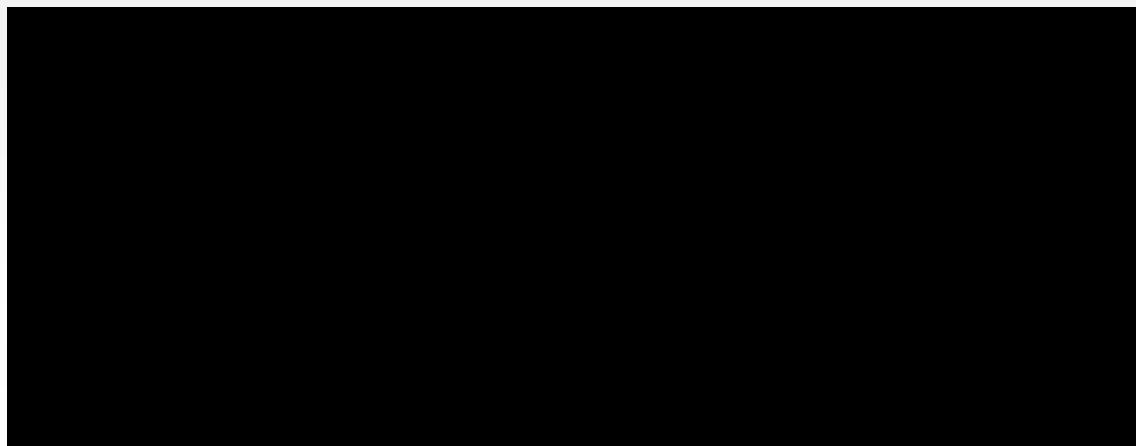
Let's generate our wrapper cookbook named `myhaproxy`. Traditionally we would name the cookbook with a prefix of the name of our company and then follow it by the cookbook name '`company-cookbook`'.

Slide 14

## Returning the Chef Repository Directory



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



Change to your chef-repo directory

## Generating a New Cookbook



```
$ chef generate cookbook cookbooks/myhaproxy
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy] action create
    - create new directory C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy
  * template[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb] action
create_if_missing
    - create new file C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb
    - update content in file C:/Users/sdelfante/chef-
repo/cookbooks/myhaproxy/metadata.rb from none to 899276
      (diff output suppressed by config)
  * template[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/README.md] action
create_if_missing
```

Generate your new cookbook.

## Creating a Dependency in the Cookbook



~/chef-repo/cookbooks/myhaproxy/metadata.rb

```
name          'myhaproxy'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures myhaproxy'  
long_description 'Installs/Configures myhaproxy'  
version        '0.1.0'  
  
depends 'haproxy', '~> 1.6.6'
```

Set up a dependency within your haproxy cookbook. Establishing this dependency informs the Chef Server that whenever you deliver this cookbook to a node, you should also deliver with it the mentioned dependent cookbooks.

This is important because your cookbook is simply going to set up new default values and then execute the recipes defined in the original cookbook.

Slide 17

# EXERCISE

## Load Balancer



*Adding a load balancer will allow us to better grow our infrastructure.*

**Objective:**

- ✓ Find or create a cookbook to manage a load balancer
- Configure the load balancer to send traffic to the new node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy cookbook

Now that you have the dependency on the haproxy cookbook in your wrapper cookbook, you need to learn what new default values you need to add to the recipe.

# Supermarket Cookbooks

Currently, the haproxy cookbook assumes that there are two different services running on the localhost at port 4000 and port 4001.

In a moment, you'll need to change that.

## Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4000,
  "ssl_port" => 4000
}, {
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4001,
  "ssl_port" => 4001
}]
```

- `node['haproxy']['member_port']` - the port that member systems will be listening on if not otherwise

<https://docs.chef.io/supermarket.html#wrapper-cookbooks>

Currently the haproxy cookbook assumes that there are two different services running on the localhost at port 4000 and port 4001. The haproxy process will relay messages to itself to those two ports.

That is not our configuration. First, we currently only have one system that we want to route traffic. Second, we want to have the traffic routed not to localhost but instead to our webserver, node1, which will have a completely different hostname and IP address.

Slide 19

# CONCEPT

## 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.

So we want to apply the default recipe that sets up the proxy server but we want to make some adjustments. We cannot change the original cookbook itself so we are instead going to load the contents of the original recipe in a recipe that we do control in our new cookbook. This is possible through the helper method `include_recipe`.

## Including the haproxy's default in recipe in this recipe

```
~/chef-repo/cookbooks/myhaproxy/recipes/default.rb
```

```
#  
# Cookbook Name:: myhaproxy  
# Recipe:: default  
#  
# Copyright (c) 2016 The Authors, All Rights  
Reserved.
```

```
include_recipe 'haproxy::default'
```

First, within the myhaproxy cookbook you will use the include\_recipe method to specify the fully-qualified name of the cookbook and recipe that you want to execute. In this case, when you run your wrapped cookbooks recipe, you'll want it to run the original cookbook's default recipe.

This is often called wrapping a recipe because our recipe calls the original recipe. The benefit is that we can define content before this recipe gets applied, to override functionality, or after this recipe gets applied, to replace functionality.

## Beginning to Replace Load Balancer Members

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
#  
  
node['haproxy']['members'] = [  
{  
    'hostname' => 'localhost',  
    'ipaddress' => '127.0.0.1',  
    'port' => 4000,  
    'ssl_port' => 4000  
, {  
    'hostname' => 'localhost',  
    'ipaddress' => '127.0.0.1',  
    'port' => 4001,  
    'ssl_port' => 4001  
}  
  
include_recipe 'haproxy::default'
```

Without changing anything any further, using this cookbook will simply execute the original cookbooks' recipe with all the same default values. Before you execute that recipe, you'll need to override the default values with your own.

Copy and paste the original default values into your recipe, as shown here. This is not the real information of our servers. We now want to find that information and replace this content with the true value of our node.

## Viewing Help on the Node Show Subcommand



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

```
knife node show NODE (options)
  -a ATTR1 [--attribute ATTR2] ,  Show one or more attributes
      --attribute
  -s, --server-url URL          Chef Server URL
      --chef-zero-host HOST       Host to start chef-zero on
      --chef-zero-port PORT      Port (or port range) to start chef-zero on.

Port ranges
  -k, --key KEY                 API Client Key
      --[no-]color              Use colored output, defaults to false on

Windows, true
  -c, --config CONFIG           The configuration file to use
      --defaults                Accept default values for all questions
  -d, --disable-editing         Do not open EDITOR, just accept the data as is
```

This new default value for the haproxy members needs to define the information about the webserver node, node1. So you need to capture the node's public host name and public IP address.

The 'knife node show' command will display information about the node. You can ask to see a specific attribute on a node with the -a flag or the --attribute flag.

## Viewing the Node's IP Address



```
$ knife node show node1 -a ipaddress
```

```
node1:  
  ipaddress: 172.31.8.68
```

You can display the IP address of node1 with the '-a' flag and specifying the attribute 'ipaddress'.

With cloud providers that generate machines for you often assign internal IP addresses, those values may not work properly.

Slide 24

# PROBLEM

## Amazon EC2 Instances



The IP address and host name are unfortunately not how we can address these nodes within our recipes.

The reason you may need to ask the node for a different set of attributes is that we are using Amazon as a cloud provider for our instances. These instances are displaying the internal IP address when we ask for the ipaddress attribute.

Ohai collects attributes from the current cloud provider and makes them available in an attribute named 'cloud'. We can look at the cloud attribute on our first node and see that it returns for us information about the node.

## Viewing the Node's Cloud Details



```
$ knife node show node1 -a cloud
```

```
node1:
  cloud:
    local_hostname: ip-172-31-8-68.ec2.internal
    local_ipv4: 172.31.8.68
    private_ips: 172.31.8.68
    provider: ec2
    public_hostname: ec2-54-175-46-24.compute-1.amazonaws.com
    public_ips: 54.175.46.24
    public_ipv4: 54.175.46.24
```

If you use 'knife node show' to display the 'cloud' attribute for node1, you will see the local, private, and public connection information.

Capture and write down the public hostname and the public ipv4 address of node1. You will need this in the recipe you are going to write.

## Inserting Real Node Data into the Attributes



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node['haproxy']['members'] = [
  {
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4000,
    'ssl_port' => 4000
  },
  {
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',
    'ipaddress' => '52.8.71.11',
    'port' => 80,
    'ssl_port' => 80
  }
]

include_recipe 'haproxy::default'
```

+

Remove one of the entries within the members array (shown in red).

Then update the information for the remaining member to include the public ipaddress and hostname for node1 (shown in green).

## Setting the Default Attributes Precedence Level



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{  
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',  
    'ipaddress' => '52.8.71.11',  
    'port' => 80,  
    'ssl_port' => 80  
}]  
  
include_recipe 'haproxy::default'
```

To replace a default attribute in a recipe you have to use:

'node.**default**['haproxy']['members']...'

So you need to change: 'node['haproxy']['members']' to

'node.**default**['haproxy']['members']'

Slide 28

## Viewing the Complete Recipe



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{  
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',  
    'ipaddress' => '52.8.71.11',  
    'port' => 80,  
    'ssl_port' => 80  
}]  
  
include_recipe 'haproxy::default'
```

The final default recipe for the wrapper cookbook 'myhaproxy' looks like the above.

Save your recipe file.

Slide 29

# EXERCISE

## Load Balancer



*Adding a load balancer will allow us to better grow our infrastructure.*

**Objective:**

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy cookbook

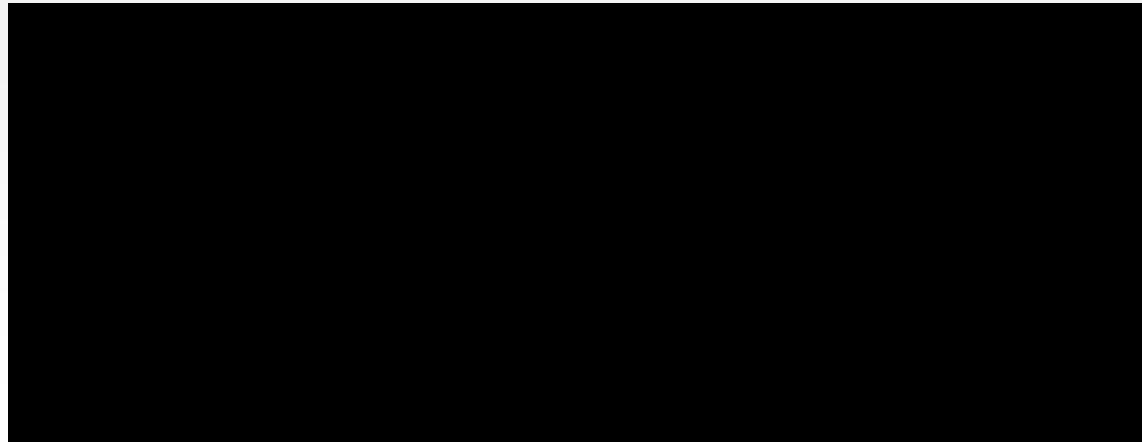
You have completed creating the wrapper cookbook. It is time to upload to the Chef Server.

Slide 30

## Moving to the Cookbook's Directory



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



Let's review that lab.

You change into the directory for the 'myhaproxy' cookbook.

Slide 31

## Installing the Cookbook's Dependencies



```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myhaproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using build-essential (2.2.3)
Using cpu (0.2.0)
Using haproxy (1.6.6)
Using myhaproxy (0.1.0) from source at .
```

We use the Berkshelf to upload our cookbooks. This is where Berkshelf really shines as a tool.

Run the command "berks install". When you run this command for a cookbook that has a dependency, you'll see that Berkshelf will download the haproxy cookbook and its dependencies as well. The haproxy cookbook is dependent on the build-essential cookbook and the cpu cookbook. If any of those cookbooks had dependencies, berkshelf would find those and download them as well.

Slide 32

## Uploading the Cookbook and Dependencies



```
$ berks upload
```

```
Uploaded build-essential (2.2.3) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded cpu (0.2.0) to: 'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded haproxy (1.6.6) to: 'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded myhaproxy (0.1.0) to: 'https://api.opscode.com:443/organizations/steveessentials2'
```

After installing all the necessary dependent cookbooks, we used 'berks upload' to send the cookbook and all its dependencies to the Chef Server. This is again an easier method to manage dependencies instead of manually identifying the dependencies and then uploading each single cookbook at a time.

Slide 33

## Verifying the Cookbook has Been Uploaded



```
$ knife cookbook list
```

apache	0.2.1
build-essential	2.2.3
cpu	0.2.0
haproxy	1.6.6
myhaproxy	0.1.0

When that is complete you can verify that you've uploaded your cookbook and all of its dependencies.

Slide 34

# EXERCISE

## Load Balancer



*Adding a load balancer will allow us to better grow our infrastructure.*

**Objective:**

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- ✓ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the haproxy cookbook

The myhaproxy cookbook's default recipe is ready to be assigned to a run list of a node. So we'll need another node. The new load balancer node.

## Bootstrapping a New Node



```
$ knife bootstrap FQDN2 -x USER -P PWD --sudo -N node2
```

```
Creating new client for node2
Creating new node for node2
Connecting to ec2-54-210-192-12.compute-1.amazonaws.com
ec2-54-210-192-12.compute-1.amazonaws.com Starting first Chef
Client run...
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: []
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com Compiling Cookbooks...
```

First you bootstrap a new node named node2.

Slide 36

## Viewing the New Node's Data



```
$ knife node show node2
```

```
Node Name:    node2
Environment:  _default
FQDN:        ip-172-31-0-128.ec2.internal
IP:          54.210.192.12
Run List:
Roles:
Recipes:
Platform:    centos 6.6
Tags:
```

After the node is bootstrapped, validate that it was added correctly to the organization.

Slide 37

## Defining a Run List for the Node



```
$ knife node run_list add node2 "recipe[myhaproxy]"
```

```
node2:  
  run_list: recipe[myhaproxy]
```

Define an initial run list for that node to converge the default recipe of the myhaproxy cookbook.

Slide 38

## Validating the Run List has been Set



```
$ knife node show node2
```

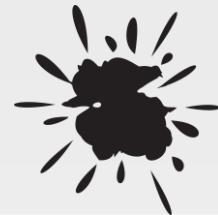
```
Node Name:    node2
Environment:  _default
FQDN:        ip-172-31-0-128.ec2.internal
IP:          54.210.192.12
Run List:    recipe[myhaproxy]
Roles:
Recipes:
Platform:   centos 6.6
Tags:
```

Ensure the run list has been set correctly for node2.

Slide 39

# PROBLEM

## SSH Woes



Logging into both systems is a pain. We can use another knife tool to allow us to send commands to all of our nodes.

We asked you to login to that remote node and run 'sudo chef-client' to apply the new run list defined for that node. This does in fact work but considering that we may need to execute this command for this node and many future nodes, it seems like a lot of windows and commands that we would need to execute.

## Viewing the Help for the ssh Subcommand



```
$ knife ssh --help
```

```
knife ssh QUERY COMMAND (options)
  -a, --attribute ATTR           The attribute to use for opening the connection
  - default depends on the context

  -s, --server-url URL          Chef Server URL
    --chef-zero-host HOST        Host to start chef-zero on
    --chef-zero-port PORT        Port (or port range) to start chef-zero on.
Port ranges like 1000,1010 or 8889-9999 will try all given ports until one works.

  -k, --key KEY                 API Client Key
    --[no-]color                Use colored output, defaults to false on
Windows, true otherwise

  -C, --concurrency NUM         The number of concurrent connections
  -c, --config CONFIG           The configuration file to use
    --defaults                  Accept default values for all questions
```

To make our lives easier, the 'knife' command provides a subcommand named 'ssh' that allows us to execute a command across multiple nodes that match a specified search query.

Slide 41

## Running a Command Across All Nodes



```
$ knife ssh "*:*" -x USERNAME -P PASSWORD "sudo chef-client"

ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-175-46-24.compute-1.amazonaws.com resolving cookbooks for run list:
["apache"]
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-54-175-46-24.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com      - apache
ec2-54-175-46-24.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-175-46-24.compute-1.amazonaws.com Converging 3 resources
ec2-54-175-46-24.compute-1.amazonaws.com Recipe: apache::server
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com      - build-essential
```

There are a lot of options for defining the search criteria that we will continue to explore. The most important criteria in this instance is star-colon-star. This means that we want to issue a command to all nodes.

So if you want to execute a "sudo chef-client" run for all of your nodes, you should write out this command. You would need to provide the user name to log into the system, the password for that system, and then finally the command to execute. In this way, you could easily ask your nodes to update from your current workstation as long as they all have the same login credentials. For more security, you should likely use SSH keys and forego specifying a username and password

Slide 42

## Viewing the Website is Being Proxied

URL of load balancer.

Output from the web server.



Point a web browser to the URL or public IP address of your load balancer. It should display the web page of the web server node that the load balancer is configured to serve.

Slide 43

# EXERCISE

## Load Balancer



*Adding a load balancer will allow us to better grow our infrastructure.*

**Objective:**

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- ✓ Upload cookbook to Chef Server
- ✓ Bootstrap a new node that runs the haproxy cookbook

With your node running the myhaproxy's cookbook's default recipe--relaying traffic to your first node running the apache cookbook's default recipe--you have moved closer to creating the original topology we set out to define today.

Slide 44

# DISCUSSION



## Discussion

What are the benefits and drawbacks of the Chef Super Market?

Is your team able to leverage community cookbooks? Is the team able to contribute to community cookbooks?

Why do you use a wrapper cookbook? When might you decide to not wrap the cookbook?

Answer these questions.

Slide 45

# DISCUSSION



## Q&A

What questions can we help you answer?

- Chef Super Market
- Wrapper Cookbooks
- Node Attributes
- knife ssh

What questions can we help you answer?

In general or about specifically about Chef Super Market, wrapper cookbooks, node attributes, the 'knife ssh' command.

Slide 46



©2016 Chef Software Inc.

## 11: Managing Multiple Nodes

### Managing Multiple Nodes

Create another web server and add it as a proxy member

This section's goal is to have you bootstrap another node, this time a web server, and add it to the proxy members.

## Slide 2

# Objectives

After completing this module, you should be able to

- Bootstrap, update the run\_list, and run chef-client on a node
- Append values to an attribute within a recipe
- Version a cookbook and upload it to the Chef Server

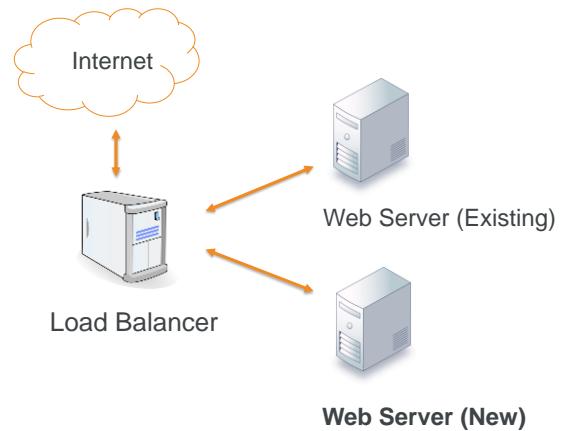
In this module you will learn how to bootstrap, update the run list, and run chef-client on a node. You will also learn how to update a default attribute within a recipe, version and upload a cookbook.

Slide 3

## Managing User Traffic

You already configured the load balancer and one web server node.

In this module you'll add another node to the load balancer's list of web server's it is serving.



After completing this module, you will have configured three nodes:

- Node 1: A web server
- Node 2: The load balancer
- Node 3: Another web server

Slide 4

# LAB



## Lab: Another Web Node

- Bootstrap a new node
- Update the run list of the new node to include the web server cookbook
- Run chef-client on that system
- Verify that the node's web server is functional

Now it's time to create a third node. The third node will be the second web server node.

We will provide you with a new node for the following exercise.

## Bootstrapping a New Node



```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node3
```

```
Connecting to ec2-54-210-86-164.compute-1.amazonaws.com
ec2-54-210-86-164.compute-1.amazonaws.com Starting first Chef Client run...
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: []
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com [2016-09-16T17:36:14+00:00] WARN: Node
node3 has an empty run list.
ec2-54-210-86-164.compute-1.amazonaws.com Converging 0 resources
ec2-54-210-86-164.compute-1.amazonaws.com
ec2-54-210-86-164.compute-1.amazonaws.com Running handlers:
ec2-54-210-86-164.compute-1.amazonaws.com Running handlers complete
ec2-54-210-86-164.compute-1.amazonaws.com Chef Client finished, 0/0 resources
```

Bootstrap the new node and name it node3.

## Viewing the Details of the New Node



```
$ knife node show node3
```

```
Node Name:    node3
Environment:  _default
FQDN:        ip-172-31-0-127.ec2.internal
IP:          54.210.86.164
Run List:
Roles:
Recipes:
Platform:    centos 6.6
Tags:
```

Verify that you bootstrapped the node.

## Setting the Run List of the New Node



```
$ knife node run_list add node3 "recipe[apache]"
```

```
node3:  
  run_list: recipe[apache]
```

Set the run list for this node by running the apache cookbook's default recipe.

## Slide 8

## Converging the Run List for All Nodes



```
$ knife ssh "*:*" -x USERNAME -P PWD "sudo chef-client"
```

```
ec2-54-175-46-24.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-192-12.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
ec2-54-175-46-24.compute-1.amazonaws.com resolving cookbooks for run list: ["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list: ["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list: ["myhaproxy"]
ec2-54-175-46-24.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com      - apache
ec2-54-175-46-24.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-175-46-24.compute-1.amazonaws.com Converging 3 resources
ec2-54-175-46-24.compute-1.amazonaws.com Recipe: apache::server
```

Apply that run list by logging into that node and running sudo chef-client or remotely administer the node with the 'knife ssh' command as shown here.

Slide 9

## Verifying that the New Node Serves the Page



Verify that the node serves up the default html page that contains the node's internal IP address and hostname.

Slide 10

# LAB



## Lab: Another Web Node

- ✓ Bootstrap a new node
- ✓ Update the run list of the new node to include the web server cookbook
- ✓ Run chef-client on that system
- ✓ Verify that the node's web server is functional

That's it! You've bootstrapped for the final time. You have three nodes which will allow you to demonstrate a working load balancer.

Slide 11

# LAB



## Lab: Update the Load Balancer

- Update the wrapped proxy server cookbook to include the new web node as a member.
- Upload that cookbook to the Chef Server
- Run chef-client on that system
- Verify that the load balancer delivers traffic to both web server nodes.

Now that you have the third node, it is time to add that node to the member's list for the load balancer.

## Displaying the New Node's IP and Hostname



```
$ knife node show node3 -a cloud
```

```
node1:  
cloud:  
  local_hostname: ip-172-31-8-64.ec2.internal  
  local_ipv4:    172.31.8.64  
  private_ips:   172.31.8.64  
  provider:     ec2  
  public_hostname: ec2-54-176-64-173.us-west-1.compute.amazonaws.com  
  public_ips:    54.175.46.48  
  public_ipv4:   54.175.46.48
```

If you use 'knife node show' to display the 'cloud' attribute for node3, you will see the local, private, and public connection information.

Capture and write down the public hostname and the public ipv4 address of node3. You will need this in the recipe you are going to write.

## Adding the New Node to the Load Balancer



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{  
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',  
    'ipaddress' => '52.8.71.11',  
    'port' => 80,  
    'ssl_port' => 80  
}, {  
    'hostname' => 'ec2-54-176-64-173.us-west-1.compute.amazonaws.com',  
    'ipaddress' => '54.175.46.48',  
    'port' => 80,  
    'ssl_port' => 80  
}  
]  
include_recipe 'haproxy::default'
```

Add the second web server (node3) to the load balancer's (LB) members list. You may need to run 'knife node show node3 -a cloud' to get the hostname and ipaddress values.

## Updating the Cookbook's Version



~/chef-repo/cookbooks/myhaproxy/metadata.rb

```
name          'myhaproxy'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures myhaproxy'  
long_description 'Installs/Configures myhaproxy'  
version        '0.2.0'  
  
depends 'haproxy', '~> 1.6.6'
```

Update the version number in myhaproxy cookbook's metadata.

## Installing the Cookbook's Dependencies



```
$ cd ~/chef-repo/cookbooks/myhaproxy  
$ berks install  
  
Resolving cookbook dependencies...  
Fetching 'myhaproxy' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Using build-essential (2.2.3)  
Using cpu (0.2.0)  
Using haproxy (1.6.6)  
Using myhaproxy (0.2.0) from source at .
```

Change into the 'myhaproxy' cookbook directory and then run 'berks install' to install any dependencies for the 'myhaproxy' cookbook.

## Uploading the Cookbook to the Chef Server



```
$ berks upload
```

```
Uploaded build-essential (2.2.3) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded cpu (0.2.0) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded haproxy (1.6.6) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded myhaproxy (0.2.0) to:  
'https://api.opscode.com:443/organizations/steveessentials2'
```

Run 'berks upload' to upload the myhaproxy cookbook to Chef Server.

## Converging the Run List for All Nodes



```
$ knife ssh "*:*" -x USERNAME -P PWD "sudo chef-client"
```

```
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
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
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list:
["apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com    - apache
ec2-54-175-46-24.compute-1.amazonaws.com  Compiling Cookbooks...
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com  Converging 3 resources
ec2-54-175-46-24.compute-1.amazonaws.com  Recipe: apache::server
```

Converge the cookbook by logging into that node and running 'sudo chef-client' or remotely administer the node with the 'knife ssh' command as shown here.

Within the output you should see the haproxy configuration file will update with a new entry that contains the information of the second member (node3).

## Slide 18

## Lab: Test the Load Balancer

The image shows three separate browser windows, each displaying a "Hello, world!" page with some configuration details. A blue arrow points from the middle window to the bottom window, indicating a transition or comparison between them.

- Top window: URL: ec2-204-236-155-223.us-west-1.compute.amazonaws.com, Content: Hello, world!  
ipaddress: 10.198.51.26  
hostname: ip-10-198-51-26
- Middle window: URL: ec2-50-18-19-208.us-west-1.compute.amazonaws.com, Content: Hello, world!  
ipaddress: 10.198.51.26  
hostname: ip-10-198-51-26
- Bottom window: URL: ec2-54-176-64-173.us-west-1.compute.amazonaws.com, Content: Hello, world!  
ipaddress: 10.197.105.148  
hostname: ip-10-197-105-148

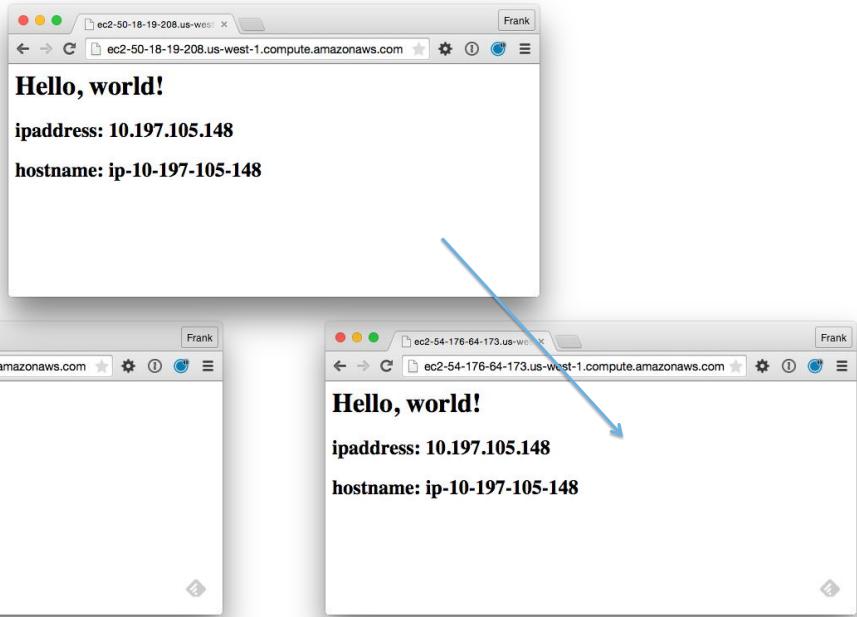
©2016 Chef Software Inc. 11-18 

Point a web browser to the URL of your Proxy server and then click Refresh a few times. You should see each of your web server's HTML page as the Proxy server switches between each web node.

This is not a very scientific way of seeing that the proxy server is balancing requests between these two web nodes.

## Slide 19

# Lab: Test the Load Balancer



Slide 20

# LAB



## Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new web node as a member.
- ✓ Upload that cookbook to the Chef Server
- ✓ Run chef-client on that system
- ✓ Verify that the load balancer delivers traffic to both web server nodes.

Great! Now the load balancer successfully delivers traffic to both of the web server nodes.

Slide 21

# DISCUSSION

## Discussion



What is the process to setup a third web node?

What is the process for removing a web node?

What is the most manual part of the process?

Answer these questions.

Slide 22

# DISCUSSION



## Q&A

What questions can we help you answer?

Slide 23



## 12: Roles

### Roles

Giving Your Nodes a Role

## Slide 2

# 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.

## Slide 3

# 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.

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 can have zero or more roles assigned to it.

## Slide 4

# 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.

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.

# EXERCISE

## 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.

## Viewing the Help for Knife Role Subcommand



```
$ 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]
```

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.

Slide 7

## Viewing the Roles on the Chef Server



```
$ knife role list
```



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

Slide 8

## Creating the roles Directory



```
$ mkdir roles
```



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

## Defining the Load Balancer Role



~/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.

Slide 10

## Uploading the Role 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.

Slide 11

## Viewing the Roles on the Chef Server



```
$ 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.

## Viewing the Details about 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:
```

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`.

## Viewing the Help for the knife node subcommand



```
$ 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)
```

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

## Setting the Run List for the Node



```
$ 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.

## Viewing the Details about the Node



```
$ 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'.

## Slide 16

## Converging all Load Balancers



```
$ 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.

Slide 17

# EXERCISE

## 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.

## Defining the Web Role

~/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.

Slide 19

## Uploading the Web Role to the Chef Server



```
$ 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.

Slide 20

## Verifying the Roles on the Chef Server



```
$ knife role list
```

```
load_balancer
```

```
web
```

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

Slide 21

## Viewing the Details 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?

Slide 22

## Setting the Node'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.

Slide 23

## Setting the Other Node'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.

## Converging 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.

# EXERCISE

## 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.

Slide 26

# 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?

Answer these questions.

Slide 27

# DISCUSSION



## Q&A

What questions can we help you answer?

Slide 28



©2016 Chef Software Inc.

## 13: Search

# Search

Update a Cookbook to Dynamically Use Nodes with the Web Role

## Slide 2

## Objectives

After completing this module, you should be able to

- Describe the query syntax used in search
- Build a search into your recipe code
- Create a Ruby Array and Ruby Hash
- Update the myhaproxy wrapper cookbook (for the load balancer) to dynamically use nodes with the web role

In this module you will learn how to describe the query syntax used in search, build a search into your recipe code, create a ruby array and ruby hash, and update the myhaproxy wrapper cookbook to dynamically use nodes with the web role.

Slide 3

# CONCEPT

## Search



So far we have seen how Chef is able to manage the policy of the nodes.

We have two web servers and one load balancer.

So far we have seen how Chef is able to manage the policy of the nodes within our infrastructure.

We have two web servers and one load balancer. As more customers come to our website we can continue scale up to meet that demand.

## Slide 4

# CONCEPT

## Search



To add new servers as load balancer members, we would need to bootstrap a new web server and then update our load balancer's myhaproxy cookbook recipe.

That seems inefficient to have to update a cookbook recipe.

To add new servers as load balancer members, we would need to bootstrap a new web server and then update our myhaproxy cookbook to include that new web server. But that seems dramatically inefficient to have to update a cookbook recipe.

A more ideal solution would be for the recipe to instead discover all of the web servers within our organization and automatically add them to a list of available members for our load balancer.

Slide 5

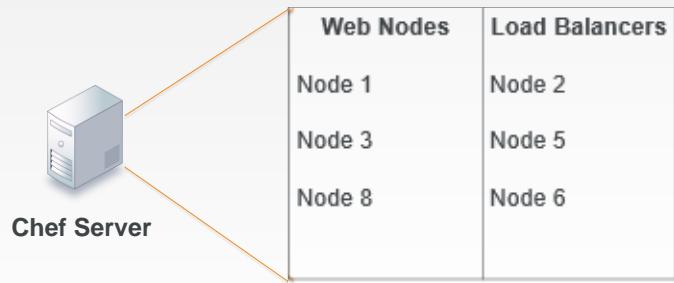
## The Chef Server and Search

Chef Server maintains a representation of all the nodes within our infrastructure that can be searched on.

Search is a service discovery tool that allows us to query the Chef Server.

[https://docs.chef.io/chef\\_search.html](https://docs.chef.io/chef_search.html)

[https://docs.chef.io/chef\\_search.html#search-indexes](https://docs.chef.io/chef_search.html#search-indexes)



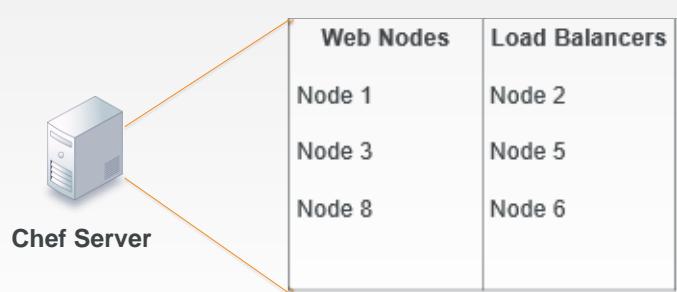
The Chef Server maintains a representation of all the nodes within an infrastructure and provides a way for us to discover these systems through Search.

Search is a service discovery tool that allows us to query the Chef Server across a few indexes. One such index is on our nodes.

Slide 6

## The Chef Server and Search

We can ask the Chef Server to return all the nodes or a subset of nodes based on the query syntax that we provide it through `knife search` or within our recipes through `search`.



We can ask the Chef Server to return back to us all the nodes or a subset of nodes based on the query syntax that we provide it through the knife command `knife search` or within our recipes through the `search` method.

## Slide 7

## Search Criteria

The search criteria that we have been using up to this point is "`*:*`"

Querying and returning every node is not what we need to solve our current problem.

Scenario: We want only to return a subset of our nodes--only the nodes that are web servers.



We have been using a form of the search criteria already when we have employed the ``knife ssh`` command. The search criteria that we have been using up to this point is "`*:*`" which we explained matched every node within our infrastructure.

Querying and returning every node is not exactly what we need to solve our current problem. Scenario: We want only to return a subset of our nodes--only the nodes that are web servers.

Let's examine the search criteria more so we can understand how it works and how we can use it to find a subset of the nodes--only the nodes that are web servers.

## Search Syntax

A search query is comprised of two parts: the key and the search pattern. A search query has the following syntax:

`key:search_pattern`

...where key is a field name that is found in the JSON description of an indexable object on the Chef server and search\_pattern defines what will be searched for,

A search query is comprised of two parts: the key and the search pattern. A search query has the following syntax:

`key:search_pattern`

...where key is a field name that is found in the JSON description of an indexable object on the Chef server (a role, node, client, environment, or data bag) and search\_pattern defines what will be searched for.

## Search Syntax within a Recipe

```
all_web_nodes = search('node', 'role:web')
```

creates and names a variable

assigns the value of the  
operation on the right  
into the variable on the left

invokes the search method

the index or items to search

the search criteria - key:value

Search within a recipe is done through a `search` method that is available within the recipe.

The `search` method accepts two arguments. The first argument is a string or variable that contains the index or item to search on the Chef Server. These are: nodes; roles; and environments. The second argument is a string or variable that contains the search criteria to scope the results. This is using the notation 'key:value'.

The result of the search method is stored in a local variable that is named 'all\_web\_nodes'. Variables within Ruby are created immediately when you assign them.

## Search Syntax within a Recipe

```
all_web_nodes = search('node', 'role:web')
```

Search the Chef Server for all node objects that have the role equal to 'web' and store the results into a local variable named 'all\_web\_nodes'.

This example syntax could be translated to mean: Search the Chef Server for all node objects that have the role equal to 'web' and store the results into a local variable named 'all\_web\_nodes'.

## Hard Coding Example

```
node.default['haproxy']['members'] = [{
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',
    'ipaddress' => '52.8.71.11',
    'port' => 80,
    'ssl_port' => 80
},
{
    'hostname' => 'ec2-54-176-64-173.us-west-1.compute.amazonaws.com',
    'ipaddress' => '54.175.46.48',
    'port' => 80,
    'ssl_port' => 80
}
]
include_recipe 'haproxy::default'
```

Previously, we had been hard coding the hostname and ipaddress values in our wrapped haproxy recipe. We can request these values from the Chef Server through the `knife node show` command. The hostname and ipaddress values are captured by Ohai and sent to the Chef Server. On the Chef Server we can query those values when we ask about a specific attribute about the node. We do that by providing the `-a` flag with the name of the attribute. Because the nodes that we manage are hosted in the cloud, these attributes are stored under a parent attribute named 'cloud'.

Slide 12

# EXERCISE



## Dynamic Web Load Balancer

*Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!*

### Objective:

- Update the myhaproxy cookbook to dynamically use nodes with the web role
- Update the major version of the myhaproxy cookbook
- Upload the Cookbook
- Run chef-client on the load balancer node
- Verify the load balancer node relays requests to both web nodes

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.

## Showing node1 Cloud Attributes



```
$ knife node show node1 -a cloud
```

```
node1:  
  cloud:  
    local_hostname: ip-10-198-51-26.us-west-1.compute.internal  
    local_ipv4: 10.198.51.26  
    private_ips: 10.198.51.26  
    provider: ec2  
      public_hostname: ec2-204-236-155-223.us-west-  
1.compute.amazonaws.com  
      public_ips: 204.236.155.223  
      public_ipv4: 204.236.155.223
```

Here we are asking for all the 'cloud' attributes for 'node1'.

## Showing node3 Cloud Attributes



```
$ knife node show node3 -a cloud
```

```
node3:
  cloud:
    local_hostname: ip-10-197-105-148.us-west-1.compute.internal
    local_ipv4: 10.197.105.148
    private_ips: 10.197.105.148
    provider: ec2
    public_hostname: ec2-54-176-64-173.us-west-
1.compute.amazonaws.com
    public_ips: 54.176.64.173
    public_ipv4: 54.176.64.173
```

Here we are asking for all the 'cloud' attributes for 'node3'.

## Removing the Hard-Coded Members



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{  
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',  
    'ipaddress' => '52.8.71.11',  
    'port' => 80,  
    'ssl_port' => 80  
},  
{  
    'hostname' => 'ec2-54-176-64-173.us-west-1.compute.amazonaws.com',  
    'ipaddress' => '54.175.46.48',  
    'port' => 80,  
    'ssl_port' => 80  
}  
]  
include_recipe 'haproxy::default'
```

Edit the 'myhaproxy' cookbook's default recipe and remove the current default recipe where you hard-coded the members.

## Using Search to Find the Web Servers



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
all_web_nodes = search('node', 'role:web')

#TODO: Convert all found nodes into hashes with ipaddress,
#      hostname, port, ssl_port
#TODO: Assign all the hashes to the node's haproxy members
#      attribute.

include_recipe 'haproxy::default'
```

Replace it with an updated recipe that searches for all nodes that have the 'web' role defined.

The search method's first parameter is asking the Chef Server to look at all the nodes within our organization.

The search method's second parameter is asking the Chef Server to only return the nodes that have been assigned the role web.

All of those nodes are stored in a local variable named `all\_web\_nodes`. This is an array of node objects. It may contain zero or more nodes that match the search criteria.

## Creating an Array to Store the Converted Members

```
~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

all_web_nodes = search('node', 'role:web')

members = []

#TODO: Convert all found nodes into hashes with ipaddress,
#      hostname, port, ssl_port

node.default['haproxy']['members'] = members

include_recipe 'haproxy::default'
```

Unfortunately we cannot simply assign our array of web nodes into the haproxy's `members` attributes because it needs a hash that contains the keys 'hostname', 'ipaddress', 'port', and 'ssl\_port'. We will need to convert each of the web node objects into a structure that the haproxy member's attribute expects.

First we create an empty array and assign that empty array into a local variable named `members`. `members` is an array that we will populated with the hashes we will create later; until then we will write a TODO for us. Then we will assign that array into the `node.default['haproxy']['members']` .

## Populating the Members with Each New Member

```
~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

all_web_nodes = search('node', 'role:web')

members = []

all_web_nodes.each do |web_node|
  member = {}
  # TODO: Populate the hash with hostname, ipaddress, port, and
  #        ssl_port
  members.push(member)
end

node.default['haproxy']['members'] = members

include_recipe 'haproxy::default'
```

So we need to loop through the array of all the web nodes stored in `all\_web\_nodes`. We do that through a method available on every array object named 'each'. With the each method a block of code is provided -- you see it here from the first 'do' right after the each to the 'end' later in the file.

A block of code is an operation that you want perform on every item in the array. In our case we want to take each of the node objects and convert them into a hash object.

So every member of the array is visited and every member of the array runs through the block of code.

## Populating the Hash with Node Details

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
# ... BEFORE THE LOOP IN THE RECIPE ...

all_web_nodes.each do |web_node|
  member = {
    'hostname' => web_node['cloud']['public_hostname'],
    'ipaddress' => web_node['cloud']['public_ipv4'],
    'port' => 80,
    'ssl_port' => 80
  }
  members.push(member)
end

# ... AFTER THE LOOP IN THE RECIPE ...
```

Between the pipes we see a local variable that we are defining that exists only in the block `web\_node`. This local variable, `web\_node`, is a name we came up with to refer to each node in our array of `all\_web\_nodes`. Each web node in the array is sent through the block. When inside the block of code it is referred to as `web\_node`. Inside the block the first thing that is created is another local variable named `member` which is assigned a hash that contains the web\_node's hostname and the web\_node's ipaddress. Then the local variable `member`, which contains that hash is pushed into the array of members. This adds the member to the end of the array. When we are done looping through every web node the `members` array contains a list of all these hash objects.

## Viewing the Final Recipe

```
~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

all_web_nodes = search('node', 'role:web')

members = []

all_web_nodes.each do |web_node|
  member = {
    'hostname' => web_node['cloud']['public_hostname'],
    'ipaddress' => web_node['cloud']['public_ipv4'],
    'port' => 80,
    'ssl_port' => 80
  }
  members.push(member)
end

node.default['haproxy']['members'] = members

include_recipe 'haproxy::default'
```

This is the complete recipe source code.

A completed example can be found at:

<https://raw.githubusercontent.com/chef-training/chef-essentials-repo/myhaproxy-complete/cookbooks/myhaproxy/recipes/default.rb>

Slide 21

# EXERCISE



## Dynamic Web Load Balancer

*Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!*

### Objective:

- ✓ Update the myhaproxy cookbook to dynamically use nodes with the web role
- ❑ Update the major version of the myhaproxy cookbook
- ❑ Upload the Cookbook
- ❑ Run chef-client on the load balancer node
- ❑ Verify the load balancer node relays requests to both web nodes

The default recipe of the myhaproxy recipe is now dynamic. Every time a load balancer checks in with the Chef Server, when you run 'chef-client', it will ask the Chef Server if there are any new nodes that are web servers.

As you add nodes, your load balancer will dynamically grow to accommodate them, returning them as node objects, which are then converted to hashes, and then assigned as members.

As you remove nodes, your load balancer will dynamically shrink to accommodate them, returning a smaller set of node objects, which are then converted to hashes, and then assigned as members.

## Updating the Cookbook's Version Number



~/chef-repo/cookbooks/myhaproxy/metadata.rb

```
name          'myhaproxy'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures myhaproxy'  
long_description 'Installs/Configures myhaproxy'  
version        '1.0.0'  
  
depends 'haproxy', '~> 1.6.6'
```

First we update the version to the next major release. We set the version number to 1.0.0.

Slide 23

# EXERCISE



## Dynamic Web Load Balancer

*Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!*

### Objective:

- ✓ Update the myhaproxy cookbook to dynamically use nodes with the web role
- ✓ Update the major version of the myhaproxy cookbook
- Upload the Cookbook
- Run chef-client on the load balancer node
- Verify the load balancer node relays requests to both web nodes

The cookbook has new content with an appropriate new version. It is time to upload it to the Chef Server.

## Installing the Cookbook's Dependencies



```
$ cd ~/chef-repo/cookbooks/myhaproxy  
$ berks install
```

```
Resolving cookbook dependencies...  
Fetching 'myhaproxy' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Using build-essential (2.2.3)  
Using cpu (0.2.0)  
Using haproxy (1.6.6)  
Using myhaproxy (1.0.0) from source at .
```

Change into the cookbook's directory and then install any new dependencies that your cookbook may need at version 1.0.0.

We have no new dependencies but this is required by berkshelf whenever you update the version of the cookbook.

## Uploading the Cookbook



```
$ berks upload
```

```
Uploaded build-essential (2.2.3) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded cpu (0.2.0) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded haproxy (1.6.6) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
Uploaded myhaproxy (1.0.0) to:  
'https://api.opscode.com:443/organizations/steveessentials2'  
PS C:\Users\sdelfante\chef-repo\cookbooks\myhaproxy>
```

Upload the cookbook using the `berks upload` command.

Slide 26

# EXERCISE



## Dynamic Web Load Balancer

*Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!*

### Objective:

- ✓ Update the myhaproxy cookbook to dynamically use nodes with the web role
- ✓ Update the major version of the myhaproxy cookbook
- ✓ Upload the Cookbook
- Run chef-client on the load balancer node
- Verify the load balancer node relays requests to both web nodes

The latest cookbook is on the Chef Server but it is not yet on the node that is running the load balancer software. It is time to ask that node to converge.

## Converging the Load Balancer Node



```
$ 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
```

Use `knife ssh` and ask only the nodes with the role `load_balancer` to run `sudo chef-client`. This is more efficient than targeting all of the nodes as we did before and more accurate than targeting the node2 `"name:node2"`.

This ensures that all nodes that are also load balancers check in with the Chef Server--similar to how we are targeting only the web server nodes in the recipe.

Slide 28

# EXERCISE



## Dynamic Web Load Balancer

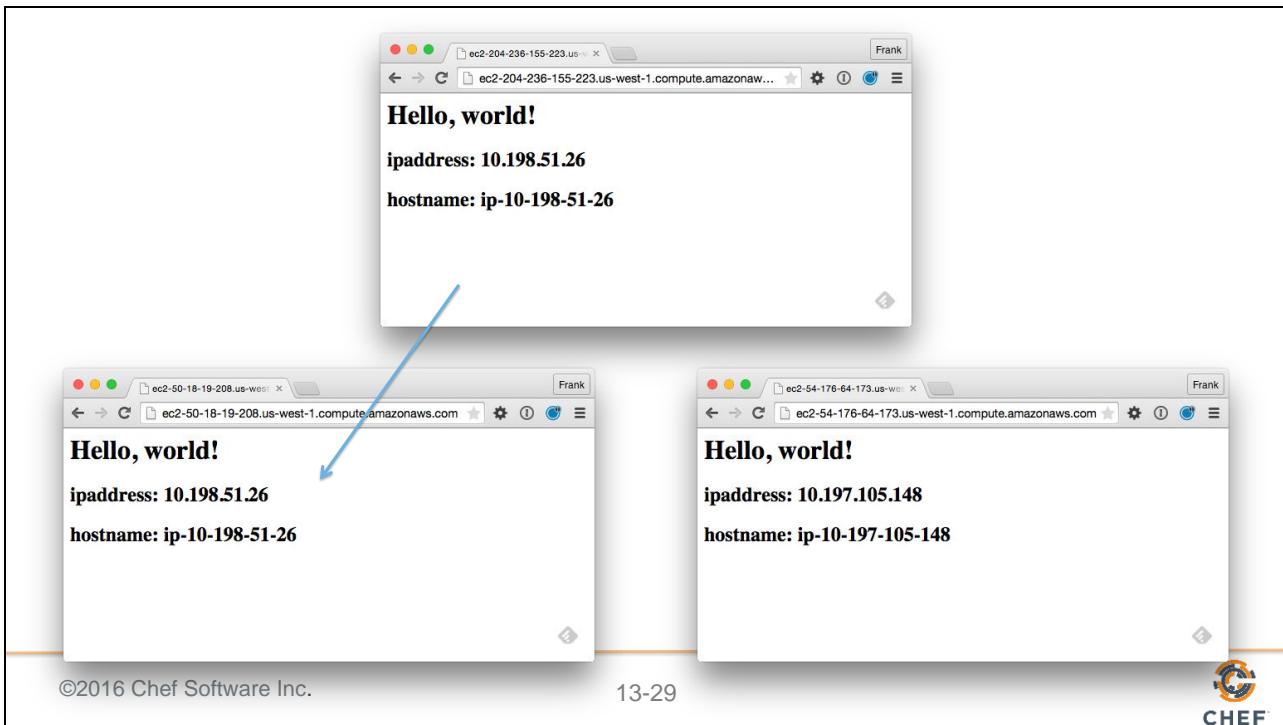
*Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!*

### Objective:

- ✓ Update the myhaproxy cookbook to dynamically use nodes with the web role
- ✓ Update the major version of the myhaproxy cookbook
- ✓ Upload the Cookbook
- ✓ Run chef-client on the load balancer node
- ❑ Verify the load balancer node relays requests to both web nodes

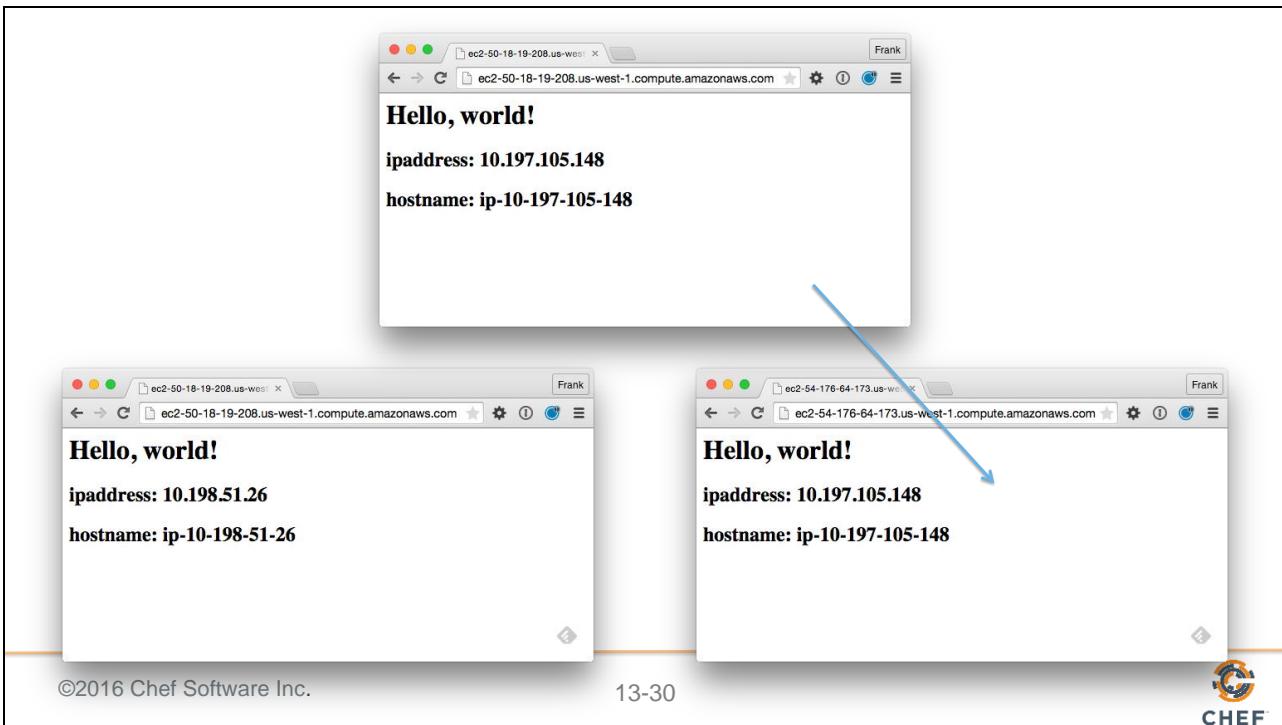
Now the last step is to verify that the load balancer still delivers traffic to the two nodes.

## Slide 29



Nothing should change externally. You may see some differences in the logs as the `load_balancer` configuration file might change the order of the two entries but the end results is that our load balancer node is still delivering traffic to our two web server nodes.

## Slide 30



Slide 31

# EXERCISE



## Dynamic Web Load Balancer

*Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!*

### Objective:

- ✓ Update the myhaproxy cookbook to dynamically use nodes with the web role
- ✓ Update the major version of the myhaproxy cookbook
- ✓ Upload the Cookbook
- ✓ Run chef-client on the load balancer node
- ✓ Verify the load balancer node relays requests to both web nodes

The load balancer performs the same function as before: sending traffic to the two nodes.

Slide 32

# DISCUSSION



## Discussion

What happens when new web nodes are added to the organization?  
Removed?

What happens if you were to terminate a web node instance without  
removing it from the Chef Server?

Answer these questions.

"Terminate" here means to turn off the machine or have the cloud provider disable the machine so that it is no longer online and network addressable.

Slide 33

# DISCUSSION



## Q&A

What questions can we help you answer?

Slide 34



## 14: Environments

# Environments

Using Environments to Reflect Organization Patterns and Workflow

## Slide 2

# Objectives

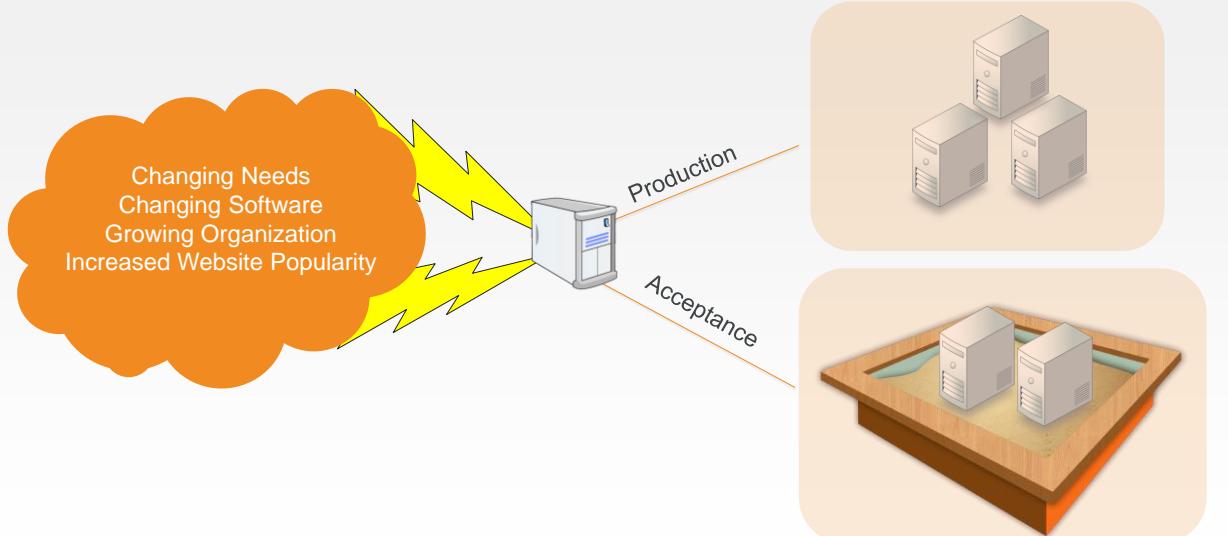
After completing this module, you should be able to

- Create a production and acceptance environment
- Deploy a node to an environment
- Update a search query to be more exact

In this section, you will learn how to create an environment, deploy a node to an environment, and update a search query to be more exact.

Slide 3

## Keeping Your Infrastructure Current



©2016 Chef Software Inc.

14-3



So, we have updated our load balancer's myhaproxy cookbook to dynamically search for and update nodes. Everything is as it should be. But our system is like a living, breathing thing that must grow and be updated to fit our changing needs. We need to find a way to update and test new tools, features and settings without impacting our current production system.

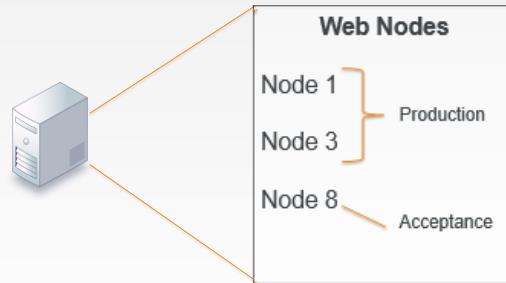
Of course, we have local testing tools like Test Kitchen to help us verify that our individual cookbooks work before we upload them to the Chef Server. But, that is not always enough. We may want to build, test, and release new features to our cookbooks but we do not immediately want all of our nodes to immediately use them. For example, what if we had a requirement to update our apache cookbook with a new front page for our application? The release date of our new service with the sign up page does not go live for a week. So, we want to build, test, and upload that cookbook to the Chef Server without actually applying the cookbook until the release date. How would we accomplish that?

This is where environments are useful.

## Slide 4

# Environments

Environments can define different functions of nodes that live on the same system.

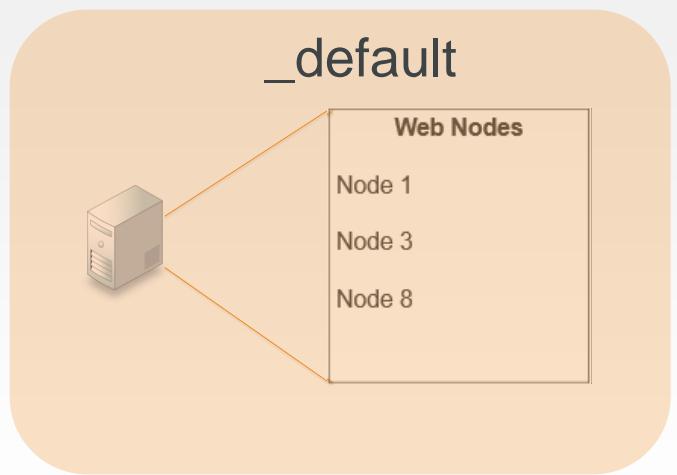


You likely are familiar with the concept of environments. An environment can best be defined as a logical separation of nodes that most often describe the life-cycle of an application. Each environment signifies different behaviors and policies to which a node adheres for a given application or platform.

For example, environments can be separated into 'acceptance' and 'production'. "Acceptance" would be where we may make allowances for constant change and updates and for applications to be deployed with each release. "Production" might be where we lock down our infrastructure and policies. Production would be what the outside world sees, and would remain unaffected by changes and upgrades until you specifically release them. Chef also has a concept of an environment. A Chef environment allows us to define a list of policies that we will allow by defining a cookbook.

# Environments

Every organization or infrastructure starts with the \_default environment.



Chef also has a concept of an environment. Chef uses environments to map an organization's real-life workflow to what can be configured and managed using the Chef server.

Every organization begins with a single environment called the \_default (underscore default) environment, which cannot be modified or deleted.

Therefore, you must create custom environments to define your organization's workflow.

Slide 6

# EXERCISE

## Production



*Let's create a reliable environment for our nodes.*

**Objective:**

- Create a Production Environment
- Add a web node to Production
- Add a load balancer node to Production

First, we need to create a Production environment. This is where we lock down our infrastructure and policies to a specific version of the myhaproxy cookbook.

## Viewing the Help of the environment Subcommand



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

** ENVIRONMENT COMMANDS **

knife environment compare [ENVIRONMENT...] (options)
knife environment create ENVIRONMENT (options)
knife environment delete ENVIRONMENT (options)
knife environment edit ENVIRONMENT (options)
knife environment from file FILE [FILE...] (options)
knife environment list (options)
knife environment show ENVIRONMENT (options)
```

Because we still are communicating with the Chef server, let's ask Chef for help regarding available environment commands.

So change into chef-repo and then run 'knife environment --help'.

Slide 8

## Viewing the List of Environments



```
$ knife environment list
```

```
_default
```

Remember, we use 'list' to view existing environments.

As previously stated, we see the `_default` environment has already been created.

Slide 9

## Viewing the Details of an Environment



```
$ knife environment show _default
```

```
chef_type:           environment
cookbook_versions:
default_attributes:
description:        The default Chef environment
json_class:         Chef::Environment
name:               _default
override_attributes:
```

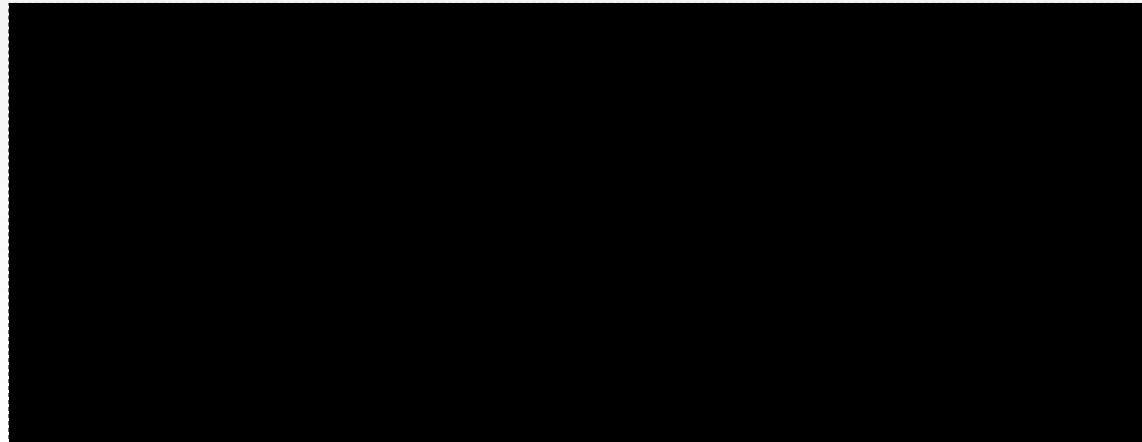
Let's see how this environment looks.

Slide 10

## Creating an Environments Directory



```
$ mkdir environments
```



First, we need to make a new environments directory. (Be sure you are still in the chef-repo before you do this.)

## Defining a Production Environment

```
~/chef-repo/environments/production.rb

name 'production'
description 'Where we run production code'

cookbook 'apache', '= 0.2.1'
cookbook 'myhaproxy', '= 1.0.0'
```

Then we need to create a production.rb file. Like in the roles.rb files, we must provide a name and description. Additionally, we need to define cookbook restrictions to lock down specific versions of both the apache and myhaproxy cookbooks.

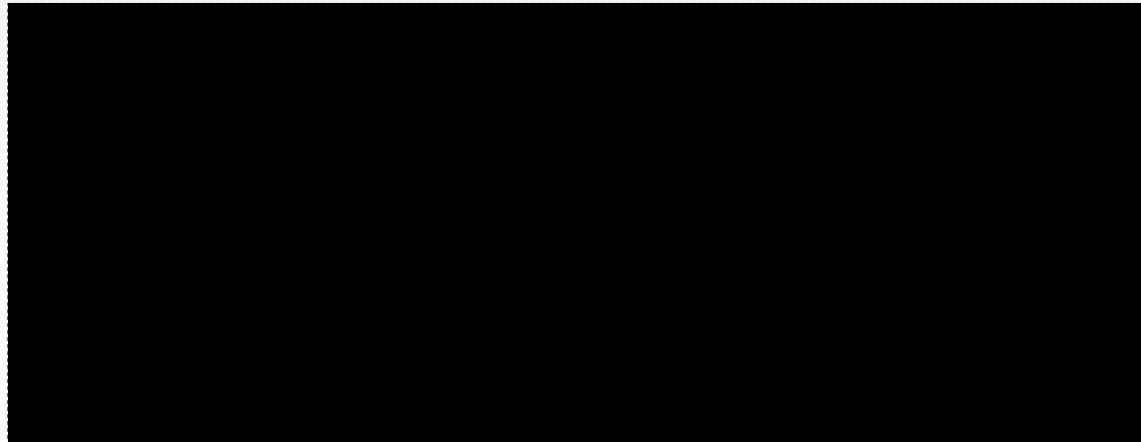
By adding this information to production.rb, we are telling our nodes to use these specific versions of these specific cookbooks. Obviously, what this means is that as we work on newer versions of these cookbooks, we won't break anything in the production let's save it and upload it.

Slide 12

## Uploading the Production Environment



```
$ knife environment from file production.rb
```



Using the knife environment command, let's upload the production.rb file. This should be familiar because it is just like the command we used to upload roles.

Slide 13

## Viewing the List of Environments



```
$ knife environment list
```

```
_default  
production
```

Okay, let's use our list command to make sure the file uploaded correctly.

## Viewing the Details of an Environment



```
$ knife environment show production
```

```
chef_type:           environment
cookbook_versions:
  apache:      = 0.2.1
  myhaproxy:   = 1.0.0
default_attributes:
description:        Where we run production code
json_class:         Chef::Environment
name:               production
override_attributes:
```

If we use the knife environment show command, we can see how the production.rb file looks.

Note the cookbook versions that we set are shown here.

Slide 15

# EXERCISE

## Production



*Let's create a reliable environment for our nodes.*

**Objective:**

- ✓ Create a Production Environment
- ❑ Add a web node to Production
- ❑ Add a load balancer node to Production

Now that we have the environment defined it is time to move one of our web nodes into that environment.

## Searching for All Nodes



```
$ knife search node "*:*"
```

```
3 items found
```

```
Node Name:    node1
```

```
Environment: _default
```

```
FQDN:          ip-172-31-8-68.ec2.internal
```

```
IP:            54.175.46.24
```

```
Run List:      role[web]
```

```
Roles:         web
```

```
Recipes:       apache, apache::default, apache::server
```

```
Platform:     centos 6.7
```

```
Tags:
```

If we search our nodes, we see that all three nodes have been set to the `_default` environment. How do we change this?

## Viewing the Help of the node Subcommand



```
$ 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)
```

Now, we need to set the environments for our nodes. Let's ask Chef for help on that as well.

Let's use the knife node environment set command.

## Viewing the Help of the environment Subcommand



```
$ knife node environment set --help
```

```
knife node environment set NODE ENVIRONMENT
  -s, --server-url URL          Chef Server URL
      --chef-zero-host HOST       Host to start chef-zero on
      --chef-zero-port PORT      Port (or port range) to start chef-zero on.
Port ranges like 1000,1010 or 8889-9999 will try all given ports until one works.
  -k, --key KEY                 API Client Key
      --[no-]color               Use colored output, defaults to false on
Windows, true otherwise
  -c, --config CONFIG           The configuration file to use
      --defaults                Accept default values for all questions
  -d, --disable-editing         Do not open EDITOR, just accept the data as is
  -e, --editor EDITOR          Set the editor to use for interactive commands
```

But how does that command work, exactly?

It looks like we just add the environment name at the end of the command to set that environment on a node.

Slide 19

## Setting the Node's Environment to Production



```
$ knife node environment set node1 production
```

```
node1:  
  chef_environment: production
```

So, let's do that for node1.

The results don't really tell us much, so let's take a look at node1.

## Viewing the Details about the Node



```
$ knife node show node1
```

```
Node Name:    node1
Environment:  production
FQDN:        ip-172-31-8-68.ec2.internal
IP:          54.175.46.24
Run List:    role[web]
Roles:       web
Recipes:     apache, apache::default, apache::server
Platform:   centos 6.7
Tags:
```

Using knife node show, we can see node1's attributes. Note that it has indeed been set to the production environment.

Slide 21

# EXERCISE

## Production



*Let's create a reliable environment for our nodes.*

**Objective:**

- ✓ Create a Production Environment
- ✓ Add a web node to Production
- ❑ Add a load balancer node to Production

Now we need to move the Load Balancer node into the same environment.

## Setting the Node's Environment to Production



```
$ knife node environment set node2 production
```

```
node2:  
  chef_environment: production
```

## Viewing the Details about the Node



```
$ knife node show node2
```

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

And, it looks like node2 was successfully set to the production environment.

Slide 24

# EXERCISE

## Production



*Let's create a reliable environment for our nodes.*

**Objective:**

- ✓ Create a Production Environment
- ✓ Add a web node to Production
- ✓ Add a load balancer node to Production

We have created a Production environment and added a web node and the load balancer node to it.

Slide 25

# EXERCISE

## Acceptance

**Objective:**

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- ✓ Move node3 into the acceptance environment
- Run chef-client on all the nodes

Now, it is time for you to create the environment we can use to change and update the cookbooks without affecting our production environment. This will be a sandbox environment where we will have no cookbook restrictions. This will be the environment where we can make sure that cookbooks all work together as we continue to update them and release new versions.

Create this new environment add the remaining web node to it. Then make sure that all the nodes converge successfully.

## Defining the Acceptance Environment

```
~/chef-repo/environments/acceptance.rb
```

```
name 'acceptance'  
description 'Where code and apps are tested'  
# No Cookbook Restrictions
```

First, let's create a new rb file in our chef-repo/environments directory. Let's name it acceptance.

In the Acceptance environment, we don't want to lock-down the cookbook versions, so we are not going to place restrictions on the cookbooks.

Slide 27

## Uploading the Environment



```
$ knife environment from file acceptance.rb
```

```
Updated Environment acceptance
```

Let's upload that .rb file to the Chef server.

Slide 28

## Viewing the List of Environments



```
$ knife environment list
```

```
_default  
production  
acceptance
```

And let's make sure that this environment file was added properly.

## Viewing the Details of the Environment



```
$ knife environment show acceptance
```

```
chef_type:           environment
cookbook_versions:
default_attributes:
description:        Where code and applications are tested
json_class:         Chef::Environment
name:               acceptance
override_attributes:
```

And last, but not least, let's ask the Chef Server to show us the acceptance environment.

Slide 30

# EXERCISE

## Acceptance

**Objective:**

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- ❑ Move node3 into the acceptance environment
- ❑ Run chef-client on all the nodes

Slide 31

## Setting the Node to Acceptance Environment



```
$ knife node environment set node3 acceptance
```

```
node3:  
  chef_environment: acceptance
```

Okay, let's set node3 to the acceptance environment.

## Viewing Details about the Node



```
$ knife node show node3
```

```
Node Name:    node3
Environment:  acceptance
FQDN:        ip-172-31-0-127.ec2.internal
IP:          54.210.86.164
Run List:    role[web]
Roles:       web
Recipes:     apache, apache::default, apache::server
Platform:   centos 6.6
Tags:
```

And confirm that it has been set properly.

Slide 33

# EXERCISE

## Acceptance

**Objective:**

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- ✓ Move node3 into the acceptance environment
- ❑ Run chef-client on all the nodes

## Converging All the Nodes



```
$ knife ssh "*:*" -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
ec2-54-210-192-12.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"]
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
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
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com   - build-essential
```

Using the knife ssh let's run chef client on all the nodes.

Slide 35

# EXERCISE

## Acceptance

**Objective:**

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- ✓ Move node3 into the acceptance environment
- ✓ Run chef-client on all the nodes

The Acceptance environment is setup and node3 is now its only member.

Slide 36

# EXERCISE

## Separating Search



*The search criteria that we defined in the Load Balancer does not consider where the nodes are located. That needs to change.*

**Objective:**

- Update the load balancer's search criteria to respect environments
- Update the load balancer cookbook's version number and upload it
- Update the Production environment to allow new cookbook version
- Converge all the nodes

Now that we have created our two environments and set each node to a specific environment, we need to update the search that we use to find the web nodes to consider the environment to ensure that the load balancer only communicates with the nodes that share its environment.

## Viewing the Existing Search Criteria



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
#  
# Cookbook Name:: myhaproxy  
# Recipe:: default  
#  
# Copyright (c) 2016 The Authors, All Rights Reserved.  
  
all_web_nodes = search('node', 'role:web')  
  
members = []  
  
#...
```

Looking at the existing recipe in the load balancer's myhaproxy cookbook, we can review the original search syntax. If we want to search by environments, what would we need to add here?

## Updating the Search to Consider Environment



~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
#  
# Cookbook Name:: myhaproxy  
# Recipe:: default  
#  
# Copyright (c) 2016 The Authors, All Rights Reserved.  
  
all_web_nodes = search('node', "role:web AND chef_environment:#{node.chef_environment}")  
  
members = []  
  
#...
```

Search the Chef Server for all node objects that have the role equal to 'web' and also share the same environment as the current node applying this recipe. The nodes currently applying this recipe are the nodes with the role set to `load_balancer`.

Now that we've made our changes, let's save this file.

Slide 39

# EXERCISE

## Separating Search



*The search criteria that we defined in the Load Balancer does not consider where the nodes are located. That needs to change.*

**Objective:**

- ✓ Update the load balancer's search criteria to respect environments
- Update the load balancer cookbook's version number and upload it
- Update the Production environment to allow new cookbook version
- Converge all the nodes

The Load Balancer's search criteria has been updated. It is time to update the version number and upload the cookbook to the Chef Server.

## Updating the Version of the Cookbook



~/chef-repo/cookbooks/myhaproxy/metadata.rb

```
name          'myhaproxy'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures myhaproxy'  
long_description 'Installs/Configures myhaproxy'  
version        '1.0.1'  
  
depends 'haproxy', '~> 1.6.6'
```

Before we upload the new myhaproxy cookbook to the server, we probably want to update the version number. What type of change have we made here?

Answer: Patch

Because we are performing a patch, let's set the version number to 1.0.1.

Slide 41

## Installing Cookbook Dependencies



```
$ cd cookbooks/myhaproxy  
$ berks install
```

```
Resolving cookbook dependencies...  
Fetching 'myhaproxy' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Using build-essential (2.2.3)  
Installing haproxy (1.6.6)  
Using cpu (0.2.0)  
Using myhaproxy (1.0.1) from source at .
```

We are going to need to use Berks to upload this cookbook because it has dependencies. So first we need to cd into the cookbook. Then run 'berks install'.

Slide 42

## Uploading the Cookbook to Chef Server



```
$ berks upload
```

```
Skipping build-essential (2.2.3) (frozen)
Skipping cpu (0.2.0) (frozen)
Skipping haproxy (1.6.6) (frozen)
Uploaded myhaproxy (1.0.1) to:
  'https://api.opscode.com:443/organizations/vogue'
```

And finally berks upload.

Slide 43

# EXERCISE

## Separating Search



*The search criteria that we defined in the Load Balancer does not consider where the nodes are located. That needs to change.*

**Objective:**

- ✓ Update the load balancer's search criteria to respect environments
- ✓ Update the load balancer cookbook's version number and upload it
- ❑ Update the Production environment to allow new cookbook version
- ❑ Converge all the nodes

The Load Balancer's search criteria has been updated. It is time to update the version number and upload the cookbook to the Chef Server.

Slide 44

# CONCEPT



## A Brief Recap

We restricted the production environment to specific cookbook version and then we updated the Load Balancer cookbook's version number.

The environment needs to be updated before it will use the new cookbook that we have defined.

Before we run 'chef-client' to bring everything up to date, let's think about what we've done. First, in the production environment, we restricted our cookbooks to a specific version. Second, we created an acceptance environment with no cookbook restrictions. Third, we set specific nodes to each of these environments. Fourth, we updated the myhaproxy default.rb to include environment search criteria. And lastly, we changed the version number in the myhaproxy metadata.rb file.

Because we have updated the version number we need to update the cookbook version restrictions in the Production environment. Otherwise it will continue to use the previous version of the cookbook.

## Updating the Version Constraints for the Environment

```
~/chef-repo/environments/production.rb
```

```
name 'production'
description 'Where we run production code'

cookbook 'apache', '= 0.2.1'
cookbook 'myhaproxy', '= 1.0.1'
```

So let's go back into our production.rb and update it to include the new version number.

Slide 46

## Uploading the Environment



```
$ cd ~/chef-repo  
$ knife environment from file production.rb
```

```
Updated Environment production
```

Change to ~/chef-repo and then run 'knife environment from file production.rb'.

## Verifying the Version Number



```
$ knife environment show production
```

```
chef_type:           environment
cookbook_versions:
  apache:      = 0.2.1
  myhaproxy:   = 1.0.1
default_attributes:
description:        Where we run production code
json_class:         Chef::Environment
name:               production
override_attributes:
```

And let's make sure that the production.rb on Chef server has the correct version of myhaproxy designated.

Slide 48

# EXERCISE

## Separating Search



*The search criteria that we defined in the Load Balancer does not consider where the nodes are located. That needs to change.*

**Objective:**

- ✓ Update the load balancer's search criteria to respect environments
- ✓ Update the load balancer cookbook's version number and upload it
- ✓ Update the Production environment to allow new cookbook version
- Converge all the nodes

The Production environment has now been updated and uploaded. Finally it is time to update all the nodes.

Slide 49

## Converging all the Nodes



```
$ knife ssh "*:*" -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
ec2-54-210-192-12.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"]
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
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
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com   - build-essential
```

And use 'sudo chef-client' to converge all nodes.

Slide 50

# EXERCISE

## Separating Search



*The search criteria that we defined in the Load Balancer does not consider where the nodes are located. That needs to change.*

**Objective:**

- ✓ Update the load balancer's search criteria to respect environments
- ✓ Update the load balancer cookbook's version number and upload it
- ✓ Update the Production environment to allow new cookbook version
- ✓ Converge all the nodes

While converging the nodes you should have seen the Load Balancer node's configuration updated only to define a single node instead of the two that it previously defined.

Slide 51

# DISCUSSION



## Discussion

What is the benefit of constraining cookbooks to a particular environment?

What are the benefits of **not** constraining cookbooks to a particular environment?

Answer these questions.

Slide 52

# DISCUSSION



## Q&A

What questions can we help you answer?

Slide 53



## 15: Further Resources

### Further Resources

Other Places to Talk About, Practice, and Learn Chef

## Slide 2

# MOTIVATION



## Going Forward

There are many Chef resources available to you outside this class. During this module we will talk about just a few of those resources.

But...remember what we said at the beginning of this class:

*The best way to learn Chef is to use Chef*

Slide 3

# MOTIVATION



## Practice Chef

First, let's talk about stuff you can read to help you learn Chef.

Slide 4

# REFERENCE



[learn.chef.io](https://learn.chef.io)

Interactive learning for those new to Chef.

---

©2016 Chef Software Inc.

15-4



Another great place to practice Chef is our awesome Learn Chef site. These interactive modules provide you with an opportunity to run through exercises similar to those we did in this class, and it is updated when new Chef features are introduced. This is one of the most robust self-guided tutorial sites out there.

Slide 5

# REFERENCE



## Beyond Essentials

- What happens during a knife bootstrap?
- What happens during a chef-client run?
- What is the security model used by chef-client
- Explains Attribute Precedence

<https://learn.chef.io/skills>

There is a special section of Learn Chef called the Skills Library where you will find some additional content that will answer some of the questions that you may have after completing this content. It is included there on Learn Chef to provide you with a resource that you can return back to again-and-again after this training.

Slide 6

# REFERENCE



## Community Resources

Example Cookbooks, Articles, Podcasts, and More

<https://github.com/obazoud/awesome-chef>

---

©2016 Chef Software Inc.

15-6



This project contains a living repository of resources curated by the community that showcase some of the better cookbooks to review and use, articles that cover basic and advanced topics, links to podcasts and videos, etc.

Slide 7

# MOTIVATION



## Resources You Can Read

A lot of people in the Chef community have written about Chef.

Here are just a few of those resources.

## Slide 8

# REFERENCE

[docs.chef.io](http://docs.chef.io)



Docs are available to you, 24 hours a day, 7 days a week.

Any question you have, you probably will find the answer for on our Docs site.

Remember, how often we referred to the Docs site throughout this workshop. That wasn't by accident. We wanted you to become comfortable with using our Docs site to resolve issues and learn about the many Chef tools out there.

Docs are there, available to you, 24 hours a day, 7 days a week. Any question you have, you probably will find the answer on our Docs site.

Slide 9

The book cover for "Learning Chef" features a large title "REFERENCE" in white, outlined letters at the top. Below it, the subtitle "Learning Chef" is written in orange. A small illustration of three books is in the top right corner. The main cover area is blue with a black and white illustration of a bird perched on a branch. The title "Learning Chef" is printed in white on the bird's chest. Below the bird, the subtitle "A GUIDE TO CONFIGURATION MANAGEMENT AND AUTOMATION" is written in smaller white text. At the bottom of the cover, the authors' names "Mischa Taylor & Seth Vargo" are listed. The O'Reilly logo is in the top left corner of the cover image.

©2016 Chef Software Inc. 15-9

 CHEF

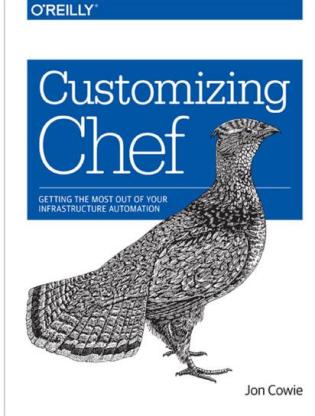
Some people who have used Chef for years have written some excellent books about Chef. Check out Mischa Taylor's and Seth Vargo's *Guide to Configuration Management and Automation*. You can find it on O'Reilly. It's a great book.

Slide 10

# REFERENCE

## Customizing Chef

Getting the Most Out of Your Infrastructure Automation



©2016 Chef Software Inc. 15-10



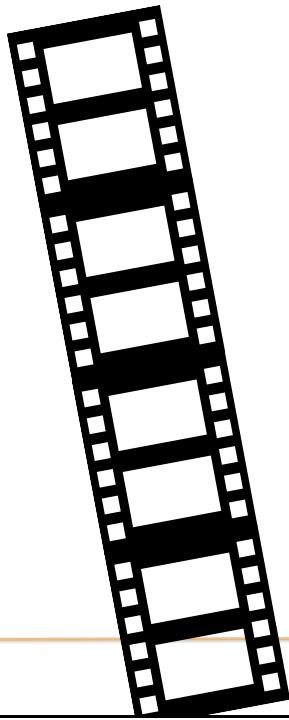
Additionally, you may want to read Jon Cowie's *Getting the Most Out of Your Infrastructure Automation*. It's also available on O'Reilly.

Slide 11

## YouTube Channel

- ChefConf Talks
- Training Videos

<https://www.youtube.com/user/getchef/playlists>



We have uploaded a number of videos to the Chef YouTube channel, including training videos and talks from past Chef conferences.

Slide 12

[foodfightshow.org](http://foodfightshow.org)



The Podcast where DevOps chefs do battle

©2016 Chef Software Inc.

15-12



Food Fight is a bi-weekly podcast for the Chef community. We bring together the smartest people in the Chef community and the broader DevOps world to discuss the thorniest issues in system administration.

Slide 13

## Chef Developers' IRC Meeting

<https://github.com/chef/chef-community-irc-meetings>



Join members of the Chef Community in a weekly meeting for Chef Developers where we'll discuss the future of the Chef project and other things pertinent to the community. The agenda and schedule can be found at this link.

Slide 14

## Chef Product Feedback Forum

Create your product feature ideas for the Chef engineering teams. As a registered user, you'll be able to vote on your features and the features proposed by others...



<https://feedback.chef.io>

---

©2016 Chef Software Inc.

15-14



Help us build the best product. If you have an idea we would love to hear more about it. Or come and vote on other features proposed by others.

Slide 15



©2016 Chef Software Inc.

15-15



ChefConf is a gathering of hundreds of Chef community members. We get together to learn about the latest and greatest in the industry (both the hows and the whys), as well as exchange ideas, brainstorm solutions, and give hugs, which has become the calling card of the DevOps community, and the Chef community in particular.

ChefConf 2016 will be held in Austin, Texas during July.

Slide 16

## Chef Community Summit - 2016 Dates to be Announced

<https://www.chef.io/summit/>



*Seattle Summit*

OCTOBER 14-15, 2015



*London Summit*

NOVEMBER 3-4, 2015



©2016 Chef Software Inc.

15-16



The Chef Community will gather for two days of open space sessions and brainstorm on Chef best practices.

The Chef Community Summit is a facilitated Open Space event. The participants of the summit propose topics, organize an agenda, and discuss and work on the ideas that are most important to the community.

Dates for the 2016 Summit will be announced in the spring of 2016.

Slide 17

