



CHEFTM

Chef Training Services

DevOps Foundation

Participant Guide

DevOps Foundation

Introduction

Course v1.0.0

©2017 Chef Software Inc.

1-1



This 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.

Instructor Note: Be sure to read Appendix Z for training lab set up notes and additional instructor notes



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.

Introduce Ourselves

Name

Current job role

Previous job roles/background

Experience with Chef and/or config management

Favorite Text Editor

Location

Instructor Note: Often times with groups I opt to have people introduce themselves to each other in a more casual fashion. Asking the learners to face one another and ask each other these questions. During that time you can walk around and insert yourself into the introductions to help odd numbers and learn more about the learners.

Chef



Chef can automate how you build, deploy, and manage your infrastructure.

Chef can integrate with cloud-based platforms such as Azure and Amazon Elastic Compute Cloud to automatically provision and configure new machines.

Chef can automate how you build, deploy, and manage your infrastructure. Your infrastructure becomes as versionable, testable, and repeatable as application code enabling you to automate the process of configuring, deploying and scaling servers and applications



Chef

Chef is a large set of tools that are able to be used on multiple platforms and in numerous configurations.

Learning Chef is like learning a language. You will learn the basic concepts very fast but it will take practice until you become comfortable.

A great way to learn Chef is to use Chef

Chef is a large set of tools that are able to be used on multiple platforms and in numerous configurations. We will have time to only explore some of its most fundamental pieces.

Learning Chef is like learning a language. You will learn the basic concepts very fast but it will take practice until you become comfortable.

Agenda: Day 1

- What is DevOps?
- Using Chef Resources
- Building Chef Cookbooks
- Git and Version control
- Collecting details about the system
- Managing data with templates

Agenda: Day 2

- Set up an Apache web server
- Install the ChefDK and sign up for a Managed Chef account
- Communicate with a Chef Server
- Use Attribute Files and Dependencies
- Community cookbooks
- Manage multiple nodes

Agenda: Day 3

- Roles
- Use Search within a recipe
- Set up chef-client to run as a service/task
- Data Bags
- Environments
- Further Resources

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



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



EXERCISE

Pre-built Workstation

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

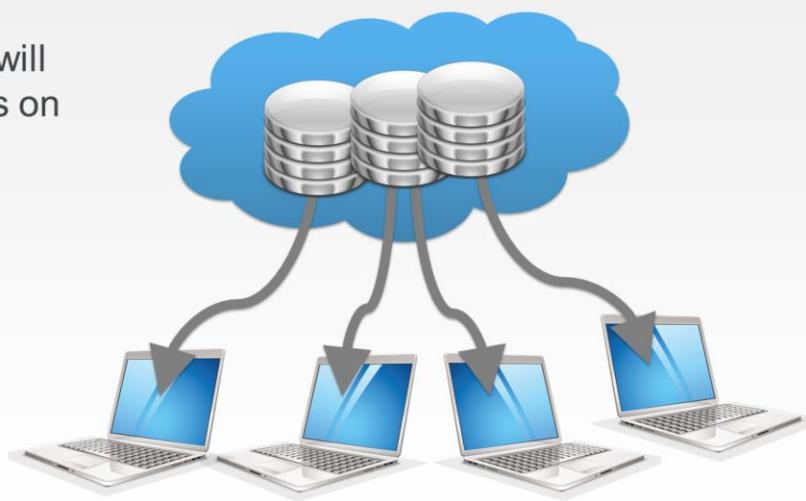
Objective:

- Login to the Remote Workstation

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.

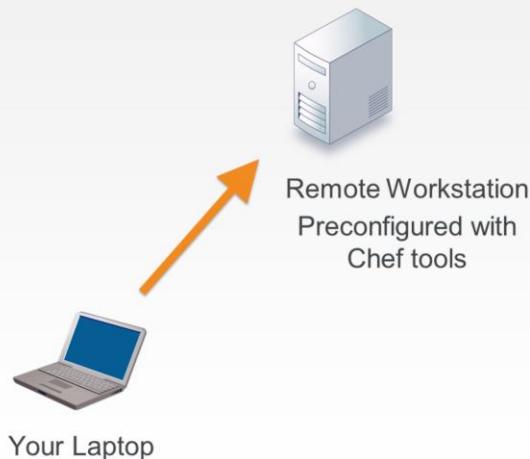
Setting up Your Workstation

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



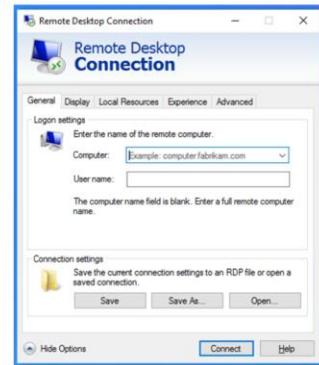
Near the beginning of Day 2 we have set aside time to install the necessary tools on your local computer. If you already have the tools we will ensure that they are working correctly and troubleshoot any issues to ensure you have a smooth experience for when you leave this training.

Chef Lab System Architecture



Logging in to the Workstation

Use the **address**, **user name**, and **password** to connect to the remote workstation.



We will provide you with the address, username and password of the workstation. With that information you will need to use a Remote Desktop Connection tool that you have installed to connect that workstation.

Instructor Note: You should assign the participants their Day 1 virtual workstations (AMIs) at this time. The login credentials and password for the virtual workstations is
user: Administrator
password: Cod3Can!



EXERCISE

Pre-built Workstation

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

Objective:

- ✓ Login to the Remote Workstation



CHEFTM

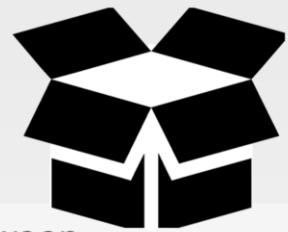
©2017 Chef Software Inc.

What is DevOps?

Blending Skill Sets to Deploy more Quickly

CONCEPT

What is DevOps?



No longer is there a boundary between the group that writes and develops code and applications from group that deploys, manages and supports:

- Collaboration between developers and operations teams

What is DevOps, not just as a coin-term but why is it valuable? The term DevOps means a blending of roles that were previously separate. It's not just a buzzword that a boss or innovator uses to persuade people to develop or move faster toward deployment. It's a change in the cultural norm of an organization. Just as the Agile movement set smaller goals to deploy more quickly, DevOps aims at the same task, yet this time it's a blending of roles and skill sets. Specifically, it attempts to break down the barriers between the roles that have been traditionally assigned to developers and operations.



DevOps Understanding

*Let's better understand **why** the DevOps strategy is prevalent.*

Objective:

- Collaboration; Development & Operations
- Key Components to DevOps
- Stages of DevOps

So let's look at a few key points surrounding DevOps



Benefits of DevOps Strategy

- Empowerment of developers to launch and destroy VMs for testing
- Separation from production environment to sandbox new feature/application
- Reduction of time needed for software development
- Operational understanding of how the application/code will affect the ecosystem

In an older modality, developers would work toward a product goal, some feature of code that needed to be created based on a manager's or client's needs. In DevOps, the goals haven't changed, but the manner in which they are reached has.

Provisioning a new machine takes time in a legacy organization, managerial approval, cross party monitoring, and a timeframe for the developer to test their finished code. This is not a well prescribed method for a quick, effective solution for developers to see the end result of their new feature. In addition, through refactoring of code, the existing machine functions just fine refinement after refinement. Yet a new machine, with a fresh OS may not fair the same. This is due to pre-existing configurations from previous iterations of the feature.



DevOps Understanding

What makes DevOps work?

Objective:

- ✓ Collaboration; Development & Operations
- ❑ Key Components to DevOps
- ❑ Stages of DevOps

Key Components to DevOps



- Automation
 - ↳ QA (quality assurance) and testing automation
 - ↳ Infrastructure automation and management
- Monitoring: Staying on top of all changes being pushed into the production environment
- Increased Deployment Frequency
- Increased Deployment Reliability
- Increased Business Value
 - ↳ Getting the product to the customer faster with better reliability and uptime

A DevOps cultural shift within an organization happens most often from a top down approach. Its key aspect is to remove the boundaries between the group that writes and develops code and applications from the group that deploys, manages and supports the same code and applications during uptime. It doesn't make sense to have the 2 groups separated and not collaborating. A third group, quality assurance (QA), was also brought into the mix once the DevOps movement started taking hold.

Why should these departments be separate instead of having them be on the same team? Operations needs to know how code and applications are developed to help them manage issues as they arise. Developers need to know about integrations that are necessary in the operations context. Code or Applications can not be pushed into production unless QA signs off, so developers need to know the criteria that need to be met when creating new features or functionality. All three groups are bound together during the lifecycle of code.



DevOps Understanding

What is the DevOps workflow?

Objective:

- ✓ Collaboration; Development & Operations
- ✓ Key Components to DevOps
- ❑ Stages of DevOps



Stages of DevOps

- Code: development, review, management, merge
- Build: continuous integration, build status
- Test: continuous testing, automated testing
- Package: artifact repository, pre-deployment staging



CONCEPT Stages of DevOps (continued)

- Release: change management, approval, automated release
- Configure: Infrastructure as Code, i.e. Chef
- Monitor: Performance and end-user experience



DevOps Understanding

What is the DevOps workflow?

Objective:

- ✓ Collaboration; Development & Operations
- ✓ Key Components to DevOps
- ✓ Stages of DevOps



Discussion

Is your team utilizing or moving towards practicing a DevOps approach?

If so, what are some of the challenges that this has presented?



CHEFTM

©2017 Chef Software Inc.

Chef Resources

Chef's Fundamental Building Blocks



Objectives

After completing this module, you should be able to:

- Use Chef to set the policy on your workstation
- Use the chef-client command
- Create a basic Chef recipe file
- Define Chef Resources

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

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.

Instructor Note: This may sound unusual to ask people to read the documentation site but it is important that they learn to refer to the documentation. This page is an important reference page.

Example: powershell_script

```
powershell_script 'Install IIS' do
  code 'Add-WindowsFeature Web-Server'
  action :run
end
```

The powershell_script named 'Install IIS' is run with the code 'Add-WindowsFeature Web-Server'.

https://docs.chef.io/resource_powershell_script.html

Here is an example of the powershell_script resource. The powershell_script named 'Install IIS' is run with the code 'Add-WindowsFeature Web-Server'

Example: service

```
service 'w3svc' do
  action [ :enable, :start ]
end
```

The service named 'w3svc' is enabled (start on reboot) and started.

https://docs.chef.io/resource_service.html

In this example, the service named 'w3svc' is enabled and started.

Service resources are often defined with two actions. The action method can only take one parameter so to provide two actions you need to specify the two actions within an Array.

Example: file

```
file 'C:\inetpub\wwwroot\Default.htm' do
  content 'Hello, world!'
  rights :read, 'Everyone'
end
```

The file 'C:\inetpub\wwwroot\Default.htm' with the content 'Hello, world!' and grants 'read' rights for 'Everyone'.

https://docs.chef.io/resource_file.html

In this example, the file named 'C\inetpub\wwwroot\Default.htm' with the content 'Hello, world!' and has allowed Everyone rights to read the file.

The default action for the file resource is to create the file.

Example: file

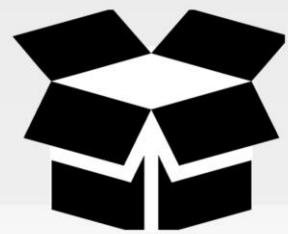
```
file 'C:\PHP\php.ini' do
  action :delete
end
```

The file name 'c:\PHP\php.ini' is deleted.

https://docs.chef.io/resource_file.html

In this example, the file named 'C:\PHP\php.ini' is deleted.

Instructor Note: A resource's default action is based on the principle of least surprise. So they are often creative actions towards the system. This is why the file resource specified here has the action specified. It is not the default action.



Resource Definition

```
file 'C:\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.

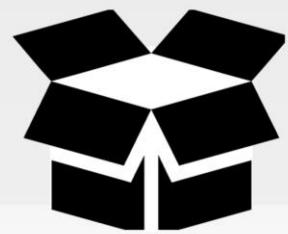


Resource Definition

```
file 'C:\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'. We showed you an example of 'service' earlier.



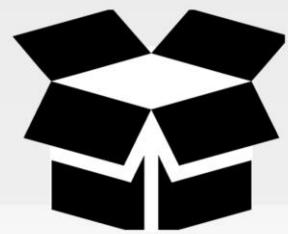
Resource Definition

```
file 'C:\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.



Resource Definition

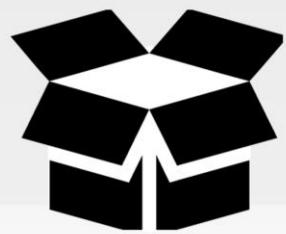
```
file 'C:\hello.txt' do
  content 'Hello, world!'
end
```

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 properties 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.

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



Resource Definition

```
file 'C:\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.



Group Lab: Hello, World?

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

Objective:

- Create a recipe file writes out 'Hello, world!' to a text file
- Apply the recipe to the workstation

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 'C:\hello.txt' with the contents 'Hello, world!'.

GL: Create and Open a Recipe File



```
> dir
```

```
Directory: C:\Users\Administrator

Mode          LastWriteTime    Length Name
----          -----          ----  --
d---          2/16/2017  6:31 PM        .atom
d---          2/16/2017  6:20 PM        .berkshelf
d---          2/16/2017  6:21 PM        .chef
d---          2/16/2017  6:23 PM        .kitchen
d---          2/16/2017  6:22 PM        .VirtualBox
d---          2/16/2017  6:21 PM        chef
d-r--         2/16/2017  6:16 PM        Contacts
d---          2/16/2017  6:44 PM        cookbooks
d-r--         2/16/2017  6:32 PM        Desktop
d-r--         2/16/2017  6:21 PM        Documents
d-r--         2/16/2017  6:16 PM        Downloads
```

GL: Create and Open a Recipe File



```
> atom hello.rb
```

Now that we have seen a few examples of resources let's get to work creating a text file. 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.

GL: Create a Recipe File Named hello.rb

```
~\hello.rb  
  
file 'C:\hello.txt' do  
  content 'Hello, world!'  
end
```

The file named 'C:\hello.txt' is created with the content
'Hello, world!'

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

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

Save the file and return to the command prompt.

Instructor Note: The default action is to create the file.



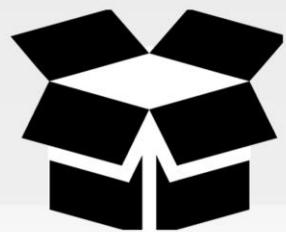
Group Lab: Hello, World?

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

Objective:

- Create a recipe file writes out 'Hello, world!' to a text file
- Apply the recipe to the workstation

Now the file is created with the resource that will create the file with the content we want to see. It is time to apply that recipe to the system.



chef-client

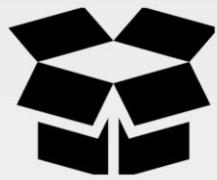
chef-client is an agent that runs locally on every node that is under management by Chef.

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

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.



--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.

GL: Apply a Recipe File



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

```
Converging 1 resources
Recipe: @recipe_files::C:/Users/Administrator/hello.rb
  * file[C:\hello.txt] action create[2017-10-30T19:23:52+00:00] INFO: Processing
    file[C:\hello.txt] action create (@reci
    pe_files::C:/Users/Administrator/hello.rb line 1)
  [2017-10-30T19:23:52+00:00] INFO: file[C:\hello.txt] created file C:\hello.txt
    - create new file C:\hello.txt[2017-10-30T19:23:52+00:00] INFO:
      file[C:\hello.txt] updated file contents C:\hello.txt

    - update content in file C:\hello.txt from none to 4ae7c3
      --- C:\hello.txt      2017-10-30 19:23:52.000000000 +0000
      +++ C:\chef-hello20171030-756-1kj8npo.txt      2017-10-30 19:23:52.000000000
```

Type the specified command to apply the recipe file. You should see that a file named 'hello.txt' was created and the contents updated to include your 'Hello, world!' text.

The output that shows the contents of the file have been modified is being displayed in a format similar to a git diff (<http://stackoverflow.com/questions/2529441/how-to-read-the-output-from-git-diff>).

GL: What Does hello.txt Say?



```
> gc C:\hello.txt
```

```
Hello, world!
```

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



Group Lab: Hello, World?

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

Objective:

- ✓ Create a recipe file writes out 'Hello, world!' to a text file
- ✓ Apply the recipe to the workstation



Discussion

What would happen if you ran the command again?

What would happen if the file were removed?

What happens when I run the command again?

Again, before you run the command -- think about it. What are your expectations now from the last time you ran it? What will the output look like?

And, of course, what would happen if the file was removed?



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 is first tested on the system before it takes action.



Test and Repair

```
file 'C:\hello.txt'
```

Yes

Do Nothing

Is file
'C:\hello.txt'
created and
has the right
content?
(test)

No

Bring resource to
desired state
(repair)

If the file is already created and not modified, then the resource does not need to take action.

If the file is not created, then the resource NEEDS to take action to create the file.
If the file is not in the desire state, then the resource NEEDS to take action to modify the file.



Lab: Goodbye Recipe

*We know how to create a file with Chef,
what about removing it?*

Objective:

- Create a recipe named 'goodbye.rb' that removes 'C:\hello.txt'
- Apply the recipe to the workstation

We wrote and applied a recipe that creates a file on the workstation. Now, let's create a recipe that removes that same file. We will define a new recipe and then apply it to the workstation.

Lab: Adding a file Resource to Delete a File

```
~\goodbye.rb
```

```
file 'C:\hello.txt' do
  action :delete
end
```

The file resources default action is to create the file. So if we want to remove a file we need to explicitly define the action.

The following policy will delete the 'C:\hello.txt' file when applied.

Lab: Apply a Recipe File



```
> chef-client --local-mode goodbye.rb
```

```
[2017-10-30T19:31:13+00:00] INFO: file[C:\hello.txt] backed up to  
C:/Users/Administrator\.chef\local-mode-cache\backup\h  
ello.txt.chef-20171030193113.498382  
[2017-10-30T19:31:13+00:00] INFO: file[C:\hello.txt] deleted file at  
C:\hello.txt  
  
      - delete file C:\hello.txt  
[2017-10-30T19:31:13+00:00] INFO: Chef Run complete in 0.059047 seconds  
  
Running handlers:  
[2017-10-30T19:31:13+00:00] INFO: Running report handlers  
Running handlers complete
```

Type the specified command to apply the recipe file. You should see that a file named 'C:\hello.txt' was deleted.

Lab: Test that the File was Deleted



```
> Test-Path C:\hello.txt
```

```
False
```

To test if that file was removed from the file system successfully we can run the following command.



Lab: Goodbye Recipe

*We know how to create a file with Chef,
what about removing it?*

Objective:

- ✓ Create a recipe that removes 'C:\hello.txt'
- ✓ Apply the recipe to the workstation

The recipe has been defined and now it is time to apply it to the workstation.



GL: Disable Limited User Account

Managing files is nice but what about Registry keys?

Objective:

- Create a recipe that disables Limited User Account
- Apply that recipe

Managing files is useful but when managing Windows systems we are often more concerned with managing the keys within the registry.

To help setup our system to be more 'user friendly' we want to disable some of the User Access Control (UAC) features that are initially enabled on a Windows system.

GL: Disable the Limited User Account

```
~\disable-uac.rb
```

```
system_policies = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'

registry_key system_policies do
  values [{{
    :name => 'EnableLUA',
    :type => :dword,
    :data => 0
  }}]
end
```

Here we are defining a variable named 'system_policies'. With Ruby you can define variables instantly whenever you need them. Here we define this variable to store our registry key in case we need to use the same registry key to set more values.

Instructor Note: The lab that follows this group exercise will use the same registry key so the learner will need to use the variable. The use of the variable here also makes the column length smaller so that the font size on the slide can remain at a reasonable size without the content breaking across multiple lines.

GL: Disable the Limited User Account

```
~\disable-uac.rb
```

```
system_policies = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'

registry_key system_policies do
  values [{{
    :name => 'EnableLUA',
    :type => :dword,
    :data => 0
  }}]
end
```

Here we are using a new resource named 'registry_key' that takes the name of a registry key. We then provide to the values attribute the values we want to set/insert in the registry. Here we are setting the EnableLUA key to have a dword value of 0. This will make it so that Windows will no longer notify the user when programs try to make changes to the computer.

See the following documentation for more information:
<https://technet.microsoft.com/en-us/library/ff715520.aspx>.



GL: Disable Limited User Account

Managing files is nice but what about Registry keys?

Objective:

- Create a recipe that disables Limited User Account
- Apply that recipe

The policy you defined in the recipe has now updated the system and you should now be prompted that Limited User Account has been disabled.

GL: Apply a Recipe File



```
> chef-client --local-mode disable-uac.rb
```

```
Converging 1 resources
Recipe: @recipe_files::C:/Users/Administrator/disable-uac.rb
  * registry_key[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System]
action create[2017-10-30T19:41:15+00:00]
] INFO: Processing
registry_key[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System]
action create (@recipe_f
iles::C:/Users/Administrator/disable-uac.rb line 3)

  - set value {:name=>"EnableLUA", :type=>:dword, :data=>0}
[2017-10-30T19:41:15+00:00] INFO: Chef Run complete in 0.060001 seconds
Running handlers:
[2017-10-30T19:41:15+00:00] INFO: Running report handlers
Running handlers complete
```

Type the specified command to apply the recipe file. This should make a change to the registry key and alert you that you need to restart Windows to disable UAC.



GL: Disable Limited User Account

Managing files is nice but what about Registry keys?

Objective:

- ✓ Create a recipe that disables Limited User Account
- ✓ Apply that recipe

The policy you defined in the recipe has now updated the system and you should now be prompted that Limited User Account has been disabled.



Lab: Disable Consent Prompt

- Update the recipe file named "disable-uac.rb" to also define:

The registry_key named
'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System' has the
values:

```
[ { :name => 'ConsentPromptBehaviorAdmin', :type => :dword, :data => 0 } ]
```

- Use **chef-client** to apply the recipe file named "disable-uac.rb"

Changing the previous registry key only disables some of UAC. To finish the work return to the recipe file that you created and add another registry resource with the following values.

Instructor Note: Allow 10 minutes to complete this exercise.

Lab: Disable the Limited User Account

```
~\disable-uac.rb
```

```
system_policies = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'

registry_key system_policies do
  # ... ENABLE LUA VALUES (NOT SHOWN HERE TO CONSERVE SPACE)
end

registry_key system_policies do
  values [{{
    :name => 'ConsentPromptBehaviorAdmin',
    :type => :dword,
    :data => 0
  }}]
end
```

This is the final recipe that contains the two registry keys. This new registry key uses the same variable that we defined before and sets a different values to disable the consent prompt.

Instructor Note: The previous registry key resource is represented here with a comment to allow more space for the new registry key being added.



Lab: Disable Consent Prompt

- ✓ Update the recipe file named "disable-uac.rb" to also define:

The registry_key named
'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System' has the
values:

```
[ { :name => 'ConsentPromptBehaviorAdmin', :type => :dword, :data => 0 } ]
```

- Use **chef-client** to apply the recipe file named "disable-uac.rb"

Lab: Apply a Recipe File



```
> chef-client --local-mode disable-uac.rb
```

```
Recipe: @recipe_files::C:/Users/Administrator/disable-uac.rb
  * registry_key[HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System] action create[2017-10-30T19:44:33+00:00]
    ] INFO: Processing
registry_key[HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System] action create
(@recipe_files::C:/Users/Administrator/disable-uac.rb line 3)
  (up to date)
  * registry_key[HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System] action create
create[2017-10-30T19:44:33+00:00]
    ] INFO: Processing
registry_key[HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System] action create
(@recipe_files::C:/Users/Administrator/disable-uac.rb line 12)
      - set value {:name=>"ConsentPromptBehaviorAdmin", :type=>:dword, :data=>0}
[2017-10-30T19:44:34+00:00] INFO: Chef Run complete in 0.063 seconds
```

Type the specified command to apply the recipe file. The first registry key should report that it is up-to-date. The second registry key will be updated to disable the consent prompt.



Lab: Disable Consent Prompt

- ✓ Update the recipe file named "disable-uac.rb" to also define:

The registry_key named
'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System' has the
values:

```
[ { :name => 'ConsentPromptBehaviorAdmin', :type => :dword, :data => 0 } ]
```

- ✓ Use **chef-client** to apply the recipe file named "disable-uac.rb"



Let's Talk About Resources

Capture your answers because we're going to talk about them as a group.

Let's finish this Resources module with a discussion.

Write down or type out a few words for each of these questions. Talk about your answers with each other.

Remember that the answer "I don't know! That's why I'm here!" is a great answer.



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?

Answer these four questions:

What is a resource?

What are some other possible examples of resources?

How did the examples 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?

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



Q&A

What questions can we answer for you?

- chef-client
- Resources
- Resource - default actions and default properties
- Test and Repair

What questions can we answer for you?

About anything or specifically about: chef-client; resources; a resources default action and default properties; and Test and Repair



CHEFTM

©2017 Chef Software Inc.

Cookbooks

Organizing Recipes

©2017 Chef Software Inc.

4-1





Objectives

After completing this module you should be able to:

- Generate a Chef cookbook
- Apply a run-list of recipes to a system
- Define a Chef cookbook that sets up a web server

In this module you will learn how to generate a cookbook with the Chef command-line application and applying multiple recipes to a system through a run-list.



Cookbooks

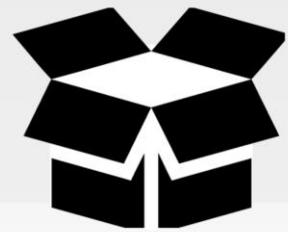
A Chef cookbook is the fundamental unit of configuration and policy distribution.

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

Read the first three paragraphs here: <http://docs.chef.io/cookbooks.html>



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



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.



GL: Setting up a Workstation

How are we going to manage this file? Does it need a README?

Objective:

- Create a cookbook
- Copy the disable-uac recipe within the cookbook
- Apply the recipe to the workstation

The disable-uac recipe is one of many recipes that we could define to setup our workstations. But before we throw this recipe file into a directory with our other scripts we should look at a concept in Chef called a cookbook.

What is a cookbook? How do we create one? Let's ask 'chef'.

Locating the cookbooks Directory



```
> dir
```

```
Directory: C:\Users\Administrator
```

Mode	LastWriteTime	Length	Name
d----	10/23/2017 12:51 AM		.atom
d----	2/16/2017 6:20 PM		.berkshelf
d----	10/23/2017 12:48 AM		.chef
d----	2/16/2017 6:23 PM		.kitchen
d----	2/16/2017 6:22 PM		.VirtualBox
d----	2/16/2017 6:21 PM		chef
d-r--	2/16/2017 6:16 PM		Contacts
d----	2/16/2017 6:44 PM		cookbooks
d-r--	10/23/2017 12:40 AM		Desktop
d-r--	2/16/2017 6:21 PM		Documents
d-r--	2/16/2017 6:16 PM		Downloads



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.

What can 'chef' do?



```
> 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
cookbook set
  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.

What Can 'chef generate' Do?



```
> 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
  policyfile    Generate a Policyfile for use with the install/push commands
  (experimental)
```

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'.

What Can 'chef generate cookbook' Do?



```
> chef generate cookbook --help
```

```
Usage: chef generate cookbook NAME [options]
  -C, --copyright COPYRIGHT           Name of the copyright holder - default...
  -m, --email EMAIL                  Email address of the author - defaults...
  -a, --generator-arg KEY=VALUE      Use to set arbitrary attribute KEY to ...
  -I, --license LICENSE              all_rights, httpd, mit, gplv2, gplv3 -
  -g GENERATOR_COOKBOOK_PATH,        Use GENERATOR_COOKBOOK_PATH for the
  --generator-cookbook
```

Let's ask the 'chef generate cookbook' command for help to see how it is used.

To generate a cookbook, all we have to do is provide it with a name.

There are two hard things in Computer Science and one of those is giving something a name.

Let's Create a Cookbook



```
> chef generate cookbook cookbooks\workstation
```

```
Generating cookbook workstation
- Ensuring correct cookbook file content
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content

Your cookbook is ready. Type `cd cookbooks/workstation` to enter it.
```

```
There are several commands you can run to get started locally developing and
testing your cookbook.
```

```
Type `delivery local --help` to see a full list.
```

```
Why not start by writing a test? Tests for the default recipe are stored at:
test/recipes/default_test.rb
```

We have you covered. Call the cookbook workstation. That's a generic enough name.

We want you to use 'chef generate' to generate a cookbook named workstation.

The Cookbook Has a README



```
> tree /f cookbooks\workstation
```

```
Folder PATH listing
Volume serial number is B04A-119C
C:\USERS\ADMINISTRATOR\COOKBOOKS\WORKSTATION
|
|   ...
|   .kitchen.yml
|   Berksfile
|   chefignore
|   metadata.rb
|   README.md
|
|---recipes
|     default.rb
|
```

Aren't you curious what's inside it? Let's take a look with the help of the 'tree' command. If we provide 'tree' with a path we will see all the visible files in the specified directory.

So the chef cookbook generator created an outline of a cookbook with a number of default files and folders. The first one we'll focus on is the README.



README.md

The description of the cookbook's features written in Markdown.

<http://daringfireball.net/projects/markdown/syntax>

All cookbooks that 'chef' will generate for you will include a default README file. The extension .md means that the file is a markdown file.

Markdown files are text documents that use various punctuation characters to provide formatting. They are meant to be easily readable by humans and can be easily be rendered as HTML or other formats by computers.

Let's Take a Look at the README



```
> gc cookbooks\workstation\README.rb
```

```
# workstation
```

```
TODO: Enter the cookbook description here.
```

If you view the contents of your new cookbook's metadata, you'll see a number of details that help describe the cookbook:

The name of the cookbook, its maintainer, a way to reach them, how the cookbook is licensed, descriptions, and the cookbook's version number.

The Cookbook Has Some Metadata



```
> tree /f cookbooks\workstation
```

```
Folder PATH listing
Volume serial number is B04A-119C
C:\USERS\ADMINISTRATOR\COOKBOOKS\WORKSTATION
|
|   ...
|   .kitchen.yml
|   Berksfile
|   chefignore
|   metadata.rb
|   README.md
|
|---recipes
|     default.rb
|
```

The cookbook also has a metadata file.

metadata.rb



Every cookbook requires a small amount of metadata. Metadata is stored in a file called `metadata.rb` that lives at the top of each cookbook's directory.

http://docs.chef.io/config_rb_metadata.html

This is a ruby file that contains its own domain specific language (DSL) for describing the details about the cookbook.

Let's Take a Look at the Metadata



```
> gc 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        '0.1.0'
```

If you view the contents of your new cookbook's metadata, you'll see a number of details that help describe the cookbook:

The name of the cookbook, its maintainer, a way to reach them, how the cookbook is licensed, descriptions, and the cookbook's version number.

The Cookbook Has a Folder for Recipes



```
> tree /f cookbooks\workstation
```

```
Folder PATH listing
Volume serial number is B04A-119C
C:\USERS\ADMINISTRATOR\COOKBOOKS\WORKSTATION
|
|   ...
|   .kitchen.yml
|   Berksfile
|   chefignore
|   metadata.rb
|   README.md
|
|---recipes
|     default.rb
|
```

The cookbook also has a folder named `recipes`. This is where we store the recipes in our cookbook. You'll see that the generator created a `default` recipe in our cookbook. What does it do?

The Cookbook Has a Default Recipe



```
> gc cookbooks\workstation\recipes\default.rb
```

```
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2017 The Authors, All Rights Reserved.
```

Looking at the contents of the default recipe you'll find it's empty except for some ruby comments.

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



GL: Setting up a Workstation

How are we going to manage this file? Does it need a README?

Objective:

- Create a cookbook
- Copy the disable-uac recipe within the cookbook
- Apply the recipe to the workstation

Copy the Recipe into the Cookbook



```
> mv disable-uac.rb cookbooks\workstation\recipes
```

From the Home directory, move your 'disable-uac.rb' recipe to the workstation cookbook and place it alongside our default recipe.



GL: Setting up a Workstation

How are we going to manage this file? Does it need a README?

Objective:

- Create a cookbook
- Copy the disable-uac recipe within the cookbook
- Apply the recipe to the workstation

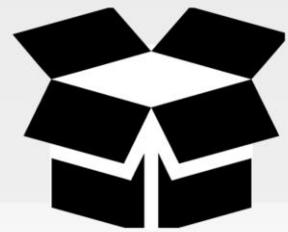


chef-client

```
$ sudo chef-client --local-mode RECIPE_FILE
```

How would we apply the workstation's setup recipe?

We have used 'chef-client' to apply recipes but we now face a new problem. How do we use this tool to apply multiple recipes to configure the state of our infrastructure? Combing the recipes seems like it goes against the concept that cookbooks map one-to-one to a piece of software. Running the command twice seems like it would make managing the system difficult to remember.



chef-client

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

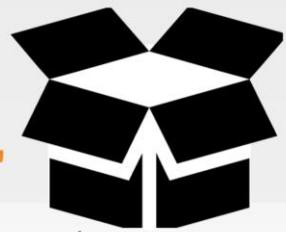
When a **chef-client** is run, it will perform all of the steps that are required to bring the node into the expected state.

https://docs.chef.io/chef_client.html

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.

--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.

GL: Applying the workstation's disable-uac recipe



```
> chef-client --local-mode --runlist "recipe[workstation::disable-uac]"
```

```
Synchronizing Cookbooks:
[2017-10-30T20:09:06+00:00] INFO: Storing updated
cookbooks/workstation/recipes/disable-uac.rb in the cache.
  - workstation (0.1.0)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::disable-uac
  *
registry_key[HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System]
action create[2017-10-30T20:09:06+00:00]
] INFO: Processing
registry_key[HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System]
```

Let's apply the workstation cookbook's recipe named 'disable-uac'.



GL: Setting up a Workstation

How are we going to manage this file? Does it need a README?

Objective:

- ✓ Create a cookbook
- ✓ Copy the disable-uac recipe within the cookbook
- ✓ Apply the recipe to the workstation

The recipe has been applied to the workstation.



Lab: Setting up a Web Server

- Use `chef generate` to create a cookbook named "myiis".
- Write and apply a recipe named "`server.rb`" with the policy:
The `powershell_script` named 'Install IIS' is run with the code '`Add-
WindowsFeature Web-Server`'.
The `file` named '`C:\inetpub\wwwroot\Default.htm`' is created with the content
`'<h1>Hello, world!</h1>'`
The `service` named '`w3svc`' is started and enabled.
- Apply the recipe and verify with `Invoke-WebRequest localhost`

Now. Here is your last challenge: Deploying a Web Server with Chef.

We need a cookbook named iis that has a server recipe. Within that server recipe we need to use a powershell_script to install the IIS windows feature, write out an example HTML file, and then start and enable the w3svc service.

Then we should apply that recipe and make sure the site is up and running by running a command to visit that site.

Instructor Note: Allow 15 minutes to complete this exercise.

Lab: Create a Cookbook



```
> chef generate cookbook cookbooks\myiis
```

```
Generating cookbook iis
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master
```

```
Your cookbook is ready. Type `cd iis` to enter it.
```

```
There are several commands you can run to get started locally developing and
testing your cookbook.
```

From the Chef home directory, run the command 'chef generate cookbook cookbooks\myiis'. This will place the myiis cookbook alongside the workstation cookbook.



Lab: Setting up a Web Server

- ✓ Use `chef generate` to create a cookbook named "`myiis`".
- ❑ Write and apply a recipe named "`server.rb`" with the policy:
The `powershell_script` named 'Install IIS' is run with the code '`Add-
WindowsFeature Web-Server`'.
The `file` named '`C:\inetpub\wwwroot\Default.htm`' is created with the content
`'<h1>Hello, world!</h1>'`
The `service` named '`w3svc`' is started and enabled.
- ❑ Apply the recipe and verify with `Invoke-WebRequest localhost`

Lab: Create a Recipe



```
> chef generate recipe cookbooks\myiis server
```

```
Compiling Cookbooks...
Recipe: code_generator::recipe
  * directory[./cookbooks/iis/spec/unit/recipes] action create (up to date)
  * cookbook_file[./cookbooks/iis/spec/spec_helper.rb] action create_if_missing
    (up to date)
  * template[./cookbooks/iis/spec/unit/recipes/server_spec.rb] action
    create_if_missing
    - create new file ./cookbooks/iis/spec/unit/recipes/server_spec.rb
    - update content in file ./cookbooks/iis/spec/unit/recipes/server_spec.rb
      from none to 93f1e7
      (diff output suppressed by config)
  * template[./cookbooks/iis/recipes/server.rb] action create
    - create new file ./cookbooks/iis/recipes/server.rb
    - update content in file ./cookbooks/iis/recipes/server.rb from none to
```

From the Chef home directory, run the command 'chef generate recipe myiis server'. This will create a recipe called server.rb in the myiis cookbook.

Lab: Create Server Recipe

```
~\cookbooks\myiis\recipes\server.rb
```

```
powershell_script 'Install IIS' do
  code 'Add-WindowsFeature Web-Server'
end

file 'C:\inetpub\wwwroot\Default.htm' do
  content '<h1>Hello, world!</h1>'
end

service 'w3svc' do
  action [:enable, :start]
end
```

The server recipe, found at ~/myiis/recipes/server.rb, defines the policy:

Install the web server feature, create an example html file, and ensure the w3svc service is started and enabled

Instructor Note: The service action defines two actions within a Ruby array. Ruby arrays are ordered, integer-indexed collections of any object. Each element in an array is associated with and referred to by an index.



Lab: Setting up a Web Server

- ✓ Use `chef generate` to create a cookbook named "`myiis`".
- ✓ Write and apply a recipe named "`server.rb`" with the policy:
The `powershell_script` named 'Install IIS' is run with the code '`Add-
WindowsFeature Web-Server`'.
The `file` named '`C:\inetpub\wwwroot\Default.htm`' is created with the content
`'<h1>Hello, world!</h1>'`
The `service` named '`w3svc`' is started and enabled.
 Apply the recipe and verify with `Invoke-WebRequest localhost`

Lab: Apply the Server Recipe



```
> chef-client -z --runlist "recipe[myiis::server]"  
  
Starting Chef Client, version 12.13.37  
resolving cookbooks for run list: []  
Synchronizing Cookbooks:  
Compiling Cookbooks...  
[2016-03-28T21:20:12+00:00] WARN: Node WIN-14DV1I4A82F.ec2.internal has an empty run  
list.  
Converging 3 resources  
Recipe: @recipe_files::C:/Users/Administrator/cookbooks/iis/recipes/server.rb  
  * powershell_script[Install IIS] action run  
    - execute "C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe" -NoLogo -  
NonInteractive -NoProfile -ExecutionP  
olicy Bypass -InputFormat None -File "C:/Users/ADMINI~1/AppData/Local/Temp/2/chef-  
script20160328-2492-q111ef.ps1"  
  * file[C:\inetpub\wwwroot\Default.htm] action create  
    - create new file C:\inetpub\wwwroot\Default.htm
```

When applying the recipe with 'chef-client', you need to specify the partial path to the recipe file within the myiis cookbook's recipe folder.

Lab: Verify That the Website is Available



> Invoke-WebRequest localhost

```
StatusCode        : 200
StatusDescription : OK
Content          : <h1>Hello, world</h1>
RawContent       : HTTP/1.1 200 OK
                  Accept-Ranges: bytes
                  Content-Length: 21
                  Content-Type: text/html
                  Date: Mon, 21 Dec 2016 20:59:13 GMT
                  ETag: "954c2066323cd11:0"
                  Last-Modified: Mon, 21 Dec 2016 20:58:52 GMT
                  Server...
```

So verify that the website is available and returns the content we expect to see.



Lab: Setting up a Web Server

- ✓ Use `chef generate` to create a cookbook named "myiis".
- ✓ Write and apply a recipe named "`server.rb`" with the policy:
The `powershell_script` named 'Install IIS' is run with the code '`Add-
WindowsFeature Web-Server`'.
The `file` named '`C:\inetpub\wwwroot\Default.htm`' is created with the content
`'<h1>Hello, world!</h1>'`
The `service` named '`w3svc`' is started and enabled.
- ✓ Apply the recipe and verify with `Invoke-WebRequest localhost`

Now. Here is your last challenge: Deploying a Web Server with Chef.

We need a cookbook named iis that has a server recipe. Within that server recipe we need to use a powershell_script to install the IIS windows feature, write out an example HTML file, and then start and enable the w3svc service.

Then we should apply that recipe and make sure the site is up and running by running a command to visit that site.

Instructor Note: Allow 15 minutes to complete this exercise.



Discussion

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

What other recipes might you include in the myiis or workstation cookbooks?

How do resources accept multiple actions?

Answer these questions.

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



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.



CHEFTM

©2017 Chef Software Inc.

Version Control with Git

Recording changes to our cookbooks



Objectives

After completing this module you should be able to:

- Use version control
- Setup a GitHub repo
- Include recipes from within another recipe
- Update the version of a cookbook

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



Questions You May Have

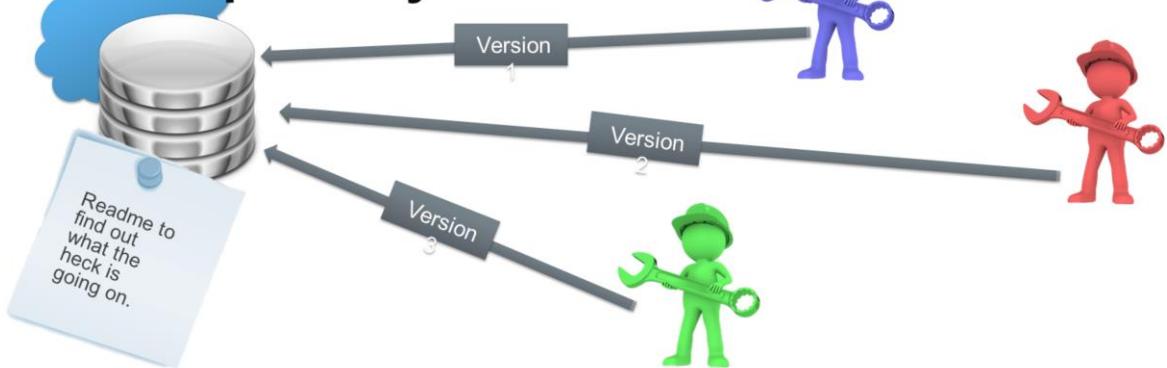
- If we have multiple versions - how might we better track these different versions and changes?
- Where would we store these different versions?

1. Recipes on their own would be difficult to manage. Imagine handing a copy of a recipe that you develop to an individual. A few weeks later they come back to you and ask you why it no longer works. If you had made a number of changes since the last time you talked with them how do you know which version you gave them? Adding a version number and a README would allow you to better document the features of a particular recipe.
2. When developing recipes it is important to consider using a version control solution that allows us to track the changes that we make.
3. The recipe that you put together to disable UAC showed that Chef is powerful enough to manage the registry. Chef is powerful enough to install and configure a web server.



Collaboration and Version Control

Central Repository



Before we answer that question, let's talk about collaboration. Usually, none of us work in a vacuum, and it's important that systems are in place to make collaboration easier. One such system is versioning. Versioning will make it easier to share the recipes that we create.

A versioning system should include:

A Central Repository into which all the developers publish their work. Each revision should be stored as a new version. For each change, a commit message should be added so that everyone knows what has or has not been changed

Versioning Pros and Cons

```
> cp hello.rb hello.rb.bak  
or  
> cp hello.rb hello-20170401.rb  
or  
> cp setup.rb setup-20170401-username.rb
```

Saving a copy of the original file as another filename.

Let's explore this first option of renaming the file by adding a quick extension, like in the first example shown here. In this way we can keep working on the original file as we add more features. As a group let's talk about the pros and cons of using this strategy.

So obviously a single backup won't do. We need backups more often as we are going to be iterating quickly. We could use the current date and time down to the minute like in the second example. As a group let's talk about the pros and cons of using this strategy.

Would adding the user's name to the end of the file, like in the third example, solve the problems we are facing with other choices? Again what are the pros and cons of this new approach?



Git Version Control

git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.

We will be using **git** throughout the rest of this course.



How about we use git? What are the pros and cons of this approach?

For the rest of this course we will be using git. This may not be the version control software you use on your teams or within your organization and that is alright. Our use of git within this course is used solely to demonstrate the use of version control when developing Chef code. When you develop with Chef you are welcome to use the version control system of your choice.

Instructor Note: It is not important that the learners understand and learn all of git during this course. It is more important that the learners understand when and where to use version control to save their work. This is about training them on making changes, testing, and then committing their work. Version control is an instrumental piece of the workflow when you adopt Infrastructure as code. There are some benefits of learning and using git because Chef uses git and GitHub to do almost all development of Chef. The majority of the Chef community uses git and GitHub.



Git Version Control with GitHub

If you use git versioning you should ultimately push the local git repository to a shared remote git repository.

In this way others could collaborate with you from a centralized location.

The screenshot shows a GitHub repository page for 'chef-training / chefdk-fundamentals-repo'. The 'apache' cookbook's commit history is displayed under the 'Branch: master' dropdown. The commits are as follows:

File	Message	Author	Date
recipes	Removed the 'yum update' it was causing problems	burtio	9 days ago
spec	ChefDK Fundamentals Course - Day 1 Repo		4 months ago
templates/default	ChefDK Fundamentals Course - Day 1 Repo		4 months ago
test/integration/default/serverspec	ChefDK Fundamentals Course - Day 1 Repo		4 months ago
.gitignore	ChefDK Fundamentals Course - Day 1 Repo		4 months ago
kitchen.yml	ChefDK Fundamentals Course - Day 1 Repo		4 months ago

git tracks all our commits, all those closed up boxes, locally on the current system. If we wanted to share those commits with other individuals we would need to push those changes to a central repository where we could collaborate with other members of the team. GitHub is an example of a central git repository.



GL: Configure git

How do we keep track of our changes?

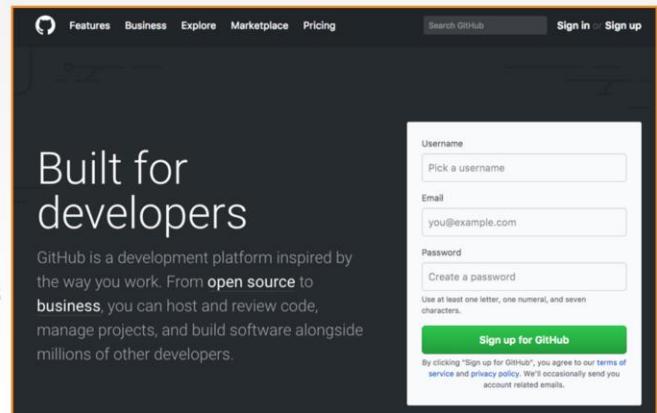
Objective:

- Create a GitHub account
- Create a GitHub repository
- Setup git version control for our 'workstation' cookbook
- Commit our work with git
- Push work up to GitHub repo

Signing Up for a GitHub Account

Steps

1. Navigate to <https://github.com/join>
2. Fill out the form as indicated in this image selecting a username and a valid email address and then click **Get Started**.
3. Select the **free public repositories** plan
4. Click **Submit** to create your account



Signing Up for a GitHub Account

Steps

1. Navigate to <https://github.com/join>
2. Fill out the form as indicated in this image selecting a username and a valid email address and then click **Get Started**.
3. Select the free public repositories plan
4. Click **Submit** to create your account

The best way to design, build, and ship software.

Step 1: Create personal account Step 2: Choose your plan Step 3: Tailor your experience

Create your personal account

Username
SomeGenericUserName

This will be your username — you can enter your organization's username next.

Email Address
youremail@email.com

You will occasionally receive account related emails. We promise not to share your email with anyone.

Password

Use at least one lowercase letter, one numeral, and seven characters.

By clicking on "Create an account" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

[Create an account](#)

You'll love GitHub

- Unlimited collaborators
- Unlimited public repositories
- Great communication
- Frictionless development
- Open source community

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

Signing Up for a GitHub Account

Steps

1. Navigate to <https://github.com/join>
2. Fill out the form as indicated in this image selecting a username and a valid email address and then click **Get Started**.
3. Select the **free public repositories** plan
4. Click **Submit** to create your account

Welcome to GitHub

You've taken your first step into a larger world, @SomeGenericUserName.

<input checked="" type="checkbox"/> Completed Set up a personal account	<input type="checkbox"/> Step 2: Choose your plan	<input type="checkbox"/> Step 3: Tailor your experience
--	--	--

Choose your personal plan

Unlimited public repositories for free.

Unlimited private repositories for \$7/month.

Don't worry, you can cancel or upgrade at any time.

Help me set up an organization next
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations](#)

Send me updates on GitHub news, offers, and events
Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

Signing Up for a GitHub Account

Steps

1. Navigate to <https://github.com/join>
2. Fill out the form as indicated in this image selecting a username and a valid email address and then click **Get Started**.
3. Select the **free public repositories** plan
4. Click **Submit** to create your account

Welcome to GitHub
You'll find endless opportunities to learn, code, and create,
@SomeGenericUserName.

Completed Set up a personal account Step 2: Choose your plan Step 3: Tailor your experience

How would you describe your level of programming experience?

Totally new to programming Somewhat experienced Very experienced

What do you plan to use GitHub for? (check all that apply)

Research School projects Development
 Design Project Management Other (please specify)

Which is closest to how you would describe yourself?

I'm a student I'm a professional I'm a hobbyist
 Other (please specify)

What are you interested in?

chef!

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

Submit skip this step



GL: Configure git

How do we keep track of our changes?

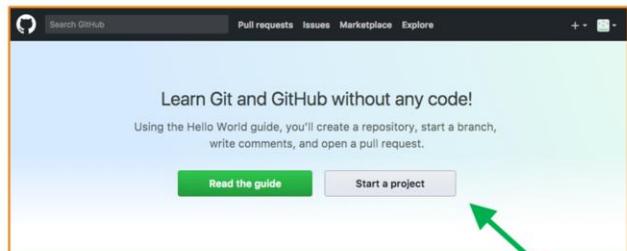
Objective:

- Create a GitHub account
- Create a GitHub repository
- Setup git version control for our 'workstation' cookbook
- Commit our work with git
- Push work up to GitHub repo

GL: Create a GitHub Repository for ‘workstation’

Steps

1. Navigate to GitHub and click **Start a project**
2. Fill out the form as indicated in this image using workstation as the repository name, fill out a description if desired and click **Create repository**
3. Copy the url for the repository



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address. An email containing verification instructions was sent to youremail@email.com.

Note: If you haven't already you will be prompted to verify your email address



GL: Create a GitHub Repository for ‘workstation’

Steps

1. Navigate to GitHub and click **Start a project**
2. Fill out the form as indicated in this image using **workstation** as the repository name, fill out a description if desired and click **Create repository**
3. Copy the url for the repository

Create a new repository
A repository contains all the files for your project, including the revision history.

Owner: SamMc87 / Repository name: workstation 

Great repository names are short and memorable. Need inspiration? How about [congenial-waffle](#).

Description (optional): This is the ‘workstation’ cookbook for the Chef DevOps Foundation class

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you’re importing an existing repository.

Add .gitignore: None | Add a license: None | ⓘ

Create repository

©2017 Chef Software Inc.

5-15

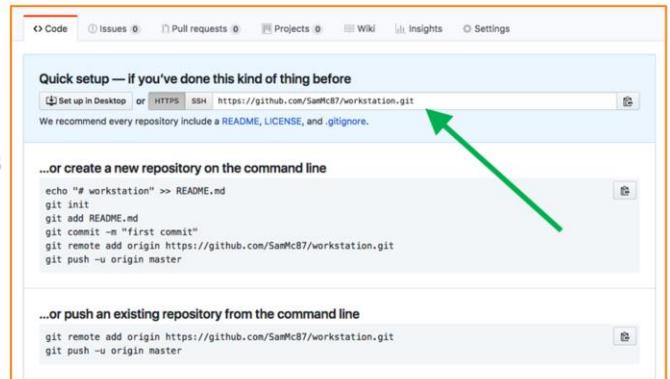


Make sure students DO NOT select the ‘Initialize this repository with a README’. This will cause conflicts with the README found in the cookbook’s directory.

GL: Create a GitHub Repository for ‘workstation’

Steps

1. Navigate to GitHub and click **Start a project**
2. Fill out the form as indicated in this image using workstation as the repository name, fill out a description if desired and click **Create repository**
3. Copy the url for the repository





GL: Configure git

How do we keep track of our changes?

Objective:

- ✓ Create a GitHub account
- ✓ Create a GitHub repository
- Setup git version control for our 'workstation' cookbook
- Commit our work with git
- Push work up to GitHub repo

Use 'git config' to set configuration variables



```
> git config --global user.email "you@example.com"
```

The command-line tool chef will automatically configure your cookbook as a Git repository. For this to work correctly you will need to setup your email address and name in the global git configuration.

Set up your email you want to associate with your git commits.

Use 'git config' to set configuration variables



```
> git config --global user.name "Username"
```

The command-line tool chef will automatically configure your cookbook as a Git repository. For this to work correctly you will need to setup your email address and name in the global git configuration.

Set up your user name you want to associate with your git commits.

Move into the workstation Directory



```
> cd cookbooks\workstation
```

Change into the workstation cookbook directory.

Move to Cookbook Directory and Initialize as a git Repository



```
> git init
```

```
Reinitialized existing Git repository in C:/Users/Administrator/cookbooks/workstation/.git/
```

We want git to start tracking the entire contents of this folder and any content in the subfolders. To do that with git, you need to execute the command 'git init' in the parent directory of the cookbook that you want to start tracking.

You will notice that git will say that the repository has been 'Reinitialized'. This is because the chef cookbook generator detected that we have git installed and automatically initialized the cookbook as a git repository.



GL: Configure git

How do we keep track of our changes?

Objective:

- ✓ Create a GitHub account
- ✓ Create a GitHub repository
- ✓ Setup git version control for our 'workstation' cookbook
- Commit our work with git
- Push work up to GitHub repo



Staging Area

The staging area has a file, generally contained in your Git directory, that stores information about what will go into your next commit.

It's sometimes referred to as the "index", but it's also common to refer to it as the staging area.

<http://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

You can think of the staging area as a box in which to put a bunch of items -- like a care package you would send to someone.

Staging files means to put them in the box, but don't close it up because you may add a few things, and don't close it up because you may replace or remove a few things. But put the items in the box because eventually we are going to close that box when it is ready to send it off.

Use 'git add' to Stage Files to be Committed



```
> git add .
```

Now we need to tell git which files it should start tracking in source control. In our case, we want to add all the files to the repository and we can do that by executing 'git add .' (dot).

This will place all the files into a staging area.

Use 'git status' to View the Staged Files



```
> git status
```

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:  recipes/disable-uac.rb
```

Let's see what changes we have placed in the staging area.

Thinking about our care package example, this is like looking inside the box and taking an inventory, allowing us to figure out if we need to move more things in or remove things we accidentally threw in there.

Running `git status` allows us to see in the box. Git reports back to us the changes that will be committed.

Instructor Note: Git helpfully tries to show you the command you can use to remove an item from that box. This is useful if you want to include all items excepts for one or simply manage everything before you commit.

Use 'git commit' to Save the Staged Changes



```
> git commit -m "Initial Commit"
```

```
[master 2c933a8] Initial Commit
 1 file changed, 17 insertions(+)
 create mode 100644 recipes/disable-uac.rb
```

If everything that is staged looks correct, then we are ready to commit the changes.



GL: Configure git

How do we keep track of our changes?

Objective:

- ✓ Create a GitHub account
- ✓ Create a GitHub repository
- ✓ Setup git version control for our 'workstation' cookbook
- ✓ Commit our work with git
- Push work up to GitHub repo



Working with Remotes

For us to contribute to a project found on GitHub, we must be able to manage our remote repositories. Primarily for us, we must be able to push our cookbooks up to their respective repositories found on GitHub.

<https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes>

Add remote for workstation Cookbook



```
> git remote add origin https://github.com/username/workstation.git
```

Push workstation Cookbook to Repo



```
> git push -u origin master
```

```
Username for 'https://github.com': username
Password for 'https://username@github.com':
Counting objects: 62, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (45/45), done.
Writing objects: 100% (62/62), 10.19 KiB | 0 bytes/s, done.
Total 62 (delta 14), reused 0 (delta 0)
remote: Resolving deltas: 100% (14/14), done.
To https://github.com/SamMc87/workstation.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

[-u | --set-upstream]

Verify Push in a Browser

This is the 'workstation' cookbook for the Chef DevOps Foundation class

5 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

SamMcB7 Initial Commit

		Latest commit 2c933a8 an hour ago
delivery	Add generated delivery build cookbook	2 hours ago
recipes	Initial Commit	an hour ago
spec	Add generated cookbook content	2 hours ago
test/smoke/default	Add generated cookbook content	2 hours ago
.gitignore	Add generated cookbook content	2 hours ago
kitchen.yml	Add generated cookbook content	2 hours ago
Berksfile	Add generated cookbook content	2 hours ago
README.md	Add generated cookbook content	2 hours ago
chefignore	Add generated cookbook content	2 hours ago
metadata.rb	Add generated cookbook content	2 hours ago

workstation

TODO: Enter the cookbook description here.



GL: Configure git

How do we keep track of our changes?

Objective:

- ✓ Create a GitHub account
- ✓ Create a GitHub repository
- ✓ Setup git version control for our 'workstation' cookbook
- ✓ Commit our work with git
- ✓ Push work up to GitHub repo



Version Control through Git

We now have a system in place that can record all our changes to the workstation cookbook...

So what if we go ahead introduce some new changes?

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>



A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- Load the workstation cookbook's disable-uac recipe in the default recipe
- Apply the workstation cookbook's default recipe
- Update the version of the "workstation" cookbook
- Commit and push the changes

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

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



-r "recipe[COOKBOOK(:default)]"

When you are referencing the default recipe within a cookbook you may optionally specify only the name of the cookbook.

chef-client understands that you mean to apply the default recipe from within that cookbook.

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

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

Similar to how resources have default actions and default properties, Chef uses the concept of providing sane defaults. A cookbook doesn't have to have a default recipe but most every cookbook has one. It's called default because when you think of a cookbook it is the recipe that defines the most common configuration policy.

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

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.

<https://docs.chef.io/recipes.html#include-recipes>

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

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

Demo: Including a Recipe

```
include_recipe 'workstation::disable-uac'
```

Include the 'disable-uac' recipe from the 'workstation' cookbook in this recipe

In this example we are including the 'workstation' cookbook's 'disable-uac' recipe.

Demo: Including a Recipe

```
include_recipe 'myiis::server'
```

Include the 'server' recipe from the 'myiis' cookbook in this recipe

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

GL: The Default Recipe Includes the Disable Recipe

```
~\cookbooks\workstation\recipes\default.rb

#
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2017 The Authors, All Rights Reserved.

include_recipe 'workstation::disable-uac'
```

We are interested in having the default recipe for our workstation cookbook run the contents of the 'disable-uac' recipe.

Within the default recipe, define the `include_recipe` method and provide one parameter, which is the name of our recipe as it appears within a run list: `cookbook_name::recipe_name`.



A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- ✓ Load the workstation cookbook's disable-uac recipe in the default recipe
- Apply the workstation cookbook's default recipe
- Update the version of the "workstation" cookbook
- Commit and push the changes

Apply the Cookbook's Default Recipe



```
> chef-client --local-mode -r "recipe[workstation]"
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation (0.1.0)
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::disable-uac
  * registry_key[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System]
action create (up to date)
  * registry_key[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System]
action create (up to date)

Running handlers:
```

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

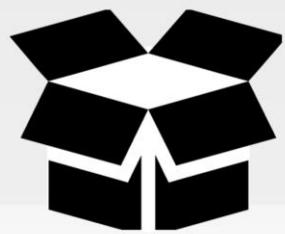


A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- ✓ Load the workstation cookbook's disable-uac recipe in the default recipe
- ✓ Apply the workstation cookbook's default recipe
- ❑ Update the version of the "workstation" cookbook
- ❑ Commit and push the changes



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

©2017 Chef Software Inc.

5-43



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 hello message. Now the page displays the hello message with additional details about the system. The changes that we finished are new features of the cookbook.



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 and refactoring of code

<http://semver.org>

©2017 Chef Software Inc.

5-44



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.



Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

©2017 Chef Software Inc.

5-45



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

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. That sounds like a Minor changes based on the definitions provided by the Semantic Versioning documentation.

GL: Update 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        '0.1.1'
```

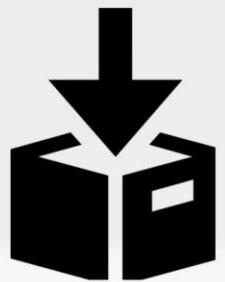


A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- ✓ Load the workstation cookbook's disable-uac recipe in the default recipe
- ✓ Apply the workstation cookbook's default recipe
- ✓ Update the version of the "workstation" cookbook
- Commit and push the changes



GL: Commit Your Work

```
> cd ~\cookbooks\workstation  
> git add .  
> git status  
> git commit -m "Version 0.1.1, included disable-uac recipe within default"  
> git push
```

GL: Verify Changes to Repo in a Browser

This is the 'workstation' cookbook for the Chef DevOps Foundation class

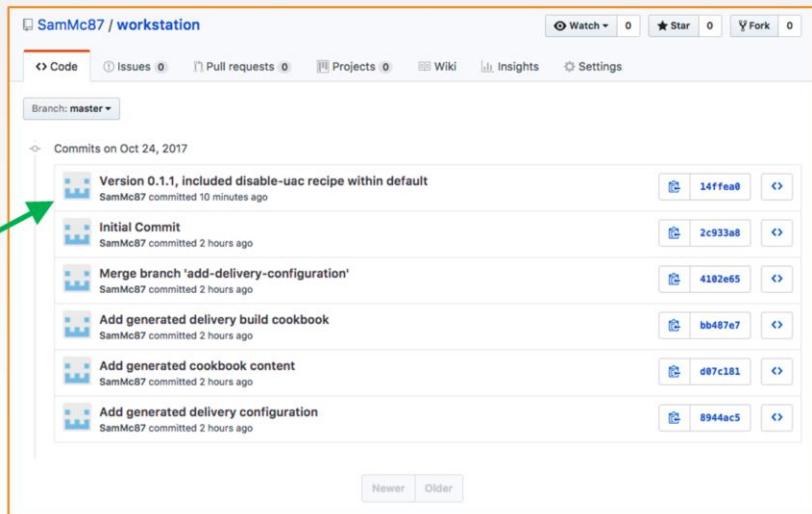
Branch: master ▾ New pull request

6 commits 1 branch 0 releases 1 contributor

SamMc87 Version 0.1.1, included disable-uac recipe within default

File	Description	Time Ago
.delivery	Add generated delivery build cookbook	2 hours ago
recipes	Version 0.1.1, included disable-uac recipe within default	a minute ago
spec	Add generated cookbook content	2 hours ago
test/smoke/default	Add generated cookbook content	2 hours ago
.gitignore	Add generated cookbook content	2 hours ago
.kitchen.yml	Add generated cookbook content	2 hours ago
Berksfile	Add generated cookbook content	2 hours ago
README.md	Add generated cookbook content	2 hours ago
chefignore	Add generated cookbook content	2 hours ago
metadata.rb	Version 0.1.1, included disable-uac recipe within default	a minute ago

GL: Verify Changes to Repo in a Browser



The screenshot shows a GitHub repository page for 'SamMcB7 / workstation'. The 'Code' tab is selected. The 'Branch: master' dropdown is set to 'master'. Below it, a list of commits is shown for 'Commits on Oct 24, 2017'. The commits are:

- Version 0.1.1, included disable-uac recipe within default
SamMcB7 committed 10 minutes ago
- Initial Commit
SamMcB7 committed 2 hours ago
- Merge branch 'add-delivery-configuration'
SamMcB7 committed 2 hours ago
- Add generated delivery build cookbook
SamMcB7 committed 2 hours ago
- Add generated cookbook content
SamMcB7 committed 2 hours ago
- Add generated delivery configuration
SamMcB7 committed 2 hours ago

At the bottom of the commit list are 'Newer' and 'Older' buttons.

GL: Verify Changes to Repo in a Browser

Version 0.1.1, included disable-uac recipe within default
by master

SamMcB7 committed 12 minutes ago 1 parent 2c933a8 commit 14ffea0cd7d2f4e87d50dcc37b9e465c36b30392

Showing 2 changed files with 3 additions and 1 deletion.

Unified Split

View ▾

2 metadata.rb

```
@@ -4,7 +4,7 @@  
4 license 'all_rights'  
5 description 'Installs/Configures workstation'  
6 long_description 'Installs/Configures workstation'  
7 -version '0.1.0'  
7 +version '0.1.1'  
8  
9 # The 'issues_url' points to the location where issues for this cookbook are  
10 # tracked. A 'View Issues' link will be displayed on this cookbook's page when  
@@
```

View ▾

2 recipes/default.rb

```
@@ -3,3 +3,5 @@  
3 # Recipe: default  
4 #  
5 # Copyright:: 2017, The Authors, All Rights Reserved.  
6 +  
7 +include_recipe 'workstation::disable-uac'
```



A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- ✓ Load the workstation cookbook's disable-uac recipe in the default recipe
- ✓ Apply the workstation cookbook's default recipe
- ✓ Update the version of the "workstation" cookbook
- ✓ Commit and push the changes



Lab: Update the myiis Cookbook

Create the myiis repository and include_recipe

Objective:

- Create a GitHub repo for the "myiis" cookbook and push it up to the remote repository
- Update the "myiis" cookbook's "default" recipe to include the "server" recipe from the "myiis" cookbook
- Run chef-client and locally apply the run_list: "recipe[myiis]"
- Update the patch version of the "myiis" cookbook
- Commit and push the changes with git

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

Instructor Note: Allow 10 minutes to complete this exercise.

Lab: Create myiis GitHub Repo and copy url

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: SamMcB7 / Repository name: myiis 

Great repository names are short and memorable. Need inspiration? How about bug-free-octo-giggle.

Description (optional): This is the 'myiis' cookbook for the Chef DevOps Foundation class

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None | 

Create repository

Code Issues Pull requests Projects Wiki Insights Settings

Quick setup — if you've done this kind of thing before
[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/SamMcB7/myiis.git> 

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# myiis" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SamMcB7/myiis.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/SamMcB7/myiis.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Lab: Move into the myiis Directory



```
> cd cookbooks\myiis
```



Change into the myiis cookbook directory.

Lab: Move to Cookbook Directory and Initialize as a git Repository



```
> git init
```

```
Reinitialized existing Git repository in C:/Users/Administrator/cookbooks/myiis/.git/
```

We want git to start tracking the entire contents of this folder and any content in the subfolders. To do that with git, you need to execute the command 'git init' in the parent directory of the cookbook that you want to start tracking.

You will notice that git will say that the repository has been 'Reinitialized'. This is because the chef cookbook generator detected that we have git installed and automatically initialized the cookbook as a git repository.

Lab: Use 'git commit' to Save the Staged Changes



```
> git commit -m "Initial Commit"
```

```
On branch master
nothing to commit, working directory clean
```

If everything that is staged looks correct, then we are ready to commit the changes.

Lab: Add remote for workstation Cookbook



```
> git remote add origin https://github.com/username/myiis.git
```

Lab: Push workstation Cookbook to Repo



```
> git push -u origin master
```

```
Username for 'https://github.com': username
Password for 'https://username@github.com':
Counting objects: 70, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (50/50), done.
Writing objects: 100% (70/70), 10.46 KiB | 0 bytes/s, done.
Total 70 (delta 16), reused 0 (delta 0)
remote: Resolving deltas: 100% (16/16), done.
To https://github.com/SamMc87/myiis.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

git push -u "Upstream" would refer to the main repo that other people will be pulling from, e.g. your GitHub repo. The -u option automatically sets that upstream for you, linking your repo to a central one. That way, in the future, Git "knows" where you want to push to and where you want to pull from, so you can use git pull or git push without arguments.



Lab: Update the myiis Cookbook

Create the myiis repository and include_recipe

Objective:

- ✓ Create a GitHub repo for the "myiis" cookbook and push it up to the remote repository
- Update the "myiis" cookbook's "default" recipe to include the "server" recipe from the "myiis" cookbook
- Run chef-client and locally apply the run_list: "recipe[myiis]"
- Update the patch version of the "myiis" cookbook
- Commit and push the changes with git

Lab: The default Recipe Includes the server Recipe

```
~\cookbooks\myiis\recipes\default.rb

#
# Cookbook Name:: myiis
# Recipe:: default
#
# Copyright (c) 2017 The Authors, All Rights Reserved.

include_recipe 'myiis::server'
```

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

Within the default recipe, define the `include_recipe` method and provide one parameter, which is the name of our recipe as it appears within a run list: `cookbook_name::recipe_name`.



Lab: Update the myiis Cookbook

Create the myiis repository and include_recipe

Objective:

- ✓ Create a GitHub repo for the "myiis" cookbook and push it up to the remote repository
- ✓ Update the "myiis" cookbook's "default" recipe to include the "server" recipe from the "myiis" cookbook
 - Run chef-client and locally apply the run_list: "recipe[myiis]"
 - Update the patch version of the "myiis" cookbook
 - Commit and push the changes with git

Lab: Applying the myiis Default Recipe



```
> chef-client --local-mode -r "recipe[myiis]"
```

```
[2017-09-15T15:23:18+00:00] WARN: No config file found or specified on command line, using command line options.  
Starting Chef Client, version 12.3.0  
resolving cookbooks for run list: ["myiis"]  
Synchronizing Cookbooks:  
  - myiis  
Compiling Cookbooks...  
Converging 3 resources  
  
Running handlers:  
Running handlers complete  
Chef Client finished, 1/3 resources updated in 3.310768509 seconds
```

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



Lab: Update the myiis Cookbook

Create the myiis repository and include_recipe

Objective:

- ✓ Create a GitHub repo for the "myiis" cookbook and push it up to the remote repository
- ✓ Update the "myiis" cookbook's "default" recipe to include the "server" recipe from the "myiis" cookbook
- ✓ Run chef-client and locally apply the run_list: "recipe[myiis]"
- Update the patch version of the "myiis" cookbook
- Commit and push the changes with git

GL: Update the Cookbook Version

```
~\cookbooks\myiis\metadata.rb
```

```
name          'myiis'
maintainer    'The Authors'
maintainer_email 'you@example.com'
license        'all_rights'
description    'Installs/Configures workstation'
long_description 'Installs/Configures workstation'
version        '0.1.1'
```

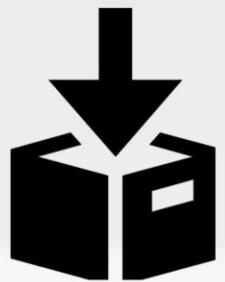


Lab: Update the myiis Cookbook

Create the myiis repository and include_recipe

Objective:

- ✓ Create a GitHub repo for the "myiis" cookbook and push it up to the remote repository
- ✓ Update the "myiis" cookbook's "default" recipe to include the "server" recipe from the "myiis" cookbook
- ✓ Run chef-client and locally apply the run_list: "recipe[myiis]"
- ✓ Update the patch version of the "myiis" cookbook
- Commit and push the changes with git



Lab: Commit Your Work

```
> cd ~\cookbooks\myiis  
> git add .  
> git commit -m "Version 0.1.1, include server  
recipe within default"  
> git push
```



Lab: Update the myiis Cookbook

Create the myiis repository and include_recipe

Objective:

- ✓ Create a GitHub repo for the "myiis" cookbook and push it up to the remote repository
- ✓ Update the "myiis" cookbook's "default" recipe to include the "server" recipe from the "myiis" cookbook
- ✓ Run chef-client and locally apply the run_list: "recipe[myiis]"
- ✓ Update the patch version of the "myiis" cookbook
- ✓ Commit and push the changes with git



Discussion

Why should we use a tool like Git for version control?

How often should you commit changes with version control?

Answer these questions.

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



Q&A

What questions can we answer for you?

- Cookbooks
- include_recipe method
- Versions
- Version control

What questions can we help you answer?

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



CHEFTM

©2017 Chef Software Inc.

Ohai and the Node Object

Finding and Displaying Information About Our System



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

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



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.

Some Useful System Data

- platform
- hostname
- memory
- CPU - MHz

Thinking about some of the scenarios that we mentioned at the start of the session makes us think that it would be useful to capture:

The platform, hostname, memory, and CPU megahertz of our current system.

Instructor Note: The next series of steps often seem like an obvious mistake to the learners. It may be helpful to have the learners watch as you perform these steps and comment on the process.

Demo: Finding Platform Info



```
> Get-WMIObject Win32_OperatingSystem
```

```
SystemDirectory : C:\Windows\system32
Organization    : Amazon.com
BuildNumber     : 9600
RegisteredUser  : EC2
SerialNumber    : 00252-70000-00000-AA535
Version         : 6.3.9600
```

Demo: Finding the Hostname



```
> hostname
```

```
WIN-KRQSVD3RFM7
```

Demo: Finding the Total Memory



```
> wmic ComputerSystem get TotalPhysicalMemory
```

```
TotalPhysicalMemory
```

```
8052654080
```

Running the following command will return to the total physical memory of the system.

Demo: Finding the CPU MHz



```
> wmic cpu get name
```

```
Name  
Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz
```

Running the following command will return to the cpu name which includes the clock speed of the CPU.



Capturing System Data

What are the limitations of the way we captured this data?

How accurate will our recipe be if we hard code this information within our recipes?

Now that we've defined these values, let's reflect:

What are the limitations of the way we captured this data?

How accurate will our Default page be when we deploy it on other systems?

Are these values we would want to capture in our tests?



Hard Coded Values

The values that we have derived at this moment may not be the correct values when we deploy this recipe again even on the same system!

If you have worked with systems for a while, the general feeling is that hard-coding the values in our file resource's attribute probably is not sustainable because the results are tied specifically to this system at this moment in time.



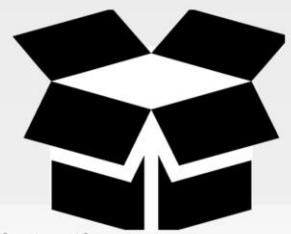
Data In Real Time

How could we capture this data in real-time?

So how can we capture this data in real-time?

Capturing the data in real-time on each system is definitely possible. One way would be to execute each of these commands, parse the results, and then insert the dynamic values within the file resource's content attribute. We could also figure out a way to run system commands within our recipes. Before we start down this path, we'd like to introduce you to Ohai.

Instructor Note: There are many ways within Ruby to escape out and run a system command. Someone new to Chef and Ruby may reach for those and this section is important in showing that most of the work has already been done.

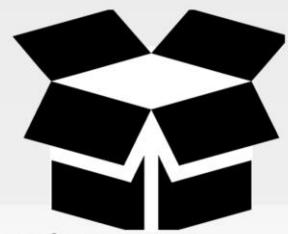


Ohai!

Ohai is a tool that already captures all the data that we similarly demonstrated finding.

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

Ohai is a tool that detects and captures attributes about our system. Attributes like the ones we spent our time capturing already.



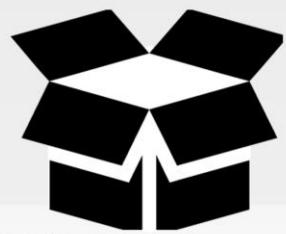
All About The System

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

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

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

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



The Node Object

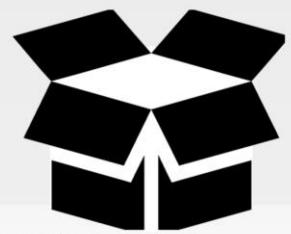
The node object is a representation of our system.
It stores all the attributes found about the system.

<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 platform, hostname, total memory, and cpu megahertz.



ohai + chef-client = <3

chef-client and chef-apply 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` and `chef-apply` automatically execute Ohai. This information is stored within a variable we call 'the node object'



GL: Details About the Node

Displaying system details in the default web page definitely sounds useful.

Objective:

- Discover attributes about the system with Ohai
- Update the web page file contents, in the "myiis" cookbook, to include system details
- Update the cookbook's version number
- Apply the updated recipe and verify the results
- Commit the changes to the "myiis" cookbook to version control

When deploying our IIS server it would be nice if the test page displayed some details about the system. Together, let's walk through finding out these details and then updating the content attribute of the file resource to include this new content.

GL: Running Ohai!



> ohai

```
{  
  "kernel": {  
    "os_info": {  
      "boot_device": "\Device\HarddiskVolume1",  
      "build_number": "9600",  
      "build_type": "Multiprocessor Free",  
      "caption": "Microsoft Windows Server 2012 R2 Standard",  
      "code_set": "1252",  
      "country_code": "1",  
      "creation_class_name": "Win32_OperatingSystem",  
      "cs_creation_class_name": "Win32_ComputerSystem",  
      "csd_version": null,  
      "cs_name": "WIN-DCK5NTVVLBH",  
    }  
  }  
}
```

Ohai is also a command-line application that is part of the Chef Development Kit (ChefDK).

Running Ohai to Show the IP Address



```
> ohai platform
```

```
[  
  "windows"  
]
```

Running Ohai to Show the Hostname



```
> ohai hostname
```

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

Running Ohai to Show the Memory



```
> ohai memory
```

```
{
  "swap": {
    "total": "0kB",
    "free": "0kB"
  },
  "total": "604308kB",
  "free": "297940kB"
}
```

Running Ohai to Show the Total Memory



```
> ohai memory/total
```

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

Running Ohai to Show the CPU



```
> ohai cpu
```

```
{
  "0" : {
    "cores": 4,
    "vendor_id": "GenuineIntel",
    "family": 1,
    "model": "14857",
    "stepping": "9",
    "physical_id": "CPU0",
    "model_name": "Intel64 Family 6 Model 58 Stepping 9",
    "mhz": "3201"
  },
  "total": 4,
```

Running Ohai to Show the First CPU



```
> ohai cpu/0
```

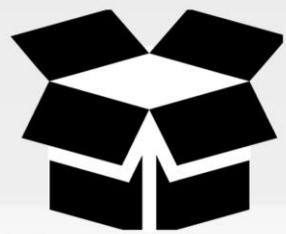
```
{  
  "cores": 4,  
  "vendor_id": "GenuineIntel",  
  "family": 1,  
  "model": "14857",  
  "stepping": "9",  
  "physical_id": "CPU0",  
  "model_name": "Intel64 Family 6 Model 58 Stepping 9",  
  "mhz": "3201"  
}
```

Running Ohai to Show the First CPU MHz



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

```
[  
  "3201"  
]
```



The Node Object

The node object is accessible within recipes as well.

Let's take a look at the syntax.

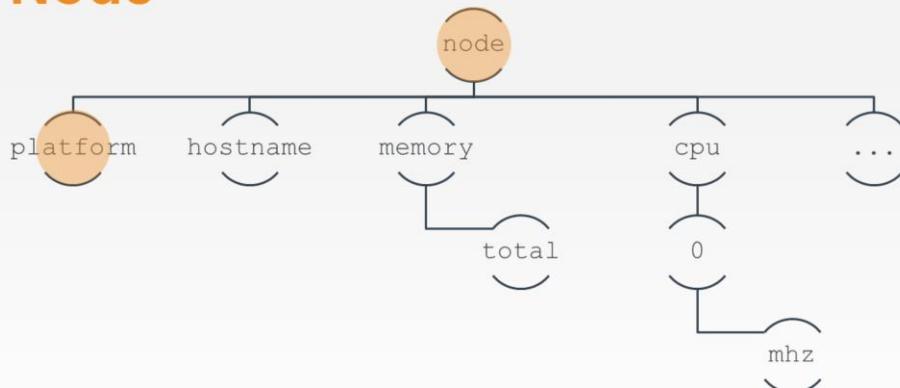
<http://docs.chef.io/nodes.html#attributes>

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.

The Node

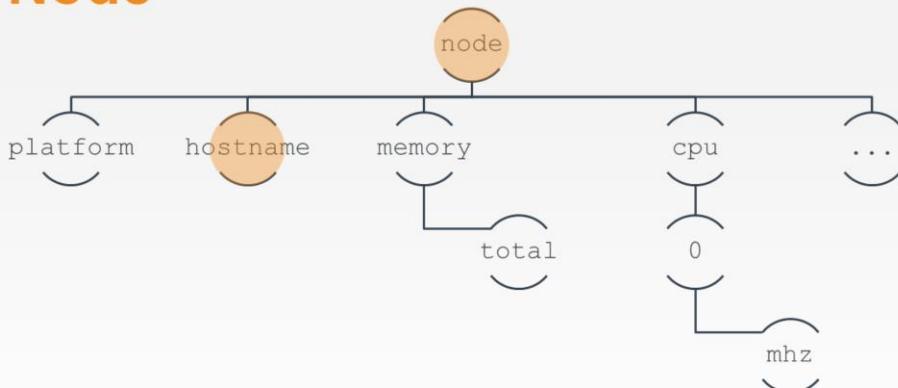


```
CLI: ohai platform
```

```
RECIPE: node['platform']
```

```
OUTPUT: windows
```

The Node



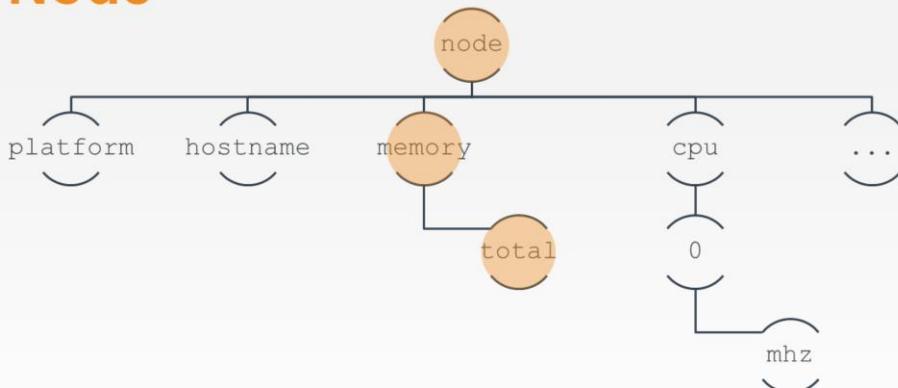
```
CLI: ohai hostname
```

```
RECIPE: node['hostname']
```

```
OUTPUT: WIN-KRQSVD3RFM7
```

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

The Node



```
CLI: ohai memory/total
```

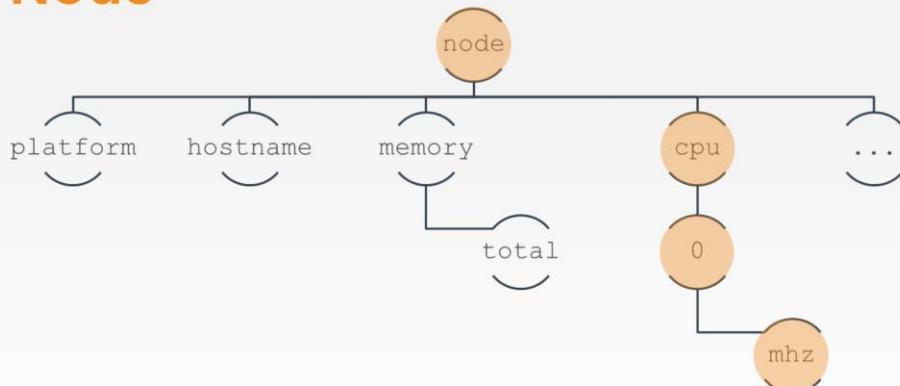
```
RECIPE: node['memory']['total']
```

```
OUTPUT: 7863920kB
```

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 the subset of keys and values at that level, and then immediately select to return the total value.

The Node

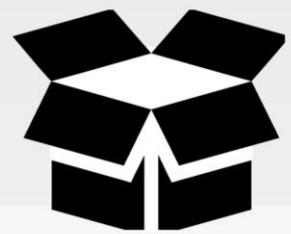


```
CLI: ohai cpu/0/mhz
```

```
RECIPE: node['cpu']['0']['mhz']
```

```
OUTPUT: 2900
```

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

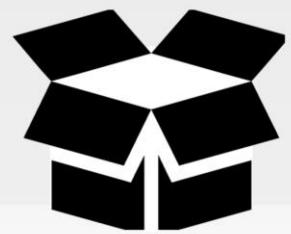


String Interpolation

```
I have 4 apples
```

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

http://en.wikipedia.org/wiki/String_interpolation#Ruby



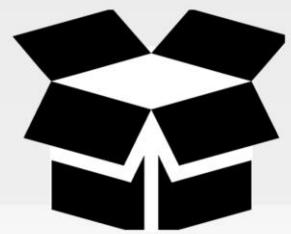
String Interpolation

```
I have 4 apples
```

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

http://en.wikipedia.org/wiki/String_interpolation#Ruby

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



String Interpolation

```
I have 4 apples
```

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



GL: Details About the Node

Displaying system details in the default web page definitely sounds useful.

Objective:

- ✓ Discover attributes about the system with Ohai
- Update the web page file contents, in the "myiis" cookbook, to include system details
- Update the cookbook's version number
- Apply the updated recipe and verify the results
- Commit the changes to the "myiis" cookbook to version control

GL: Details About the Node

~\cookbooks\myiis\recipe\server.rb

```
# ... POWERSHELL_SCRIPT RESOURCE ...

file 'c:\inetpub\wwwroot\Default.htm' do
  content "<h1>Hello, world!</h1>
<h2>PLATFORM: #{node['platform']}</h2>
<h2>HOSTNAME: #{node['hostname']}</h2>
<h2>MEMORY:   #{node['memory']['total']}</h2>
<h2>CPU Mhz:  #{node['cpu'][0]['mhz']}</h2>""
end

# ... SERVICE RESOURCE ...
```

Update the content attribute of the file resource to include the node details. Remember to include the details about the node in the content string we need to use string interpolation. String interpolation requires that the string be a double-quoted string. Ensure you change the single-quotes to double quotes. Then use `#{}` to escape out of the double-quoted string. Between the curly braces you can define the node object and specify which attribute you would like to display.



GL: Details About the Node

Displaying system details in the default web page definitely sounds useful.

Objective:

- Discover attributes about the system with Ohai
- Update the web page file contents, in the "myiis" cookbook, to include system details
- Update the cookbook's version number
- Apply the updated recipe and verify the results
- Commit the changes to the "myiis" cookbook to version control



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?

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. That sounds like a Minor changes based on the definitions provided by the Semantic Versioning documentation.

GL: Update the Cookbook Version

```
~\cookbooks\myiis\metadata.rb
```

```
name          'myiis'
maintainer    'The Authors'
maintainer_email 'you@example.com'
license        'all_rights'
description    'Installs/Configures iis'
long_description 'Installs/Configures iis'
version        '0.2.0'
```

Edit the "myiis" cookbook's metadata file. Find the line that specifies the version and increase the minor value by one.



GL: Details About the Node

Displaying system details in the default web page definitely sounds useful.

Objective:

- ✓ Discover attributes about the system with Ohai
- ✓ Update the web page file contents, in the "myiis" cookbook, to include system details
- ✓ Update the cookbook's version number
- ❑ Apply the updated recipe and verify the results
- ❑ Commit the changes to the "myiis" cookbook to version control

Return Home and Apply iis Cookbook



```
> cd ~  
> chef-client --local-mode -r "recipe[myiis]"
```

```
[2014-12-23T22:45:48+00:00] WARN: No config file found or specified on command  
line, using command line options.  
Starting Chef Client, version 12.13.37  
resolving cookbooks for run list: ["myiis"]  
Synchronizing Cookbooks:  
- myiis (0.2.0)  
Compiling Cookbooks...  
Converging 3 resources  
Recipe: myiis::server  
* powershell_script[Install IIS] action run  
- execute "C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe" -NoLogo  
-NonInteractive -NoProfile -ExecutionPolicy Bypass -InputFormat None -File
```

©2017 Chef Software Inc.

6-39



If everything looks good, then we want to use `chef-client`. `chef-client` is not run on a specific cookbook--it is a tool that allows us to apply recipes for multiple cookbooks that are stored within a cookbooks directory.

1. So we need to return home to the parent directory of all our cookbooks.
2. Then use `chef-client` to locally apply the run list defined as: the 'iis' cookbook's default recipe.

Verify the Default Page Returns the Details



```
> Invoke-WebRequest localhost
```

```
StatusCode      : 200
StatusDescription : OK
Content         : <h1>Hello, world!</h1>
                  <h2>PLATFORM: windows</h2>
                  <h2>HOSTNAME: WIN-8694LT97S51</h2>
                  <h2>MEMORY: 8388208kB</h2>
                  <h2>CPU Mhz: 2400</h2>
RawContent      : HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Length: 137
```

Verify that the Default page is returning the new updated content with the details about the system.

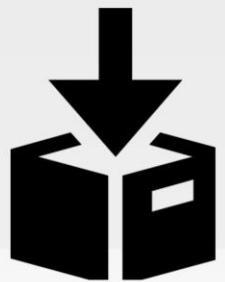


GL: Details About the Node

Displaying system details in the default web page definitely sounds useful.

Objective:

- ✓ Discover attributes about the system with Ohai
- ✓ Update the web page file contents, in the "myiis" cookbook, to include system details
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results
- Commit the changes to the "myiis" cookbook to version control



GL: Commit Your Work

```
> cd ~\cookbooks\myiis  
> git add .  
> git status  
> git commit -m "Version 0.2.0, display system info within  
webpage content"  
> git push
```

The last thing to do is commit our changes to source control. Change into the directory, add all the changed files, and then commit them.



GL: Details About the Node

Displaying system details in the default web page definitely sounds useful.

Objective:

- ✓ Discover attributes about the system with Ohai
- ✓ Update the web page file contents, in the "myiis" cookbook, to include system details
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results
- ✓ Commit the changes to the "myiis" cookbook to version control



Discussion

What is the node object and when is this generated?

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

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

Answer these questions.

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



Q&A

What questions can we help you answer?

- Ohai
- Node Object
- Node Attributes
- String Interpolation

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.



CHEFTM

©2017 Chef Software Inc.

Templates

Desired State vs Data



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.



Cleaner Recipes

In the last section we updated our ‘myiis’ web server cookbook to display information about our node.

We added this content to the file resource in the server recipe.

We were successful in displaying the details about the system instead of using hard-coded values. We added that content to the content attribute of the file resource that generates the Default page for the webserver.

Demo: The 'myiis' Server Recipe

~/cookbooks/myiis/recipes/server.rb

```
powershell script 'Install IIS' do
  code 'Add-WindowsFeature Web-Server'
end

file 'c:\inetpub\wwwroot\Default.htm' do
  content "<h1>Hello, world!</h1>
<h2>PLATFORM: #{node['platform']}</h2>
<h2>HOSTNAME: #{node['hostname']}</h2>
<h2>MEMORY:  #{node['memory']['total']}</h2>
<h2>CPU Mhz: #{node['cpu'][0]['mhz']}</h2>"
```

```
service 'w3svc' do
  action [ :enable, :start ]
end
```

What if new changes are given to us for the website splash page?

For each new addition we would need to return to this recipe and carefully paste the contents of the new HTML into the string value of the content attribute.

Let's talk about some ways in which this method of keeping the content within the recipe will be problematic as we make more changes.

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.

If any of the text that we paste into this content field has double quotes it is going to have to be escaped. This can possibly become a problem if we wanted to add some additional style or design to our splash page.



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.

Here is the fixed version where all the double-quotes within the string are escaped to ensure that Ruby will read this as a single string of characters.



Backslash

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

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



That also brings up an issue with the backslash character. This is particularly important on Windows where the file separator is often represented with a backslash character. You will need to keep an eye out for backslash characters because backslash characters are now the escape character in a double-quoted string.

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



Backslash

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

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

So every time text you use a double-quoted string, you will need to find and replace all backslashes with double-backslashes and then replace all double-quotes with backslash double-quotes.

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

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.

Instructor Note: The Message of the Day file is not a white-space important file. Other configuration files that could be managed with Chef may be white-space important.



Manual Editing

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

So if you are copying and pasting large amounts of text content into a double-quoted string it will be important to check that every time.

This sounds 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.



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.



GL: Cleaner Recipes

Adding the node attributes to our recipes did make it harder to read.

Objective:

- Decide which resource will help us address this issue

To solve this problem, we need to read up on the file resource more or see if Chef provides alternatives.



GL: Let's Check the Docs...

Use the file resource to manage files directly on a node.

Use the **cookbook_file** resource to copy a file from a cookbook's `/files` directory. Use the **template** resource to create a file based on a template in a cookbook's `/templates` directory. And use the **remote_file** resource to transfer a file to a node from a remote location.

https://docs.chef.io/resource_file.html

Let's start from what we know--the file resource. Open the documentation and see what it says and see if it gives us a clue to finding alternatives.

The file resource documentation suggests a couple of alternatives to using the file resource: cookbook_file resource; template resource; and remote_file resource.

Let's start with the remote_file resource.



GL: `remote_file`

Use the `remote_file` resource to transfer a file from a remote location using file specificity. This resource is similar to the `file` resource.

https://docs.chef.io/resource_remote_file.html

Reading the documentation for `remote_file`, it seems that `remote_file` is similar to `file`. Except `remote_file` is used to specify a file at a remote location that is copied to a specified file path on the system.

So we could define our index file or message-of-the-day file on a remote system. But that does not allow us to insert attributes about the node we are currently on.



GL: `cookbook_file`

Use the `cookbook_file` resource to transfer files from a sub-directory of COOKBOOK_NAME/files/ to a specified path located on a host that is running the chef-client.

https://docs.chef.io/resource_cookbook_file.html

Reading the documentation for `cookbook_file`, after the boiler-plate resource definition, it sounds as though a cookbook file is capable of...

Demo: cookbook_file's Source Match Up

```
> tree /f cookbooks\myiis\files\default
Folder PATH listing
Volume serial number is B04A-119C
C:\users\Administrator\cookbooks\myiis\files\default
    Default.htm
No subfolders exist

cookbook_file 'C:\inetpub\wwwroot\Default.htm' do
    source 'Default.htm'
end
```

...allowing us to store a file within our cookbook and then have that file transferred to a specified file path on the system.

While it sounds like it allows us to write a file in its native format, it does not sound as though the ability exists to escape out to access the node object and dynamically populate data.



Template

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

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

Let's explore templates.

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

Demo: Template File's Source Matches Up

```
> tree /f cookbooks\myiis\templates\default
Folder PATH listing
Volume serial number is B04A-119C
C:\USERS\ADMINISTRATOR\COOKBOOKS\MYIIS\TEMPLATES
    Default.htm.erb
No subfolders exist
```

```
template 'C:\inetpub\wwwroot\Default.htm' do
  source 'Default.htm.erb'
end
```

A template can be placed in a particular directory within the cookbook and it will be delivered to a specified file path on the system.

The biggest difference is that it says templates can contain ruby expressions and statements. This sounds like what we wanted: A native file format with the ability to insert information about our node.



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

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.



GL: Cleaner myiis Recipe

Adding the node attributes to the index page did make it harder to read the recipe.

Objective:

- Create a template with chef generate
- Define the contents of the ERB template
- Change the file resource to the template resource in the 'myiis' cookbook

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.

GL: What Can chef generate Do?



```
> 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
policyfile	Generate a Policyfile for use with the install/push commands (experimental)

And let's ask for help about the 'generate' subcommand.

GL: What Can chef generate template Do?



```
> 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, 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
      --generator-cookbook
```

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.

GL: Use chef to Generate a Template



```
> cd ~  
> chef generate template cookbooks\myiis Default.htm
```

```
Recipe: code_generator::template  
  * directory[.\cookbooks/myiis/templates/default] action create  
    - create new directory .\cookbooks/myiis/templates/default  
  * template[.\cookbooks/myiis/templates/Default.htm.erb] action create  
    - create new file .\cookbooks/myiis/templates/Default.htm.erb  
    - update content in file .\cookbooks/myiis/templates/Default.htm.erb from  
      none to e3b0c4  
      (diff output suppressed by config)
```

Use '**chef generate template**' to create a template in the myiis cookbook found in the cookbooks\myiis directory and the file we want to create is named Default.htm.

GL: Lets Look at the Template File



```
> tree /f cookbooks\myiis\templates
cookbooks/myiis/templates/
|   Default.htm.erb
|
└── default
```

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



Cleaner Recipes

Adding the node attributes to the default page did make it harder to read the recipe.

Objective:

- Create a template with chef generate
- Define the contents of the ERB template
- Change the file resource to the template resource in the 'myiis' cookbook

Now we need to understand what ERB means.



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>

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.

Text Within an ERB Template

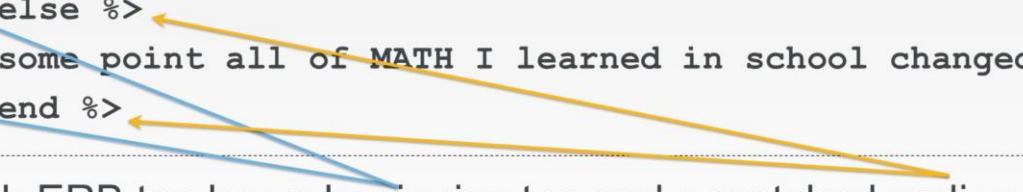
```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```

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

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

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.

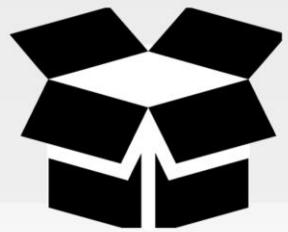
These tags are used to execute ruby but the results are not displayed.

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.



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.

GL: Move Our Source to the Template

```
~\cookbooks\myiis\templates\Default.htm.erb
```

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>PLATFORM: #{node['platform']}</h2>
    <h2>HOSTNAME: #{node['hostname']}</h2>
    <h2>MEMORY:   #{node['memory']['total']}</h2>
    <h2>CPU Mhz:  #{node['cpu'][0]['mhz']}</h2>
  </body>
</html>
```

With that in mind let's update the template with the current value of the file resource's content field.

Copying this literally into the file does not work because we no longer have the ability to use string interpolation within this html file. String interpolation only works within a ruby file between a double-quoted String.

GL: Replace String Interpolation with ERB

```
~\cookbooks\myiis\templates\Default.htm.erb
```

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>PLATFORM: <%= node['platform'] %></h2>
    <h2>HOSTNAME: <%= node['hostname'] %></h2>
    <h2>MEMORY:   <%= node['memory']['total'] %></h2>
    <h2>CPU Mhz: <%= node['cpu']['0']['mhz'] %></h2>
  </body>
</html>
```

We are going to need to change string interpolation sequence with the ERB template syntax. And it seems for this content we want to display the output so we want to make sure that we are using ERB's angry squid opening tag.



Cleaner Recipes

Adding the node attributes to the default page did make it harder to read the recipe.

Objective:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ❑ Change the file resource to the template resource in the 'myiis' cookbook

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

GL: Remove the Existing Content Attribute

```
~\cookbooks\myiis\recipes\server.rb

# ... POWERSHELL_SCRIPT RESOURCE ...

file 'c:\inetpub\wwwroot\Default.htm' do
  content "<h1>Hello, world!</h1>
<h2>ipaddress: #{node['ipaddress']}</h2>
<h2>hostname: #{node['hostname']}</h2>
<h2>total memory: #{node['memory']['total']}</h2>
<h2>CPU Mhz: #{node['cpu'][0]['mhz']}</h2>"
end
# ... SERVICE RESOURCE ...
```

Let's open the myiis cookbook's recipe named 'server'.

We will want to remove the content attribute from the file resource. Because that content is now in the template. But only if we use a template resource.

GL: Change the File Resource to a Template

```
~\cookbooks\myiis\recipes\server.rb
```

```
# ... POWERSHELL_SCRIPT RESOURCE ...
```

```
template 'c:\inetpub\wwwroot\Default.htm' do
```

```
end
```

```
# ... SERVICE RESOURCE ...
```

So it's time to change the file resource to a template resource so that it can use the template file that we have defined.

GL: Change the File Resource to a Template

```
~\cookbooks\myiis\recipes\server.rb

template 'c:\inetpub\wwwroot\Default.htm' do
  source 'Default.htm.erb'
end
```

Now we have the path to our template so we can update the template resource's source attribute value.



Cleaner Recipes

Adding the node attributes to the default page did make it harder to read the recipe.

Objective:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource in the 'myiis' cookbook

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



Lab: Apply the Changes

- Use chef-client to apply the "myiis" cookbook's "default" recipe
- Update the "myiis" cookbook's version for this patch
- Commit and push the changes

In this lab, you will use 'kitchen' to verify the cookbook and use 'chef-client' to apply the cookbook. If everything is working then update the patch number and commit the changes to version control.

Instructor Note: Allow 8 minutes to complete this exercise.

Lab: Change Directories and Apply the Cookbook



```
> cd ~  
> chef-client --local-mode -r "recipe[myiis]"
```

```
[2017-08-30T19:48:15+00:00] WARN: No config file found or specified on command line,  
using command line options.  
Starting Chef Client, version 12.13.37  
resolving cookbooks for run list: ["myiis"]  
Synchronizing Cookbooks:  
  - myiis (0.2.0)  
Installing Cookbook Gems:  
Compiling Cookbooks...  
Converging 3 resources  
Recipe: myiis::server  
  * powershell_script[Install IIS] action run  
    - execute "C:\Windows\system32\WindowsPowerShell\v1.0\powershell.exe" -NoLogo -  
NonInteractive -NoProfile -ExecutionPolicy Bypass -InputFormat None -File ...
```

When all the tests pass, return to the home directory, so you can execute 'chef-client'.

And then apply the myiis cookbook's default recipe to the local system.

Lab: Update the Cookbook's Patch Number

```
~\cookbooks\myiis\metadata.rb
```

```
name          'myiis'
maintainer    'The Authors'
maintainer_email 'you@example.com'
license        'all_rights'
description    'Installs/Configures myiis'
long_description 'Installs/Configures myiis'
version        '0.2.1'
```

If everything converges correctly, update the version number. As mentioned previously, this is a patch fix.

Lab: Commit the Changes



```
> cd ~\cookbooks\myiis  
> git add .  
> git status  
> git commit -m "Version 0.2.1, utilize a template for  
webpage content"  
> git push
```

Return to the cookbook directory and add all the changed files and commit them with a message.



Lab: Apply the Changes

- ✓ Use chef-client to apply the "myiis" cookbook's "default" recipe
- ✓ Update the "myiis" cookbook's version for this patch
- ✓ Commit and push the changes

Great you now have the myiis cookbook updated to a new version with a template.



Discussion

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

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

What do each of the ERB tags accomplish?

Answer these questions.

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



Q&A

What questions can we help you answer?

- Resources (file, cookbook_file, template, and remote_file)
- Templates
- ERB

What questions can we help you answer?

Generally or specifically about resources, templates, and ERB.



CHEFTM

©2017 Chef Software Inc.

Lab

Building an Apache Webserver Cookbook

©2017 Chef Software Inc.

8-1



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



Group Lab: Pre-built Linux Node

We will provide for you a Linux node with all the tools installed.

Objective:

- Login to the Linux Node

Login 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.

Instructor Note: You should assign the participants their Day 2 'apache_web' virtual machines at this time. The login credentials and password for the virtual workstations is chef/Cod3Can!



Choose an Editor

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

In your 'participant guide', found below, you can find tips for using these editors:

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!
```



Group Lab: Pre-built Workstation

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

Objective:

- ✓ Login to the Remote Workstation

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



Setting up an Apache Web Server

- Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- Create a '**server**' recipe that defines the following policy:
 - o The package named 'httpd' is installed.
 - o The template named '/var/www/html/index.html' is created with the source 'index.html.erb'
 - o The service named 'httpd' is both started and enabled.
- Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- Use the `include_recipe` method to include the '**server**' recipe within the '**default**' recipe
- Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
- Verify the site is available by running `curl localhost`
- Create GitHub repo for '**apache**' and commit/push changes with git

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.

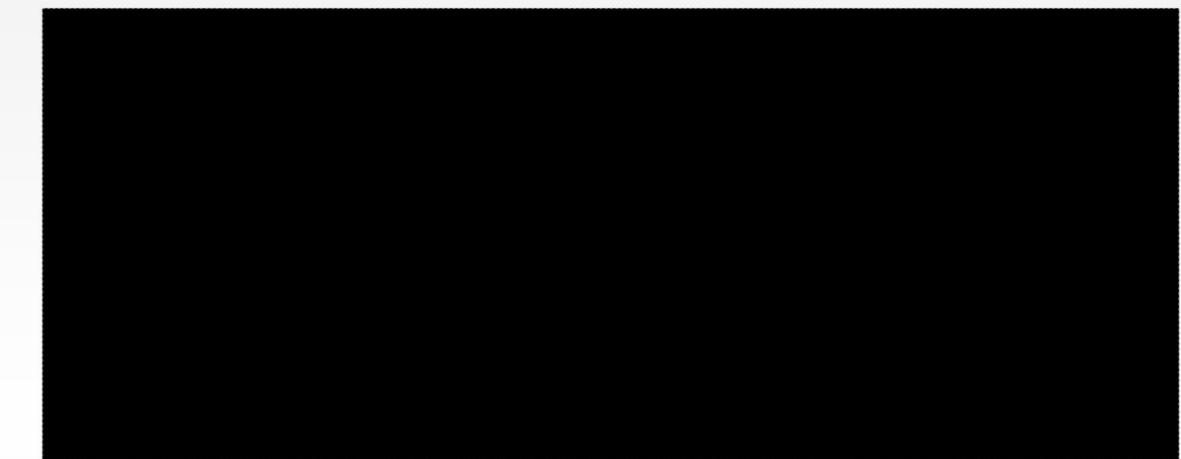
The lab's activity here is to challenge you to employ all that you have done already but in a new context. When everyone has completed the activity we will review the commands and code in the following slides.

Instructor Note: Allow 30 minutes to complete this exercise.

Working within Home Directory



```
$ cd ~
```



Before we get started let's return to the home directory.

GL: Create a Cookbooks Directory



```
$ 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.

Creating the apache Cookbook



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

Generating cookbook apache

- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master

Your cookbook is ready. Type `cd cookbooks/apache` to enter it.



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ❑ Create a '**server**' recipe that 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 both started and enabled.
- ❑ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ❑ Use the `include_recipe` method to include the '**server**' recipe within the '**default**' recipe
- ❑ Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
- ❑ Verify the site is available by running `curl localhost`
- ❑ Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.

Creating the server Recipe



```
> chef generate recipe cookbooks/apache server

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
  e5ca2c
```

Defining the Policy in the server Recipe

```
~/cookbooks/apache/recipes/server.rb
```

```
#  
# Cookbook Name:: apache  
# Recipe:: server  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
package 'httpd'  
  
template '/var/www/html/index.html' do  
  source 'index.html.erb'  
end  
  
service 'httpd' do  
  action [:enable, :start]  
end
```



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ✓ Create a '**server**' recipe that 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 both started and enabled.
- ❑ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ❑ Use the `include_recipe` method to include the '**server**' recipe within the '**default**' recipe
- ❑ Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
- ❑ Verify the site is available by running `curl localhost`
- ❑ Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.

Creating the html Template



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

Recipe: code_generator::template
  * directory[cookbooks/apache/templates/default] action create
    - create new directory cookbooks/apache/templates/default
  * template[cookbooks/apache/templates/index.html.erb] action create
    - create new file cookbooks/apache/templates/index.html.erb
    - update content in file
  cookbooks/apache/templates/index.html.erb from none to e3b0c4
  (diff output suppressed by config)
```

Defining the index.html Template

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

```
<html>
  <body>
    <h1>Welcome Home!</h1>
    <h2>PLATFORM: <%= node['platform'] %></h2>
    <h2>HOSTNAME: <%= node['hostname'] %></h2>
    <h2>MEMORY:   <%= node['memory']['total'] %></h2>
    <h2>CPU Mhz:  <%= node['cpu'][0]['mhz'] %></h2>
  </body>
</html>
```



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ✓ Create a '**server**' recipe that 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 both started and enabled.
- ✓ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ❑ Use the `include_recipe` method to include the '**server**' recipe within the '**default**' recipe
- ❑ Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
- ❑ Verify the site is available by running `curl localhost`
- ❑ Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.

Including the server Recipe

```
~/cookbooks/apache/recipes/default.rb
```

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
#  
# Copyright (c) 2016 The Authors, All Rights Reserved.  
  
include_recipe 'apache::server'
```



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ✓ Create a '**server**' recipe that 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 both started and enabled.
- ✓ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ✓ Use the `include_recipe` method to include the '**server**' recipe within the '**default**' recipe
 - Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
 - Verify the site is available by running `curl localhost`
 - Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.

Applying the apache Cookbook's default Recipe



```
> sudo chef-client --local-mode --runlist "recipe[apache]"  
Starting Chef Client, version 13.2.20  
resolving cookbooks for run list: ["apache"]  
Synchronizing Cookbooks:  
  - apache (0.1.0)  
Installing Cookbook Gems:  
Compiling Cookbooks...  
Converging 3 resources  
Recipe: apache::server  
  * yum_package[httpd] action install  
    - install version 2.2.15-60.el6.centos.6 of package httpd
```



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ✓ Create a '**server**' recipe that 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 both started and enabled.
- ✓ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ✓ Use the include_recipe method to include the '**server**' recipe within the '**default**' recipe
- ✓ Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
 - Verify the site is available by running `curl localhost`
 - Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.

Verifying the Default Website is Available



```
> curl localhost
```

```
<html>
  <body>
    <h1>Welcome Home!</h1>
    <h2>PLATFORM: centos</h2>
    <h2>HOSTNAME: ip-172-31-29-209</h2>
    <h2>MEMORY: 604192kB</h2>
    <h2>CPU Mhz: 1800.000</h2>
  </body>
</html>
```



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ✓ Create a '**server**' recipe that 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 both started and enabled.
- ✓ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ✓ Use the include_recipe method to include the '**server**' recipe within the '**default**' recipe
- ✓ Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
- ✓ Verify the site is available by running **curl localhost**
- Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.

Create apache GitHub Repo and copy url

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: SamMcB7 ✓ Repository name: apache 

Great repository names are short and memorable. Need inspiration? How about super-duper-octo-waffle.

Description (optional): This is the 'apache' cookbook for the Chef DevOps Foundation class

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None

Code Issues Pull requests Projects Wiki Insights Settings

Quick setup — if you've done this kind of thing before
Set up in Desktop or HTTPS SSH https://github.com/SamMcB7/apache.git 

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# apache" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin https://github.com/SamMcB7/apache.git  
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/SamMcB7/apache.git  
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Use 'git config' to set configuration variables



```
> git config --global user.email "you@example.com"
```

The command-line tool chef will automatically configure your cookbook as a Git repository. For this to work correctly you will need to setup your email address and name in the global git configuration.

Set up your email you want to associate with your git commits.

Use 'git config' to set configuration variables



```
> git config --global user.name "Username"
```

The command-line tool chef will automatically configure your cookbook as a Git repository. For this to work correctly you will need to setup your email address and name in the global git configuration.

Set up your user name you want to associate with your git commits.

Move into the apache Directory



```
> cd cookbooks/apache
```

Change into the workstation cookbook directory.

Move to Cookbook Directory and Initialize as a git Repository



```
> git init
```

```
Reinitialized existing Git repository in /home/chef/cookbooks/apache/.git/
```

We want git to start tracking the entire contents of this folder and any content in the subfolders. To do that with git, you need to execute the command 'git init' in the parent directory of the cookbook that you want to start tracking.

You will notice that git will say that the repository has been 'Reinitialized'. This is because the chef cookbook generator detected that we have git installed and automatically initialized the cookbook as a git repository.

Add Work to Staging Area



```
> git add .
```

Use 'git commit' to Save the Staged Changes



```
> git commit -m "Initial Commit"
```

```
On branch master
5 files changed, 66 insertions(+)
 create mode 100644 recipes/server.rb
 create mode 100644 spec/unit/recipes/server_spec.rb
 create mode 100644 templates/index.html.erb
 create mode 100644 test/smoke/default/server_test.rb
```

If everything that is staged looks correct, then we are ready to commit the changes.

Add remote for apache Cookbook



```
> git remote add origin https://github.com/username/apache.git
```

Push apache Cookbook to Repo



```
> git push -u origin master  
Username for 'https://github.com': username  
Password for 'https://username@github.com':  
Counting objects: 73, done.  
Compressing objects: 100% (54/54), done.  
Writing objects: 100% (73/73), 11.07 KiB | 0 bytes/s, done.  
Total 73 (delta 18), reused 0 (delta 0)  
remote: Resolving deltas: 100% (18/18), done.  
To https://github.com/SamMc87/apache.git  
 * [new branch]      master -> master  
Branch master set up to track remote branch master from origin.
```

Verify Push in a Browser

The screenshot shows a GitHub repository page for 'SamMcB7 / apache'. The repository is described as 'This is the "apache" cookbook for the Chef DevOps Foundation class'. It has 5 commits, 1 branch, 0 releases, and 0 contributors. The commits are listed as follows:

File	Description	Time Ago
.delivery	Add generated delivery build cookbook	an hour ago
.recipes	Initial Commit	15 minutes ago
.spec	Initial Commit	15 minutes ago
.templates	Initial Commit	15 minutes ago
.test/smoke/default	Initial Commit	15 minutes ago
.gitignore	Add generated cookbook content	an hour ago
.kitchen.yml	Add generated cookbook content	an hour ago
Berkfile	Add generated cookbook content	an hour ago
README.md	Add generated cookbook content	an hour ago
.chefignore	Add generated cookbook content	an hour ago
metadata.rb	Add generated cookbook content	an hour ago
README.md		

Below the commits, there is a section titled 'apache' with a 'TODO' note: 'TODO: Enter the cookbook description here.'



Setting up an Apache Web Server

- ✓ Create a '**cookbooks**' directory and generate a cookbook named '**apache**'
- ✓ Create a '**server**' recipe that 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 both started and enabled.
- ✓ Create a template named '**index.html.erb**' and populate it with a welcome message, the node's platform, hostname, total memory, and CPU speed.
- ✓ Use the `include_recipe` method to include the '**server**' recipe within the '**default**' recipe
- ✓ Use '**sudo chef-client**' to locally apply the apache cookbook's default recipe
- ✓ Verify the site is available by running `curl localhost`
- ✓ Create GitHub repo for '**apache**' and commit/push changes with git

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.

Instructor Note: Allow 30 minutes to complete this exercise.



Q&A

What questions can we help you answer?

What questions can we help you answer?



CHEFTM

©2017 Chef Software Inc.

Workstation Installation

Configuring Your Laptop as a Workstation

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.



EXERCISE

Installing the ChefDK

Installing the tools on your system

Objective:

- Install the ChefDK
- Install Git
- Open Terminal / Command Prompt
- Execute a series of commands to ensure everything is installed
- 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.



CHEF 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>.

Instructor Note: Prior to attending this course they may have received correspondence that informed them to setup the ChefDK on their systems. It is possible that they did not and this slides acts as a good reminder to ensure that they have the necessary tools before continuing on to the next section. **IMPORTANT:** This course was tested against ChefDK version 0.17.17. If you use a different version, the exercises and labs may not work properly.



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.

Installing ChefDK



©2017 Chef Software Inc.

9-6



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.



CONCEPT Download Git

Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

<https://git-scm.com/downloads>

Installing Git

The screenshot shows the official Git website's "Downloads" page. At the top, there's a search bar labeled "Search entire site...". On the left, a sidebar includes links for "About", "Documentation", "Blog", "Downloads" (which is currently selected), and "Community". Below the sidebar, there's a note about the "Pro Git book" available online for free. The main content area has three sections: "Downloads" (with links for Mac OS X, Windows, and Linux/Unix), "GUI Clients" (with a note about built-in tools like git-gui and gitk), and "Logos" (with a link to view various Git logos). A large image of a Mac computer monitor displaying the latest source release version 2.15.0 is prominently featured.

Verify installation of ChefDK



```
> chef --version
```

```
Chef Development Kit Version: 2.0.28
chef-client version: 13.2.20
delivery version: master (17c1b0fed9be4c70f69091a6d21a4cbf0df60a23)
berks version: 6.2.0
kitchen version: 1.16.0
inspec version: 1.31.1
```

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.

Verify installation of Git



```
> git --version
```

```
git version 2.14.1 (Apple Git-94)
```



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.



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>

Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running and version control. - <https://code.visualstudio.com>

Sublime Text

A sophisticated text editor for code, markup and prose - <https://www.sublimetext.com/>

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.



EXERCISE

Installing the ChefDK

Installing the tools on your system

Objective:

- ✓ Install the ChefDK
- ✓ Execute a series of commands to ensure everything is installed
- ✓ 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.



Q&A

What questions can we answer for you?



CHEF TM

©2017 Chef Software Inc.

Signing Up For Managed Chef

Get to Know the Benefits

Objectives

After completing this module, you should be able to

- Utilize a Manage Chef Account and create a new organization
- 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.



Flavors of Chef Server

**Open Source
Chef Server**

**Chef Server
(Support +
Premium
Features)**

**Multi-tenant
Hosted Chef Server**

At the core we offer Chef Server as an open source project freely available for anyone to deploy. We offer support and additional premium features. Lastly, we have Hosted Chef Server, which is a multi-tenant Chef Server that you host as a service. This by far is the quickest way to get started with and is free as long as you remain under the reasonable node amount.



EXERCISE

Hosted Chef

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

Objective:

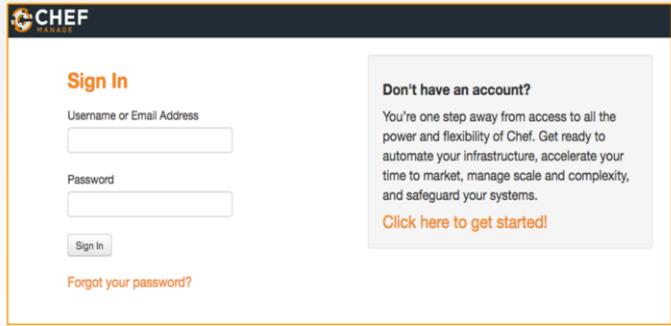
- Create a Hosted Chef Account

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

Signing Up for a Hosted Chef Account

Steps

1. Navigate to
<https://manage.chef.io>
2. Fill out the form as indicated in this image using your name and a valid email address and then click **Get Started**.



The screenshot shows the 'Sign In' page for the Chef Server. At the top left is the 'CHEF SERVER' logo. Below it, the word 'Sign In' is centered above two input fields: 'Username or Email Address' and 'Password'. To the right of these fields is a button labeled 'Sign In'. Below the password field is a link 'Forgot your password?'. On the right side of the page, there is a sidebar with the heading 'Don't have an account?'. It contains a brief description: '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.' Below this text is a blue button labeled 'Click here to get started!'. The entire page has a light gray background with orange borders around the main form area and the sidebar.

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

Signing Up for a Hosted Chef Account

Steps

1. Navigate to <https://manage.chef.io>
2. 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	<input type="text" value="Your Name"/>
Company	<input type="text" value="Your Company"/>
Email	<input type="text" value="youremail@gmail.com"/>
Username	<input type="text" value="your_username"/>
<input type="checkbox"/> I'm not a robot  <small>reCAPTCHA Privacy - Terms</small>	
<input checked="" type="checkbox"/> I agree to the Terms of Service and the Master License and Services Agreement . Get Started	

Instructor Note: Learners that already have an account will login instead of creating an account. The learner's account may be tied to an production organization. The learner can create a new organization with their existing account. If there are still concerns they can create a new login with a unique email address.

Signing Up for a Hosted Chef Account

Steps

3. From the resulting page, click the [Create New Organization](#) button.
4. Fill out the resulting Create Organization form and then click Create Organization.
5. From the resulting page, click your new organization to highlight it and then click Starter Kit.
6. From the resulting window, click the Download Starter Kit button.

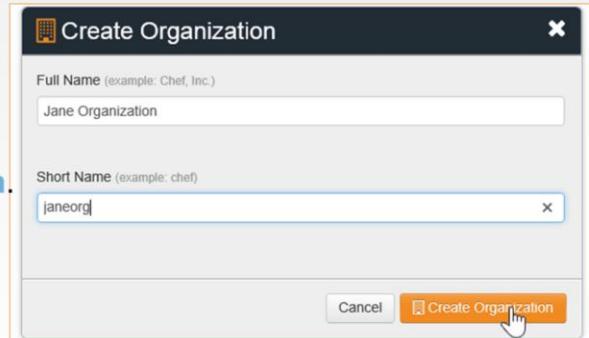


Instructor Note: Learners that already have an account will already have an organization. They are welcome to use that organization if it does not have any cookbooks or nodes from previous exercises or from their production systems. It is often easier to have them create a new organization for the purposes of this training. That can be done through the website.

Signing Up for a Hosted Chef Account

Steps

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



The screenshot shows a modal dialog titled "Create Organization". It has two input fields: "Full Name (example: Chef, Inc.)" with the value "Jane Organization" and "Short Name (example: chef)" with the value "janeorg". At the bottom right are two buttons: "Cancel" and "Create Organization", with a hand cursor icon hovering over "Create Organization".

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.

Signing Up for a Hosted Chef Account

Steps

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



Instructor Note: By far the easiest way to get setup with Managed Chef Server is download the Starter Kit. The most important pieces of the starter kit are found in a hidden directory (".`chef`") which contains the organization key, user key, and organization configuration file.

Signing Up for a Hosted Chef Account

Steps

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

The screenshot shows the Chef Manage web interface. In the top navigation bar, there are links for 'Nodes', 'Reports', and 'Policy'. Below the navigation, there's a sidebar with options like 'Create', 'Reset Validation Key', 'Generate Knife Config', 'Invite User', 'Leave Organization', and 'Starter Kit'. The main content area displays a message: 'Thank you for choosing CHEF!' followed by 'Follow these steps to be on your way'. A prominent orange button labeled 'Download Starter Kit' is highlighted with a mouse cursor. To the right of the button is a link 'What's next?'. Further down, there are links for 'Chef Documentation' and 'Browse CHEF cookbooks'. At the bottom of the page, there's a footer note: 'The best place to start learning about Chef in general...'. Below the main content is a modal dialog box with a black header containing an orange exclamation mark icon and the text 'Are you certain?'. The body of the dialog says 'Your user and organization keys will be reset. Are you sure you want to do this?'. At the bottom of the dialog are two buttons: 'Cancel' and a red 'Proceed' button, which also has a mouse cursor pointing at it.

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.

Instructor Note: If the learner already has an account and an organization tied to that account this will reset their personal key and organizational key. This means that the other chef repository that they were previously maintaining will no longer be able to communicate with the Chef Server.

Signing Up for a Hosted Chef Account

Steps

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



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

Instructor Note: The reason that spaces are not suggested is that Ruby tools have a hard time with file paths that contain spaces.



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.



Q&A

What questions can we answer for you?



CHEFTM

©2017 Chef Software Inc.

The Chef Server

A Hub for Configuration Data



Objectives

After completing this module, you should be able to

- Connect your local workstation (laptop) to a Chef Server
- Clone a GitHub repository
- 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.



More Web Servers?

More easily manage multiple nodes

Objective:

- Utilize a Chef Server via a Hosted Chef Account
- Clone the 'myiis' and 'apache' cookbooks
- Upload your cookbooks to the Hosted Chef Server
- Add a new node as a managed node

Currently, your cookbook exists on one webserver. If you wanted to setup additional web servers to serve additional traffic for your soon-to-be highly successful website, what steps would you need to take to setup an identical system?



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 or myiis cookbook.
4. Run chef-client on the new node to apply the apache or myiis 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.

Instructor Note: This exercise is to show the value of using a Chef Server with regard to managing multiple systems. It can be done with the group, with individuals, or done in pairs.



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.

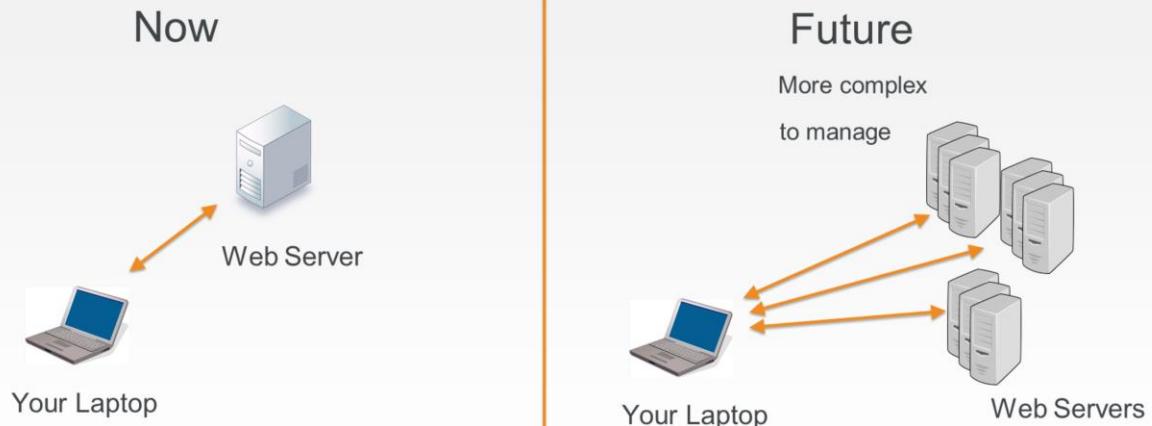
Chef provides a one-line curl install for the Chef Development Kit (ChefDK).

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.



Managing Additional Systems



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.

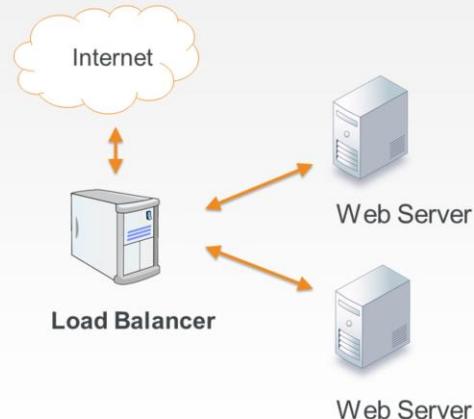
Some may think 10 minutes is not so bad. But what if there were 10 new nodes? 20 new nodes?

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.



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.

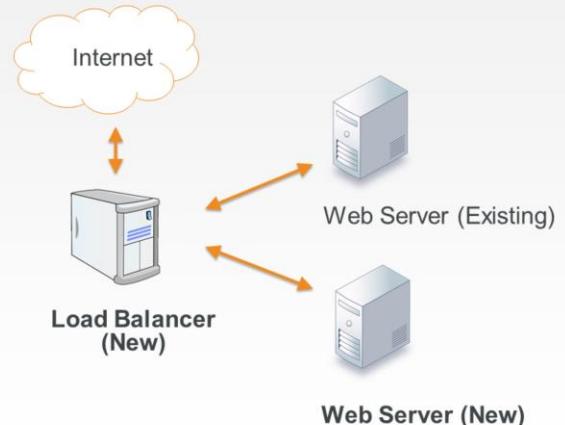
In addition to the complexities of configuring and managing multi-server infrastructure, such as web servers, you also need to develop a way to route incoming traffic to each of those web servers and other nodes. There are many ways that you can route the traffic from one node to a group of similar nodes. 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.

Instructor Note: The preceding three slides covered the complexity of configuring and managing multi-server infrastructure. This slide and page is now talking about managing user web traffic, via a load balancer.



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.

Steps to Setup 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

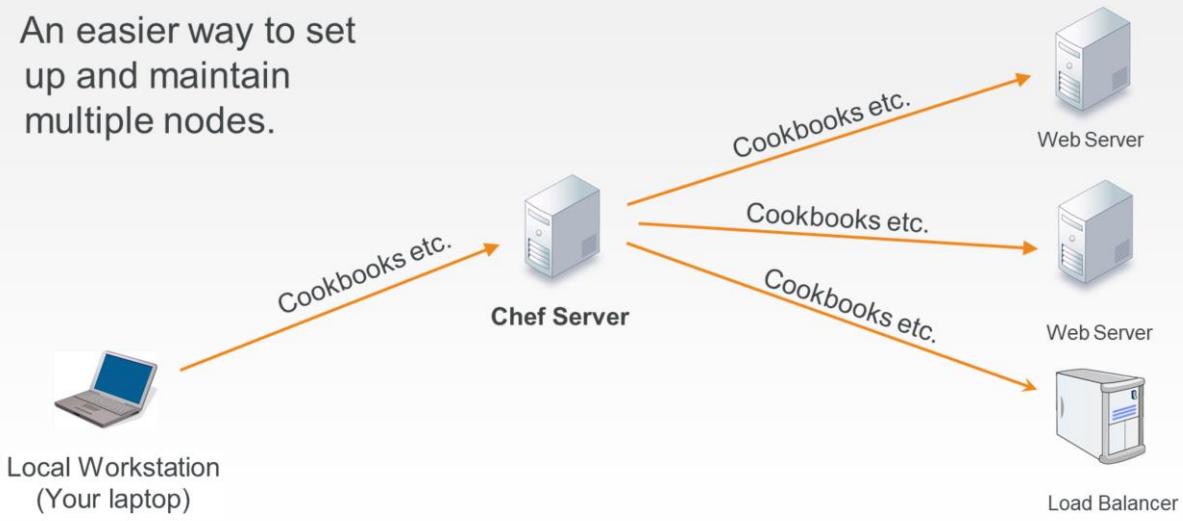
Whether you tackle installing, configuring, or running a load balancer or recreate a second instance running the apache cookbook's default recipe, you will need to solve the problem of how you can manage multiple systems. Each system would need to have Chef installed, the cookbooks copied onto each system,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

Instructor Note: The left side is setting up a new web server that is like the one they created yesterday. The right is new work that the learner will be accomplishing today. Note that the haproxy is actually a load balancer.

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.



GL: 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:

- Clone the 'myiis' cookbook
- Upload the 'myiis' cookbook to the Hosted Chef Server



Git Clone

At this point we have GitHub repositories containing our ‘myiis’ and ‘apache’ cookbooks. What if we want to work with them from another location or on a new machine?

With ‘git clone’ we can download a copy of an existing Git Repository and work from there.

<https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository>



GL: Code Repository

You may wish to clone your own repositories or you may download repositories containing a similar copy of the work you did previously in this course which can be found here:

<https://github.com/SamMc87/myiis>

<https://github.com/SamMc87/apache>

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.

Instructor Note: A learner may want to use the exact copy of the cookbooks that they developed. You may need to coordinate with the learners on using git or other methods to retrieve those cookbooks from those remote workstations.

Navigate to the cookbooks Directory



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

```
chefignore    starter
```

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.

Remove the starter Example Cookbook



```
$ rm -rf starter
```

The 'starter' cookbook is a generic example of a cookbook and is not necessary to keep for class.

GL: Copy the URL of the 'myiis' GitHub Repo

This is the 'myiis' cookbook for the Chef DevOps Foundation class

Add topics

7 commits · 1 branch · 0 releases · 1 contributor

Branch: master · New pull request · Create new file · Upload files · Find file · Clone or download

Clone with HTTPS Use SSH
https://github.com/SamMcB87/myiis.git

File	Description	Last Commit
.delivery	Add generated delivery build cookbook	2 days ago
recipes	0.2.0 added system details	2 days ago
spec	This is the initial commit of files to our repo	2 days ago
test/smoke/default	This is the initial commit of files to our repo	2 days ago
.gitignore	Add generated cookbook content	2 days ago
.kitchen.yml	Add generated cookbook content	2 days ago
Berkfile	Add generated cookbook content	2 days ago
README.md	Add generated cookbook content	2 days ago
chefignore	Add generated cookbook content	2 days ago
metadata.rb	Version 0.1.1, include server recipe within default	23 hours ago

©2017 Chef Software Inc.

11-16



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

Clone the myiis GitHub Repo



```
$ git clone https://github.com/username/myiis.git
```

```
Cloning into 'myiis'...
remote: Counting objects: 79, done.
remote: Compressing objects: 100% (40/40), done.
remote: Total 79 (delta 21), reused 77 (delta 19), pack-reused 0
Unpacking objects: 100% (79/79), done.
```

Navigate to the myiis Directory



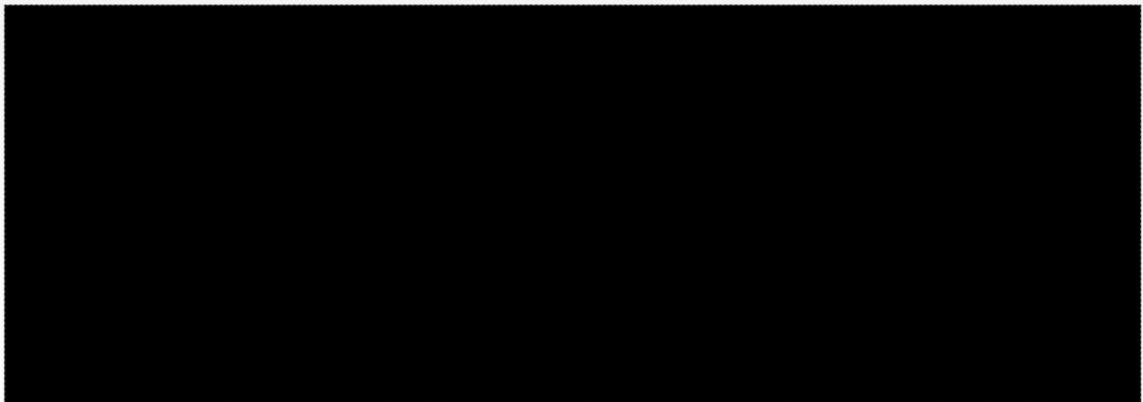
```
$ cd ~/chef-repo/cookbooks/myiis  
$ ls -l (or dir if using Powershell)
```

```
-rw-r--r-- 1 technotrainer staff    47 Oct 25 22:03 Berksfile  
-rw-r--r-- 1 technotrainer staff    53 Oct 25 22:03 README.md  
-rw-r--r-- 1 technotrainer staff  1133 Oct 25 22:03 chefignore  
-rw-r--r-- 1 technotrainer staff   740 Oct 25 22:03 metadata.rb  
drwxr-xr-x 4 technotrainer staff   136 Oct 25 22:03 recipes  
drwxr-xr-x 4 technotrainer staff   136 Oct 25 22:03 spec  
drwxr-xr-x 3 technotrainer staff   102 Oct 25 22:03 test
```

Use ‘git config’ to set configuration variables



```
$ git config --global user.email "you@example.com"  
$ git config --global user.name "Username"
```



The command-line tool chef will automatically configure your cookbook as a Git repository. For this to work correctly you will need to setup your email address and name in the global git configuration. Note that it is not necessary to use the --global flag if only wanting to set it for the single repo.

Set up your email you want to associate with your git commits.

Instructor note: students do not need to configure git globally if they do not want to, leaving out the --global flag will only configure the variables for that single repository.



GL: 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:

- Clone the 'myiis' cookbook
- Upload the 'myiis' cookbook to the Hosted Chef Server



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.

knife --help



```
$ 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)
knife client edit 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'.

knife cookbook --help



```
$ 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.

knife cookbook list



```
$ knife cookbook list
```



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.

Change to the cookbooks/myiis Directory

```
 $ cd cookbooks/myiis
```



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



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.

<http://berkshelf.com>

Run berks --help



```
$ berks --help
```

Commands:

```
berks apply ENVIRONMENT      # Apply version locks from Berksfile.lock to a ...
berks contingent COOKBOOK    # List all cookbooks that depend on the given c...
berks cookbook NAME [PATH]   # Create a skeleton for a new cookbook
berks help [COMMAND]         # Describe available commands or one specific c...
berks info [COOKBOOK]        # Display name, author, copyright, and dependen...
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 specifi...
berks outdated [COOKBOOKS]   # List dependencies that have new versions avai...
berks package [PATH]          # Vendor and archive the dependencies of a Berk...
berks search NAME            # Search the remote source for cookbooks matchi...
berks shelf SUBCOMMAND       # Interact with the cookbook store
berks show [COOKBOOK]         # Display the path to a cookbook on disk
```

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

To see full command descriptions you can include the berks sub commands. For example: `berks apply ENVIRONMENT --help`

Run berks install



```
$ berks install
```

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

See the Berksfile.lock



```
$ ls -al (or dir -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'.

See the Contents of the Berksfile.lock

 \$ cat Berksfile.lock

```
DEPENDENCIES
apache
  path: .
  metadata: true

GRAPH
myiis (0.2.1)
```

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

Upload the Cookbook to the Chef Server



```
$ berks upload
```

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

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 myiis cookbook to the Chef Server.

Display Cookbooks within Your Org



```
$ knife cookbook list
```

```
myiis      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.

Note: You can also use the -a flag to see all versions of a cookbook. For example, if you had multiple versions of a cookbook:

```
$ knife cookbook list -a
```

GL: 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:

- ✓ Clone the 'myiis' cookbook
- ✓ Upload the 'myiis' cookbook to the Hosted Chef Server



Lab: Clone and Upload the Apache Cookbook

Let's get the cookbook on our local machine and the Chef Server

Objective:

- Clone the 'apache' cookbook from the GitHub repo into the 'cookbooks' directory
- Upload the 'apache' cookbook to the chef server
- Verify the cookbook is uploaded

As a lab, upload the remaining cookbooks within the cookbooks directory. After you have done that verify that the cookbooks have been uploaded.

Instructor Note: Allow 5 minutes to complete this exercise.

Navigate to the cookbooks Directory

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

```
chefignore    myiis
```

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.

GL: Copy the URL of the ‘apache’ GitHub Repo

This is the 'apache' cookbook for the Chef DevOps Foundation class

Branch: master • New pull request

Clone with HTTPS ⓘ Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/SamMc87/apache.git>

Open in Desktop Download ZIP

File	Description	Time Ago
.delivery	Add generated delivery build cookbook	7 hours ago
recipes	Initial Commit	7 hours ago
spec	Initial Commit	8 hours ago
templates	Initial Commit	8 hours ago
test/smoke/default	Initial Commit	8 hours ago
.gitignore	Add generated cookbook content	8 hours ago
.kitchen.yml	Add generated cookbook content	8 hours ago
Berksfile	Add generated cookbook content	8 hours ago
README.md	Add generated cookbook content	8 hours ago
.chefignore	Add generated cookbook content	8 hours ago
metadata.rb	Add generated cookbook content	8 hours ago

©2017 Chef Software Inc.

11-36



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

Clone the apache GitHub Repo



```
$ git clone https://github.com/username/apache.git
```

```
Cloning into 'apache'...
remote: Counting objects: 73, done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 73 (delta 18), reused 73 (delta 18), pack-reused 0
Unpacking objects: 100% (73/73), done.
```



Lab: Clone and Upload the Apache Cookbook

Let's get the cookbook on our local machine and the Chef Server

Objective:

- ✓ Clone the 'apache' cookbook from the GitHub repo into the 'cookbooks' directory
- ❑ Upload the 'apache' cookbook to the chef server
- ❑ Verify the cookbook is uploaded

As a lab, upload the remaining cookbooks within the cookbooks directory. After you have done that verify that the cookbooks have been uploaded.

Instructor Note: Allow 5 minutes to complete this exercise.

Lab: Navigate to the apache Directory



```
$ cd ~/chef-repo/cookbooks/apache  
$ ls -l
```

```
-rw-r--r-- 1 technotrainer staff    47 Oct 25 22:03 Berksfile  
-rw-r--r-- 1 technotrainer staff    53 Oct 25 22:03 README.md  
-rw-r--r-- 1 technotrainer staff 1133 Oct 25 22:03 chefignore  
-rw-r--r-- 1 technotrainer staff   740 Oct 25 22:03 metadata.rb  
drwxr-xr-x 4 technotrainer staff   136 Oct 25 22:03 recipes  
drwxr-xr-x 4 technotrainer staff   136 Oct 25 22:03 spec  
drwxr-xr-x 3 technotrainer staff   102 Oct 25 22:03 test
```

Lab: Install the Cookbook Dependencies



```
$ berks install
```

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

Run "berks install" to install all the cookbook dependencies.

Lab: Upload the Cookbook to the Chef Server



```
$ berks upload
```

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

Run "berks upload" to upload the cookbook and all its dependencies to the Chef Server.



Lab: Clone and Upload the Apache Cookbook

Now let's make sure that all worked.

Objective:

- ✓ Clone the 'apache' cookbook from the GitHub repo into the 'cookbooks' directory
- ✓ Set the git variables for your username and email
- ✓ Upload the 'apache' cookbook to the chef server
- Verify the cookbook is uploaded

As a lab, upload the remaining cookbooks within the cookbooks directory. After you have done that verify that the cookbooks have been uploaded.

Instructor Note: Allow 5 minutes to complete this exercise.

Lab: Is the workstation Cookbook Uploaded?

 \$ knife cookbook list

```
apache      0.1.0
myiis      0.2.1
```

Lastly, run "knife cookbook list" to validate that the apache cookbook is now uploaded to the Chef Server.



Lab: Clone and Upload the Apache Cookbook

Good work. Let's move on to understand some key concepts.

Objective:

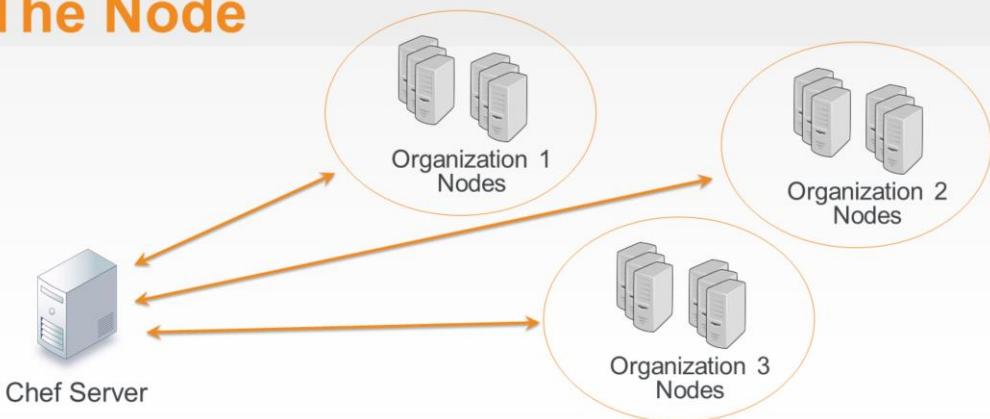
- ✓ Clone the 'apache' cookbook from the GitHub repo into the 'cookbooks' directory
- ✓ Set the git variables for your username and email
- ✓ Upload the 'apache' cookbook to the chef server
- ✓ Verify the cookbook is uploaded

As a lab, upload the remaining cookbooks within the cookbooks directory. After you have done that verify that the cookbooks have been uploaded.

Instructor Note: Allow 5 minutes to complete this exercise.



The Node



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.



GL: Bootstrap Your Node

In this lab you will use a new instance and bootstrap it as a managed node.

You'll need the FQDN or Public IP of that instance to perform this lab.



Bootstrapping a Node

Often, the node you are bootstrapping 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 a new instance as a node within our organization. Often times, the node you are bootstrapping 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.

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

Change to the chef-repo



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



Return to the root of the chef repository.

Run 'knife node –help'



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

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

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.

Run 'knife node list'



```
$ 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.

Run 'knife bootstrap –help'



```
$ 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.

Bootstrap Your Node



```
> knife bootstrap windows winrm IP -x USER -P PWD -N iis_web
```

```
Creating new client for node1
Creating new node for node1
Connecting to node1 via winrm...
Fully Qualified Domain Name or IP: ec2-54-175-46-24.compute-1.amazonaws.com
ws... user name: Chef
password: [REDACTED]
node name: node1; run.

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:
```

©2017 Chef Software Inc.

11-52



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.

Run 'knife node list' Again



```
$ knife node list
```

```
iis_web
```

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.

View More Information About Your Node



```
$ knife node show iis_web
```

```
Node Name:    iis_web
Environment:  _default
FQDN:        ip-172-31-8-68.ec2.internal
IP:          54.175.46.24
Run List:
Roles:
Recipes:
Platform:    windows 6.3.9200
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.

Add a Recipe to a Run List



```
$ knife node run_list add iis_web "recipe[myiis]"
```

```
iis_web:  
  run_list: recipe[myiis]
```

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.



WinRM Woes

Logging into 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.

Running a Command with Knife



```
$ knife winrm "IP" -m -x USERNAME -P PASSWORD "chef-client"
```

```
54.159.197.193 Starting Chef Client, version 13.6.0
54.159.197.193
54.159.197.193 [2017-10-30T23:33:23+00:00] INFO: *** Chef 13.6.0 ***
54.159.197.193 [2017-10-30T23:33:23+00:00] INFO: Platform: x64-mingw32
54.159.197.193 [2017-10-30T23:33:23+00:00] INFO: Chef-client pid: 3016
54.159.197.193 [2017-10-30T23:33:23+00:00] INFO: The plugin path
C:\chef\ohai\plugins does not exist. Skipping...
54.159.197.193 [2017-10-30T23:33:27+00:00] INFO: Run List is [recipe[myiis]]
54.159.197.193 [2017-10-30T23:33:27+00:00] INFO: Run List expands to [myiis]
54.159.197.193 [2017-10-30T23:33:27+00:00] INFO: Starting Chef Run for iis_web
54.159.197.193 [2017-10-30T23:33:27+00:00] INFO: Running start handlers
54.159.197.193 [2017-10-30T23:33:27+00:00] INFO: Start handlers complete.
```

©2017 Chef Software Inc.

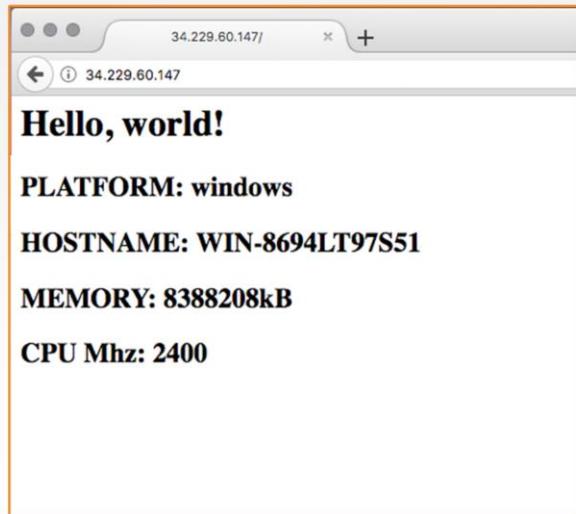
11-57



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

Verify 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.



More Web Servers?

More easily manage multiple nodes

Objective:

- ✓ Utilize a Chef Server via a Hosted Chef Account
- ✓ Clone the 'myiis' and 'apache' cookbooks
- ✓ Upload your cookbooks to the Hosted Chef Server
- ✓ Add a new node as a managed node

Currently, your cookbook exists on one webserver. If you wanted to setup additional web servers to serve additional traffic for your soon-to-be highly successful website, what steps would you need to take to setup an identical system?



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.

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



Q&A

What questions can you help you answer?

- Chef Server
- Managed Chef
- Git Clone
- Berkshelf
- Bootstrapping Nodes

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



CHEFTM

©2017 Chef Software Inc.

Cookbook Attributes, Attribute Files and Dependencies

Setting Attributes within a Cookbook

Your goal in this module is to bootstrap another node, this time a web server, and add it to the proxy members.

Objectives

After completing this module, you should be able to

- Explain where cookbook attributes reside
- Create a wrapper cookbook
- Configure dependencies between cookbooks



Attribute Files

The Node Object contains many automatic attributes generated by OHAI.

You can also maintain attributes within a cookbook.

These are like variables or parameters for your cookbook and allow recipes to be data driven.

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



Best Practices

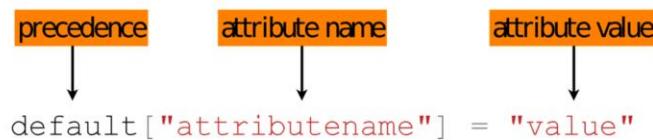
- ❖ Well-written cookbooks change behavior based on attributes.
- ❖ Ideally, you don't have to modify the contents of a cookbook to use it for your specific use case.
- ❖ Look at the attributes directory for things you can override through roles to affect behavior of the cookbook.
- ❖ Of course, well written cookbooks have sane defaults, and a README to describe all this.

Setting Attributes in Attribute Files

Cookbook attributes are set in the attributes file

```
./cookbooks/<cookbook>/attributes/default.rb
```

Format is:



We'll look at precedence later.

Example: Setting package name to an attribute

```
cookbooks/apache/attributes/default.rb
```

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

```
cookbooks/apache/recipes/default.rb
```

```
package node['apache']['package_name'] do
  action :install
end
```

We can set the name of a particular package to an attribute and then call that attribute within a recipe

Example: Setting package name to an attribute

```
cookbooks/apache/attributes/default.rb

case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
else
  default['apache']['package_name'] = 'httpd'
end
```

Implementing conditional statements allows us to alter the control flow permitting our cookbooks to be data driven.



Reconfigure Welcome Message

Currently a 'Hello world!' message is hard coded in both web server cookbooks.

What if we wanted to display a message that includes our company name utilizing a node attribute?

How could we implement this node attribute within both our 'myiis' and 'apache' cookbooks?



GL: Reconfigure Welcome Message

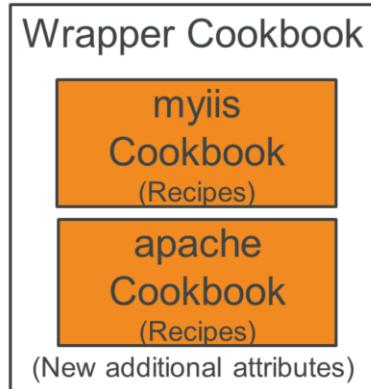
*So we want both our web server cookbooks to display
our company name...*

- Objective:**
- Create a 'company_web' wrapper cookbook that can apply either the 'myiis' or 'apache' default recipe based on platform
 - Create a node attribute that contains your company name
 - Implement the edit_resource method to update the template resource for both the 'myiis' and 'apache' server recipes
 - Upload cookbook to the Chef Server
 - Update the run list of the iis_web node to use the default recipe of the company_web cookbook and converge the node

Wrapper Cookbooks

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

It can access all of the recipes, cookbook components, and attributes found in the original cookbook(s) and implement them in new ways.



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

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

©2017 Chef Software Inc.

12-10



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 company_web.

Generate the Wrapper Cookbook



```
$ cd ~/chef-repo  
$ chef generate cookbook cookbooks/company_web
```

```
Generating cookbook company_web  
- Ensuring correct cookbook file content  
- Committing cookbook files to git  
- Ensuring delivery configuration  
- Ensuring correct delivery build cookbook content  
- Adding delivery configuration to feature branch  
- Adding build cookbook to feature branch  
- Merging delivery content feature branch to master
```

```
Your cookbook is ready. Type `cd cookbooks/company_web` to enter
```

GL: Create Dependency on apache and myiis

```
cookbooks/company_web/metadata.rb

name 'company_web'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'All Rights Reserved'
description 'Installs/Configures company_web'
long_description 'Installs/Configures company_web'
version '0.1.0'
chef_version '>= 12.1' if respond_to?(:chef_version)

depends 'myiis'
depends 'apache'
```



Configuring Berks

`berks install` retrieves dependencies from the Chef Supermarket by default.

We want to override this to use the local 'myiis' and 'apache' cookbooks instead of the community versions.

GL: Edit Berksfile

```
chef-repo/cookbooks/company_web/Berksfile

source 'https://supermarket.chef.io'

metadata

cookbook 'myiis', path: '../myiis'
cookbook 'apache', path: '../apache'
```

GL: Include Recipe Based on Platform

```
~/company_web/recipes/default.rb

# Cookbook:: company_web
# Recipe:: default
#
# Copyright:: 2017, The Authors, All Rights Reserved.

case node['platform']
when 'windows'
  include_recipe 'myiis::default'
else
  include_recipe 'apache::default'
end
```



GL: Reconfigure Welcome Message

*So we want both our web server cookbooks to display
our company name...*

- Objective:**
- ✓ Create a 'company_web' wrapper cookbook that can apply either the 'myiis' or 'apache' default recipe based on platform
 - Create a node attribute that contains your company name
 - Implement the edit_resource method to update the template resource for both the 'myiis' and 'apache' server recipes
 - Upload cookbook to the Chef Server
 - Update the run list of the iis_web node to use the default recipe of the company_web cookbook and converge the node

Generate the default Attribute File



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

```
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
```

Generate the default Attribute File



```
$ chef generate attribute --help
```

```
Usage: chef generate attribute [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
      -h, --help                         Show this message
```

Generate the default Attribute File



```
$ chef generate attribute cookbooks/company_web default
```

```
Recipe: code_generator::attribute
  * directory[cookbooks/company_web/attributes] action create
    - create new directory cookbooks/company_web/attributes
  * template[cookbooks/company_web/attributes/default.rb] action create
    - create new file cookbooks/company_web/attributes/default.rb
    - update content in file
  cookbooks/company_web/attributes/default.rb from none to e3b0c4
```

GL: Set the Company Name as an Attribute

```
cookbooks/company_web/attributes/default.rb
```

```
default['company_web']['company_name'] = 'Your Company Name'
```

It's good practice to include the name of the cookbook in the attribute name – this helps trace where the value is set, although it is not enforced.



GL: Reconfigure Welcome Message

*So we want both our web server cookbooks to display
our company name...*

- Objective:**
- ✓ Create a 'company_web' wrapper cookbook that can apply either the 'myiis' or 'apache' default recipe based on platform
 - ✓ Create a node attribute that contains your company name
 - ❑ Implement the edit_resource method to update the template resource for both the 'myiis' and 'apache' server recipes
 - ❑ Upload cookbook to the Chef Server
 - ❑ Update the run list of the iis_web node to use the default recipe of the company_web cookbook and converge the node



Using the company_name Attribute

We are now able to apply a different default recipe based on whether the node's platform is Windows or Centos, but how do we update the respective template file to display the company_name attribute for both the 'myiis' and 'apache' cookbooks?



edit_resource

A recipe can find a resource in the resource collection, and then edit it by using the `edit_resource` method. If a resource block with the same name exists in the resource collection, it will be updated with the contents of the resource block.

https://docs.chef.io/dsl_recipe.html#edit-resource

GL: View the server Recipes

~/myiis/recipes/server.rb

```
powershell_script 'Install IIS' do
  code 'Add-WindowsFeature Web-Server'
end

template 'c:\inetpub\wwwroot\Default.htm' do
  source 'Default.htm.erb'
end

service 'w3svc' do
  action [:enable, :start]
end
```

~/apache/recipes/server.rb

```
package 'httpd'

template '/var/www/html/index.html' do
  source 'index.html.erb'
end

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



We want to use a new source for the template resource for both our cookbooks

GL: Edit the Template resource for myiis

```
~/company_web/recipes/default.rb

case node['platform']
when 'windows'
  include_recipe 'myiis::default'

  edit_resource(:template, 'c:\inetpub\wwwroot\Default.htm') do
    source 'homepage.erb'
    cookbook 'company_web'
  end

#Else Statement...
```

GL: Edit the Template resource for apache

```
~/company_web/recipes/default.rb

#When Statement...
else

  include_recipe 'apache::default'

  edit_resource(:template, '/var/www/html/index.html') do
    source 'homepage.erb'
    cookbook 'company_web'
  end
end
```

GL: View the default Recipe

```
~/company_web/recipes/default.rb

case node['platform']
when 'windows'
  include_recipe 'myiis::default'

  edit_resource(:template, 'c:\inetpub\wwwroot\Default.htm') do
    source 'homepage.erb'
    cookbook 'company_web'
  end
else
  include_recipe 'apache::default'

  edit_resource(:template, '/var/www/html/index.html') do
    source 'homepage.erb'
    cookbook 'company_web'
  end
end
```

Generate the Template file



```
$ chef generate template cookbooks/company_web homepage
```

```
Recipe: code_generator::template
  * directory[/Users/technotrainer/chef-repo/cookbooks/company_web/templates]
action create
  - create new directory /Users/technotrainer/chef-
repo/cookbooks/company_web/templates
  * template[/Users/technotrainer/chef-
repo/cookbooks/company_web/templates/homepage.erb] action create
    - create new file /Users/technotrainer/chef-
repo/cookbooks/company_web/templates/homepage.erb
    - update content in file /Users/technotrainer/chef-
repo/cookbooks/company_web/templates/homepage.erb from none to e3b0c4
```

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

GL: Update the Template File

```
~/company_web/templates/homepage.erb
```

```
<html>
  <body>
    <h1><%=node['company_web']['company_name']%> Welcomes You!</h1>
    <h2>PLATFORM: <%= node['platform'] %></h2>
    <h2>HOSTNAME: <%= node['hostname'] %></h2>
    <h2>MEMORY: <%= node['memory']['total'] %></h2>
    <h2>CPU Mhz: <%= node['cpu'][0]['mhz'] %></h2>
  </body>
</html>
```



GL: Reconfigure Welcome Message

*So we want both our web server cookbooks to display
our company name...*

- Objective:**
- ✓ Create a 'company_web' wrapper cookbook that can apply either the 'myiis' or 'apache' default recipe based on platform
 - ✓ Create a node attribute that contains your company name
 - ✓ Implement the edit_resource method to update the template resource for both the 'myiis' and 'apache' server recipes
 - Upload cookbook to the Chef Server
 - Update the run list of the iis_web node to use the default recipe of the company_web cookbook and converge the node

Upload company_web to Chef Server



```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'apache' from source at ../apache
Fetching 'company_web' from source at .
Fetching 'myiis' from source at ../myiis
Fetching cookbook index from https://supermarket.chef.io...
Using apache (0.1.0) from source at ../apache
Using company_web (0.1.0) from source at .
Using myiis (0.2.1) from source at ../myiis
```

Upload company_web to Chef Server



```
$ berks upload
```

```
Skipping apache (0.1.0) (frozen)
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-
5.1.1/lib/ridley/client.rb:271:in `server_url': Ridley::Client#url_prefix at
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-
5.1.1/lib/ridley/client.rb:79 forwarding to private method
Celluloid::PoolManager#url_prefix
Uploaded company_web (0.1.0) to:
'https://api.chef.io:443/organizations/2017_oct_26'
Skipping myiis (0.2.1) (frozen)
```

List cookbooks found on Chef Server



```
$ knife cookbook list
```

apache	0.1.0
company_web	0.1.0
myiis	0.2.1



GL: Reconfigure Welcome Message

*So we want both our web server cookbooks to display
our company name...*

- Objective:**
- ✓ Create a 'company_web' wrapper cookbook that can apply either the 'myiis' or 'apache' default recipe based on platform
 - ✓ Create a node attribute that contains your company name
 - ✓ Implement the edit_resource method to update the template resource for both the 'myiis' and 'apache' server recipes
 - ✓ Upload cookbook to the Chef Server
 - Update the run list of the iis_web node to use the default recipe of the company_web cookbook and converge the node

Set Run List for iis_web Node



```
$ knife node run_list set iis_web 'recipe[company_web]'
```

```
iis_web:  
  run_list: recipe[company_web]
```

Verify Run List Update



```
$ knife node show iis_web
```

```
Node Name:    iis_web
Environment: _default
FQDN:        WIN-8694LT97S51.ec2.internal
IP:          34.229.225.40
Run List:    recipe[company_web]
Roles:
Recipes:     myiis::default, myiis::server
Platform:   windows 6.3.9600
Tags:
```

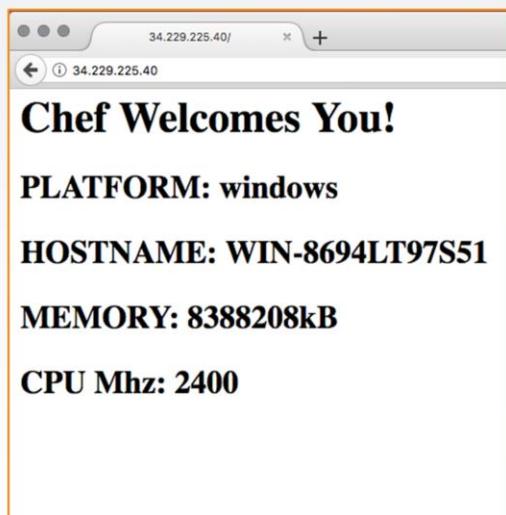
Converge the iis_web Node



```
$ cd ~/chef-repo  
$ knife winrm "IP" -m -x USERNAME -P PASSWORD "chef-client"
```

```
34.229.225.40 [2017-10-27T02:29:09+00:00] INFO: *** Chef 12.18.31 ***  
34.229.225.40 [2017-10-27T02:29:09+00:00] INFO: Platform: i386-mingw32  
34.229.225.40 [2017-10-27T02:29:09+00:00] INFO: Chef-client pid: 908  
34.229.225.40 [2017-10-27T02:29:44+00:00] INFO: Run List is  
[recipe[company_web]]  
34.229.225.40 [2017-10-27T02:29:44+00:00] INFO: Run List expands to  
[company_web]  
34.229.225.40 [2017-10-27T02:29:44+00:00] INFO: Starting Chef Run for iis_web  
34.229.225.40 [2017-10-27T02:29:44+00:00] INFO: Running start handlers  
34.229.225.40 [2017-10-27T02:29:44+00:00] INFO: Start handlers complete.
```

Verify that the Node Serves the New Page



Verify that the node serves up the default html page that contains the node's platform, hostname, memory, and CPU speed.



GL: Reconfigure Welcome Message

*So we want both our web server cookbooks to display
our company name...*

- Objective:**
- ✓ Create a 'company_web' wrapper cookbook that can apply either the 'myiis' or 'apache' default recipe based on platform
 - ✓ Create a node attribute that contains your company name
 - ✓ Implement the edit_resource method to update the template resource for both the 'myiis' and 'apache' server recipes
 - ✓ Upload cookbook to the Chef Server
 - ✓ Update the run list of the iis_web node to use the default recipe of the company_web cookbook and converge the node

Lab (optional): Commit company_web Cookbook to Version Control with Git



Objective:

- Create a GitHub repo for the "company_web" cookbook
- Commit, and push company_web to Github repository

Lab: Create company_web GitHub Repo and copy URL

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: SamMc87 / Repository name: company_web

Great repository names are short and memorable. Need inspiration? How about cuddly-octo-bassoon.

Description (optional): This is the company_web cookbook for the Chef DevOps Foundations class

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None

Create repository

SamMc87 / company_web

Code Issues Pull requests Projects Wiki Insights Settings

Quick setup — if you've done this kind of thing before
Set up in Desktop or HTTPS SSH https://github.com/SamMc87/company_web.git
We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# company_web" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SamMc87/company_web.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/SamMc87/company_web.git
git push -u origin master
```

...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

Lab (optional): Commit company_web Cookbook to Version Control with Git



Objective:

- Create a GitHub repo for the "company_web" cookbook
- Commit, and push company_web to Github repository

Lab: Move into the company_web Directory



```
$ cd cookbooks/company_web
```



Change into the company_web cookbook directory.

Lab: Move to Cookbook Directory and Initialize as a git Repository



```
$ git init
```

```
Reinitialized existing Git repository in /Users/technotrainer/chef-repo/cookbooks/company_web/.git/
```

We want git to start tracking the entire contents of this folder and any content in the subfolders. To do that with git, you need to execute the command 'git init' in the parent directory of the cookbook that you want to start tracking.

You will notice that git will say that the repository has been 'Reinitialized'. This is because the chef cookbook generator detected that we have git installed and automatically initialized the cookbook as a git repository.

Lab: Use 'git add .' to Add Modified Files to Staging



```
$ git add .
```

Lab: Use 'git commit' to Save the Staged Changes



```
$ git commit -m "Initial Commit"
```

```
[master f34c292] Initial Commit
 5 files changed, 34 insertions(+), 1 deletion(-)
  create mode 100644 attributes/default.rb
  create mode 100644 templates/homepage.erb
```

If everything that is staged looks correct, then we are ready to commit the changes.

Lab: Add remote for company_web Cookbook



```
$ git remote add origin https://github.com/username/company_web.git
```

Lab: Push company_web Cookbook to Repo



```
$ git push -u origin master
```

```
Counting objects: 68, done.  
Delta compression using up to 8 threads.  
Compressing objects: 100% (49/49), done.  
Writing objects: 100% (68/68), 10.88 KiB | 2.18 MiB/s, done.  
Total 68 (delta 18), reused 0 (delta 0)  
remote: Resolving deltas: 100% (18/18), done.  
To https://github.com/username/company_web.git  
 * [new branch]      master -> master  
Branch master set up to track remote branch master from origin.
```



Lab (optional): Commit company_web Cookbook to Version Control with Git

- ✓ Create a GitHub repo for the "company_web" cookbook
- ✓ Set git user variables, commit, and push company_web to Github repository



Discussion

Attributes are like parameters to your cookbook – no hard-coded values in recipes or templates. Can you think of some more parameters that you might want to create attributes for?

Can you imagine in complex topologies, where you could have multiple levels of dependencies between cookbooks?



Q&A

What questions can we answer for you?

Before we continue let us stop for a moment answer any questions that anyone might have at this time.



CHEFTM

©2017 Chef Software Inc.

Community Cookbooks

Find, Explore and View Chef Cookbooks

©2017 Chef Software Inc.

13-1



Objectives

After completing this module, you should be able to

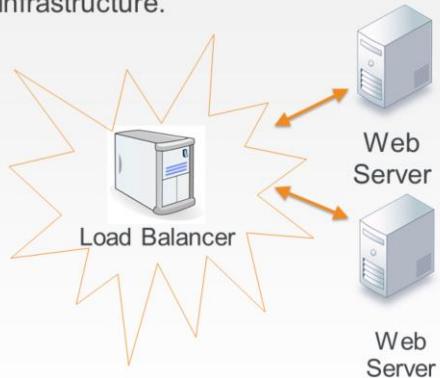
- Find cookbooks on the Chef Supermarket
- Create a wrapper cookbook for a community cookbook
- Replace the existing default values
- Upload a cookbook to the Chef Server
- Bootstrap a new node that runs the cookbook

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

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.

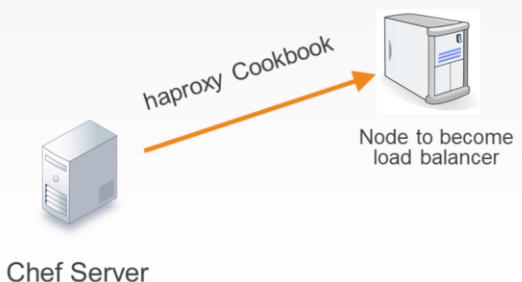
A load balancer is able to receive requests and relay them to other systems. In our case, we specifically want to use the load balancer to balance the entire traffic load between one or more systems.

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.

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.



Community Cookbooks

Someone already wrote that cookbook?

Available through the community site called the Chef Supermarket

<https://supermarket.chef.io>

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.



EXERCISE

GL: Load Balancer

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

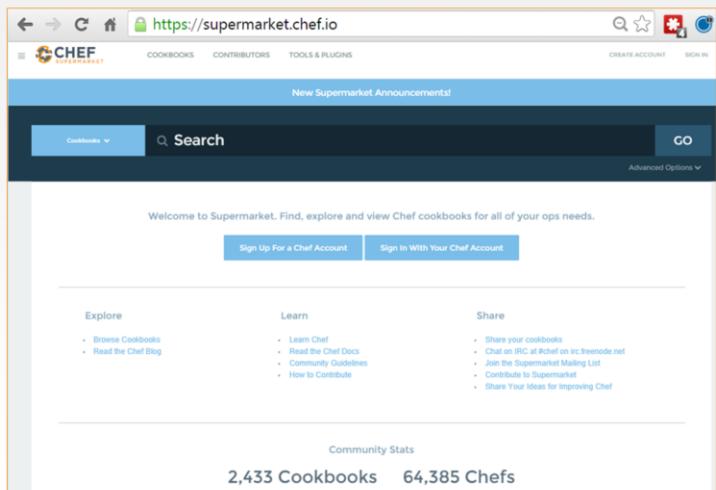
Objective:

- Find a Cookbook on the Chef Supermarket to Manage a load balancer
- Configure the load balancer to send traffic to the iis_web node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy (load balancer) cookbook
- Create GitHub repository and commit changes with Git (optional)

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

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.



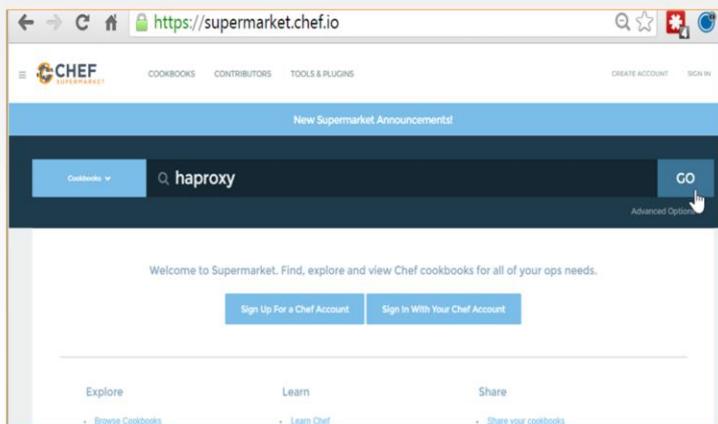
An important thing to remember is that on the community site are cookbooks managed by individuals. Chef does not verify or approve the cookbooks found in the Supermarket. These cookbooks solved problems for the original authors and then they decided to share them. This means that the cookbooks you find 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.

Searching in the Supermarket

STEPS

1. Visit 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.



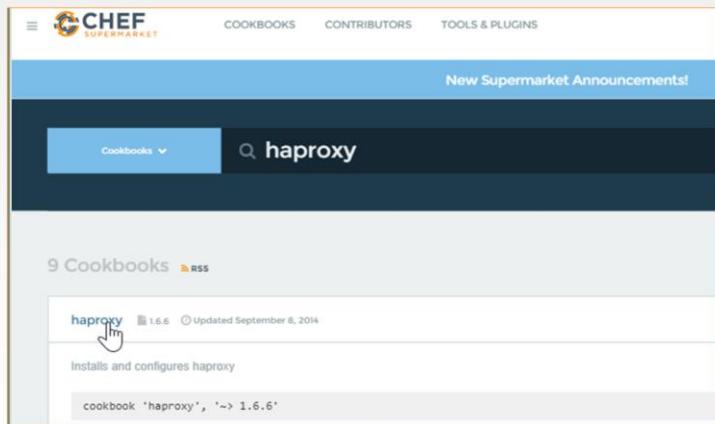
From the Supermarket main page type the search term "haproxy" and the click the GO button.

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

Searching in the Supermarket

STEPS

1. Visit 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.

Supermarket Cookbooks

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

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

The screenshot shows the Chef Supermarket page for the 'haproxy' cookbook. At the top, there's an RSS feed icon and a 'Follow' button. Below that, the title 'haproxy' is displayed with '(All) Versions 3.0.0'. A search bar contains the query 'cookbook \'haproxy\', \'= 3.0.0\''. To the right, there's a sidebar for 'sous-chefs' showing profile pictures of maintainers. Below the sidebar, sections include 'DETAILS', 'View Source', 'View Issues', 'UPDATED JANUARY 24, 2017', 'Created on October 25, 2009', 'PLATFORMS' (with icons), and 'LICENSE' (Apache 2.0). The main content area has tabs for 'README', 'Dependencies', 'Changelog', and 'Quality'. The 'README' tab is active, showing the title 'haproxy Cookbook' and a brief description: 'Installs haproxy and prepares the configuration location.' It also lists 'build passing' and 'cookbook v5.0.0'.

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.

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 a webpage for the 'haproxy Cookbook'. At the top, there are tabs for 'README', 'Dependencies', 'Changelog', and 'Quality'. Below the tabs, the title 'haproxy Cookbook' is displayed. Underneath the title, there is a green button labeled 'build passing' and a blue button labeled 'cookbook v5.0.0'. A brief description follows: 'Installs haproxy and prepares the configuration location.' Below this, two sections are listed: 'Requirements' and 'Platforms'. The 'Requirements' section includes a bullet point for 'Chef 12.1+'. The 'Platforms' section lists several supported operating systems: 'Ubuntu 12.04+', 'RHEL 6+, CentOS6+', 'RHEL 7+, CentOS7+', and 'Debian 8+'.

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 defined 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']` .

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  
}]
```

<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.



Using Community Cookbooks

Chef Community Cookbooks, can be used as is but in most cases you will want to use them as a foundation as you write your own.

Don't use forked community cookbooks in production, or you will miss out on upstream changes, and will have to rebase. Instead use **wrapper cookbooks**.

Supermarket Cookbooks

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

It can define new default values for the recipes.

Wrapper Cookbook

haproxy
Cookbook
(Attributes)

(New additional attributes)

<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'.



EXERCISE

GL: Load Balancer

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

Objective:

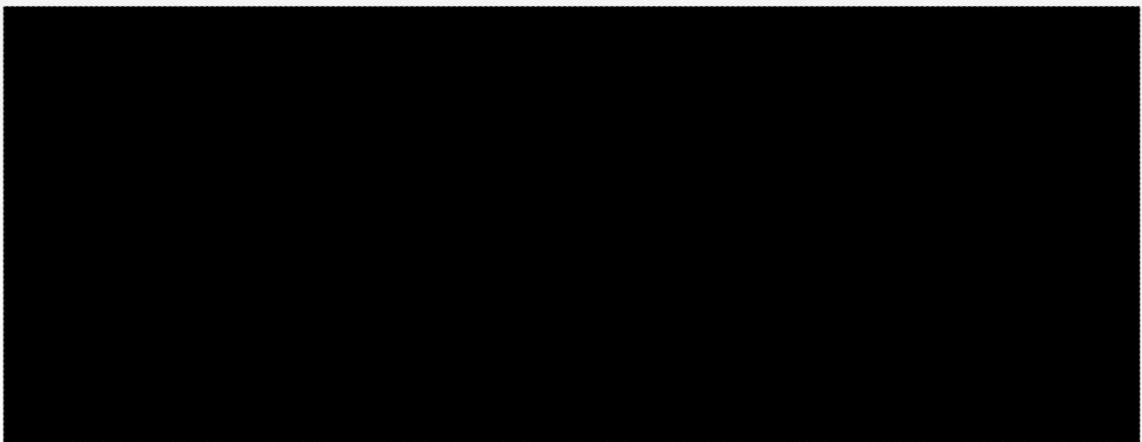
- ✓ Find a Cookbook on the Chef Supermarket to Manage a load balancer
- Configure the load balancer to send traffic to the iis_web node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy (load balancer) cookbook
- Create GitHub repository and commit changes with Git (optional)

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

Returning the Chef Repository Directory



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



Change to your chef-repo directory

Instructor Note: The generate cookbook command needs the path in the name of the cookbook here to ensure that it generates the cookbook in the cookbooks directory. Without it the cookbook will be generated within the root of the chef repository. If that happens simply have the learner move the newly generated cookbook into the cookbooks directory.

Generating a New Cookbook



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

```
Generating cookbook myhaproxy
- Ensuring correct cookbook file content
- Committing cookbook files to git
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content
- Adding delivery configuration to feature branch
- Adding build cookbook to feature branch
- Merging delivery content feature branch to master
```

```
Your cookbook is ready. Type `cd cookbooks/myhaproxy` to enter it.
```

```
There are several commands you can run to get started locally developing and testing
your cookbook.
```

Generate your new cookbook.

Instructor Note: The generate cookbook command needs the path in the name of the cookbook here to ensure that it generates the cookbook in the cookbooks directory. Without it the cookbook will be generated within the root of the chef repository. If that happens simply have the learner move the newly generated cookbook into the cookbooks directory.

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 Reserved'
description 'Installs/Configures myhaproxy'
long_description 'Installs/Configures myhaproxy'
version '0.1.0'
chef_version '>= 12.1' if respond_to?(:chef_version)

depends 'haproxy', '~> 3.0.0'
```

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.

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  
}]
```

<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.



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`.

Include the haproxy's manual recipe in default recipe

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

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

include_recipe 'haproxy::manual'
```

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 manual 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::manual'
```

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. 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 iis_web -a ipaddress
```

```
iis_web:  
  ipaddress: 172.31.8.68
```

You can display the IP address of `iis_web` 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.

Instructor Note: The IP addresses of the nodes that were used during the creation of this training were based on Amazon Web Services (AWS). The address reported by Ohai is often the private, internal address.



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 iis_web -a cloud
```

```
iis_web:  
  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 iis_web, you will see the local, private, and public connection information.

Capture and write down the public hostname and the public ipv4 address of iis_web. 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::manual'
```

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 iis_web (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::manual'
```

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']'

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::manual'
```

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

Save your recipe file.



Node Attribute Precedence

	Attribute Files	Node / Recipe	Environment	Role
default	1	2	3	4
force_default	5	6		
normal	7	8		
override	9	10	12	11
force_override	13	14		
automatic		15		



GL: Load Balancer

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

Objective:

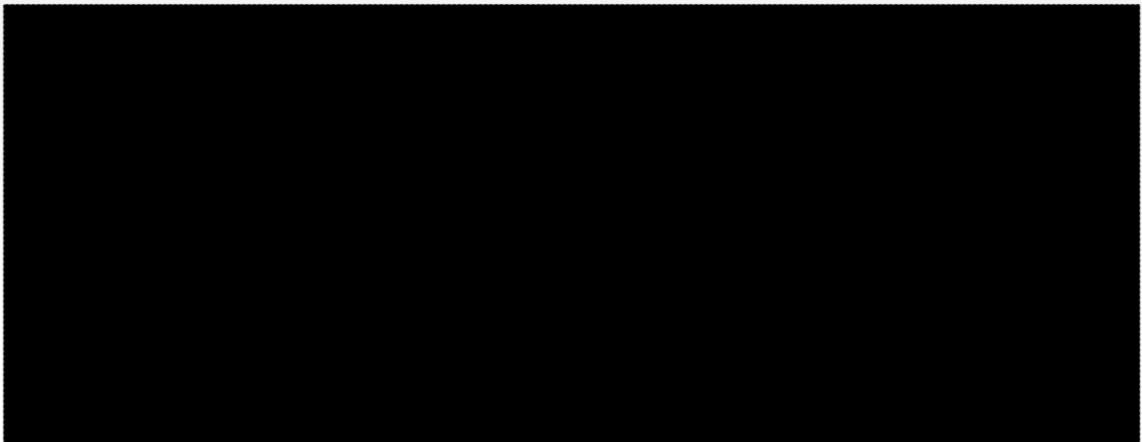
- ✓ Find a Cookbook on the Chef Supermarket to Manage a load balancer
- ✓ Configure the load balancer to send traffic to the iis_web node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy (load balancer) cookbook
- Create GitHub repository and commit changes with Git (optional)

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

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.

Installing the Cookbook's Dependencies



```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myhaproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using cpu (2.0.0)
Using build-essential (8.0.3)
Using haproxy (3.0.4)
Using windows (3.2.0)
Using ohai (5.2.0)
Using poise (2.8.1)
Using mingw (2.0.1)
Using poise-service (1.5.2)
Using seven_zip (2.0.2)
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.

Instructor Note: Berkshelf downloads these cookbooks into a common directory within your home path. They are not added alongside your other cookbooks.

Uploading the Cookbook and Dependencies



```
$ berks upload
```

```
Uploaded build-essential (8.0.3) to: 'https://api.chef.io:443/organizations/devops'
Uploaded cpu (2.0.0) to: 'https://api.chef.io:443/organizations/devops'
Uploaded haproxy (3.0.4) to: 'https://api.chef.io:443/organizations/devops'
Uploaded mingw (2.0.1) to: 'https://api.chef.io:443/organizations/devops'
Uploaded myhaproxy (0.2.0) to: 'https://api.chef.io:443/organizations/devops'
Uploaded ohai (5.2.0) to: 'https://api.chef.io:443/organizations/devops'
Uploaded poise (2.8.1) to: 'https://api.chef.io:443/organizations/devops'
Uploaded poise-service (1.5.2) to: 'https://api.chef.io:443/organizations/devops'
Uploaded seven_zip (2.0.2) to: 'https://api.chef.io:443/organizations/devops'
Uploaded windows (3.2.0) to: 'https://api.chef.io:443/organizations/devops'
```

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.

Verifying the Cookbook has Been Uploaded



```
$ knife cookbook list
```

apache	0.1.0
build-essential	8.0.3
company_web	0.1.0
cpu	2.0.0
haproxy	3.0.4
mingw	2.0.1
myhaproxy	0.1.0
myiis	0.2.1
ohai	5.2.0
poise	2.8.1
poise-service	1.5.2
seven_zip	2.0.2

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



EXERCISE

GL: Load Balancer

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

Objective:

- Find a Cookbook on the Chef Supermarket to Manage a load balancer
- Configure the load balancer to send traffic to the iis_web node
- Upload cookbook to Chef Server
- Bootstrap a new node that runs the haproxy (load balancer) cookbook
- Create GitHub repository and commit changes with Git (optional)

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

Bootstrapping a New Linux Node



```
$ knife bootstrap IP -x USER -P PWD --sudo -N lb
```

```
Creating new client for lb
Creating new node for lb
Connecting to 54.242.217.190
54.242.217.190 -----> Existing Chef installation detected
54.242.217.190 Starting the first Chef Client run...
54.242.217.190 Starting Chef Client, version 13.2.20
54.242.217.190 resolving cookbooks for run list: []
54.242.217.190 Synchronizing Cookbooks:
54.242.217.190 Installing Cookbook Gems:
54.242.217.190 Compiling Cookbooks...
54.242.217.190 [2017-10-31T01:48:06+00:00] WARN: Node lb has an empty run list.
54.242.217.190 Converging 0 resources
```

First you bootstrap a new node named lb.

Viewing the New Node's Data



```
$ knife node show lb
```

```
Node Name:    lb
Environment: _default
FQDN:        ip-172-31-0-128.ec2.internal
IP:          54.210.192.12
Run List:
Roles:
Recipes:
Platform:    centos 6.7
Tags:
```

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

Defining a Run List for the Node



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

```
lb:  
  run_list: recipe[myhaproxy]
```

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

Validating the Run List has been Set



```
$ knife node show lb
```

```
Node Name:    lb
Environment: _default
FQDN:        ip-172-31-0-128.ec2.internal
IP:          54.210.192.12
Run List:     recipe[myhaproxy]
Roles:
Recipes:
Platform:    centos 6.7
Tags:
```

Ensure the run list has been set correctly for lb.



PROBLEM

SSH Woes

Logging into 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.

Running a Command with Knife



```
$ knife ssh "IP" -m -x USERNAME -P PASSWORD "sudo chef-client"
```

```
ec2-34-230-9-106.compute-1.amazonaws.com Starting Chef Client, version 13.2.20
ec2-34-230-9-106.compute-1.amazonaws.com resolving cookbooks for run list:
[ "myhaproxy" ]
ec2-34-230-9-106.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-34-230-9-106.compute-1.amazonaws.com   - myhaproxy (0.2.0)
ec2-34-230-9-106.compute-1.amazonaws.com   - haproxy (3.0.4)
ec2-34-230-9-106.compute-1.amazonaws.com   - cpu (2.0.0)
ec2-34-230-9-106.compute-1.amazonaws.com   - build-essential (8.0.3)
ec2-34-230-9-106.compute-1.amazonaws.com   - seven_zip (2.0.2)
ec2-34-230-9-106.compute-1.amazonaws.com   - windows (3.2.0)
ec2-34-230-9-106.compute-1.amazonaws.com   - ohai (5.2.0)
```

Viewing the Website is Being Proxied

URL of load balancer.

Output from the
iis_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.



EXERCISE

GL: Load Balancer

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

Objective:

- ✓ Find a Cookbook on the Chef Supermarket to Manage a load balancer
- ✓ Configure the load balancer to send traffic to the iis_web node
- ✓ Upload cookbook to Chef Server
- ✓ Bootstrap a new node that runs the haproxy (load balancer) cookbook
- Create GitHub repository and commit changes with Git (optional)

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

Create myhaproxy GitHub Repo and copy URL

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: SamMcB7 Repository name: myhaproxy 

Great repository names are short and memorable. Need inspiration? How about jubilant-happiness.

Description (optional): This is the 'myhaproxy' cookbook for the Chef DevOps Foundation class

Public: Anyone can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None | 

Create repository

SamMcB7 / myhaproxy

Code Issues Pull requests Projects Wiki Insights Settings

Quick setup — if you've done this kind of thing before
Set up in Desktop or HTTPS SSH https://github.com/SamMcB7/myhaproxy.git 

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# myhaproxy" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SamMcB7/myhaproxy.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/SamMcB7/myhaproxy.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Commit the Changes



```
> cd ~/chef-repo/cookbooks/myhaproxy  
> git init  
> git add .  
> git commit -m "Initial Commit"  
> git remote add origin https://github.com/username/myhaproxy.git  
> git push -u origin master
```

Return to the cookbook directory and add all the changed files and commit them with a message.



EXERCISE

GL: Load Balancer

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

Objective:

- ✓ Find a Cookbook on the Chef Supermarket to Manage a load balancer
- ✓ Configure the load balancer to send traffic to the iis_web node
- ✓ Upload cookbook to Chef Server
- ✓ Bootstrap a new node that runs the haproxy (load balancer) cookbook
- ✓ Create GitHub repository and commit changes with Git (optional)

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



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.

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



Q&A

What questions can we help you answer?

- Chef Supermarket
- 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.



CHEF TM

©2017 Chef Software Inc.

Managing Multiple Nodes

Create Another Apache 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.

Objectives

After completing this module, you should be able to

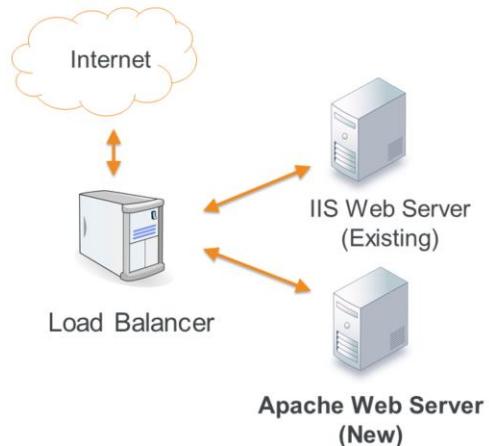
- Bootstrap, update the run_list, and run chef-client on a Linux node
- Append values to an attribute within a recipe
- Version a cookbook and upload it to the Chef Server

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.

Managing User Traffic

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

In this module you'll add another Apache web server node to the load balancer's pool of web servers it is directing traffic to.



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

- iis_web: An IIS web server
- lb: The load balancer
- apache_web: Another web server



Lab: Add a Linux Web Node

- Bootstrap a new Linux node giving it the name 'apache_web'
- Update the run list of the new node to include the company_web 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.

Instructor Note: Allow 10 minutes to complete this exercise

Bootstrapping a New Linux Node



```
$ knife bootstrap IP -x USER -P PWD --sudo -N apache_web
```

```
Connecting to 52.90.49.196
52.90.49.196 ----> Existing Chef installation detected
52.90.49.196 Starting the first Chef Client run...
52.90.49.196 Starting Chef Client, version 13.2.20
52.90.49.196 resolving cookbooks for run list: []
52.90.49.196 Synchronizing Cookbooks:
52.90.49.196 Installing Cookbook Gems:
52.90.49.196 Compiling Cookbooks...
52.90.49.196 [2017-10-20T18:25:32+00:00] WARN: Node apache_web has an empty run list.
52.90.49.196 Converging 0 resources
52.90.49.196
52.90.49.196 Running handlers:
52.90.49.196 Running handlers complete
52.90.49.196 Chef Client finished, 0/0 resources updated in 07 seconds
```

Bootstrap the new node and name it apache_web.

Viewing the Details of the New Node



```
$ knife node show apache_web
```

```
Node Name: apache_web
Environment: _default
FQDN: ip-172-31-18-193.ec2.internal
IP: 52.90.49.196
Run List:
Roles:
Recipes:
Platform: centos 6.9
Tags:
```

Verify that you bootstrapped the node.



Lab: Add a Linux Web Node

- Bootstrap a new Linux node giving it the name 'apache_web'
- Update the run list of the new node to include the company_web 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.

Instructor Note: Allow 10 minutes to complete this exercise

Setting the Run List of the New Node



```
$ knife node run_list add apache_web "recipe[company_web]"
```

```
apache_web:  
  run_list: recipe[company_web]
```

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



Lab: Add a Linux Web Node

- ✓ Bootstrap a new Linux node giving it the name 'apache_web'
- ✓ Update the run list of the new node to include the company_web 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.

Instructor Note: Allow 10 minutes to complete this exercise

Converging the Run List



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

```
34.207.100.168 -----> Existing Chef installation detected
34.207.100.168 Starting the first Chef Client run...
34.207.100.168 Starting Chef Client, version 13.2.20
34.207.100.168 resolving cookbooks for run list: ["company_web"]
34.207.100.168 Synchronizing Cookbooks:
34.207.100.168   - apache (0.1.0)
34.207.100.168   - myiis (0.2.1)
34.207.100.168   - company_web (0.1.1)
34.207.100.168 Installing Cookbook Gems:
34.207.100.168 Compiling Cookbooks...
34.207.100.168 Converging 3 resources
34.207.100.168 Recipe: apache::server
34.207.100.168   * yum_package[httpd] action install (up to date)
```

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.



Lab: Add a Linux Web Node

- ✓ Bootstrap a new Linux node giving it the name 'apache_web'
- ✓ Update the run list of the new node to include the company_web 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.

Instructor Note: Allow 10 minutes to complete this exercise

Verifying that the New Node Serves the Page



Verify that the node serves up the default html page that contains the node's platform, hostname, memory, and CPU speed.



Lab: Add a Linux Web Node

- ✓ Bootstrap a new Linux node giving it the name 'apache_web'
- ✓ Update the run list of the new node to include the company_web 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.

Instructor Note: Allow 10 minutes to complete this exercise



Lab: Update the Load Balancer

- Update the wrapped proxy server cookbook to include the new apache_web node as a member.
- Update the metadata for a minor change and upload the cookbook to the Chef Server
- Converge the load balancer
- Verify that the load balancer delivers traffic to both web server nodes.
- Commit your changes with git (optional)

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

Instructor Note: Allow 15 minutes to complete this exercise

Displaying the New Node's IP and Hostname



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

```
apache_web:  
cloud:  
  local_hostname: ip-172-31-18-193.ec2.internal  
  local_ipv4: 172.31.18.193  
  local_ipv4_addrs: 172.31.18.193  
  provider: ec2  
  public_hostname: ec2-52-90-49-196.compute-1.amazonaws.com  
  public_ipv4: 52.90.49.196  
  public_ipv4_addrs: 52.90.49.196
```

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

Capture and write down the public hostname and the public ipv4 address of apache_web. 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' => 'ec2-52-90-49-196.compute-1.amazonaws.com',
  'ipaddress' => '52.90.49.196',
  'port' => 80,
  'ssl_port' => 80
}]
include_recipe 'haproxy::manual'
```

Add the second web server (apache_web) 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.



Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new Apache web node as a member.
- ❑ Update the metadata for a minor change and upload the cookbook to the Chef Server
- ❑ Converge the load balancer
- ❑ Verify that the load balancer delivers traffic to both web server nodes.
- ❑ Commit your changes with git (optional)

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

Instructor Note: Allow 15 minutes to complete this exercise

Updating the Cookbook's Version

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

name 'myhaproxy'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'All Rights Reserved'
description 'Installs/Configures myhaproxy'
long_description 'Installs/Configures myhaproxy'
version '0.2.0'
chef_version '>= 12.1' if respond_to?(:chef_version)

depends 'haproxy', '~> 3.0.0'
```

Update the version number in myhaproxy cookbook's metadata.

Uploading the Cookbook to the Chef Server



```
$ berks upload
```

```
Uploaded myhaproxy (0.2.0) to:  
'https://api.chef.io:443/organizations/devops'  
Skipping ohai (5.2.0) (frozen)  
Skipping poise (2.8.1) (frozen)  
Skipping poise-service (1.5.2) (frozen)  
Skipping seven_zip (2.0.2) (frozen)  
Skipping windows (3.2.0) (frozen)
```

Run 'berks upload' to upload the myhaproxy cookbook to Chef Server.



Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new Apache web node as a member.
- ✓ Update the metadata for a minor change and upload the cookbook to the Chef Server
- Converge the load balancer
- Verify that the load balancer delivers traffic to both web server nodes.
- Commit your changes with git (optional)

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

Instructor Note: Allow 15 minutes to complete this exercise

Converging the Load Balancer node



```
$ knife ssh 'name:lb' -x chef -P Cod3Can! 'sudo chef-client'
```

```
ec2-52-90-49-196.compute-1.amazonaws.com Starting Chef Client, version 13.2.20
ec2-34-230-9-106.compute-1.amazonaws.com Starting Chef Client, version 13.2.20
ec2-52-90-49-196.compute-1.amazonaws.com resolving cookbooks for run list: ["apache"]
ec2-52-90-49-196.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-52-90-49-196.compute-1.amazonaws.com   - apache (0.2.1)
ec2-52-90-49-196.compute-1.amazonaws.com Installing Cookbook Gems:
ec2-52-90-49-196.compute-1.amazonaws.com Compiling Cookbooks...
ec2-52-90-49-196.compute-1.amazonaws.com Converging 3 resources
ec2-52-90-49-196.compute-1.amazonaws.com Recipe: apache::server
ec2-34-230-9-106.compute-1.amazonaws.com resolving cookbooks for run list: ["myhaproxy"]
ec2-34-230-9-106.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com   - build-essential
```

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 (apache_web).



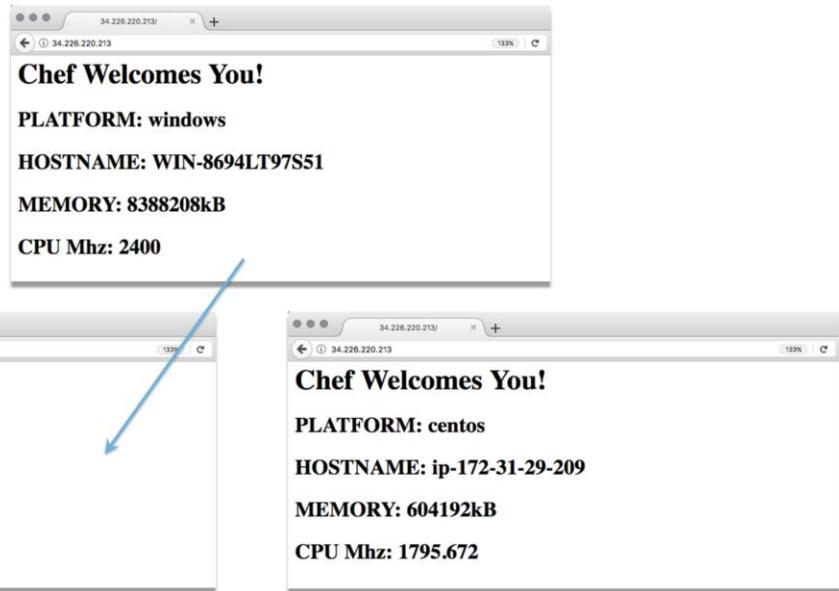
Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new Apache web node as a member.
- ✓ Update the metadata for a minor change and upload the cookbook to the Chef Server
- ✓ Converge the load balancer
- Verify that the load balancer delivers traffic to both web server nodes.
- Commit your changes with git (optional)

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

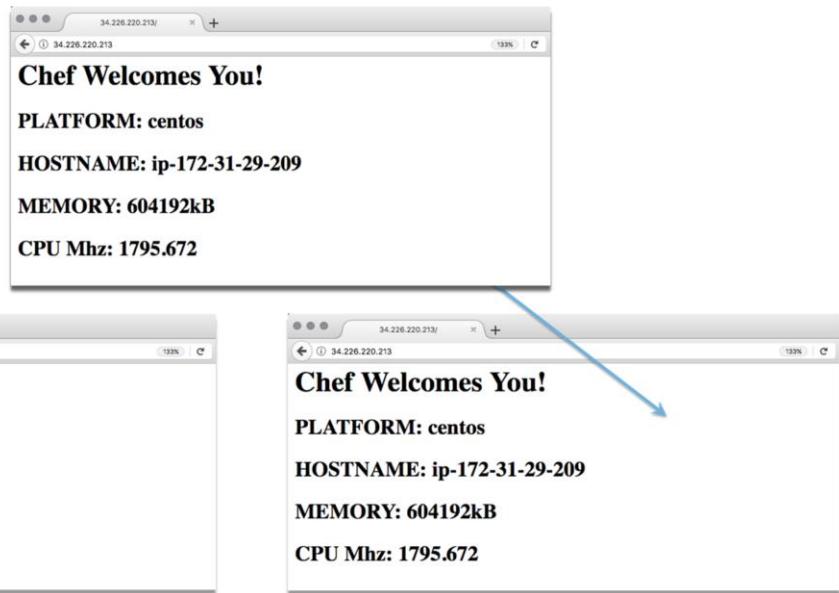
Instructor Note: Allow 15 minutes to complete this exercise

Lab: Test the Load Balancer



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 server.

Lab: Test the Load Balancer





Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new Apache web node as a member.
- ✓ Update the metadata for a minor change and upload the cookbook to the Chef Server
- ✓ Converge the load balancer
- ✓ Verify that the load balancer delivers traffic to both web server nodes.
- Commit your changes with git (optional)

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

Instructor Note: Allow 15 minutes to complete this exercise



Commit the Changes

```
> cd ~/chef-repo/cookbooks/myhaproxy  
> git add .  
> git commit -m "Version 0.2.0, added load  
balancing member"  
> git push
```

Return to the cookbook directory and add all the changed files and commit them with a message.



Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new Apache web node as a member.
- ✓ Update the metadata for a minor change and upload the cookbook to the Chef Server
- ✓ Converge the load balancer
- ✓ Verify that the load balancer delivers traffic to both web server nodes.
- ✓ Commit your changes with git (optional)

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

Instructor Note: Allow 15 minutes to complete this exercise



Discussion

What is the process to set up a third web node?

What would the process be for removing a web node?

What is the most manual part of the process?

Answer these questions.

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



Q&A

What questions can we help you answer?



CHEF TM

©2017 Chef Software Inc.

Roles

Giving Your Nodes a Role

Objectives

After completing this module, you should be able to

- Create roles that will contain a particular run list of recipes
- Assign roles to nodes so you can better describe them and configure them in a similar manner.
- Set new node attributes values with a role

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



CONCEPT Roles

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

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

Up until this point it has been a mouthful to describe the nodes within our organization.

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.



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_server" Role

In this section you will create a `load_balancer` role and assign it to the run list of `lb`. You will also will create a `web_server` role and assign it to the run list of `iis_web` and `apache_web`.

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
```

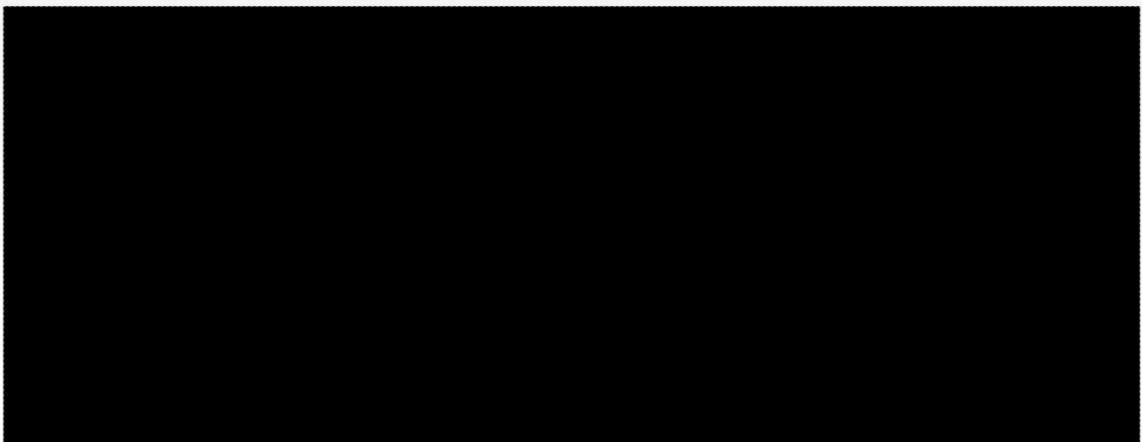
```
** ROLE COMMANDS **  
knife role bulk delete REGEX (options)  
knife role create ROLE (options)  
knife role delete ROLE (options)  
knife role edit ROLE (options)  
knife role env_run_list add [ROLE] [ENVIRONMENT] [ENTRY[,ENTRY]]  
(options)  
knife role env_run_list clear [ROLE] [ENVIRONMENT] 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

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.

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.

Locating the roles Directory



```
$ ls -l
```

```
-rw-r--r--@ 1 technotrainer  staff  2341 Oct 18 21:02 README.md
drwxr-xr-x@ 7 technotrainer  staff   238 Oct 18 16:09 cookbooks
drwxr-xr-x@ 5 technotrainer  staff   170 Oct 18 16:50 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.

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.

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:           recipe[myhaproxy]
```

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

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)  
knife node show NODE (options)
```

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

Setting the Run List for the Node



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

```
lb:  
  run_list: role[load_balancer]
```

The last step is to redefine the run list for lb. 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 lb
```

```
Node Name:    lb
Environment: _default
FQDN:        ip-172-31-25-169.ec2.internal
IP:          34.230.9.106
Run List:    role[load_balancer]

Roles:
Recipes:     myhaproxy, myhaproxy::default, haproxy::manual, haproxy::install_package
Platform:   centos 6.9
Tags:
```

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

Converging all Load Balancers



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

```
ec2-34-230-9-106.compute-1.amazonaws.com Starting Chef Client, version 13.2.20
ec2-34-230-9-106.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-34-230-9-106.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-34-230-9-106.compute-1.amazonaws.com   - myhaproxy (0.2.1)
ec2-34-230-9-106.compute-1.amazonaws.com   - haproxy (3.0.4)
ec2-34-230-9-106.compute-1.amazonaws.com   - cpu (2.0.0)
ec2-34-230-9-106.compute-1.amazonaws.com   - build-essential (8.0.3)
ec2-34-230-9-106.compute-1.amazonaws.com   - seven_zip (2.0.2)
ec2-34-230-9-106.compute-1.amazonaws.com   - windows (3.2.0)
ec2-34-230-9-106.compute-1.amazonaws.com   - ohai (5.2.0)
```

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 lb 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.

Lab: Define the `web_server` Role



- Create a `web_server` role using the `company_web` cookbook as it's run list
- Upload the role to the Chef Server
- Assign the `iis_web` node and `apache_web` node the `web_server` role
- Converge both web servers

Defining the Web Server Role

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

```
name 'web_server'  
description 'Apache and IIS Web Servers'  
run_list 'recipe[company_web]'
```

First we create a file named web_server.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 company_web cookbook's default recipe.



Lab: Define the `web_server` Role

- ✓ Create a `web_server` role using the `company_web` cookbook as it's run list
- ❑ Upload the role to the Chef Server
- ❑ Assign the `iis_web` node and `apache_web` node the `web_server` role
- ❑ Converge both web servers

Upload the Web Server Role



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

```
Updated Role web_server
```

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

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

Verifying the Roles on the Chef Server



```
$ knife role list
```

```
load_balancer
```

```
web_server
```

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

Viewing the Details about the Role



```
$ knife role show web_server
```

```
chef_type:          role
default_attributes:
description:        Apache and IIS Web Servers
env_run_lists:
json_class:         Chef::Role
name:               web_server
override_attributes:
run_list:           recipe[company_web]
```

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



Lab: Define the `web_server` Role

- ✓ Create a `web_server` role using the `company_web` cookbook as it's run list
- ✓ Upload the role to the Chef Server
- ❑ Assign the `iis_web` node and `apache_web` node the `web_server` role
- ❑ Converge both web servers

Setting the IIS Web Node's Run List



```
$ knife node run_list set iis_web "role[web_server]"
```

```
iis_web:  
  run_list: role[web_server]
```

Setting the Apache Web Node's Run List



```
$ knife node run_list set apache_web "role[web_server]"
```

```
apache_web:  
  run_list: role[web_server]
```

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



Lab: Define the `web_server` Role

- ✓ Create a `web_server` role using the `company_web` cookbook as it's run list
- ✓ Upload the role to the Chef Server
- ✓ Assign the `iis_web` node and `apache_web` node the `web_server` role
- Converge both web servers

Converging all Web Nodes (Linux)



```
$ knife ssh "role:web_server AND os:linux" -x USER -P PWD "sudo chef-client"
```

```
ec2-184-73-96-131.compute-1.amazonaws.com Starting Chef Client, version
13.2.20
ec2-184-73-96-131.compute-1.amazonaws.com resolving cookbooks for run list:
["company_web"]
ec2-184-73-96-131.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-184-73-96-131.compute-1.amazonaws.com   - company_web (0.1.0)
ec2-184-73-96-131.compute-1.amazonaws.com   - myiis (0.2.1)
ec2-184-73-96-131.compute-1.amazonaws.com   - apache (0.1.0)
ec2-184-73-96-131.compute-1.amazonaws.com Installing Cookbook Gems:
ec2-184-73-96-131.compute-1.amazonaws.com Compiling Cookbooks...
ec2-184-73-96-131.compute-1.amazonaws.com Converging 1 resources
ec2-184-73-96-131.compute-1.amazonaws.com Recipe: company_web::default
```

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 Linux nodes with the role set to web_server.

Converge All Web Nodes (Windows)



```
> knife winrm "role:web_server AND os:windows" -x USER -P PWD "chef-client"
```

```
ERROR: Network Error: A connection attempt failed because the connected party did
not properly respond after a period of time, or established connection failed
because connected host has failed to respond. - connect(2) for "172.31.10.176" port
5985 (172.31.10.176:5985)
```

```
Check your knife configuration and network settings
```

All nodes that have the web role

We can use knife winrm to log into multiple remote hosts and run commands

But it cannot seem to connect. Why?

To verify that everything is working the same as before, run 'knife winrm'. In this instance the query syntax is going to find all Windows nodes with the role set to web_server using the specified attribute on the node to match against.

Capture Nodes' IP

 > knife node show iis_web -a cloud.public_ipv4

```
iis_web:  
  cloud.public_ipv4: 34.196.63.231
```

Issue: We will use the `node['cloud']['public_ipv4']` attribute value

Lets execute the show command again for each of our nodes

Converge Web Node (Windows)



```
> knife winrm "role:web_server AND os:windows" -a cloud.public_ipv4 -x USER  
-P PWD "chef-client"
```

```
54.159.197.193 Starting Chef Client, version 13.6.0  
54.159.197.193  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: *** Chef 13.6.0 ***  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: Platform: x64-mingw32  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: Chef-client pid: 496  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: The plugin path  
C:\chef\ohai\plugins does not exist. Skipping...  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Run List is [role[web_server]]  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Run List expands to [company_web]  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Starting Chef Run for iis_web  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Running start handlers  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Start handlers complete.
```

To verify that everything is working the same as before, run 'knife winrm'. In this instance the query syntax is going to find all Windows nodes with the role set to web_server.



Lab: Define the `web_server` Role

- ✓ Create a `web_server` role using the `company_web` cookbook as it's run list
- ✓ Upload the role to the Chef Server
- ✓ Assign the `iis_web` node and `apache_web` node the `web_server` role
- ✓ Converge both web servers



EXERCISE

Roles for Everyone

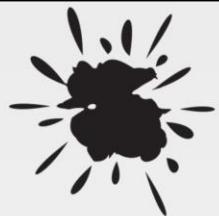
We will give our nodes a role to better describe them and so we can configure them in a similar manner.

Objective:

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

In this section you will create a `load_balancer` role and assign it to the run list of `lb`. You will also will create a `web_server` role and assign it to the run list of `iis_web` and `apache_web`.

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.



Update the Company Name

Wait!

Your company was just acquired by **E Corp** and now your website must be updated to display the correct company name.

GL: Updating the Web Server Role

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

```
name 'web_server'  
description 'Apache and IIS Web Servers'  
run_list 'recipe[company_web]'  
default_attributes 'company_web' => { 'company_name' => 'E Corp' }
```

Upload the Web Server Role



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

```
Updated Role web_server
```

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

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

Converge Web Nodes



```
$ knife winrm "name:iis_web" -a cloud.public_ipv4 -x USER -P PWD "chef-client"  
$ knife ssh "name:apache_web" -x USER -P PWD "sudo chef-client"
```

```
54.159.197.193 Starting Chef Client, version 13.6.0  
54.159.197.193  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: *** Chef 13.6.0 ***  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: Platform: x64-mingw32  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: Chef-client pid: 496  
54.159.197.193 [2017-10-31T02:49:21+00:00] INFO: The plugin path  
C:\chef\ohai\plugins does not exist. Skipping...  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Run List is [role[web_server]]  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Run List expands to [company_web]  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Starting Chef Run for iis_web  
54.159.197.193 [2017-10-31T02:49:24+00:00] INFO: Running start handlers
```

Verify the Changes to Web Page





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.

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



Q&A

What questions can we help you answer?



CHEF TM

©2017 Chef Software Inc.

Search

Update a Cookbook to Dynamically Use Nodes with the Server Role

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 dynamically

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.



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.

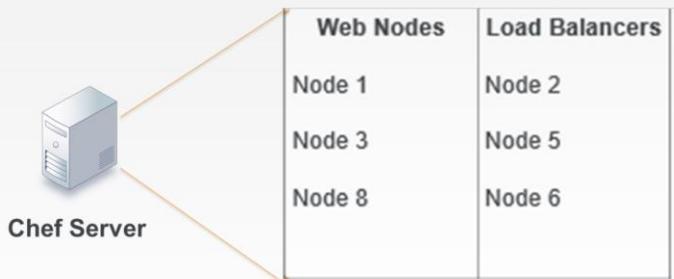
To add new servers as load balancer members, we would need to bootstrap a new web server and then update our myhaproxy cookbook to include that new web server. But that seems dramatically inefficient to have to update a cookbook recipe.

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.

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

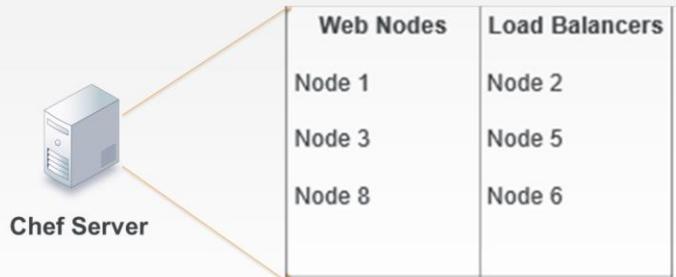
https://docs.chef.io/chef_search.html#search-indexes

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.

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.

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 Criteria

We may use wildcards within search so a search criteria that we could use 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 "`*.*`" 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.

Demo: View Information for All Nodes



```
> knife search node "*:*"
```

```
Node Name: lb
Environment: _default
FQDN: ip-172-31-23-107.ec2.internal
IP: 34.226.220.213
Run List: recipe[myhaproxy]
Roles:
Recipes: myhaproxy, myhaproxy::default, haproxy::manual, haproxy::install_package
Platform: centos 6.9
Tags:

Node Name: iis_web
Environment: _default
FQDN: WIN-8694LT97S51.ec2.internal
IP: 34.229.225.40
Run List: role[web_server]
Roles:
Recipes: company_web, company_web::default, myiis::default, myiis::server
Platform: windows 6.3.9600
Tags:
```

So run the command `knife search node "*:*`" – you should see all nodes returned.

However, querying and returning every node is not exactly what we need to solve our current problem. In our scenario we want only to return a subset of our nodes (only the nodes that are web servers) and also we only want the IP address of those nodes, not any the other attributes returned.

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, and only the info we want for those nodes.

Demo: View Information for All Server Nodes



```
> knife search node "role:web_server"
```

```
Node Name: iis_web
Environment: _default
FQDN: WIN-8694LT97S51.ec2.internal
IP: 34.229.225.40
Run List: role[web_server]
Roles:
Recipes: company_web, company_web::default, myiis::default, myiis::server
Platform: windows 6.3.9600
Tags:

Node Name: apache_web
Environment: _default
FQDN: ip-172-31-29-209.ec2.internal
IP: 34.207.100.168
Run List: role[web_server]
Roles:
Recipes: company_web, company_web::default, apache::default, apache::server
Platform: centos 6.9
Tags:
```

Demo: Return Public Hostname for Servers



```
> knife search node "role:web_server" -a cloud.public_hostname
```

```
iis_web:  
  cloud.public_hostname: ec2-34-229-225-40.compute-1.amazonaws.com  
  
apache_web:  
  cloud.public_hostname: ec2-34-207-100-168.compute-1.amazonaws.com
```

The '-a' flag allows you to specify a particular attribute from those nodes.

Search Syntax within a Recipe

```
all_web_nodes = search('node', 'role:web_server')
```

creates and names a variable

assigns the value of the operation on the right into the variable on the left

the index or items to search

the search criteria - key:value

invokes the search method

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_server')
```

Search the Chef Server for all node objects that have the role equal to 'web_server' 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_server' 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::manual'
```

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'.



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_server 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
- Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.

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::manual'
```

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_server')

#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::manual'
```

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_server`.

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_server')

members = []

#TODO: Convert all found nodes into hashes with ipaddress,
#      hostname, port, ssl_port

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

include_recipe 'haproxy::manual'
```

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_server')

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::manual'
```

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_server')

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::manual'
```

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>



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_server 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
- Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.

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 Reserved'
description 'Installs/Configures myhaproxy'
long_description 'Installs/Configures myhaproxy'
version '1.0.0'
chef_version '>= 12.1' if respond_to?(:chef_version)

depends 'haproxy', '~> 3.0.0'
```

First we update the version to the next major release. We set the version number to 1.0.0.



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_server 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
- ❑ Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.

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.

Instructor Note: During the course the learner may find they have a mistake with the cookbook and need to re-upload the cookbook. Berkshelf will 'freeze' the versions of the cookbooks that you upload. This is to prevent you from accidentally overriding cookbooks that you may have already created. It is a best practice to not re-upload a cookbook again with new changes if they share the same version. During this course, however, it is important that the version numbers be aligned to make future sections work correctly so it OK to do in the training environment. To re-upload a cookbook with Berkshelf replacing the existing cookbook can be done with `berks upload --force`



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_server 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
- Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.

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 lb "role:lb".

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.



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_server 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
- Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.

E Corp Welcomes You!

PLATFORM: windows

HOSTNAME: WIN-DQFQCUFHDCP

MEMORY: 1048176kB

CPU Mhz: 2400

E Corp Welcomes You!

PLATFORM: windows

HOSTNAME: WIN-DQFQCUFHDCP

MEMORY: 1048176kB

CPU Mhz: 2400

E Corp Welcomes You!

PLATFORM: centos

HOSTNAME: ip-172-31-26-186

MEMORY: 604192kB

CPU Mhz: 1799.999

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.

The image shows three separate browser windows, each displaying a "E Corp Welcomes You!" message followed by system configuration details. A blue arrow points from the top window down to the middle window, indicating they are the same node. The rightmost window is a different node. A large black chef's hat icon is positioned in the top right corner of the slide area.

Platform	Hostname	Memory	CPU Mhz
Windows	WIN-DQFQCUFHDCP	1048176kB	2400
Windows	WIN-DQFQCUFHDCP	1048176kB	2400
CentOS	ip-172-31-26-186	604192kB	1799.999

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.



EXERCISE

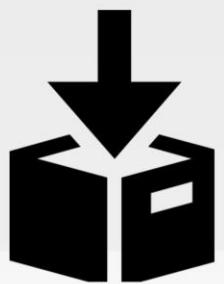
Dynamic Web Load Balancer

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_server 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
- Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.



Commit the Changes

```
> cd ~/chef-repo/cookbooks/myhaproxy  
> git add .  
> git status  
> git commit -m "Version 1.0.0, dynamically update load  
balancing pool"  
> git push
```

Return to the cookbook directory and add all the changed files and commit them with a message.



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_server 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
- ✓ Commit changes with Git (optional)

In this section we'll update the load balancer's myhaproxy cookbook to dynamically use nodes with the web role.



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.

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



Q&A

What questions can we help you answer?



CHEF TM

©2017 Chef Software Inc.

Running chef-client as a Service

Introducing the chef-client cookbook

Lesson Objectives

After completing this module, you should be able to:

- Use knife to work with the Chef Supermarket site API
- Override community cookbook defaults using wrapper cookbooks
- Run chef-client as a service/task



Step Back: How is chef-client Configured?

- ❖ How can I run chef-client as a service or Windows task?
- ❖ Where can I configure logging?
- ❖ How does chef-client know what Chef Server to connect to?
- ❖ How does chef-client authenticate with the Chef Server?
- ❖ How do I configure where chef-client caches?

GL: View How chef-client is Configured



In this group lab you will view how chef-client is configured.

Demo: View chef-client config Directory (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD "ls -F /etc/chef"
```

```
ec2-54-242-217-190.compute-1.amazonaws.com client.pem  client.rb  first-boot.json      ohai/
ec2-184-73-96-131.compute-1.amazonaws.com client.pem  client.rb  first-boot.json      ohai/
```

chef-client looks for its configuration information in the directory '/etc/chef' by default – although this is configurable itself!

This directory contains, not only its own configuration file (which we'll look at shortly), but also its key to authenticate with the Chef Server, an Ohai plugins directory and a first-boot.json file. The first-boot.json file is generated from the workstation as part of the initial knife bootstrap subcommand, and contains the initial runlist that chef-client should run after Chef has been installed, and before it first registers with the Chef Server.

Demo: View chef-client config File (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD \
"cat /etc/chef/client.rb"
```

```
ec2-184-73-96-131.compute-1.amazonaws.com chef_server_url
"https://api.chef.io/organizations/2017_october_30"
ec2-184-73-96-131.compute-1.amazonaws.com validation_client_name "chef-validator"
ec2-184-73-96-131.compute-1.amazonaws.com log_location STDOUT
ec2-184-73-96-131.compute-1.amazonaws.com node_name "apache_web"
ec2-184-73-96-131.compute-1.amazonaws.com
ec2-54-242-217-190.compute-1.amazonaws.com chef_server_url
"https://api.chef.io/organizations/2017_october_30"
ec2-54-242-217-190.compute-1.amazonaws.com validation_client_name "chef-validator"
ec2-54-242-217-190.compute-1.amazonaws.com log_location STDOUT
ec2-54-242-217-190.compute-1.amazonaws.com node_name "lb"
ec2-54-242-217-190.compute-1.amazonaws.com
```

The configuration file for chef-client is '/etc/chef/client.rb'. This file contains the URL for the Chef Server that chef-client should communicate with, i.e. the API endpoint. The validation_client_name parameter is only used with older versions of Chef Server to define what key is used for the initial authentication with the Chef Server. The file also contains the name of the node, which is used to identify the node on the Chef Server, as well as chef-client's log level and log location.

'chef-client' on the node and 'knife' on the workstation are both API client, in that they both communicate with the Chef Server over the API - the file '/etc/chef/client.rb' is the equivalent to the 'knife.rb' file on the workstation.

Demo: View chef-client config Directory (Windows)



```
$ knife winrm "os/windows" -x USER -P PWD  
-a cloud.public_ipv4 "dir C:\chef"
```

```
54.159.197.193  Directory of C:\chef  
54.159.197.193  
54.159.197.193 10/31/2017  12:59 AM    <DIR>        .  
54.159.197.193 10/31/2017  12:59 AM    <DIR>        ..  
54.159.197.193 10/31/2017  12:59 AM    <DIR>        backup  
54.159.197.193 10/30/2017  11:33 PM    <DIR>        cache  
54.159.197.193 10/30/2017  11:30 PM      1,706 client.pem  
54.159.197.193 10/30/2017  11:30 PM      335 client.rb  
54.159.197.193 10/30/2017  11:30 PM      17 first-boot.json  
54.159.197.193 02/01/2016  06:41 AM    <DIR>        ohai  
54.159.197.193 10/30/2017  11:28 PM      316 wget.ps1  
54.159.197.193 10/30/2017  11:28 PM      1,923 wget.vbs
```

Demo: View chef-client config File (Windows)



```
$ knife winrm "os/windows" -x USER -P PWD  
-a cloud.public_ipv4 "more C:\chef\client.rb"
```

```
54.159.197.193 chef_server_url "https://api.chef.io/organizations/2017_october_30"  
54.159.197.193  
54.159.197.193 validation_client_name "chef-validator"  
54.159.197.193 file_cache_path "c:/chef/cache"  
54.159.197.193 file_backup_path "c:/chef/backup"  
54.159.197.193 cache_options ({:path => "c:/chef/cache/checksums", :skip_expires => true})  
54.159.197.193 node_name "iis_web"  
54.159.197.193 log_level :info  
54.159.197.193 log_location STDOUT
```



Introducing the **chef-client** Cookbook

The chef-client cookbook allows you to manage and configure chef-client as a service on Linux-based nodes, or as a task on Windows nodes, configure logging, caching, etc.

Bootstrapping installs the chef-client executable.

The chef-client cookbook is used to configure chef-client.



Lets Examine the `chef-client` Cookbook

We're going to use one recipe on our node from the `chef-client` cookbook.

`chef-client::service`
(via `chef-client::default`)

GL: View the chef-client::default Recipe

```
~/berkshelf/cookbooks/chef-client-<version>/recipes/default.rb
```

```
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
  
#  
if platform?('windows')  
  include_recipe 'chef-client::task'  
else  
  include_recipe 'chef-client::service'  
end
```

The default recipe just makes a call to the recipe 'chef-client::service'--this is the recipe that will set chef-client to run as a service.

GL: View the chef-client::service Recipe

~/berkshelf/cookbooks/chef-client-<version>/recipes/service.rb

- The recipe supports a number of **service** providers and styles.
- It works on a lot of **platforms**.
- Everything is controllable through **attributes**.

```
supported_init_styles = %w(
  bsd
  init
  launchd
  smf
  src
  systemd
  upstart
  windows
)

init_style = node['chef_client']['init_style']

# Services moved to recipes
if supported_init_styles.include? init_style
  include_recipe "chef-client::#{init_style}_service"
else
  log 'Could not determine service init style, manual
  intervention required to start up the chef-client
  service.'
end
```

There's a lot to this recipe so we won't cover it all in detail.

There are several init styles that can be selected by setting the node attribute in the recipe. "init" is the default, and what we're going to use. Also available: smf, upstart, arch, runit, bluepill, daemontools, winsw. "cron" is a separate recipe since it is not technically running as a "service".



Introducing chef-client Cookbook

We will use the chef-client community cookbook to configure chef-client on each of our nodes to run periodically

Objective:

- Upload chef-client cookbook to Chef Server
- Configure each of our nodes to run chef-client as a service



Wrapper Cookbooks

Don't use forked community cookbooks in production, or you will miss out on upstream changes, and will have to rebase

Instead use **wrapper cookbooks** to wrap upstream cookbooks and change their behavior without forking

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

Create mychef_client Wrapper Cookbook



```
$ cd chef-repo  
$ chef generate cookbook cookbooks/mychef_client
```

```
Generating cookbook mychef_client  
- Ensuring correct cookbook file content  
- Committing cookbook files to git  
- Ensuring delivery configuration  
- Ensuring correct delivery build cookbook content  
- Adding delivery configuration to feature branch  
- Adding build cookbook to feature branch  
- Merging delivery content feature branch to master
```

```
Your cookbook is ready. Type `cd cookbooks/mychef_client` to enter it.
```

Change to your chef-repo directory and then generate your new cookbook.

Instructor Note: The generate cookbook command needs the path in the name of the cookbook here to ensure that it generates the cookbook in the cookbooks directory. Without it the cookbook will be generated within the root of the chef repository. If that happens simply have the learner move the newly generated cookbook into the cookbooks directory.

Edit mychef_client Default Recipe

cookbooks/mychef_client/recipes/default.rb

```
#  
# Cookbook:: mychef_client  
# Recipe:: default  
#  
# Copyright:: 2017, The Authors, All Rights Reserved.
```

```
include_recipe 'chef-client::default'
```

This recipe just calls the recipe `chef-client::default`

GL: Update mychef_client metadata.rb

```
cookbooks/mychef_client/metadata.rb

name 'mychef_client'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'All Rights Reserved'
description 'Installs/Configures mychef_client'
long_description 'Installs/Configures mychef_client'
version '0.1.0'
chef_version '>= 12.1' if respond_to?(:chef_version)

depends 'chef-client'
```

Install the Dependencies



```
$ cd cookbooks/mychef_client  
$ berks install
```

```
Resolving cookbook dependencies...  
Fetching 'mychef_client' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Installing chef-client (9.0.0)  
Using windows (3.2.0)  
Installing logrotate (2.2.0)  
Installing compat_resource (12.19.0)  
Using mychef_client (0.1.0) from source at .  
Installing cron (4.2.0)  
Using ohai (5.2.0)
```

Demo: View Berksfile.lock



```
$ cat Berksfile.lock
```

```
DEPENDENCIES
mychef_client
  path: .
  metadata: true

GRAPH
chef-client (9.0.0)
cron (>= 2.0.0)
logrotate (>= 1.9.0)
windows (>= 2.0.0)
compat_resource (12.19.0)
...
```

Demo: View the Berkshelf in Your Home Directory



```
$ ls -lt ~/.berkshelf/cookbooks/
```

```
drwxr-xr-x 11 technotrainer staff 374 Oct 30 21:31 chef-client-9.0.0
drwxr-xr-x  9 technotrainer staff 306 Oct 30 21:31 compat_resource-12.19.0
drwxr-xr-x 13 technotrainer staff 442 Oct 30 21:31 cron-4.2.0
drwxr-xr-x 20 technotrainer staff 680 Oct 30 21:31 logrotate-2.2.0
drwxr-xr-x 10 technotrainer staff 340 Oct 18 16:24 seven_zip-2.0.2
drwxr-xr-x 13 technotrainer staff 442 Oct 18 16:24 windows-3.2.0
drwxr-xr-x 12 technotrainer staff 408 Oct 18 16:24 build-essential-8.0.3
drwxr-xr-x  9 technotrainer staff 306 Oct 18 16:24 cpu-2.0.0
```

Note: Windows users can use `ls -force ~/.berkshelf/cookbooks/`

Berkshelf does not store dependent cookbook in the actual repo you're working in, but puts them in another location, usually under your home directory. This means your repo is not polluted with cookbooks that you neither need nor want. For example, the chef-client cookbook is designed to be multiplatform, and will work on all flavors of Linux as well as Windows. However, if you're a Linux shop you may not want Windows specific dependencies in your working repo.

Upload Cookbooks to Chef Server



```
$ berks upload
```

```
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-
5.1.1/lib/ridley/client.rb:271:in `server_url': Ridley::Client#url_prefix at
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-5.1.1/lib/ridley/client.rb:79
forwarding to private method Celluloid::PoolManager#url_prefix
Uploaded chef-client (9.0.0) to:
'https://api.chef.io:443/organizations/2017_october_30'
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-
5.1.1/lib/ridley/client.rb:271:in `server_url': Ridley::Client#url_prefix at
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-5.1.1/lib/ridley/client.rb:79
forwarding to private method Celluloid::PoolManager#url_prefix
Uploaded compat_resource (12.19.0) to:
'https://api.chef.io:443/organizations/2017_october_30'
/opt/chefdk/embedded/lib/ruby/gems/2.4.0/gems/ridley-
5.1.1/lib/ridley/client.rb:271:in `server_url': Ridley::Client#url_prefix at
```

View Cookbooks on Chef Server



```
$ knife cookbook list
```

apache	0.3.0
chef-client	7.0.0
compat_resource	12.16.2
cron	3.0.0
haproxy	0.2.1
logrotate	2.1.0
mychef_client	0.1.0
ohai	4.2.2
windows	2.1.1
workstation	0.2.1



chef-client as a Service

We will add the chef-client default recipe to the roles for each node.

GL: Create the new 'base' role

```
chef-repo/roles/base.rb
```

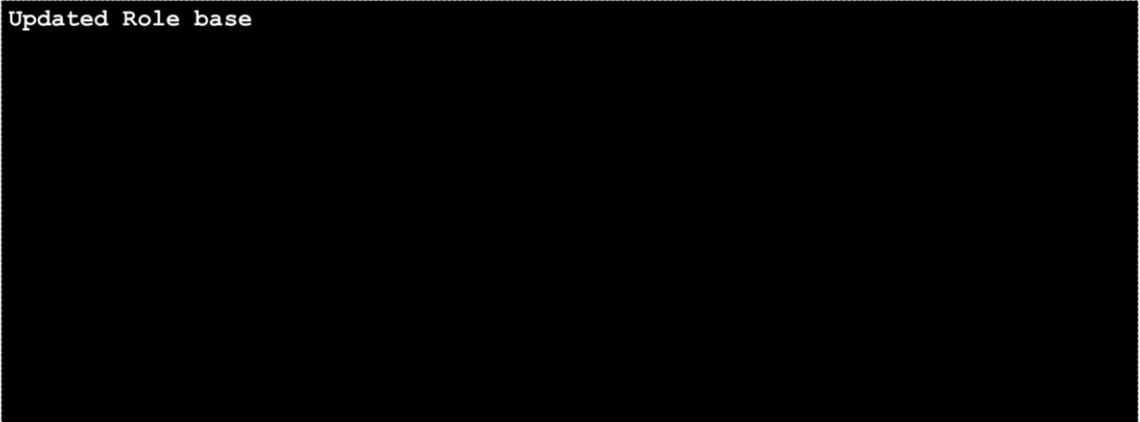
```
name 'base'  
description 'Base Role'  
run_list 'recipe[mychef_client]'
```

Upload it to the Chef Server



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

```
Updated Role base
```



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

GL: Add the base Role to load_balancer Role

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

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

GL: Add the base Role to web_server Role

```
chef-repo/roles/web_server.rb
```

```
name 'web_server'  
description 'Apache and IIS Web Servers'  
run_list 'role[base]', 'recipe[company_web]'  
default_attributes 'company_web' => {'company_name' => 'E Corp'}
```

<http://bit.ly/236fthv>

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.

Upload the roles Files



```
$ cd ~/chef-repo  
$ knife role from file load_balancer.rb web_server.rb
```

```
Updated Role load_balancer  
Updated Role web_server
```

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

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

Converge All Nodes (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD "sudo chef-client"
```

```
ec2-54-162-31-253.compute-1.amazonaws.com  Starting Chef Client, version 12.13.37
ec2-54-167-232-148.compute-1.amazonaws.com Starting Chef Client, version 12.13.37
ec2-54-205-59-139.compute-1.amazonaws.com  Starting Chef Client, version 12.13.37
ec2-54-167-232-148.compute-1.amazonaws.com resolving cookbooks for run list: ["chef-client", "apache"]
ec2-54-205-59-139.compute-1.amazonaws.com  resolving cookbooks for run list: ["chef-client", "apache"]
ec2-54-162-31-253.compute-1.amazonaws.com  resolving cookbooks for run list: ["chef-client", "haproxy"]
ec2-54-167-232-148.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-205-59-139.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-162-31-253.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-205-59-139.compute-1.amazonaws.com      - chef-client (5.0.0)
...
...
```

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 OS of linux.

Verify chef-client is Running (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD  
"ps awux | grep chef-client"
```

```
ec2-514-88-185-159.compute-1.amazonaws.com root      10369  0.0 10.1 266928 61116 ?  
S1 12:54  0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c  
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300  
...  
ec2-514-88-169-195.compute-1.amazonaws.com root      14922  0.0 10.1 267020 61092 ?  
S1 12:54  0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c  
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300  
...
```

Its using our full-stack installation path automatically, and the -d says "runs as a daemon in the background"

We log to /var/log/chef/client.log (configurable). The -i is the interval, a number of seconds (30 minutes here) (configurable) and the last bit is splay (mitigating the thundering herd problem)

Instructor Note: embedded dir path--talk about Omnibus installer if not mentioned already.

Converge All Nodes (Windows)



```
$ knife winrm "os/windows" -x USER -P PWD  
-a cloud.public_ipv4 "chef-client"
```

```
54.159.197.193 Starting Chef Client, version 13.6.0  
54.159.197.193  
54.159.197.193 [2017-10-31T03:45:21+00:00] INFO: *** Chef 13.6.0 ***  
54.159.197.193 [2017-10-31T03:45:21+00:00] INFO: Platform: x64-mingw32  
54.159.197.193 [2017-10-31T03:45:21+00:00] INFO: Chef-client pid: 2860  
54.159.197.193 [2017-10-31T03:45:21+00:00] INFO: The plugin path  
C:\chef\ohai\plugins does not exist. Skipping...  
54.159.197.193 [2017-10-31T03:45:25+00:00] INFO: Run List is [role[web_server]]  
54.159.197.193 [2017-10-31T03:45:25+00:00] INFO: Run List expands to [mychef-client,  
company_web]  
54.159.197.193 [2017-10-31T03:45:25+00:00] INFO: Starting Chef Run for iis_web
```

Verify chef-client is Running (Windows)



```
$ knife winrm 'os/windows' -x USER -P PWD -a  
cloud.public_ipv4 'schtasks /Query | find "chef-client"'
```

```
54.159.197.193 chef-client  
54.159.197.193 10/23/2017 11:39:00 PM Running
```



Introducing chef-client cookbook

We will use the chef-client community cookbook to configure chef-client on each of our nodes to run periodically

Objective:

- ✓ Upload chef-client cookbook to Chef Server
- ✓ Configure each of our nodes to run chef-client as a service



Change Default Settings

Wait!

There has just been a mandate that every node in the infrastructure must run chef-client every 5 minutes.

Example: Setting the chef-client run Interval

```
chef-repo/cookbooks/chef-client/attributes/default.rb

...
default['chef_client']['log_file']      = 'client.log'
default['chef_client']['interval']      = '1800' (highlighted)
default['chef_client']['splay']          = '300'
default['chef_client']['conf_dir']       = '/etc/chef'
default['chef_client']['bin']           = '/usr/bin/chef-client'

...
```

We need to change the value of the attribute

`default['chef_client']['interval']` from 1800 (seconds) to
300 (...but don't change it in this file.)

So we need to change the attribute `default['chef_client']['interval']` in `chef-client` cookbook.

But what if a new cookbook version is released and we want to upgrade to it?

We'd need to re-implement these changes!

Maintenance nightmare – especially if its been refactored!

Good practice would be to not edit a community cookbook that you have wrapped.

So we will set the interval attribute as shown on the next slide.

GL: Update the 'base' Role

chef-repo/roles/base.rb

```
name 'base'
description 'Base Role'
run_list 'recipe[mychef_client]'
default_attributes 'chef_client' => {'interval' => 300}
```

Edit the file named base.rb in the roles directory as shown in this slide.

```
name 'base'
description 'Base Role'
run_list 'recipe[my_chef_client]'
default_attributes({
  "chef_client" => {
    "interval" => 300
  }
})
```

Upload it to the Chef Server



```
$ cd ~/chef-repo/roles  
$ knife role from file base.rb
```

```
Updated Role base!
```

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

Converge All Nodes (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD "sudo chef-client"
```

```
ec2-54-242-217-190.compute-1.amazonaws.com Starting Chef Client, version 13.2.20
ec2-184-73-96-131.compute-1.amazonaws.com  Starting Chef Client, version 13.2.20
ec2-54-242-217-190.compute-1.amazonaws.com resolving cookbooks for run list:
["mychef-client", "myhaproxy"]
ec2-184-73-96-131.compute-1.amazonaws.com  resolving cookbooks for run list:
["mychef-client", "company_web"]
ec2-184-73-96-131.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-184-73-96-131.compute-1.amazonaws.com      - mychef-client (0.1.0)
ec2-184-73-96-131.compute-1.amazonaws.com      - chef-client (9.0.0)
ec2-184-73-96-131.compute-1.amazonaws.com      - cron (4.2.0)
ec2-184-73-96-131.compute-1.amazonaws.com      - compat_resource (12.19.0)
ec2-184-73-96-131.compute-1.amazonaws.com      - logrotate (2.2.0)
ec2-184-73-96-131.compute-1.amazonaws.com      - windows (3.2.0)
```

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.

Verify chef-client is Running (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD \  
"ps awux | grep chef-client"
```

```
ec2-514-88-185-159.compute-1.amazonaws.com root      10369  0.0 10.1 266928 61116 ?  
S1 12:54  0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c  
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300  
...  
ec2-514-88-169-195.compute-1.amazonaws.com root      14922  0.0 10.1 267020 61092 ?  
S1 12:54  0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c  
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300  
...
```

Its using our full-stack installation path automatically, and the -d says "runs as a daemon in the background"

We log to /var/log/chef/client.log (configurable). The -i is the interval, a number of seconds (30 minutes here) (configurable) and the last bit is splay (mitigating the thundering herd problem)

Instructor Note: embedded dir path--talk about Omnibus installer if not mentioned already.

Converge All Nodes (Windows)



```
$ knife winrm "os/windows" -x USER -P PWD \
-a cloud.public_ipv4 "chef-client"
```

```
54.159.197.193 Starting Chef Client, version 13.6.0
54.159.197.193
54.159.197.193 [2017-10-31T03:52:58+00:00] INFO: *** Chef 13.6.0 ***
54.159.197.193 [2017-10-31T03:52:58+00:00] INFO: Platform: x64-mingw32
54.159.197.193 [2017-10-31T03:52:58+00:00] INFO: Chef-client pid: 1424
54.159.197.193 [2017-10-31T03:52:58+00:00] INFO: The plugin path
C:\chef\ohai\plugins does not exist. Skipping...
54.159.197.193 [2017-10-31T03:53:02+00:00] INFO: Run List is [role[web_server]]
54.159.197.193 [2017-10-31T03:53:02+00:00] INFO: Run List expands to [mychef-client,
company_web]
54.159.197.193 [2017-10-31T03:53:02+00:00] INFO: Starting Chef Run for iis_web
```

Verify chef-client is Running (Windows)



```
$ knife winrm 'os/windows' -x USER -P PWD -a  
cloud.public_ipv4 'schtasks /Query | find "chef-client"'
```

```
54.159.197.193 chef-client  
54.159.197.193 10/23/2017 11:39:00 PM Ready
```

LAB

Lab (optional): Create GitHub Repo for mychef_client



©2017 Chef Software Inc.

17-42



Create mychef_client GitHub Repo and copy URL

Create a new repository
A repository contains all the files for your project, including the revision history.

Owner: SamMcB7 Repository name: mychef_client 

Great repository names are short and memorable. Need inspiration? How about `furry-eureka`.

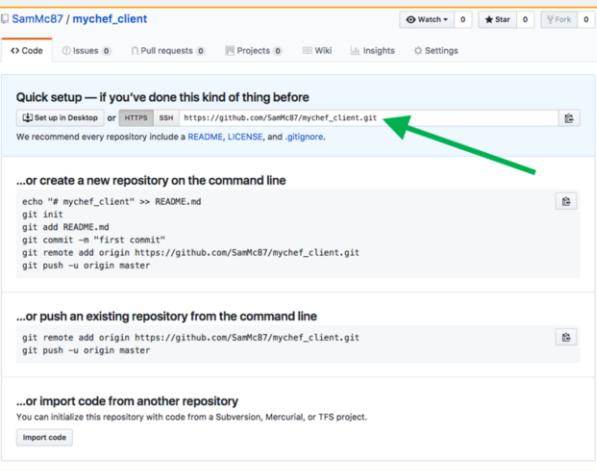
Description (optional):
This is the 'mychef_client' cookbook for the Chef DevOps Foundation class.

Public: Anyone can see this repository. You choose who can commit.
 Private: You choose who can see and commit to this repository.

Initialize this repository with a README: This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None Add a license: None 

Create repository

SamMcB7 / mychef_client 

Quick setup — if you've done this kind of thing before
Set up in Desktop or **HTTPS** https://github.com/SamMcB7/mychef_client.git 
We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line
`echo "# mychef_client" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SamMcB7/mychef_client.git
git push -u origin master` 

...or push an existing repository from the command line
`git remote add origin https://github.com/SamMcB7/mychef_client.git
git push -u origin master` 

...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code 

©2017 Chef Software Inc.

17-43



Commit the Changes



```
> cd ~/chef-repo/cookbooks/mychef_client  
> git init  
> git add .  
> git commit -m "Initial Commit"  
> git remote add origin  
  https://github.com/username/mychef_client.git  
> git push -u origin master
```



Q&A

What questions can we help you answer?

- Chef Supermarket
- Wrapper Cookbooks
- chef-client Cookbook

What questions can we help you answer?

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



CHEFTM

©2017 Chef Software Inc.

Data Bags

Working with Custom Data Sets



Objectives

After completing this module, you should be able to

- Understand how to use and manage a Data Bag
- Create a Data Bag
- Upload Data Bag to Chef Server



CONCEPT

What about user data?

Where should we store data that each node might need access to?



What about user data?

We could start by storing information about users as Node Attributes...

But this would duplicate a lot of information - every user in the company would be stored in every Node object!



Data Bags

A data bag is a container for items that represent information about your infrastructure that is not tied to a single node.

Examples:

- Users
- Groups
- Application Release Information
- Passwords (in an encrypted data bag)



Custom Data Sets

We can store sets of JSON data on our Chef Server, accessible by a node with search

Objective:

- Create “users” and “groups” data bags
- Upload data bags to Chef Server
- Create a cookbook to manage users and groups

What Can 'knife data bag' Do?



```
$ cd ~/chef-repo  
$ knife data bag --help  
  
** DATA BAG COMMANDS **  
knife data bag create BAG [ITEM] (options)  
knife data bag delete BAG [ITEM] (options)  
knife data bag edit BAG ITEM (options)  
knife data bag from file BAG FILE|FOLDER [FILE|FOLDER..] (options)  
knife data bag list (options)  
knife data bag show BAG [ITEM] (options)
```

Run 'knife data bag list'



```
$ knife data bag list
```



Create a data_bags Directory



```
$ mkdir data_bags
```



Create a data_bags/users Directory



```
$ mkdir data_bags/users
```



Create users Data Bag on Chef Server



```
$ knife data bag create users
```

```
Created data_bag[users]
```

Create centos_user.json

```
~/chef-repo/data_bags/users/centos_user.json
```

```
{  
  "id": "centos_user",  
  "comment": "I am a centos user",  
  "uid": 100,  
  "gid": 1,  
  "home": "/home/centos_user",  
  "shell": "/bin/bash",  
  "platform": "centos"  
}
```

Create windows_user.json

```
~/chef-repo/data_bags/users/windows_user.json
```

```
{  
  "id": "windows_user",  
  "comment": "I am a windows user",  
  "uid": 101,  
  "gid": 1,  
  "platform": "windows"  
}
```

Upload data bag items to Chef Server



```
$ knife data bag from file users data_bags/users/centos_user.json  
data_bags/users/windows_user.json
```

```
Updated data_bag_item[users::centos_user]  
Updated data_bag_item[users::windows_user]
```

Validate Chef Server Received It



```
$ knife data bag list
```

```
users
```

View Details of users Data Bag



```
$ knife data bag show users
```

```
centos_user  
windows_user
```

View Details of user1



```
$ knife data bag show users/centos_user
```

```
comment  
gid  
home  
id  
platform  
shell  
uid
```

Search the “users” index



```
$ knife search users "*:*"
```

```
2 items found

chef_type: data_bag_item
comment: I am a windows user
data_bag: users
gid: 1
id: windows_user
platform: windows
uid: 101
....
```

Return users with “platform:centos”



```
$ knife search users "platform:windows"
```

```
1 items found

chef_type: data_bag_item
comment: I am a windows user
data_bag: users
gid: 1
id: windows_user
platform: windows
uid: 101
```



Create Users from a Data Bag

Dynamically search through the Chef Server under the 'users' index to create users

Objective:

- Generate the 'myusers' cookbook
- Search for all users belonging to group1
- Create Users based on data bag contents
- Upload 'myusers' cookbook to Chef Server
- Update webserver roles
- Converge web nodes



Where are the users?

Since our users and groups Data Bags are now indexed on the Chef Server, we have a centralized source of truth for this information.

We can search through this information inside of our recipes.

cd and Generate the myusers Cookbook



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

Create default recipe

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

```
search("users", "platform:#{node['platform']}").each do |user_data|
  if node['platform'] == 'windows'
    user user_data['id'] do
      comment user_data['comment']
      action :create
    end
  else
    ...
  end
end
```

Create default recipe - Continued

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

```
...
else
  user user_data['id'] do
    comment user_data['comment']
    uid user_data['uid']
    gid user_data['gid']
    home user_data['home']
    shell user_data['shell']
    action :create
  end
end
end
```

Change to the cookbooks/myusers Directory

 \$ cd cookbooks/myusers



Run berks install



```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myusers' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using myusers (0.1.0) from source at .
```

Upload the Cookbook to the Chef Server



```
$ berks upload
```

```
Uploaded myusers (0.1.0) to: 'https://api.opscode.com:443/organizations/ORG_NAME'
```

Display Cookbooks within Your Org



```
$ knife cookbook list
```

apache	0.1.0
build-essential	7.0.2
compat_resource	12.16.2
cpu	1.0.0
haproxy	2.0.0
mingw	1.2.4
myhaproxy	0.1.0
myusers	0.1.0
ohai	4.2.3
seven_zip	2.0.2
windows	2.1.1
workstation	0.2.1

Update the Web Server Role to include myusers

~/chef-repo/roles/web_server.rb

```
name 'web_server'  
description 'Apache and IIS Web Servers'  
run_list 'recipe[myusers]', 'role[base]', 'recipe[company_web]'  
default_attributes 'company_web' => { 'company_name' => 'E Corp'}
```

Upload Role to the Chef Server



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

```
Updated Role web_server
```

View Details of the Role



```
$ knife role show web_server
```

```
chef_type:          role
default_attributes:
  company_web:
    company_name: E Corp
description:        Apache and IIS Web Servers
env_run_lists:
json_class:         Chef::Role
name:               web_server
override_attributes:
run_list:
  recipe[myusers]
  role[base]
  recipe[company_web]
```

Converging all Web Nodes (Linux)



```
$ knife ssh "role:web_server AND os:linux" -x USER -P PWD "sudo chef-client"
```

```
ec2-52-90-49-196.compute-1.amazonaws.com Starting Chef Client, version
13.2.20
ec2-52-90-49-196.compute-1.amazonaws.com resolving cookbooks for run list:
["myusers", "apache"]
ec2-52-90-49-196.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-52-90-49-196.compute-1.amazonaws.com   - myusers (0.1.0)
ec2-52-90-49-196.compute-1.amazonaws.com   - apache (0.2.1)
ec2-52-90-49-196.compute-1.amazonaws.com Installing Cookbook Gems:
ec2-52-90-49-196.compute-1.amazonaws.com Compiling Cookbooks...
ec2-52-90-49-196.compute-1.amazonaws.com Converging 4 resources
ec2-52-90-49-196.compute-1.amazonaws.com Recipe: myusers::default
...
```

Converge All Web Nodes (Windows)



```
> knife winrm "role:web_server AND os:windows" -a cloud.public_ipv4 -x USER  
-P PWD "chef-client"
```

```
34.196.63.231 [2017-01-20T14:34:57+00:00] INFO: *** Chef 12.13.37 ***  
34.196.63.231 [2017-01-20T14:34:57+00:00] INFO: Platform: i386-mingw32  
34.196.63.231 [2017-01-20T14:34:57+00:00] INFO: Chef-client pid: 2460  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Run List is [role[web]]  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Run List expands to [myusers,myiis]  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Starting Chef Run for node1  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Running start handlers  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Start handlers complete.  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Loading cookbooks [myusers@0.1.0,  
myiis@0.2.1]  
52.71.198.206 Synchronizing Cookbooks:  
52.71.198.206   - myusers (0.1.0)  
...
```

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

Check Local Users for Apache Server



```
$ knife ssh "name:apache_web" -x USER -P PWD cat "/etc/passwd"
```

```
....  
ec2-52-90-49-196.compute-1.amazonaws.com sshd:x:74:74:Privilege-separated  
SSH:/var/empty/sshd:/sbin/nologin  
ec2-52-90-49-196.compute-1.amazonaws.com chef:x:500:500:ChefDK  
User:/home/chef:/bin/bash  
ec2-52-90-49-196.compute-1.amazonaws.com dbus:x:81:81:System message  
bus:/sbin/nologin  
ec2-52-90-49-196.compute-1.amazonaws.com dockerroot:x:498:498:Docker  
User:/var/lib/docker:/sbin/nologin  
ec2-52-90-49-196.compute-1.amazonaws.com centos_user:x:100:1:I am a centos  
user:/home/centos_user:/bin/bash
```

Check Local Users for IIS Server



```
> knife winrm "name:iis_web" -a cloud.public_ipv4 -x USER -P PWD "net user windows_user"
```

```
52.90.49.196 User name           windows_user
52.90.49.196 Full Name          I am a windows user
52.90.49.196 Comment
52.90.49.196 User's comment
52.90.49.196 Country/region code 000 (System Default)
52.90.49.196 Account active      Yes
52.90.49.196 Account expires     Never
....
```



Create Users from Data bag

Dynamically search through the Chef Server under the 'users' index to create users

Objective:

- ✓ Generate the 'myusers' cookbook
- ✓ Create Users based on data bag contents
- ✓ Upload 'myusers' cookbook to Chef Server
- ✓ Update webserver role
- ✓ Converge web nodes



Lab: Managing Groups

- Create and upload a 'groups' data bag with your users as members
- Update the 'myusers' cookbook to add your users to the group using search
- Verify webserver roles contain 'myusers', Converge your webservers
- Verify the new group on centos in the /etc/group file
- Verify the new group on windows with 'net localgroup GROUP_NAME'

Create a data_bags/groups Directory



```
$ mkdir data_bags/groups
```



Create groups Data Bag on Chef Server



```
$ knife data bag create groups
```

```
Created data_bag[groups]
```

Create centos_group.json

```
~/chef-repo/data_bags/groups/centos_group.json
```

```
{  
  "id": "centos_group",  
  "gid": "100",  
  "members": ["centos_user"],  
  "platform": "centos"  
}
```

Create windows_group.json

```
~/chef-repo/data_bags/groups/windows_group.json
```

```
{  
  "id": "windows_group",  
  "members": ["windows_user"],  
  "platform": "windows"  
}
```

Upload data bag item to Chef Server



```
$ knife data bag from file groups data_bags/groups/centos_group.json  
data_bags/groups/windows_group.json
```

```
Updated data_bag_item[groups::centos_group]  
Updated data_bag_item[groups::windows_group]
```

Validate Chef Server Received It



```
$ knife data bag list
```

```
users  
groups
```

View Details of users Data Bag



```
$ knife data bag show groups
```

```
centos_group  
windows_group
```

View Details of centos_group



```
$ knife data bag show groups centos_group
```

```
gid:      100
id:       centos_group
members:  centos_user
platform: centos
```

Return groups with “platform:windows”

```
 $ knife search groups "platform:windows"
```

```
1 items found

chef_type: data_bag_item
data_bag:   groups
id:         windows_group
members:   windows_user
platform:  windows
```

Version the myusers metadata.rb

```
~/chef-repo/cookbooks/myusers/metadata.rb
```

```
name          'myusers'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures myusers'  
long_description 'Installs/Configures myusers'  
version        '0.2.0'
```

Create a 'groups' recipe

~/chef-repo/cookbooks/myusers/recipes/groups.rb

```
search("groups", "platform:#{{node['platform']}}").each do
|group_data|
  group group_data['id'] do
    members group_data['members']
    action :create
  end
end
```

Update the default recipe

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

```
...
else
  user user_data['id'] do
    comment user_data['comment']
    uid user_data['uid']
    gid user_data['gid']
    home user_data['home']
    shell user_data['shell']
    action :create
  end
end
end

include_recipe 'myusers::groups'
```

Change to the cookbooks/myusers Directory

 \$ cd cookbooks/myusers



Run berks install



```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myusers' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using myusers (0.2.0) from source at .
```

Upload the Cookbook to the Chef Server



```
$ berks upload
```

```
Uploaded myusers (0.2.0) to: 'https://api.opscode.com:443/organizations/ORG_NAME'
```

Display Cookbooks within Your Org



```
$ knife cookbook list
```

apache	0.2.1
build-essential	7.0.2
compat_resource	12.16.2
cpu	1.0.0
haproxy	2.0.0
mingw	1.2.4
myhaproxy	0.1.0
myusers	0.2.0
ohai	4.2.3
seven_zip	2.0.2
windows	2.1.1
workstation	0.2.1

Converging all Web Nodes (Linux)



```
$ knife ssh "role:web_server AND os:linux" -x USER -P PWD "sudo chef-client"
```

```
ec2-52-90-49-196.compute-1.amazonaws.com Starting Chef Client, version
13.2.20
ec2-52-90-49-196.compute-1.amazonaws.com resolving cookbooks for run list:
["myusers", "apache"]
ec2-52-90-49-196.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-52-90-49-196.compute-1.amazonaws.com   - myusers (0.1.0)
ec2-52-90-49-196.compute-1.amazonaws.com   - apache (0.2.1)
ec2-52-90-49-196.compute-1.amazonaws.com Installing Cookbook Gems:
ec2-52-90-49-196.compute-1.amazonaws.com Compiling Cookbooks...
ec2-52-90-49-196.compute-1.amazonaws.com Converging 4 resources
ec2-52-90-49-196.compute-1.amazonaws.com Recipe: myusers::default
...
```

Converge All Web Nodes (Windows)



```
> knife winrm "role:web_server AND os:windows" -a cloud.public_ipv4 -x USER  
-P PWD "chef-client"
```

```
34.196.63.231 [2017-01-20T14:34:57+00:00] INFO: *** Chef 12.13.37 ***  
34.196.63.231 [2017-01-20T14:34:57+00:00] INFO: Platform: i386-mingw32  
34.196.63.231 [2017-01-20T14:34:57+00:00] INFO: Chef-client pid: 2460  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Run List is [role[web]]  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Run List expands to [myusers,myiis]  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Starting Chef Run for node1  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Running start handlers  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Start handlers complete.  
34.196.63.231 [2017-01-20T14:35:34+00:00] INFO: Loading cookbooks [myusers@0.1.0,  
myiis@0.2.1]  
52.71.198.206 Synchronizing Cookbooks:  
52.71.198.206   - myusers (0.1.0)  
...
```

Check Local Groups (Linux)



```
$ knife ssh "name:apache_web" -x USER -P PWD "cat /etc/group"
```

```
....  
ec2-52-90-49-196.compute-1.amazonaws.com sshd:x:74:  
ec2-52-90-49-196.compute-1.amazonaws.com chef:x:500:  
ec2-52-90-49-196.compute-1.amazonaws.com dbus:x:81:  
ec2-52-90-49-196.compute-1.amazonaws.com cgred:x:499:  
ec2-52-90-49-196.compute-1.amazonaws.com dockerroot:x:498:chef  
ec2-52-90-49-196.compute-1.amazonaws.com centos_group:x:501:centos_user
```

Check Local Groups (Windows)



```
> knife winrm "name:iis_web" -a cloud.public_ipv4 -x USER -P PWD "net localgroup windows_group"
```

```
52.90.49.196 Alias name      windows_group
52.90.49.196 Comment
52.90.49.196
52.90.49.196 Members
52.90.49.196
52.90.49.196 -----
-----
52.90.49.196 windows_user
52.90.49.196 The command completed successfully.
```



Lab: Managing Groups

- ✓ Create and upload a 'groups' data bag with your users as members
- ✓ Update the 'myusers' cookbook to add your users to the group using search
- ✓ Verify webserver roles contain 'myusers', Converge your webservers
- ✓ Verify the new group on centos in the /etc/group file
- ✓ Verify the new group on windows with 'net localgroup GROUP_NAME'



Lab (optional): Create GitHub Repo for myusers

Create myusers GitHub Repo and copy URL

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: SamMcB87 / Repository name: myusers 

Great repository names are short and memorable. Need inspiration? How about `miniature-enigma`.

Description (optional): This is the 'myusers' cookbook for the Chef DevOps Foundation class.

Visibility: Public  Private 

Initialize this repository with a README This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None  Add a license: None 

Create repository

SamMcB87 / myusers

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Quick setup — if you've done this kind of thing before
Set up in Desktop or HTTPS SSH <https://github.com/SamMcB87/myusers.git> 
We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line
`echo "# myusers" >> README.md`
`git init`
`git add README.md`
`git commit -m "first commit"`
`git remote add origin https://github.com/SamMcB87/myusers.git`
`git push -u origin master`

...or push an existing repository from the command line
`git remote add origin https://github.com/SamMcB87/myusers.git`
`git push -u origin master`

...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

Commit the Changes



```
> cd ~/chef-repo/cookbooks/myusers  
> git init  
> git add .  
> git commit -m "Initial Commit"  
> git remote add origin https://github.com/username/myusers.git  
> git push -u origin master
```



Discussion

When should we utilize data bags instead of node attributes?

When creating a new data bag, what index on the Chef server does the data bag get added to?



Q&A

What questions can we help you answer?



©2017 Chef Software Inc.

Environments

Using Environments to Reflect Organization Patterns and Workflow

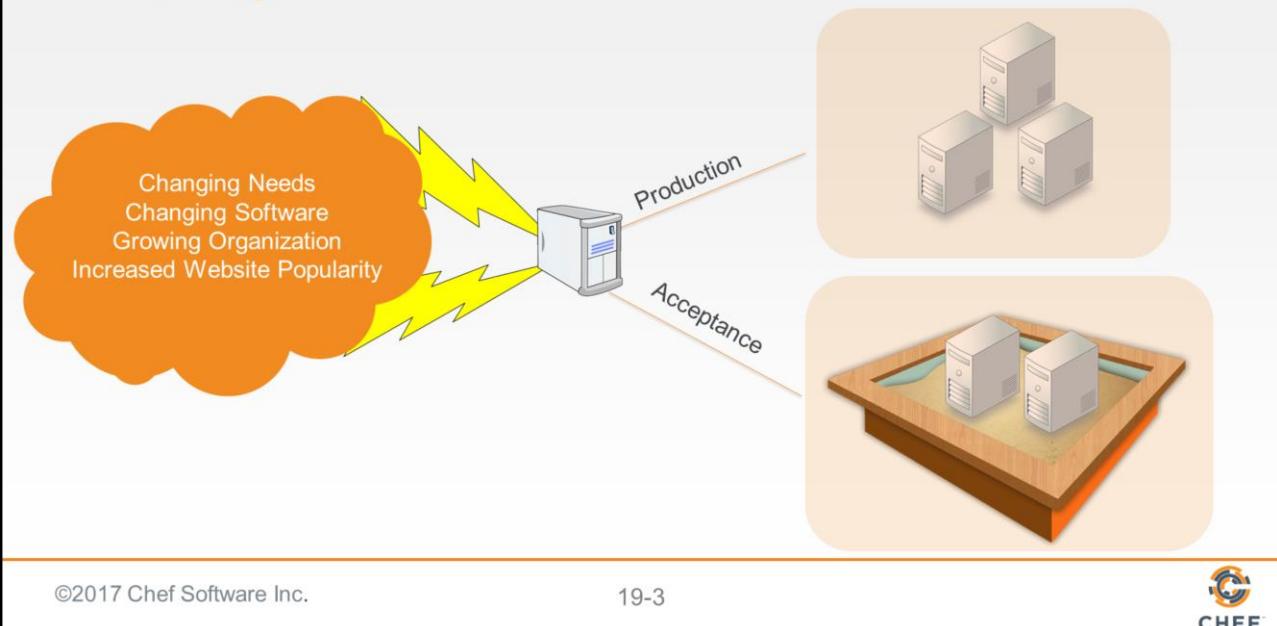
Objectives

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.

Keeping Your Infrastructure Current



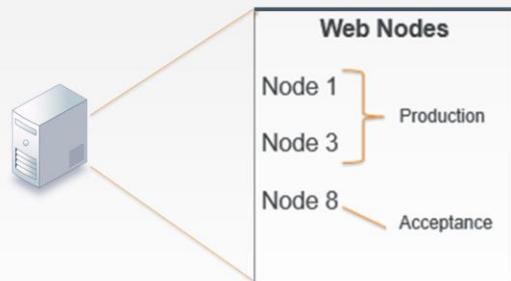
So, we have updated our load balancer's myhaproxy cookbook to dynamically search for and update nodes. Everything is as it should be. But our system is like a living, breathing thing that must grow and be updated to fit our changing needs. We need to find a way to update and test new tools, features 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.

Environments

Environments can define different functions of nodes that live on the same system.

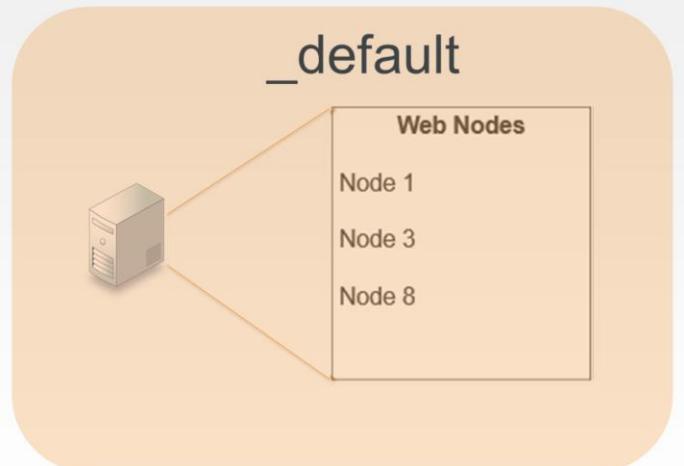


You likely are familiar with the concept of environments. An environment can best be defined as a logical separation of nodes that most often describe the life-cycle of an application. Each environment signifies different behaviors and policies to which a node adheres for a given application or platform.

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.



EXERCISE

Production

Let's create a reliable environment for our nodes.

Objective:

- Create a Production Environment
- Add the IIS web node to Production
- Add the 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'.

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.

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.

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 'company_web', '= 0.1.0'  
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 environment. Okay, so now that we have captured our 'good' environment in this file, let's save it and upload it.

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.

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:
  company_web: = 0.1.0
  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.



EXERCISE

Production

Let's create a reliable environment for our nodes.

Objective:

- Create a Production Environment
- Add the IIS web node to Production
- Add the 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:    lb
Environment: _default
FQDN:          ip-172-31-23-107.ec2.internal
IP:            34.226.220.213
Run List:      recipe[myhaproxy]
Roles:
Recipes:       myhaproxy, myhaproxy::default, haproxy::manual,
               haproxy::install_package
Platform:      centos 6.9
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)
```

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.

Setting the Node's Environment to Production



```
$ knife node environment set iis_web production
```

```
iis_web:  
  chef_environment: production
```

So, let's do that for `iis_web`.

The results don't really tell us much, so let's take a look at `iis_web`.

Viewing the Details about the Node



```
$ knife node show iis_web
```

```
Node Name: iis_web
Environment: production
FQDN: WIN-8694LT97S51.ec2.internal
IP: 34.229.225.40
Run List: role[web_server]
Roles:
Recipes: company_web, company_web::default, myiis::default,
myiis::server
Platform: windows 6.3.9600
Tags:
```

Using knife node show, we can see iis_web's attributes. Note that it has indeed been set to the production environment.



EXERCISE

Production

Let's create a reliable environment for our nodes.

Objective:

- ✓ Create a Production Environment
- ✓ Add the IIS 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 lb production
```

```
lb:  
  chef_environment: production
```

Viewing the Details about the Node



```
$ knife node show lb

Node Name:    lb
Environment:  production
FQDN:         ip-172-31-23-107.ec2.internal
IP:           34.226.220.213
Run List:     recipe[myhaproxy]
Roles:
Recipes:      myhaproxy, myhaproxy::default, haproxy::manual,
               haproxy::install_package
Platform:     centos 6.9
Tags:
```

And, it looks like lb was successfully set to the production environment.



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.



Lab: The Acceptance Environment

- Create an environment named "acceptance" that has no cookbook restrictions
- Move apache_web into the acceptance environment
- Run chef-client on all the nodes

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.

Uploading the Environment



```
$ knife environment from file acceptance.rb
```

```
Updated Environment acceptance
```

Let's upload that .rb file to the Chef server.

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.



Lab: The Acceptance Environment

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- Move apache_web into the acceptance environment
- Run chef-client on all the nodes

Setting the Node to Acceptance Environment



```
$ knife node environment set apache_web acceptance
```

```
apache_web:  
  chef_environment: acceptance
```

Okay, let's set apache_web to the acceptance environment.

Viewing Details about the Node



```
$ knife node show apache_web

Node Name: apache_web
Environment: acceptance
FQDN: ip-172-31-29-209.ec2.internal
IP: 34.207.100.168
Run List: role[web_server]
Roles:
Recipes: company_web, company_web::default, apache::default,
apache::server
Platform: centos 6.9
Tags:
```

And confirm that it has been set properly.



Lab: The Acceptance Environment

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- ✓ Move apache_web into the acceptance environment
- Run chef-client on all the nodes

Converging All the Nodes (Linux)



```
$ knife ssh "os:linux" -x USER -P PWD "sudo chef-client"
```

```
ec2-34-226-220-213.compute-1.amazonaws.com Starting Chef Client, version
13.2.20
ec2-34-207-100-168.compute-1.amazonaws.com Starting Chef Client, version
13.2.20
ec2-34-226-220-213.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-34-207-100-168.compute-1.amazonaws.com resolving cookbooks for run list:
["company_web"]
ec2-34-207-100-168.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-34-207-100-168.compute-1.amazonaws.com   - myiis (0.2.1)
ec2-34-207-100-168.compute-1.amazonaws.com   - company_web (0.1.1)
ec2-34-207-100-168.compute-1.amazonaws.com   - apache (0.1.0)
ec2-34-207-100-168.compute-1.amazonaws.com Installing Cookbook Gems:
```

Using the knife ssh let's run chef client on all the nodes.

Converging All the Nodes (Windows)



```
$ knife winrm "os/windows" -a cloud.public_ipv4 -x  
USER -P PWD "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 winrm let's run chef client on all the nodes.



Lab: The Acceptance Environment

- ✓ Create an environment named "acceptance" that has no cookbook restrictions
- ✓ Move apache_web into the acceptance environment
- ✓ Run chef-client on all the nodes



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 the load balancer node

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_server')

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) 2017 The Authors, All Rights Reserved.

all_web_nodes = search('node',"role:web_server 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.



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 the load balancer node

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', '~> 3.0.0'
```

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.

Uploading the Cookbook to Chef Server



```
$ berks upload
```

```
Uploaded myhaproxy (1.0.1) to: 'https://api.chef.io:443/organizations/2017_oct_26'
Skipping ohai (5.2.0) (frozen)
Skipping poise (2.8.1) (frozen)
Skipping poise-service (1.5.2) (frozen)
Skipping seven_zip (2.0.2) (frozen)
Skipping windows (3.2.0) (frozen)
```

And finally berks upload.



A Brief Recap

We restricted the production environment to a specific cookbook versions and we then updated the myhaproxy cookbook.

What must we do next for this change to be applied to our load balancer node?

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.



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 the load balancer node

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 Constraints for the Environment

```
~/chef-repo/environments/production.rb
```

```
name 'production'  
description 'Where we run production code'  
  
cookbook 'company_web', '= 0.1.0'  
cookbook 'myhaproxy', '= 1.0.1'
```

So let's go back into our production.rb and update it to include the new version number.

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:
  company_web: = 0.1.0
  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.



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 the load balancer node

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.

Converging the Load Balancer



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

ec2-34-226-220-213.compute-1.amazonaws.com Starting Chef Client, version
13.2.20
ec2-34-226-220-213.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-34-226-220-213.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-34-226-220-213.compute-1.amazonaws.com   - haproxy (3.0.4)
ec2-34-226-220-213.compute-1.amazonaws.com   - cpu (2.0.0)
ec2-34-226-220-213.compute-1.amazonaws.com   - build-essential (8.0.3)
ec2-34-226-220-213.compute-1.amazonaws.com   - seven_zip (2.0.2)
ec2-34-226-220-213.compute-1.amazonaws.com   - windows (3.2.0)
ec2-34-226-220-213.compute-1.amazonaws.com   - ohai (5.2.0)
ec2-34-226-220-213.compute-1.amazonaws.com   - myhaproxy (1.0.1)
```

And use 'sudo chef-client' to converge the load balancer node.



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 the load balancer node

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.



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.

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



Q&A

What questions can we help you answer?



CHEFTM

©2017 Chef Software Inc.

Further Resources

Other Places to Talk About, Practice, and Learn Chef



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.

The best way to learn Chef is to use Chef



Practice Chef

First, let's talk about stuff you can read to help you learn Chef.



Learn Chef Rally

Interactive learning for those new to Chef.

learn.chef.io





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/modules/beyond-the-basics#/>

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.



Community Resources

Example Cookbooks, Articles, Podcasts, and More

<https://github.com/obazoud/awesome-chef>

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.



Resources You Can Read

A lot of people in the Chef community have written about Chef.

Here are just a few of those resources.

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.

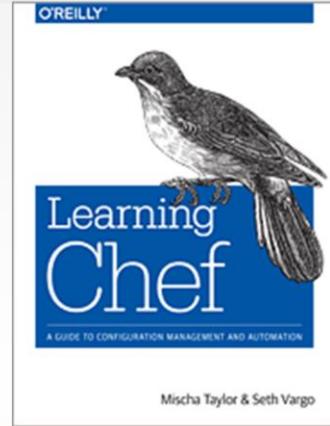
Remember, how often we referred to the Docs site throughout this workshop. That wasn't by accident. We wanted you to become comfortable with using our Docs site to resolve issues and learn about the many Chef tools out there.

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.

REFERENCE

Learning Chef

A Guide to Configuration Management and Automation

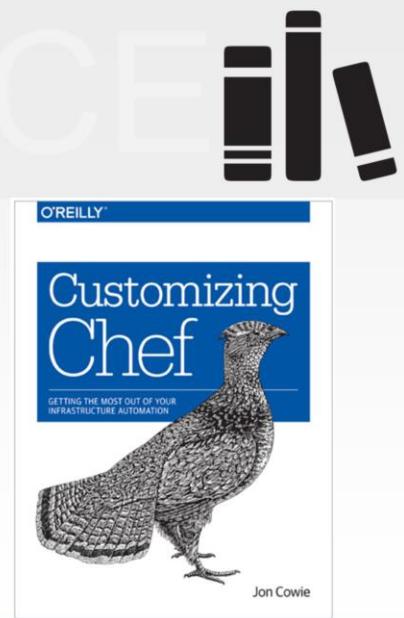


Some people who have used Chef for years have written some excellent books about Chef. Check out Mischa Taylor's and Seth Vargo's Guide to Configuration Management and Automation. You can find it on O'Reilly. It's a great book.

REFERENCE

Customizing Chef

Getting the Most Out of Your Infrastructure Automation



©2017 Chef Software Inc.

20-10

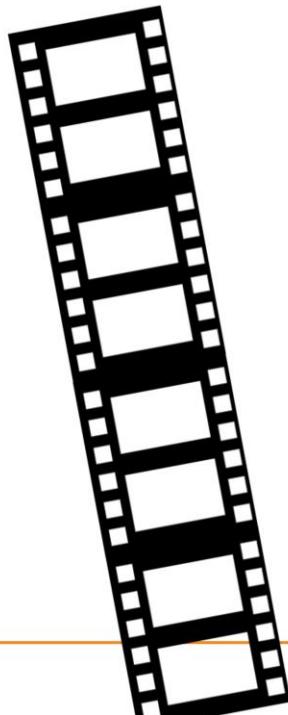


Additionally, you may want to read Jon Cowie's Getting the Most Out of Your Infrastructure Automation. It's also available on O'Reilly.

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.

foodfightshow.org



The Podcast where DevOps chefs do battle

©2017 Chef Software Inc.

20-12



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.

Chef Developers' Slack Meetings

<https://github.com/chef/chef-community-slack-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.

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://www.chef.io/feedback/>

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.



©2017 Chef Software Inc.

<https://chefconf.chef.io/>



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 2018 will be held in Chicago during May.



©2017 Chef Software Inc.

<https://www.chef.io/summit/>



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.



CHEFTM

Thank You!