

# ESSENTIALS WINDOWS



Chef Essentials Windows v 2.0.0

# Introduce Yourselves

Name

Current job role

Previous job roles/background

Experience with Chef and/or config management

**Expectations**

# Expectations

You will leave this class with a basic understanding of Chef's core components, architecture, commonly used tools, and basic troubleshooting methods.

You bring with you your own domain expertise and problems. Chef is a framework for solving those problems. Our job is to teach you how to express solutions to your problems with Chef.

# Course Objectives

After completing this course, you should be able to:

- Use Chef Resources to define the state of your system
- Write and use Chef recipes and cookbooks
- Manage multiple nodes with Chef Server
- Create Organizations
- Bootstrap nodes
- Apply Integration Testing and Unit Testing

# Agenda

## Day 1

---

Getting a Workstation  
Using Resources  
Building Cookbooks  
chef-client  
Ohai and the Node Object  
Roles  
Connecting to Chef Server  
Community Cookbooks

## Day 2

---

Chef Search  
Why Write Tests?  
Writing a Test First  
Refactoring Cookbooks with Tests  
Faster Feedback with Unit Testing  
Testing Resources in Recipes  
Refactoring to Attributes

# Chef

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

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

# 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 reach fluency very fast but it will take practice until you become comfortable.

**A great way to learn Chef is to use Chef**

# Chef Fundamentals

**Ask Me Anything:** It is important that we answer your questions and set you on the path to find more.

**Break It:** If everything works the first time go back and make some changes. Break it!

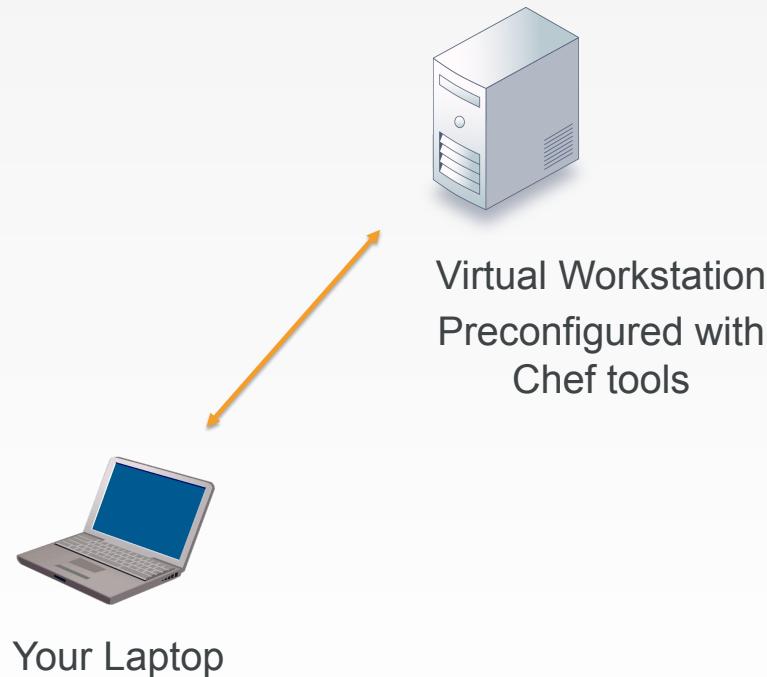
# Chef Lab System Architecture

In this course you will use two different architectures:

1. Initially, you'll use a virtual workstation so you can start using Chef right away.
2. Later, you'll use a common production type of architecture that includes a Chef Server.

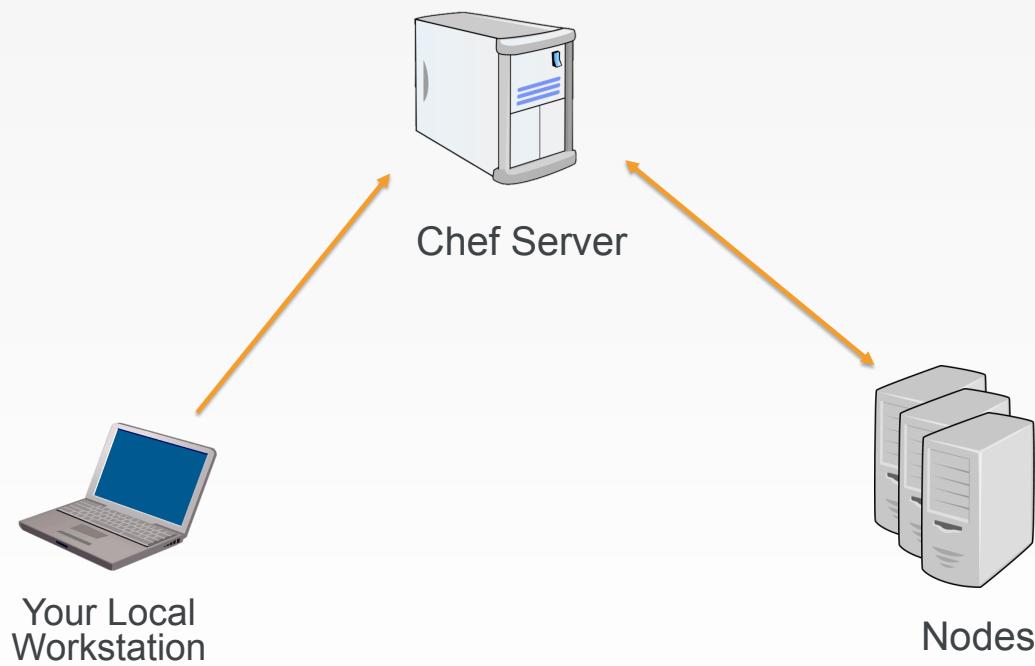
# Chef Lab System Architecture

Architecture 1



# Chef Lab System Architecture

Architecture 2



# Configuring a Workstation

We need the following:

- Chef Development Kit (ChefDK)
- Editor (we recommend Atom or Visual Studio Code)

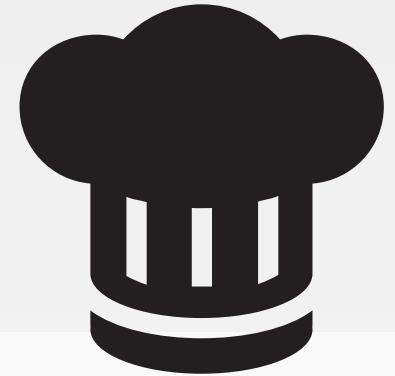
# Hands-on Legend

- GL or Group Lab: All participants and the instructor do this task together with the instructor often leading the way and explaining things as we proceed.
- Lab: You perform this task on your own.

LAB

## Group Lab: Pre-built Workstation

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

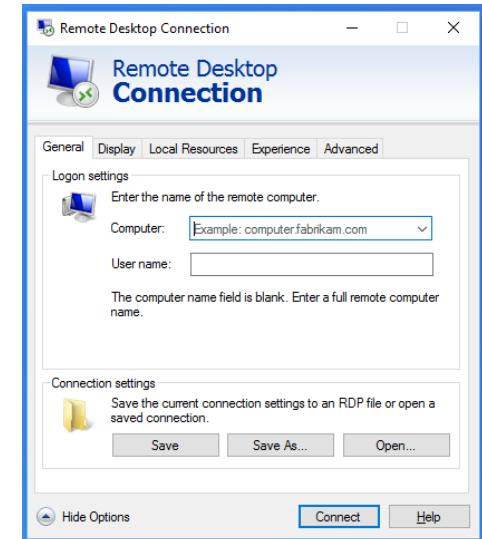


### OBJECTIVE:

- Login to the Remote Workstation

# GL: Login to the Remote Workstation

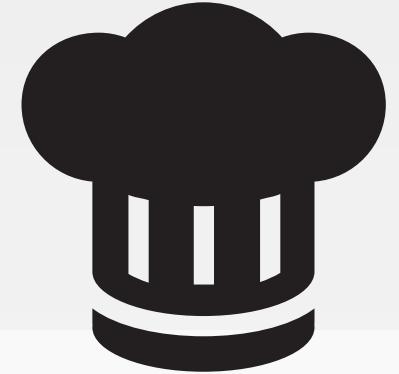
Use the **address**, **user name**, and **password** to connect to the remote workstation.



LAB

## Group Lab: Pre-built Workstation

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



### OBJECTIVE:

- ✓ Login to the Remote Workstation



CHEF™



# Chef Resources

Chef's Fundamental Building Blocks

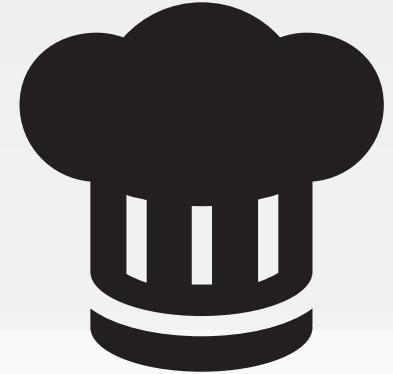
# Objectives

After completing this module, you should be able to:

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

LAB

## 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

# DOCS

## Resources



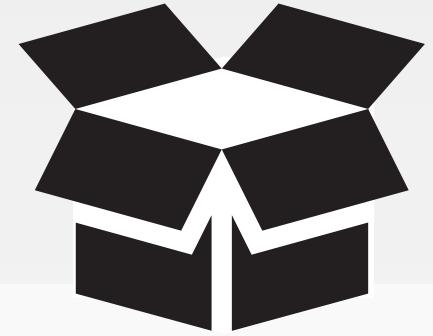
A resource is a statement of configuration policy.

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>

# CONCEPT

## Resource Definition

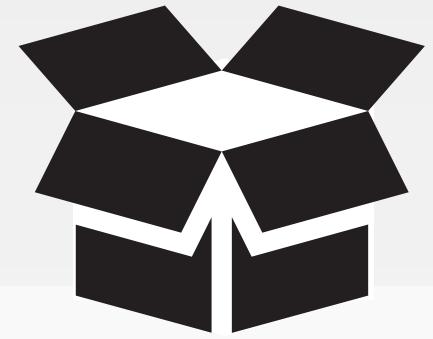


```
file 'C:\hello.txt' do
  content 'Hello, world!'
  action :create
end
```

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

# CONCEPT

## Resource Definition

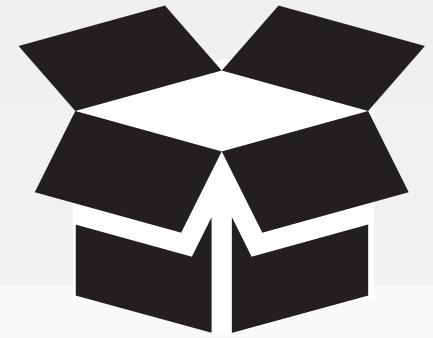


```
file 'C:\hello.txt' do
  content 'Hello, world!'
  action :create
end
```

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

# CONCEPT

## Resource Definition

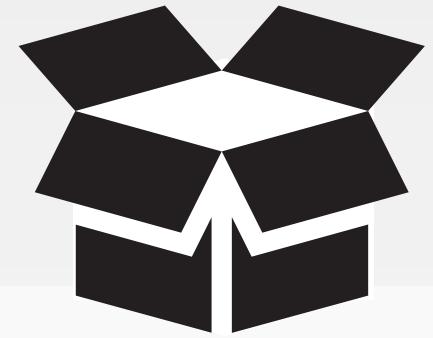


```
file 'C:\hello.txt' do
  content 'Hello, world!'
  action :create
end
```

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

# CONCEPT

## Resource Definition

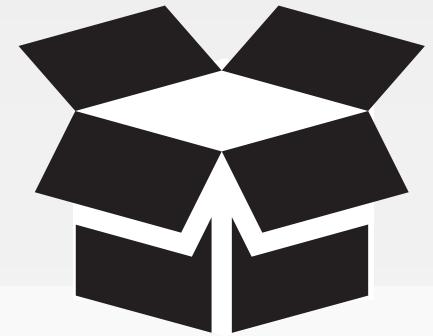


```
file 'C:\hello.txt' do
  content 'Hello, world!'
  action :create
end
```

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

# CONCEPT

## Resource Definition



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

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

# Example Resource: file creation

```
file 'C:\inetpub\wwwroot\Default.htm' do
  content 'Hello, world!'
  rights :read, 'Everyone'
  action :create
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](https://docs.chef.io/resource_file.html)

# Example Resource: file deletion

```
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](https://docs.chef.io/resource_file.html)

# Example Resource: service enable/start

```
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](https://docs.chef.io/resource_service.html)

# Example resource: 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](https://docs.chef.io/resource_powershell_script.html)

# Example: Full Recipe

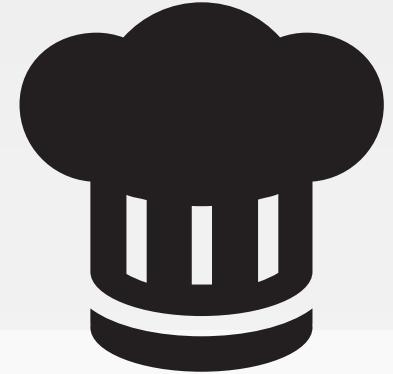
```
powershell_script 'Install IIS' do
  code 'Add-WindowsFeature Web-Server'
  action :run
end

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

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

LAB

## 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:

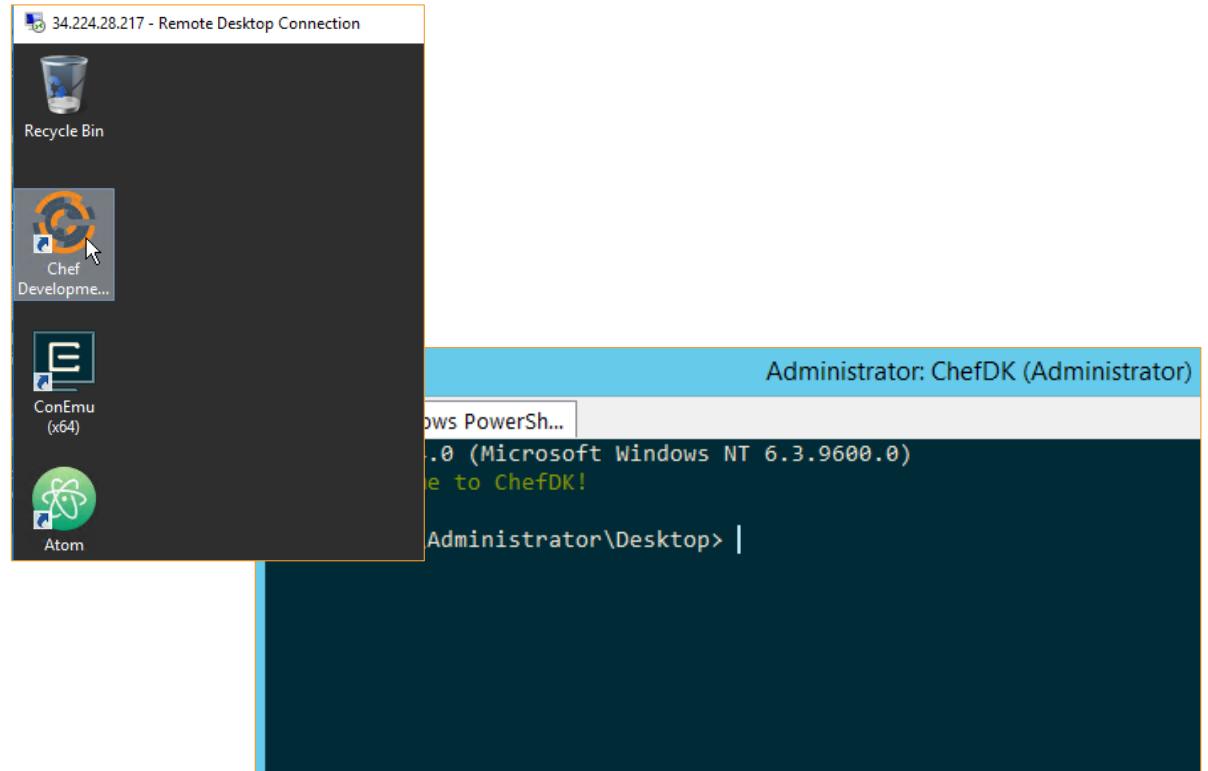
- Create a recipe file writes out 'Hello, world!' to a text file
- Apply the recipe to the workstation

# GL: Launch the ChefDK PowerShell

Double-click the  
ChefDK icon on  
your virtual  
workstation.



This will launch a  
Chef version of  
PowerShell.



# GL: Navigate to the Home Directory



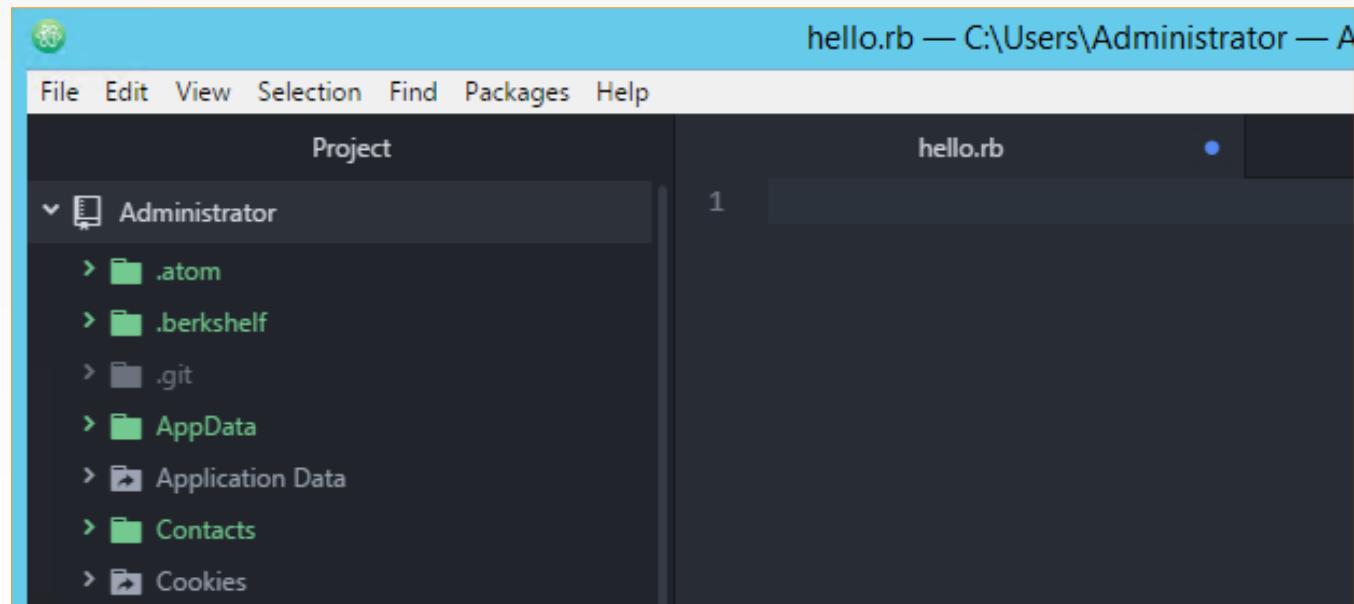
```
> cd ~
```

```
PS C:\Users\Administrator>
```

# GL: Create and Open a Recipe File



```
C:\Users\Administrator> atom hello.rb
```



# GL: Create a Recipe File Named hello.rb

```
1 ~\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>

LAB

## Group Lab: Hello, World?



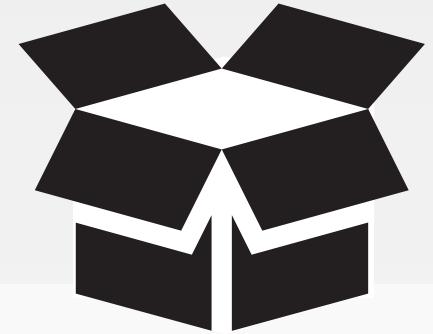
*I heard Chef is written in Ruby. If that's the case it's required that we write a quick "Hello, world!" application.*

### OBJECTIVE:

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

# CONCEPT

## chef-client



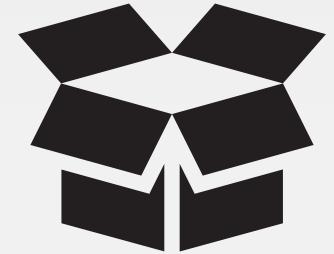
chef-client is an agent that runs locally on every node that is under management by Chef.

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

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

# CONCEPT

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



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

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

# GL: Apply a Recipe File



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

```
...
[2018-09-08T16:48:26+00:00] WARN: No cookbooks directory found at or above current directory.
Assuming C :/Users/Administrator.

Starting Chef Client, version 13.3.42
...
Converging 1 resources

Recipe: @recipe_files::C:/Users/Administrator/hello.rb
  * file[C:\hello.txt] action create
    - create new file C:\hello.txt
    - update content in file C:\hello.txt from none to 315f5b
      --- C:\hello.txt      2018-09-08 16:49:39.000000000 +0000
      +++ C:\chef-hello20180908-3804-txt4m7.txt      2018-09-08 16:49:39.000000000 +0000
      @@ -1 +1,2 @@
      +Hello, world! ...
```

# GL: What Does hello.txt Say?



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

```
Hello, world!
```

## Discussion



What would happen if you ran the command again?

What would happen if the file were removed?

# CONCEPT

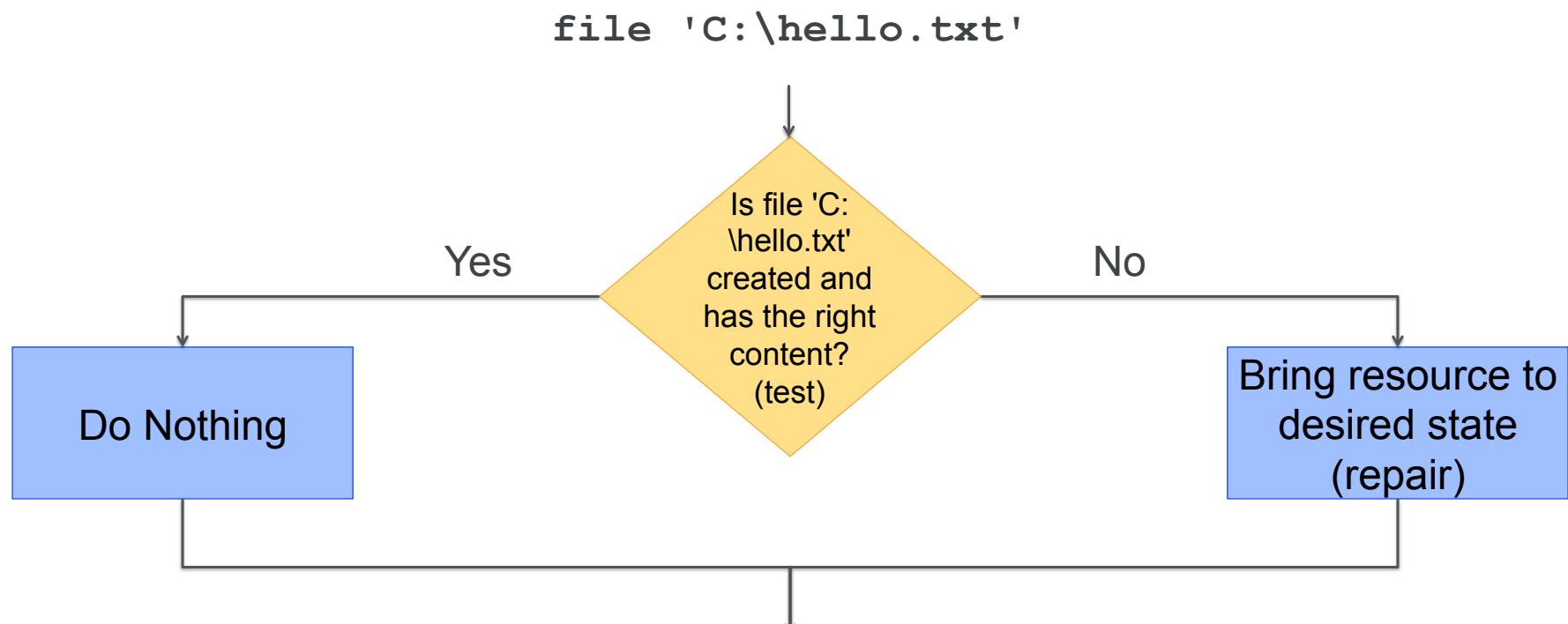
## Test and Repair



`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.

# Test and Repair





## Lab: Test and Repair

What would happen if the file contents were modified?

- Modify the contents of 'C:\hello.txt' manually with your text editor
- Run the chef-client command again

# Modify the Contents of the File

C:\hello.txt

Goodbye, world!

# GL: Now What Does hello.txt Say?



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

```
Goodbye, world!
```

# Re-apply the Policy Defined in the Recipe



```
C:\Users\Administrator> chef-client --local-mode hello.rb
```

```
Starting Chef Client, version 13.3.42
...
Converging 1 resources
Recipe: @recipe_files::C:/Users/Administrator/hello.rb
  * file[C:\hello.txt] action create
    - update content in file C:\hello.txt from 2ffd26 to 315f5b
      --- C:\hello.txt      2018-09-08 19:29:21.000000000 +0000
      +++ C:\chef-hello20180908-3068-129bjez.txt  2018-09-08 19:30:18.000000000
+0000
        @@ -1,2 +1,2 @@
        -Goodbye, world!
        +Hello, world!
```

# GL: What Does hello.txt Say?



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

```
Hello, world!
```



# Lab: The `file` Resource

- Read** <https://docs.chef.io/resources.html>

- Discover the `file` resource's:**

- default action
  - `rights` attribute

- Update the `file` policy in "hello.rb" to:**

The file named 'C:\hello.txt' is created with 'read' rights for 'Everyone'.

# Lab: The Updated file Resource

~\hello.rb

```
file 'C:\hello.txt' do
  content 'Hello, world!'
  rights :read, 'Everyone'
  action :create
end
```

This sets the file's rights to allow 'Everyone' access.

The default action is to create (not necessary to define it).

# Questions

What questions can we answer for you?





## Lab: Goodbye Recipe

- Create a recipe file named "goodbye.rb" that defines the policy:  
The file named 'C:\hello.txt' is deleted.
- Use chef-client to apply the recipe file named "goodbye.rb"

# Lab: The Updated file Resource

~\goodbye.rb

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

# Lab: Apply a Recipe File



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

```
C:/Users/Administrator.  
Starting Chef Client, version 13.3.42  
...  
Compiling Cookbooks...  
[2018-09-08T19:50:35+00:00] WARN: Node WIN-VBVUIACREJ.ec2.internal has an  
empty run list.  
Converging 1 resources  
Recipe: @recipe_files::C:/Users/Administrator/goodbye.rb  
  * file[C:\hello.txt] action delete  
    - delete file C:\hello.txt  
Running handlers:  
Running handlers complete  
Chef Client finished, 1/1 resources updated in 04 seconds
```

## Lab: Test that the File was Deleted

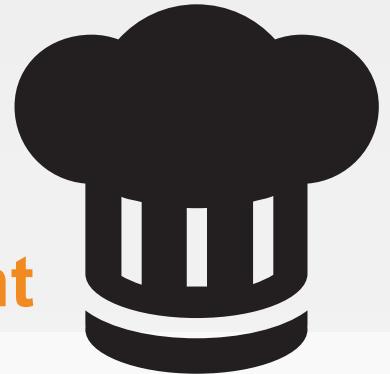


```
> Test-Path C:\hello.txt
```

```
False
```

# LAB

## 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

# GL: Disable the Limited User Account

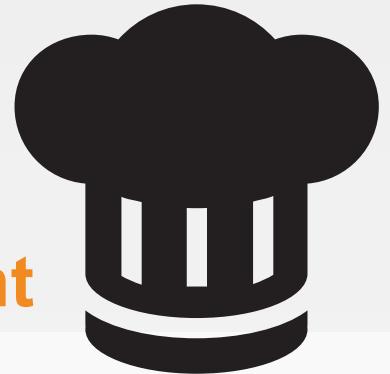
```
[ ] ~\disable-uac.rb
```

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

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

# LAB

## 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

**NOTE: PLEASE DO NOT RESTART YOUR COMPUTER WHEN PROMPTED!**

# GL: Apply a Recipe File



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

```
Starting Chef Client, version 13.3.42
resolving cookbooks for run list: []

...
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
  - set value {:name=>"EnableLUA", :type=>:dword, :data=>0}
...

Running handlers complete
Chef Client finished, 1/1 resources updated in 04 seconds
[2018-09-08T19:54:40+00:00] WARN: No config file found or specified on command
line, using command line
```

options



## 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"

# GL: 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 ON THIS SLIDE TO CONSERVE SPACE)
end

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

# Lab: Apply a Recipe File



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

```
Starting Chef Client, version 13.3.42
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks...
[2018-09-08T20:41:05+00:00] WARN: Node WIN-VBVUIIACREJ.ec2.internal has an empty run list.
Converging 2 resources
Recipe: @recipe_files::C:/Users/Administrator/disable-uac.rb
  * 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
      - set value {:name=>"ConsentPromptBehaviorAdmin", :type=>:dword, :data=>0}

Running handlers:
```

# 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?

## Q&A



What questions can we answer for you?

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



CHEF™



# Cookbooks

Organizing Recipes

©2018 Chef Software Inc.



# Objectives

After completing this module you should be able to:

- Generate a Chef cookbook
- Define a Chef recipe that sets up a web server



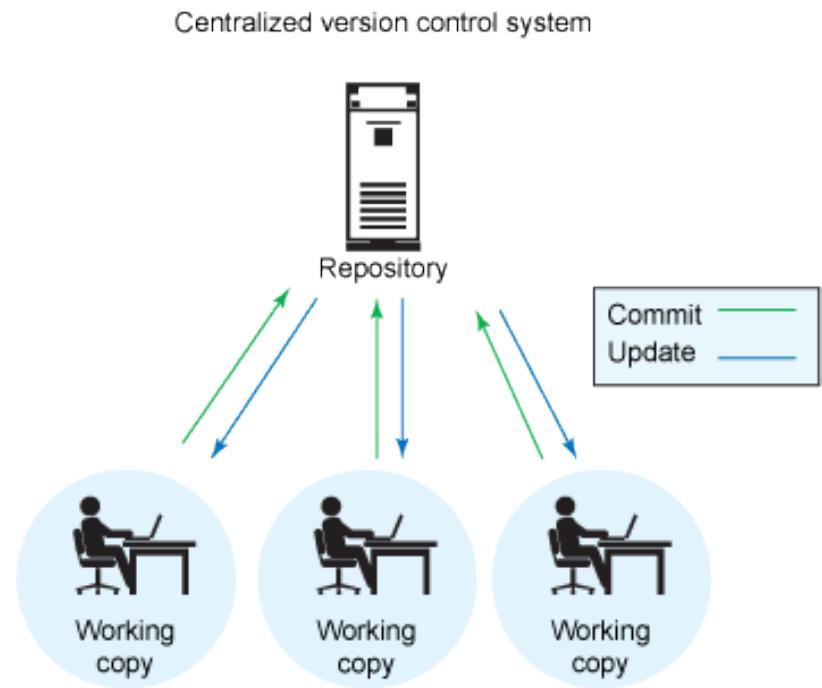
## Questions You May Have

1. Is there a way to package up recipes you create with a version number and a README?
2. If we have multiple versions - how might we better track these different changes and versions?
3. Thinking about the previous recipes, could we create a recipe to setup a web server?

# Collaboration and Version Control

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.



# Collaboration and Version Control

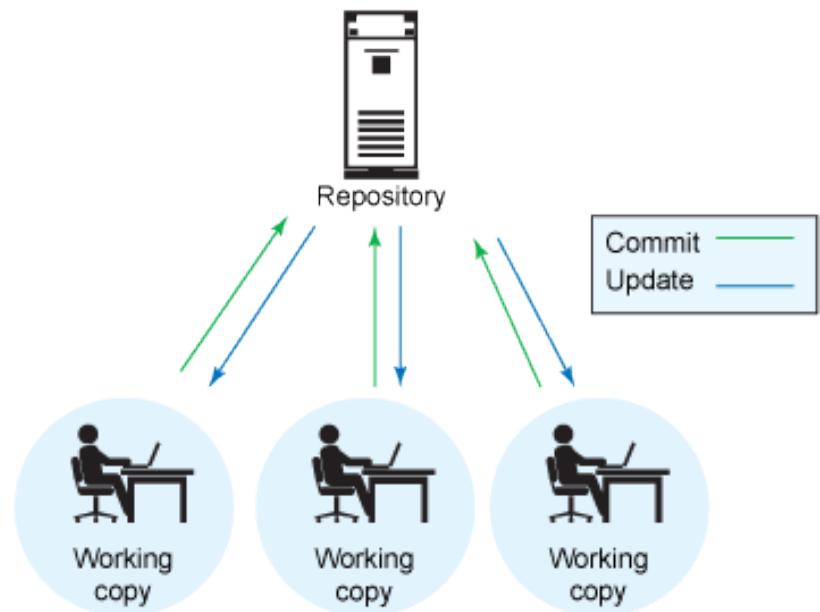
We won't cover versioning control systems, such as Git, in this course. But it's something you should eventually learn to use.

There are plenty of resources on the web that you can use to learn versioning control.

You can also get started by going through this *Learn Chef Rally* module.

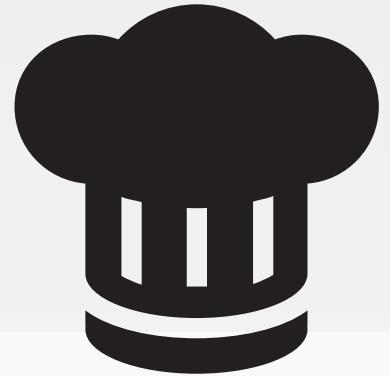
<https://learn.chef.io/modules/version-control#/>

Centralized version control system



# LAB

## GL: Create a Cookbook



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

### OBJECTIVE:

- Use chef to generate a cookbook to store our disable-uac recipe.

# CONCEPT



## What is 'chef'?

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

# 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

# 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>



## GL: Ensure You are in Your Home Directory



```
> cd ~
```

# GL: Create a Cookbooks Directory



```
> C:\Users\Administrator> mkdir cookbooks
```

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

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	9/13/2018 3:42 PM		cookbooks

# 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)

# 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 to ...	Use to set arbitrary attribute KEY
-I, --license LICENSE	all_rights, httpd, mit, gplv2, gplv3 -
-g GENERATOR_COOKBOOK_PATH, --generator-cookbook	Use GENERATOR_COOKBOOK_PATH for the

# GL: 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`

# GL: The Cookbook Has a README



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

```
Volume serial number is 2CB6-07DF
C:\USERS\ADMINISTRATOR\COOKBOOKS\WORKSTATION
|
|   .gitignore
|   .kitchen.yml
|   Berksfile
|   chefignore
|   metadata.rb
|   README.md
|
|   |--.delivery
|   |   config.json
|   |   project.toml
|   |
|   |   ...
|
```

# CONCEPT

## README.md



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

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

# GL: 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
|
```

# DOCS

## 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](http://docs.chef.io/config_rb_metadata.html)

# GL: 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 Reserved'
description 'Installs/Configures workstation'
long_description 'Installs/Configures workstation'
version '0.1.0'
chef_version '>= 13.0' if respond_to?(:chef_version)
```

# GL: 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
|   cheffignore
|   metadata.rb
|   README.md
|
|---recipes
|     default.rb
|
```

# GL: The Cookbook Has a Default Recipe



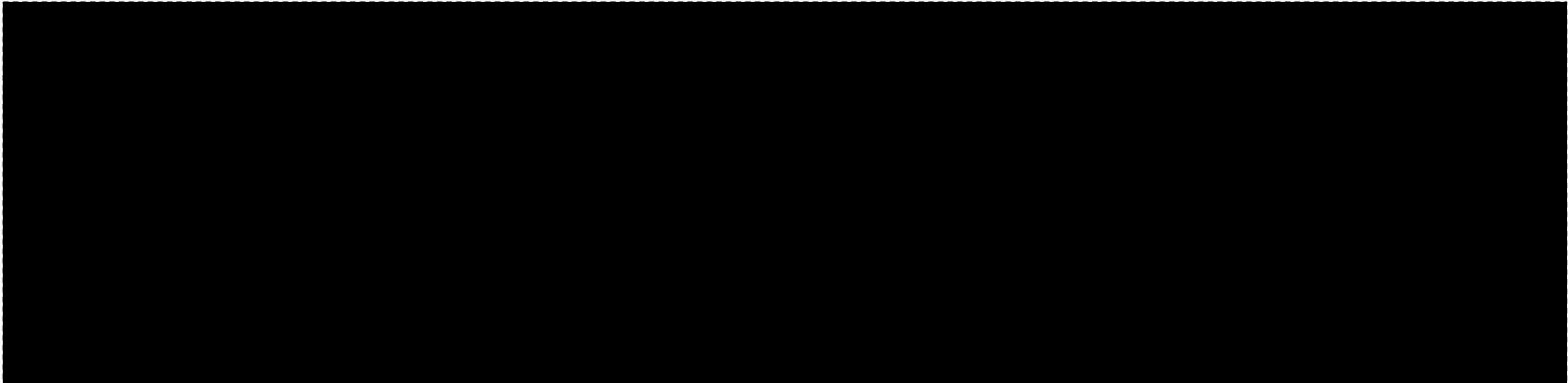
```
> gc cookbooks\workstation\recipes\default.rb
```

```
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2018 The Authors, All Rights Reserved.
```

# GL: Copy the Recipe into the Cookbook



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





## 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 the site is available by running `Invoke-WebRequest localhost`

# Lab: Create a Cookbook



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

Generating cookbook myiis

- 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\myiis` to enter it..

# Lab: Create a Recipe



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

```
...
* template[cookbooks/myiis/spec/unit/recipes/server_spec.rb] action create_if_missing
  - create new file cookbooks/myiis/spec/unit/recipes/server_spec.rb
  - update content in file cookbooks/myiis/spec/unit/recipes/server_spec.rb from none to
53eaa6
  (diff output suppressed by config)
* directory[cookbooks/myiis/test/smoke/default] action create (up to date)
* template[cookbooks/myiis/test/smoke/default/server_test.rb] action create_if_missing
  - create new file cookbooks/myiis/test/smoke/default/server_test.rb
  - update content in file cookbooks/myiis/test/smoke/default/server_test.rb from none to
abd78c
  (diff output suppressed by config)
* template[cookbooks/myiis/recipes/server.rb] action create
  - create new file cookbooks/myiis/recipes/server.rb
  - update content in file cookbooks/myiis/recipes/server.rb from none to 8fb075
  (diff output suppressed by config)
```

# Lab: Create Server Recipe

```
1 ~\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
```

# Lab: Apply the Server Recipe



```
> chef-client -z cookbooks\myiis\recipes\server.rb
```

```
...
Recipe: @recipe_files::C:/Users/Administrator/cookbooks/myiis/recipes/server.rb
"C:/Users/ADMINI~1/AppData/Local/Temp/2/chef-script2018091
pii.ps1"
  * file[C:\inetpub\wwwroot\Default.htm] action create
    - create new file C:\inetpub\wwwroot\Default.htm
    - update content in file C:\inetpub\wwwroot\Default.htm from none to 17d291
      --- C:\inetpub\wwwroot\Default.htm 2018-09-13 21:00:07.000000000 +0000
      +++ C:\inetpub\wwwroot\chef-Default20180913-3148-u3dbpm.htm 2018-09-13
21:00:07.000000000 +0000
        @@ -1 +1,2 @@
        +<h1>Hello, world!</h1>
  * windows_service[w3svc] action enable (up to date)
  * windows_service[w3svc] action start (up to date)
```

# 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
...
          ETag: "9e9c7b47d32cd31:0"
          Last-Modified: Wed, 13 Sep 2018 21:00:07 GMT
          Server...
Forms              : {}
Headers            : {[Accept-Ranges, bytes], [Content-Length, 22], [Content-Type, text/html], [Date, Wed, 13 Sep 2018 21:02:57 GMT]...}
Images             : {}
```

# Discussion



What file should you read first when examining a cookbook?

Can resources accept multiple actions?

## Q&A



What questions can we answer for you?

- Cookbooks



CHEF™



# chef-client

Applying Recipes from Cookbooks

©2018 Chef Software Inc.



# Objectives

After completing this module, you should be able to use chef-client to:

- Locally apply multiple cookbooks' recipes with chef-client
- Include a recipe from within another recipe

# PROBLEM

## chef-client

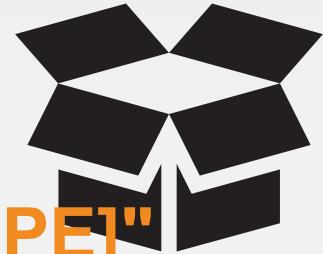


```
> chef-client --local-mode RECIPE_FILE
```

How would we apply both the workstation's setup recipe and apache's server recipe?

# CONCEPT

**--run-list "recipe[COOKBOOK::RECIPE]"**

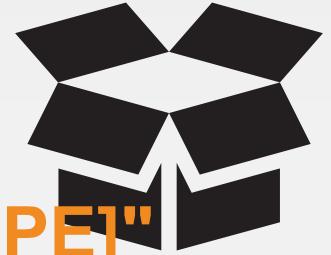


In local mode, we need to provide a list of recipes to apply to the system. This is called the **run list**, and is the ordered collection of recipes to execute

Each recipe in the run list must be addressed with the format **recipe[COOKBOOK::RECIPE]**

# CONCEPT

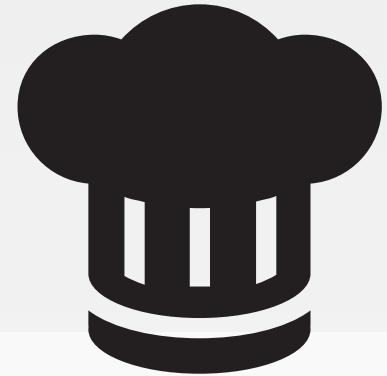
**--run-list "recipe[COOKBOOK::RECIPE]"**



-r is the shortcut for --run-list

# LAB

## Applying a Run List



*Using a run list will allow us to specify things more 'logically' instead of with paths.*

### OBJECTIVE:

- Individually apply the myiis cookbook's server recipe and workstation cookbook's disable-uac recipe
- Apply both of the recipes at the same time

## Demo: Using 'chef-client' to Locally Apply Recipes

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

Applying the following recipes locally:

The 'disable-uac' recipe from the 'workstation' cookbook

## Demo: Using 'chef-client' to Locally Apply Recipes

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

Applying the following recipes locally:

The 'server' recipe from the 'myiis' cookbook

**Bonus question: why does one resource show as updated?**

## Demo: Using 'chef-client' to Locally Apply Recipes

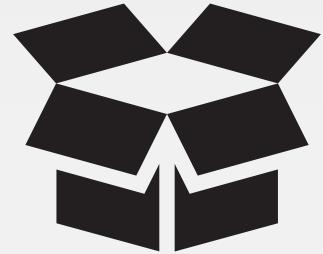
```
> chef-client --local-mode -r "recipe[workstation::disable-uac], recipe[myiis::server]"
```

Applying the following recipes locally:

- The 'disable-uac' recipe from the 'workstation' cookbook
- The 'server' recipe from the 'myiis' cookbook

# CONCEPT

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



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

**Example: -r “recipe[workstation]”**

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

# DOCS

## **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>

# Demo: Including a Recipe

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

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

# Demo: Including a Recipe

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

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

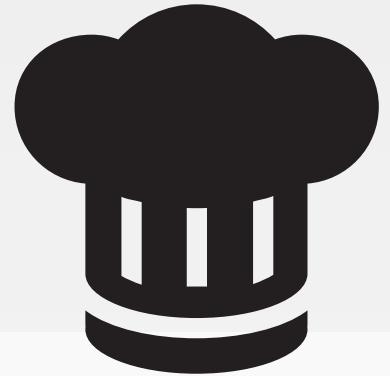
## GL: The Default Recipe Includes the Disable Recipe

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

```
#  
# Cookbook Name:: workstation  
# Recipe:: default  
#  
# Copyright (c) 2018 Chef Software Inc.  
# Copyright (c) 2018 The Authors, All Rights Reserved.  
  
include_recipe 'workstation::disable-uac'
```

# LAB

## 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

# GL: Apply the Cookbook's Default Recipe



```
> chef-client -z -r "recipe[workstation]"
```

```
...
Synchronizing Cookbooks:
  - workstation (0.1.0)
Installing Cookbook Gems:
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: ...
```



## Lab: Update the myiis Cookbook

- 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]"

## Lab: The Default Recipe Includes the myiis Recipe

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

```
#  
# Cookbook Name:: myiis  
# Recipe:: default  
#  
# Copyright (c) 2018 Chef Software Inc.  
# Copyright (c) 2018 The Authors, All Rights  
Reserved.  
  
include_recipe 'myiis::server'
```

# Lab: Applying the myiis Default Recipe



```
> chef-client -z -r "recipe[myiis]"
```

```
Synchronizing Cookbooks:
```

```
  - myiis (0.1.0)
```

```
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 -Execu
tionPolicy Bypass -InputFormat None -File "C:/Users/ADMINI~1/AppData/Local/Temp/
2/chef-script20180913-3376-d0ss08.ps1"
  * file[C:\inetpub\wwwroot\Default.htm] action create (up to date)
  * windows_service[w3svc] action enable (up to date)
```

# Discussion



Why would you want to apply more than one recipe at a time?

What are the benefits and drawbacks of using "include\_recipe" within a recipe?

## Q&A

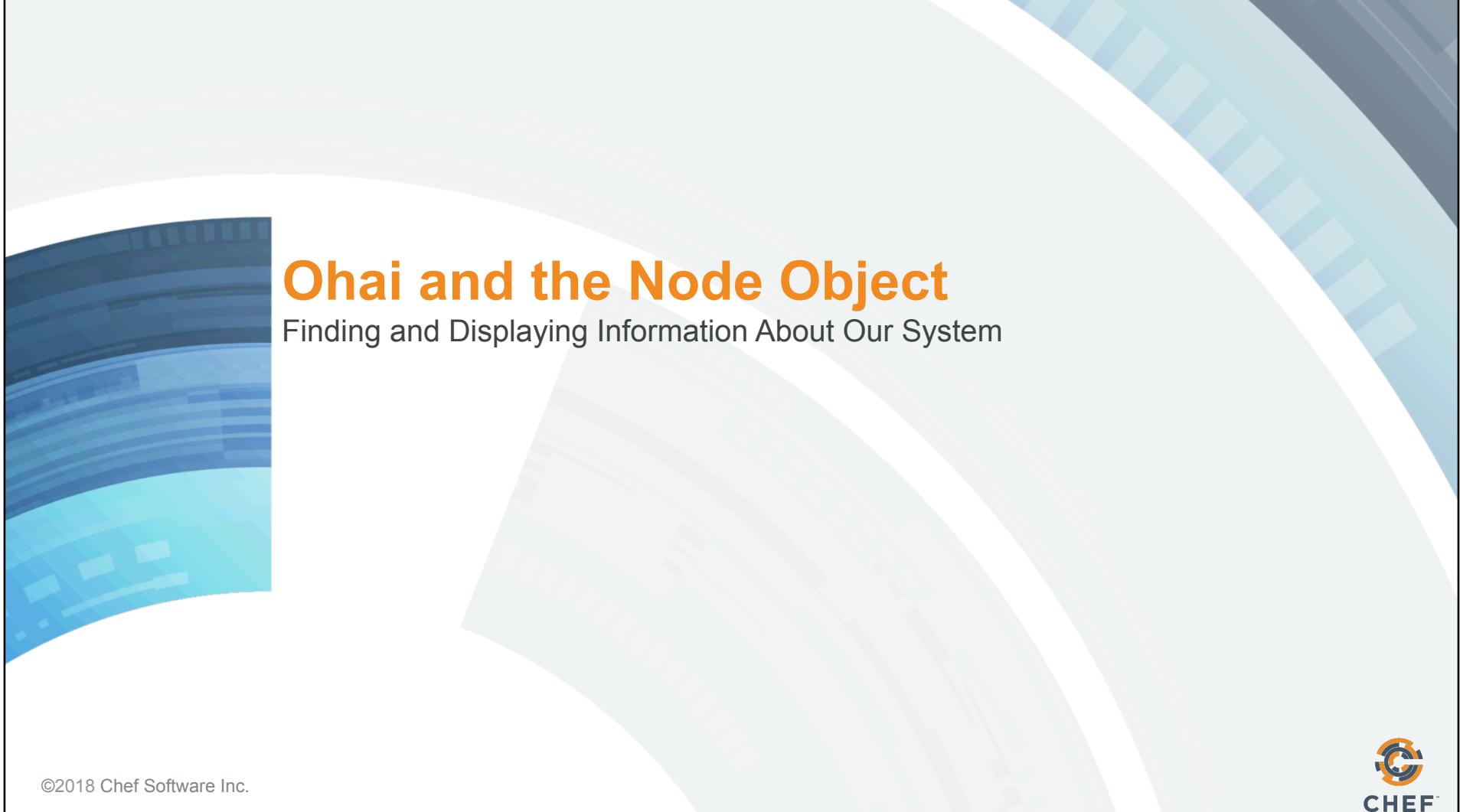


What questions can we help you answer?

- chef-client
- local mode
- run list
- include\_recipe



CHEF<sup>TM</sup>



# Ohai and the Node Object

Finding and Displaying Information About Our System

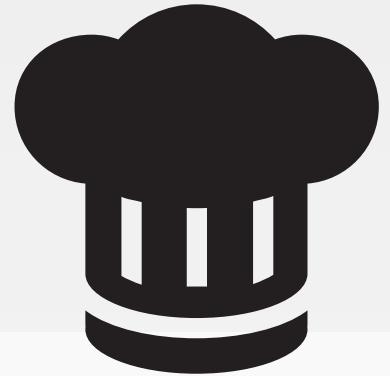
# Objectives

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
- Update the version of a cookbook

# LAB

## Details About the System



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

### OBJECTIVE:

- Find out various details about the system
- Update the web page file contents in the "myiis" cookbook to include system details
- Use chef-client to locally apply the "myiis" cookbook's default recipe

# Some Useful System Data

- IP Address
- hostname
- Memory

# GL: Finding the IP Address



> ipconfig

```
Windows IP Configuration
```

```
Ethernet adapter Ethernet 2:
```

```
Connection-specific DNS Suffix . : ec2.internal
Link-local IPv6 Address . . . . . : fe80::2da8:4ba7:45e2:e863%21
IPv4 Address . . . . . : 172.31.21.21
Subnet Mask . . . . . : 255.255.240.0
Default Gateway . . . . . : 172.31.16.1
```

# GL: Finding the Hostname



```
> hostname
```

```
WIN-KRQSVD3RFM7
```

# GL: Finding the Total Memory



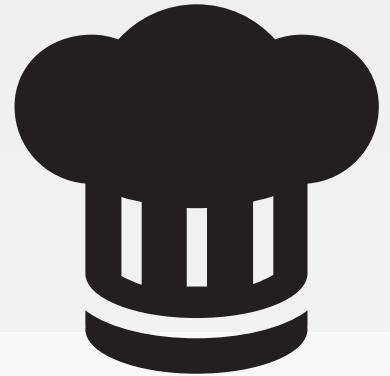
```
> wmic ComputerSystem get TotalPhysicalMemory
```

```
TotalPhysicalMemory
```

```
8052654080
```

# LAB

## Details About the System



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

### OBJECTIVE:

- ✓ Find out various details about the system
- ❑ Update the web page file contents, in the "myiis" cookbook, to include system details
- ❑ Use chef-client to locally apply the "myiis" cookbook's default recipe

# GL: Adding the CPU

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

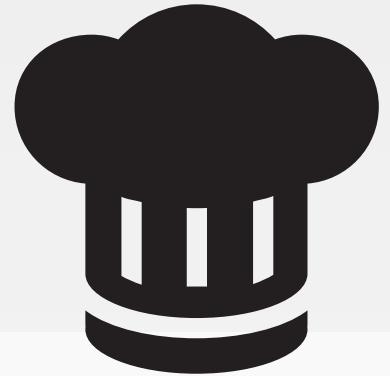
```
# ... POWERSHELL_SCRIPT RESOURCE ...

file 'C:\inetpub\wwwroot\Default.htm' do
  content '<h1>Hello, world!</h1>
<h2>ipaddress: 172.31.21.21</h2>
<h2>hostname: WIN-KRQSVD3RFM7</h2>
<h2>total memory: 8052654080</h2>'
end

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

# LAB

## Details About the Node



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

### OBJECTIVE:

- ✓ Find out various details about the system
- ✓ Update the web page file contents, in the "myiis" cookbook, to include system details
- ❑ Use chef-client to locally apply the "myiis" cookbook's default recipe

## GL: Return Home and Apply myiis Cookbook



```
> cd ~  
> chef-client -z -r "recipe[myiis]"  
  
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 Non  
      e -File "C:/tmp/chef-script20180919-996-rjuiw4.ps1"  
    ...  
  - update content in file C:\inetpub\wwwroot\Default.htm from 17d291 to f37fdd  
    --- C:\inetpub\wwwroot\Default.htm 2018-09-18 20:41:33.000000000 +0000  
    +++ C:\inetpub\wwwroot\chef-Default20180919-996-om7yot.htm 2018-09-19  
      17:25:01.000000000 +0000  
      @@ -1,2 +1,6 @@  
      <h1>Hello, world!</h1>  
      +<h2>ipaddress: 172.31.21.21</h2>
```

## GL: Verify the Default Page Returns the Details



> Invoke-WebRequest localhost

```
StatusCode      : 200
StatusDescription : OK
Content          : <h1>Hello, world!</h1>
                  <h2>ipaddress: 172.31.21.21</h2>
                  <h2>hostname: WIN-KRQSVD3RFM7</h2>
                  <h2>total memory: 8052654080</h2>
RawContent       : HTTP/1.1 200 OK
                  Accept-Ranges: bytes
                  Content-Length: 150
                  Content-Type: text/html
                  Date: Tue, 19 Sep 2018 17:27:19 GMT
```

## Capturing System Data



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

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

# PROBLEM

## 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!

## Data In Real Time

How could we capture this data in real-time?



# CONCEPT

## Ohai!



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

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

# 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-VBVUIIACREJ",  
        ...  
    }  
}
```

# CONCEPT

## All About The System



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

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

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

# CONCEPT

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

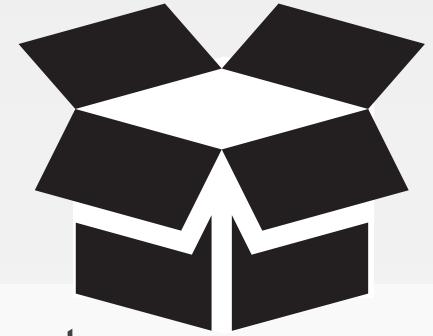


chef-client and chef-apply automatically executes ohai and stores the data about the node in an object we can use within the recipes named node.

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

# CONCEPT

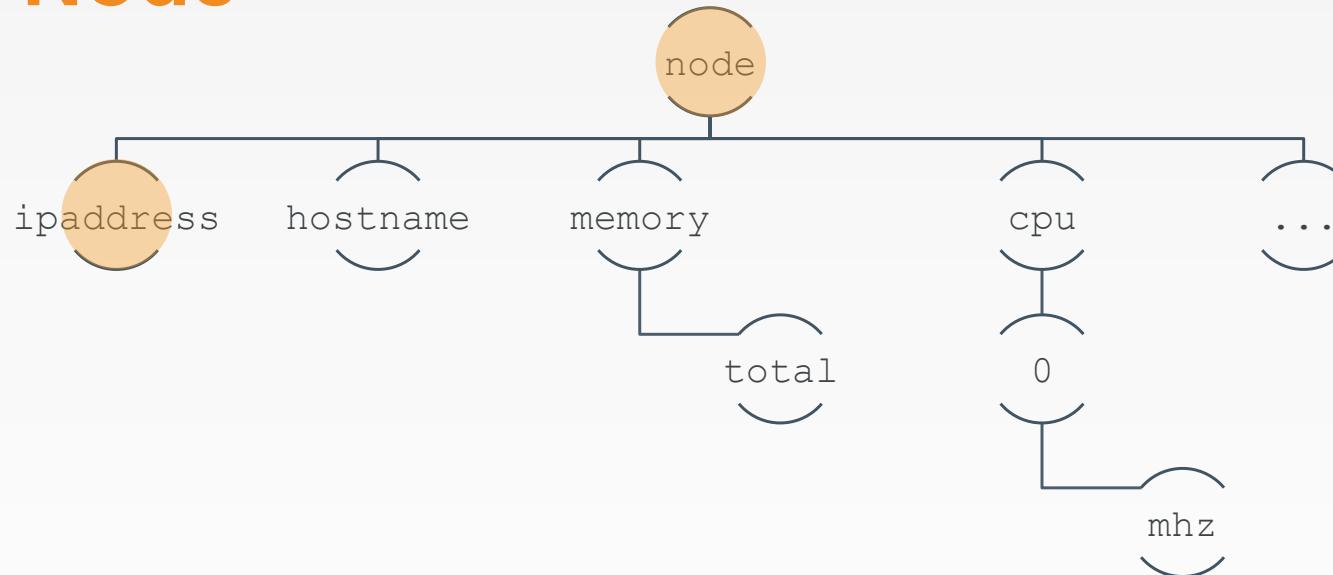
## The Node Object



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

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

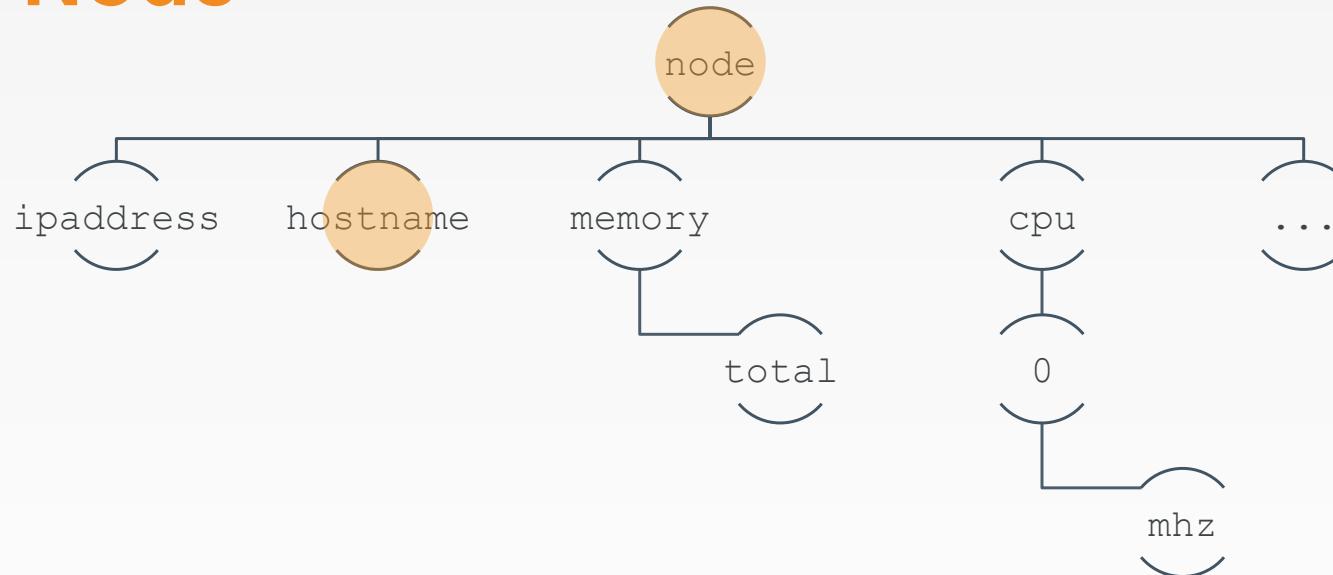
# The Node



IPADDRESS: 172.31.21.21

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

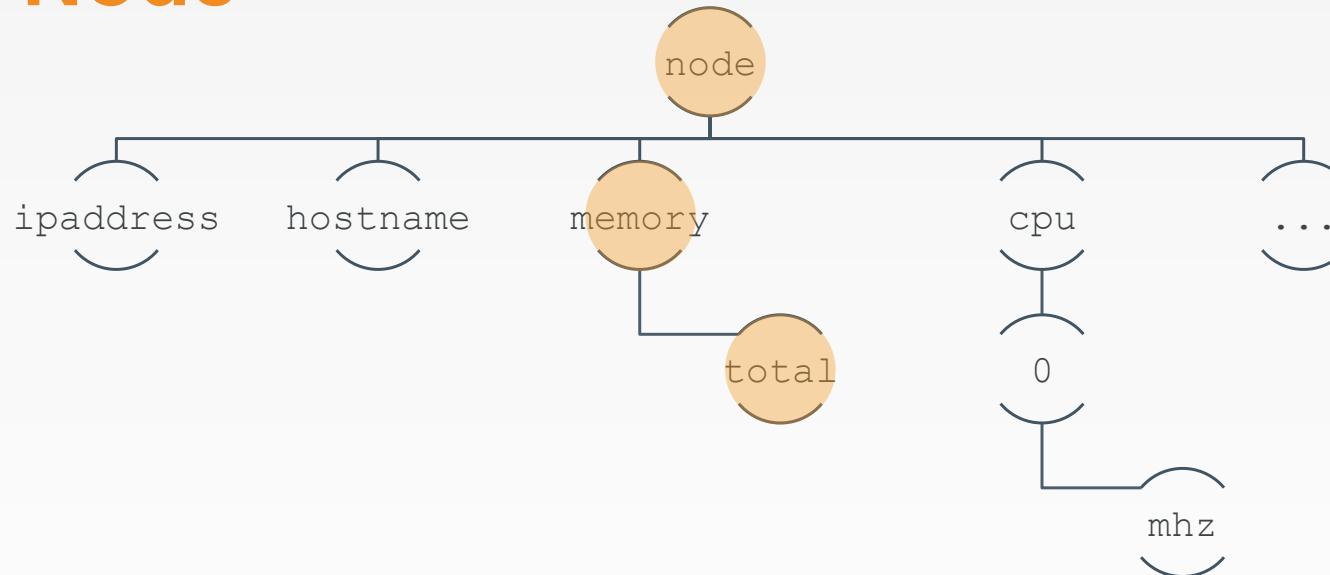
# The Node



**HOSTNAME:** WIN-KRQSVD3RFM7

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

# The Node

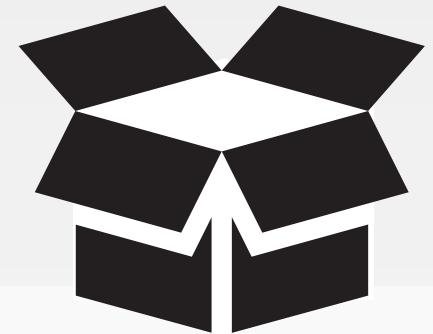


**MEMORY: 7863920kB**

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

# CONCEPT

## String Interpolation



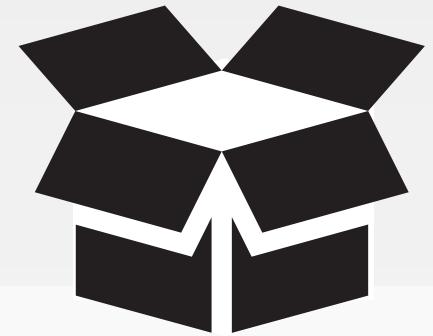
To combine strings and variables:

- Must be in double quotes
- Uses `#{variable}` notation

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

# CONCEPT

## String Interpolation



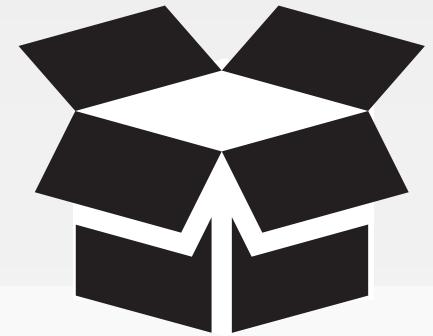
I have 4 apples

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

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

# CONCEPT

## String Interpolation



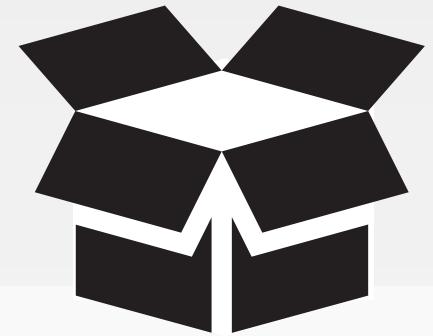
I have 4 apples

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

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

# CONCEPT

## String Interpolation



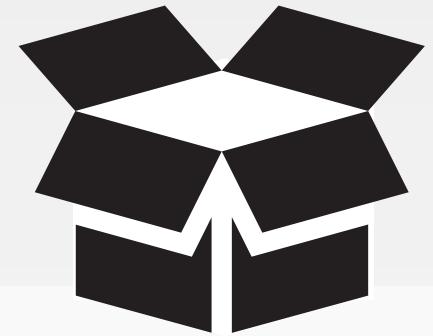
I have 4 apples

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



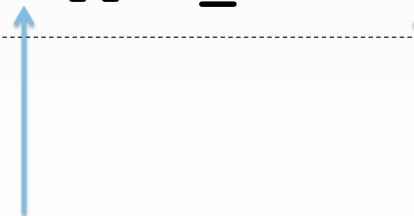
# CONCEPT

## String Interpolation



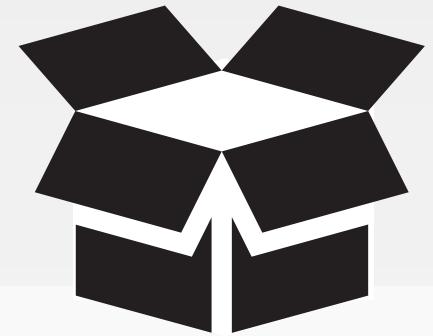
I have 4 apples

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



# CONCEPT

## String Interpolation



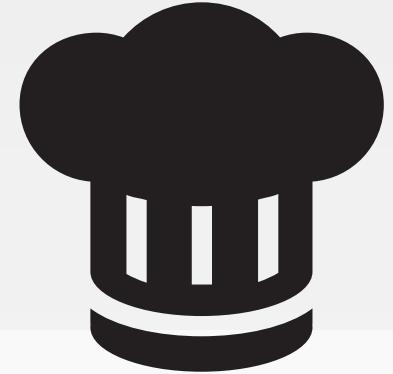
I have 4 apples

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

Output: "I have 4 apples"

# LAB

## Using Node Attributes



*Hard-coding the values was a start, but a better approach would be to replace with the dynamic values found from Ohai.*

### OBJECTIVE:

- Update the web page file contents, in the "myiis" cookbook, to include system details from node attributes.

# GL: Using the Node's Attributes

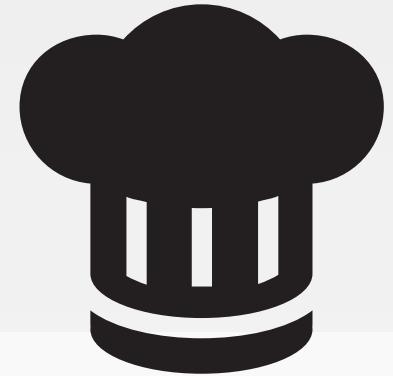
~\cookbooks\myiis\recipe\server.rb

```
# ... 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']}
```

# LAB

## Using Node Attributes



*That feels much better!*

### OBJECTIVE:

- ✓ Update the web page file contents, in the "myiis" cookbook, to include system details from node attributes.



## Lab: Verify the Changes

- Change directory into the home directory
- Run chef-client locally to verify the "myiis" cookbook's default recipe.

# Lab: Apply the 'myiis' Cookbook's Default Recipe



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

```
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 Non  
      e -File "C:/tmp/chef-script20180919-3724-1wk70vt.ps1"  
  * file[C:\inetpub\wwwroot\Default.htm] action create  
    - update content in file C:\inetpub\wwwroot\Default.htm from f37fdd to  
      38be49  
      --- C:\inetpub\wwwroot\Default.htm 2018-09-19 17:25:01.000000000 +0000  
      +++ C:\inetpub\wwwroot\chef-Default20180919-3724-1mczbah.htm  
2018-09-19 17:34:44.000000000 +0000
```

LAB

## Change Means a New Version



*Let's bump the cookbook's version number*

### OBJECTIVE:

- Determine the new version number
- Update the version of the "myiis" cookbook

# CONCEPT

## Cookbook Versions

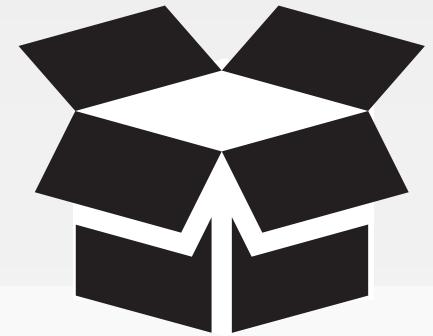


A cookbook version represents a set of functionality that is different from the cookbook on which it is based.

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

# CONCEPT

## Semantic Versions



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

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

<http://semver.org>

# Major, Minor, or Patch?

What kind of changes did you make to the cookbook?



LAB

## Change Means a New Version



*Let's bump the version number*

### OBJECTIVE:

- Determine the new version number
- Update the version of the "myiis" cookbook

# 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 myiis'  
long_description 'Installs/Configures myiis'  
version        '0.2.0'
```

# Discussion



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

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

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

## Q&A



What questions can we help you answer?

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



CHEF<sup>TM</sup>



©2018 Chef Software Inc.

2W

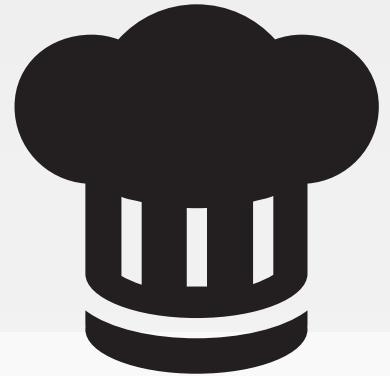
# Objectives

After completing this module, you should be able to:

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

# LAB

## More Web Servers?



*More easily manage multiple nodes*

### OBJECTIVE:

- Create a Hosted Chef Account
- Upload your cookbooks to the Hosted Chef Server
- Add your old workstation as a managed node

# 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 myiis cookbook.
4. Run chef-client on the new node to apply the myiis cookbook's default recipe.

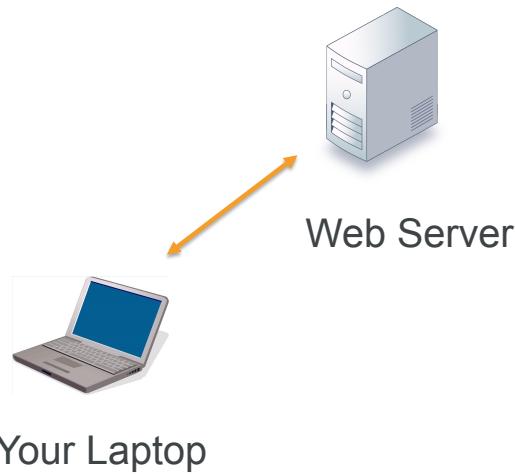
# Managing Additional Systems

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

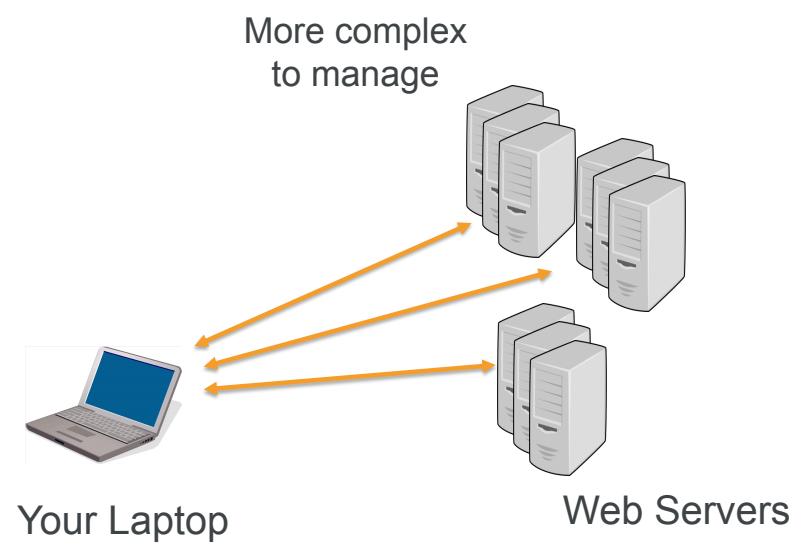
- Chef provides a one-line curl install.
- You could use **git** to clone the repository from a common **git** repository.
- Applying the run list.

# Managing Additional Systems

Now

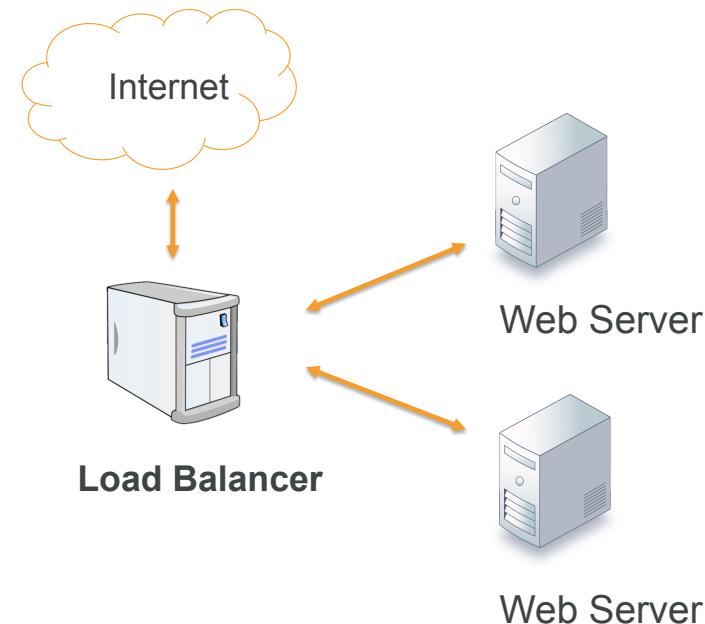


Future



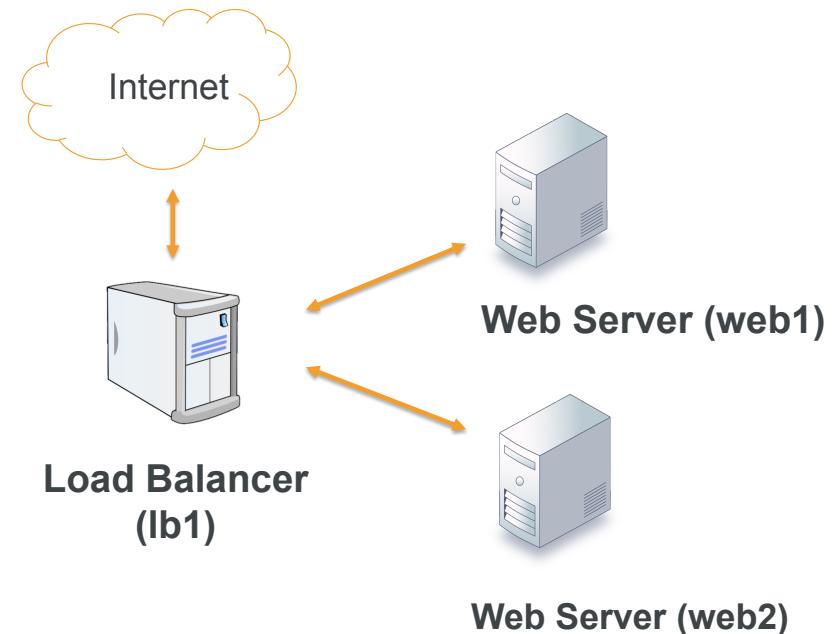
# Managing User Traffic

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



# Managing User Traffic

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



# Steps to Set up Load Balancer and Web Servers

## Web Server

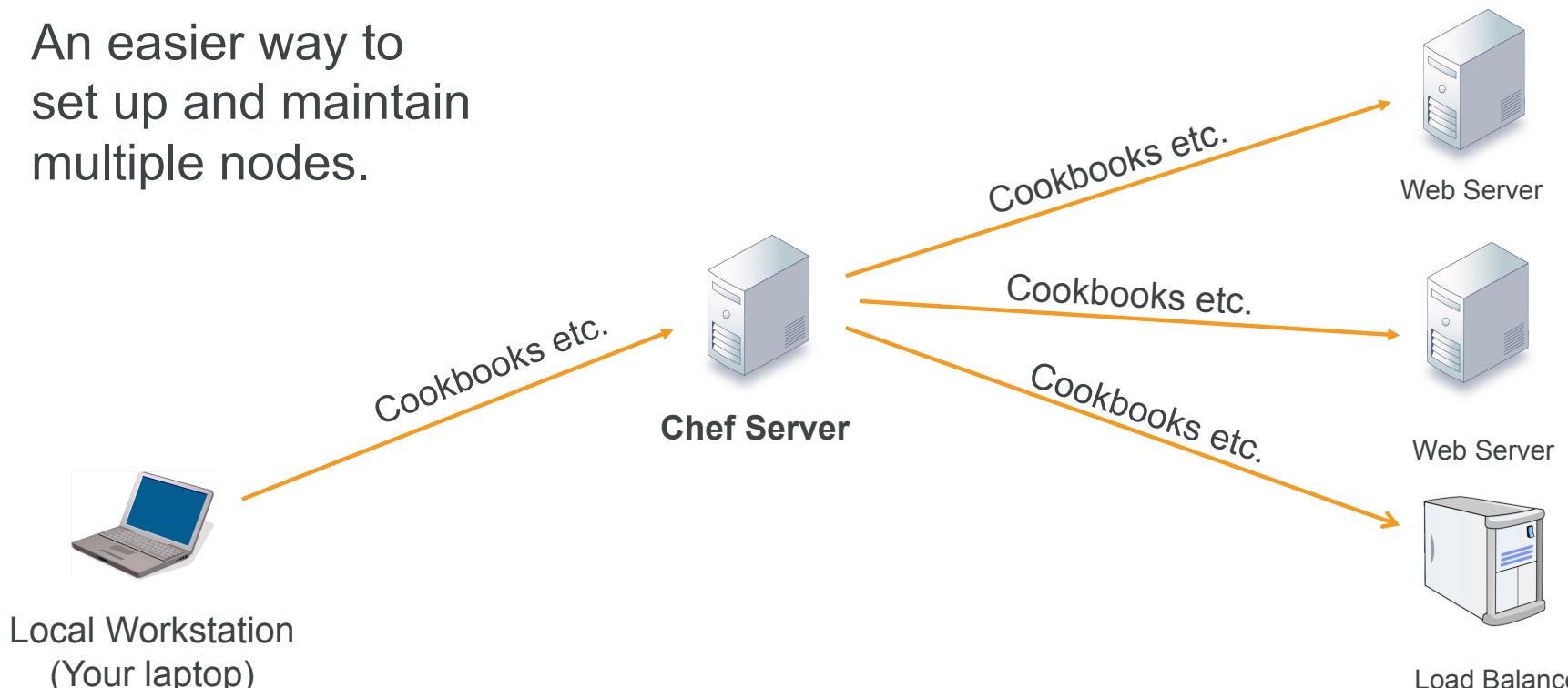
1. Provision the instance
2. Install Chef
3. Copy the Web Server cookbook
4. Apply the cookbook

## Load Balancer

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

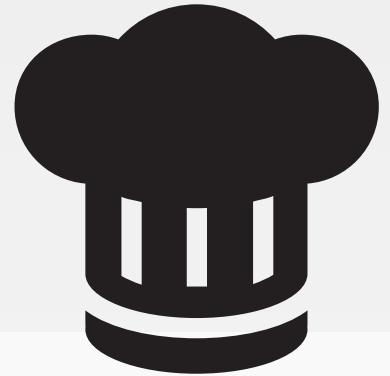
# The Chef Server

An easier way to set up and maintain multiple nodes.



LAB

## GL: Hosted Chef



*More easily manage multiple nodes*

### OBJECTIVE:

- Create a Hosted Chef Account
- Upload your cookbooks to the Hosted Chef Server
- Add your old workstation as a managed node

# GL: Signing Up for a Hosted Chef Account



## Steps

1. Navigate to <https://manage.chef.io/signup>
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 'Start your free trial of Hosted Chef' page. It features a dark header with the 'CHEF MANAGE' logo. Below it, a large orange button says 'Start your free trial of Hosted Chef'. To the right of the button, there's a vertical sidebar with partially visible text: 'Already have an account? Click here.', 'Looking for something else? Start with our Getting Started guide.', 'Join the Chef community.', and 'Join our mailing list.' The main form area has four input fields: 'Full Name' (Jane Smith), 'Company' (Chef), 'Email' (janesmith@chef.io), and 'Username' (janesmith). A checkbox labeled 'I agree to the Terms of Service and the Master License and Services Agreement.' is checked. A large orange 'Get Started' button is at the bottom, with a hand cursor icon pointing to it.

# GL: Signing Up for a Hosted Chef Account



## Steps

3. When prompted, open the email just sent to you and click the link in the email to finish the creation of your account.



Thanks for signing up!

We've just sent you an email to verify your email address. Click the link in the

# GL: Signing Up for a Hosted Chef Account



## Steps

4. Enter a password when prompted and then click **Create User**.

You should write down your password in case you forget it.

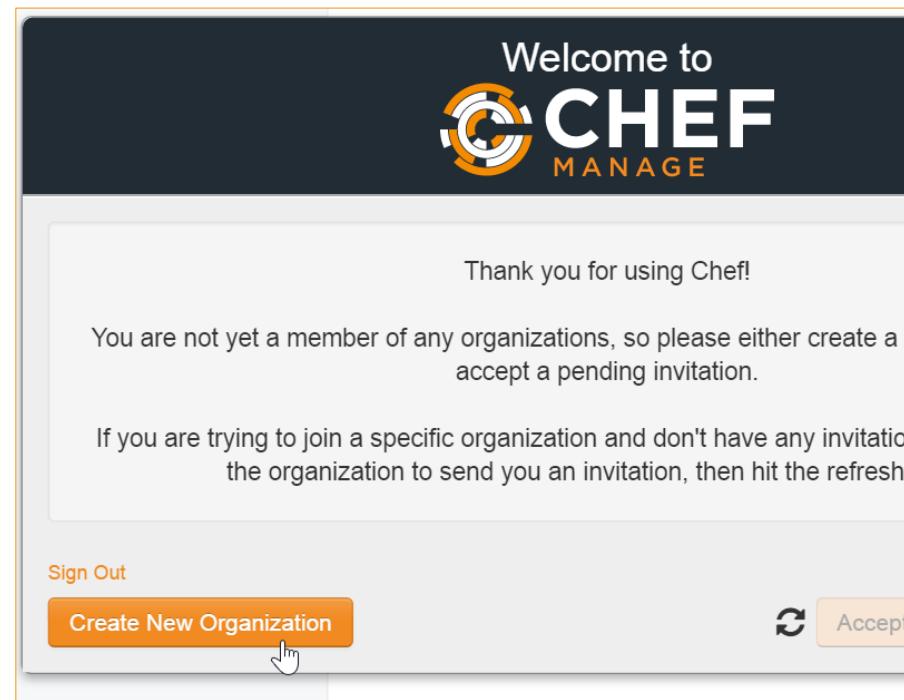
The screenshot shows a web browser window with the Chef Manage logo at the top. The main content area displays the message "Email Verification Successful". Below this, a note reads: "Thank you for verifying your email address! Please enter the password you'd like to use below and submit the form to complete the creation of your account." A password input field is shown with the placeholder "Password" and a redacted password. To the right of the input field is a small icon showing a speech bubble with the number "6". At the bottom is a large orange button labeled "Create User" with a hand cursor icon pointing to it.

# GL: Signing Up for a Hosted Chef Account



## Steps

5. From the resulting page, click the **Create New Organization** button.



# GL: Signing Up for a Hosted Chef Account



## Steps

6. Fill out the resulting Create Organization form and then click **Create Organization**.

**Create Organization**

Full Name (example: Chef, Inc.)  
Janesmith Org

Short Name (example: chef)  
janesorg



# GL: Signing Up for a Hosted Chef Account



## Steps

7. From the resulting page, click **Download Starter Kit** and then click **Proceed** when prompted.

A chef-starter zip file should download to your laptop.

The screenshot shows the Chef Manage interface with a dark header bar containing the Chef logo and navigation links for Nodes, Reports, Policy, and Admin. The main content area has a light gray background. On the left, there's a sidebar with sections for Organizations (Create, Reset Validation Key, Generate Knife Config, Invite User, Leave Organization, Starter Kit), Users, Groups, and Global Permissions. To the right, a large text area says "Thank you for choosing Hosted Chef" and "Follow these steps to be on your way". It features two buttons: "Download Starter Kit" (highlighted with a mouse cursor) and "Learn Chef". Below these are links for "Chef Documentation", "Browse Community Cookbooks", and "Contact Support". At the bottom, there's a "What's next?" section with links for "Chef", "Documentation", "Community", "Cookbooks", "Contact Support", and "Our support". A prominent orange modal dialog box is centered over the page, containing the text "Are you certain? Your user key will be reset. Are you sure you want to do this?". It has "Cancel" and "Proceed" buttons, with "Proceed" being highlighted with a mouse cursor. The entire modal is surrounded by a thick orange border.

# GL: Signing Up for a Hosted Chef Account



## Steps

8. Open the downloaded zip file and copy chef-repo folder that's contained in the zip file.
9. 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`

## GL: Navigate to the chef-repo



```
> cd ~\chef-repo
```



# GL: Move the cookbooks directory into the chef-repo



```
➤ cd ~  
➤ mv cookbooks chef-repo
```

# CONCEPT

## knife



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

# GL: 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)
```

# GL: knife client --help



```
> knife client --help
```

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

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

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

# GL: knife client list

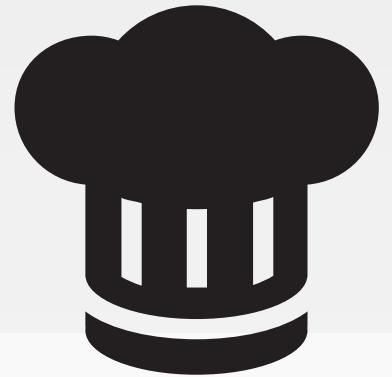


```
> knife client list
```

```
ORGNAME-validator
```

# LAB

## Hosted Chef



*More easily manage multiple nodes*

### OBJECTIVE:

- ✓ Create a Hosted Chef Account
- Upload your cookbooks to the Hosted Chef Server
- Add your old workstation as a managed node

# GL: knife cookbook --help



```
> knife cookbook --help
```

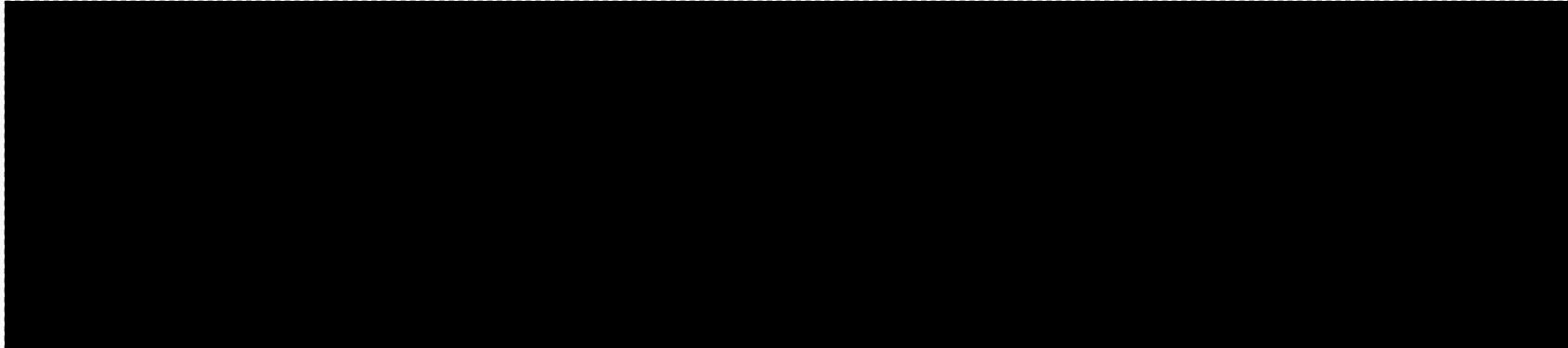
```
** COOKBOOK COMMANDS **

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)
```

# GL: knife cookbook list

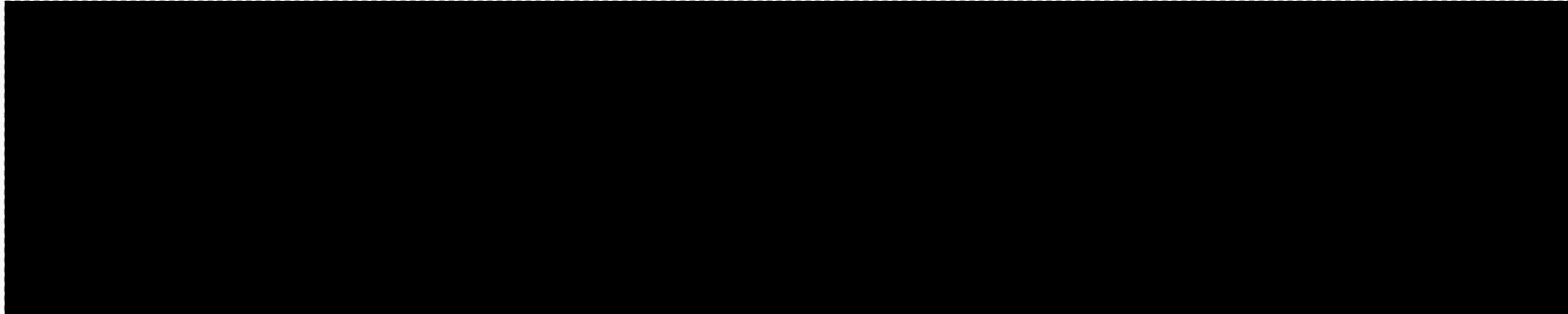


> knife cookbook list



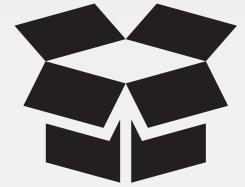
## GL: Change to the myiis Directory

 > cd cookbooks\myiis



# CONCEPT

## Berkshelf



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

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

# GL: Run berks --help



```
> berks --help
```

Commands:

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

# GL: Run berks install



```
> 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 .
```

# GL: See the Berksfile.lock



> dir

```
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
```

## GL: See the Contents of the Berksfile.lock



```
> cat Berksfile.lock
```

```
DEPENDENCIES
```

```
myiis
  path: .
  metadata: true
```

```
GRAPH
```

```
myiis (0.2.1)
```

## GL: Upload the Cookbook to the Chef Server



```
> berks upload
```

```
Uploaded myiis (0.2.1) to: 'https://api.chef.io:443/organizations/  
awesomestudent'
```

# GL: Display Cookbooks within Your Org



```
> knife cookbook list
```

```
myiis      0.2.1
```



## Lab: Upload Cookbooks

- Upload your remaining cookbooks
- Verify that all cookbooks are uploaded

## Lab: cd and Run knife cookbook list



```
> cd ~\chef-repo\cookbooks\workstation  
> knife cookbook list
```

```
myiis      0.2.1
```

# Lab: Install the Cookbook Dependencies



```
> berks install
```

```
Resolving cookbook dependencies...
Fetching 'workstation' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using workstation (0.1.0) from source at .
```

# Lab: Upload the Cookbook to the Chef Server



```
> berks upload
```

```
Uploaded workstation (0.1.0) to: 'https://api.chef.io:443/organizations/  
awesomestudent'
```

# Lab: Is the workstation Cookbook Uploaded?



```
> knife cookbook list
```

```
myiis          0.2.1
workstation    0.1.0
```

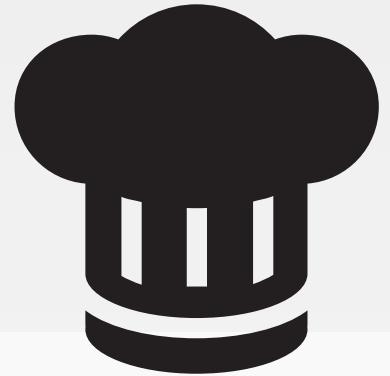


## Lab: Upload Cookbooks

- ✓ Upload your remaining cookbooks
- ✓ Verify that all cookbooks are uploaded

# LAB

## Hosted Chef



*More easily manage multiple nodes*

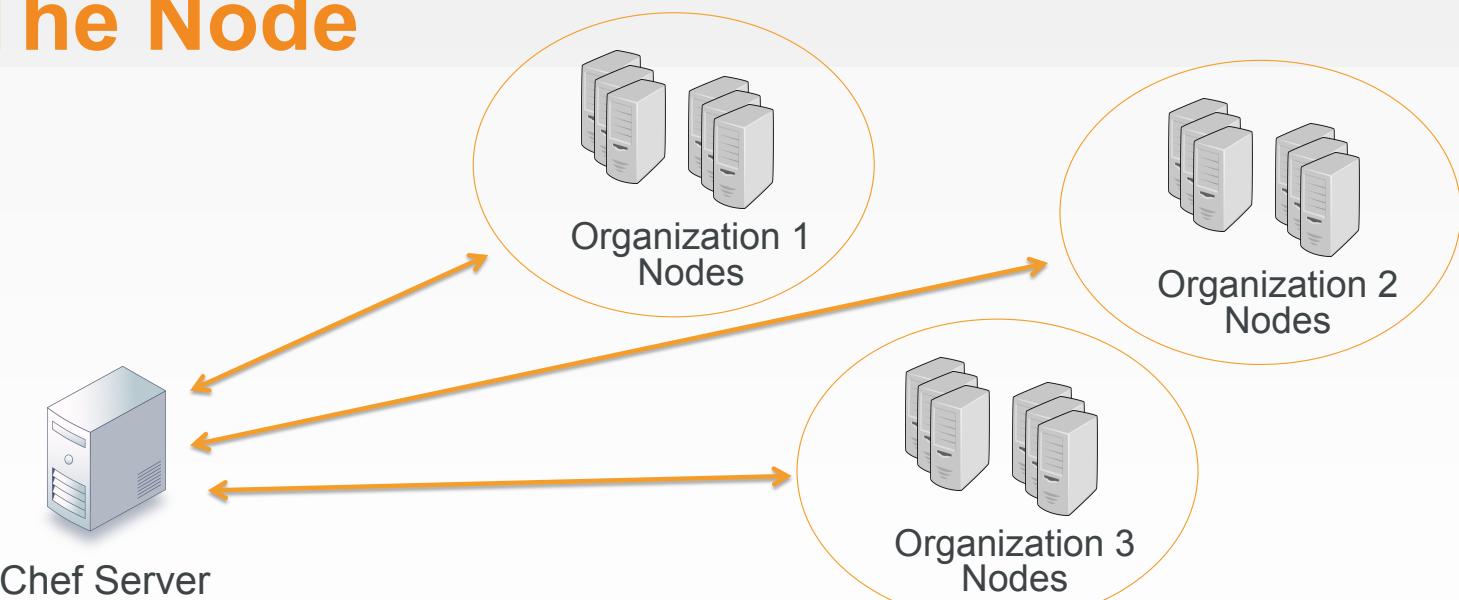
### OBJECTIVE:

- ✓ Create a Hosted Chef Account
- ✓ Upload your cookbooks to the Hosted Chef Server
- Add your web server as a managed node

# CONCEPT



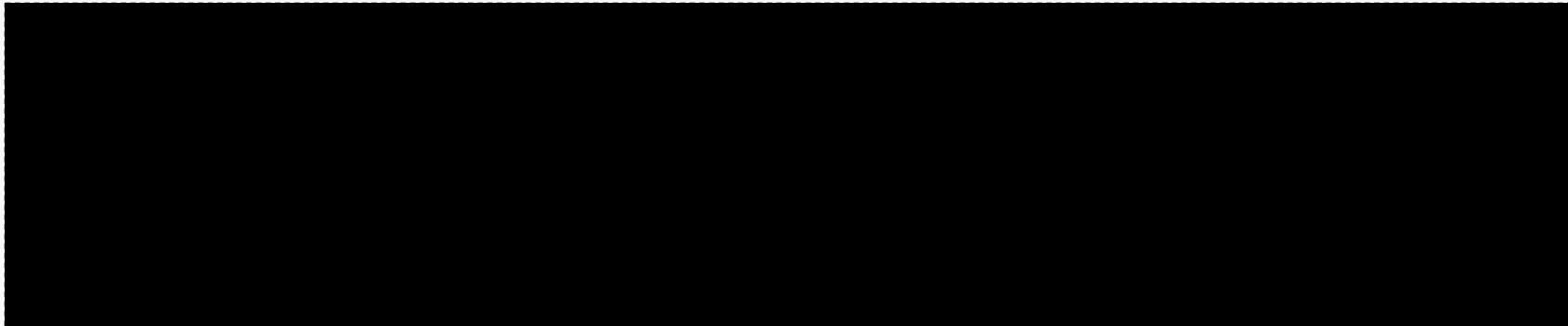
## The Node



# GL: Change to the chef-repo



```
> cd ~\chef-repo
```



# GL: Run 'knife node --help'



```
> knife node --help
```

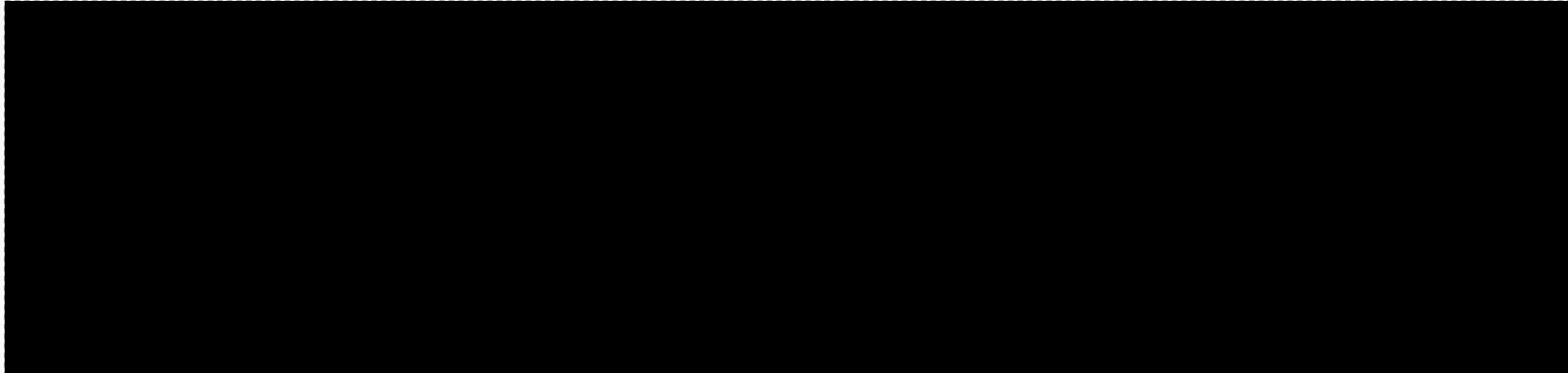
```
** NODE COMMANDS **

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)
```

## GL: Run 'knife node list'



```
> knife node list
```



# GL: 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.
```

# GL: Bootstrap Your Node



```
> knife bootstrap windows winrm IP -x USER -P PWD -N web1
```

Creating new client for web1

Creating new node for web1

Fully Qualified Domain  
Name or IP

Waiting for boot  
34.228.41.102 Response received.

Remote node responded after 0.02 minutes.

Bootstrapping Chef on 34.228.41.102

34.228.41.102 Rendering "C:

\Users\ADMINI~1\AppData\Local\Temp\bootstrap-4032-1505928424.bat" chunk 1

34.228.41.102 Rendering "C:

\Users\ADMINI~1\AppData\Local\Temp\bootstrap-4032-1505928424.bat" chunk 2

34.228.41.102 Rendering "C:

\Users\ADMINI~1\AppData\Local\Temp\bootstrap-4032-1505928424.bat" chunk 3

...

©2018 Chef Software Inc.

2210



# GL: Run 'knife node list' Again



```
> knife node list
```

```
web1
```

## GL: View More Information About Your Node



```
> knife node show web1
```

```
Node Name:      web1
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            34.228.41.102
Run List:
Roles:
Recipes:
Platform:      windows 6.3.9600
Tags:
```

# GL: Add a Recipe to a Run List



```
> knife node run_list add web1 "recipe[myiis]"
```

```
web1:  
  run_list: recipe[myiis]
```

# GL: Verify the New Node



```
> knife node show web1
```

```
Node Name:      web1
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            34.228.41.102
Run List:      recipe[myiis]
Roles:
Recipes:
Platform:      windows 6.3.9600
Tags:
```

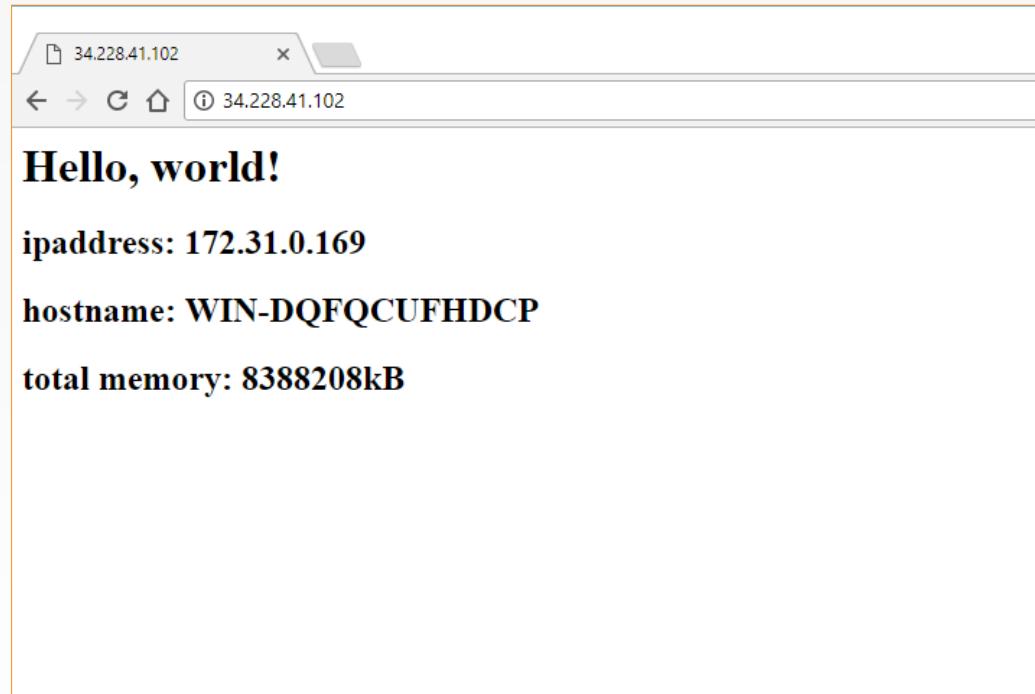
# GL: Converge web1



```
> knife winrm IP -m -x USER -P PWD "chef-client"
```

```
34.228.41.102 Starting Chef Client, version 13.4.24
34.228.41.102
34.228.41.102 [2018-08-20T17:37:02+00:00] INFO: *** Chef 13.4.24 ***
34.228.41.102 [2018-09-20T17:37:02+00:00] INFO: Platform: x64-mingw32
34.228.41.102 [2018-09-20T17:37:02+00:00] INFO: Chef-client pid: 500
34.228.41.102 [2018-09-20T17:37:02+00:00] INFO: The plugin path C:\chef\ohai\plugins
does not exist. Skipping...
34.228.41.102 [2018-09-20T17:37:05+00:00] INFO: Run List is [recipe[myiis]]
34.228.41.102 [2018-09-20T17:37:05+00:00] INFO: Run List expands to [myiis]
34.228.41.102 [2018-09-20T17:37:05+00:00] INFO: Starting Chef Run for web1
...
34.228.41.102 Chef Client finished, 2/4 resources updated in 39 seconds
34.228.41.102 [2018-09-20T17:37:41+00:00] INFO: Sending resource update report (run-
id: a0a2aa9a-cf6d-4b94-8e18-954f28c99428)
```

# Verify web1 Serves the Page





## Setting run\_list at Bootstrap

Alternatively we could have set the run\_list during the bootstrap process by adding `-r "recipe[myiis]"`.

```
knife bootstrap windows winrm IP -x USER -P PWD -N web1 -r "recipe[myiis]"
```



## Lab: Another Web Node

- Bootstrap a new node, setting the run list of the to include the web server cookbook using the `'-r` flag
- Verify that the node's web server is functional

# Lab: Bootstrap the New Node



```
> knife bootstrap windows winrm IP_OF_web2 -x USER -P PWD  
-N web2 -r "recipe[myiis]"
```

```
Creating new client for web2  
Creating new node for web2  
Waiting for remote response before bootstrap.54.146.147.44 .  
54.146.147.44 Response received.  
Remote node responded after 0.02 minutes.  
Bootstrapping Chef on 54.146.147.44  
...  
54.146.147.44 Running handlers complete  
54.146.147.44 [2018-09-20T18:06:10+00:00] INFO: Report handlers complete  
54.146.147.44 Chef Client finished, 2/4 resources updated in 01 minutes 02 seconds
```

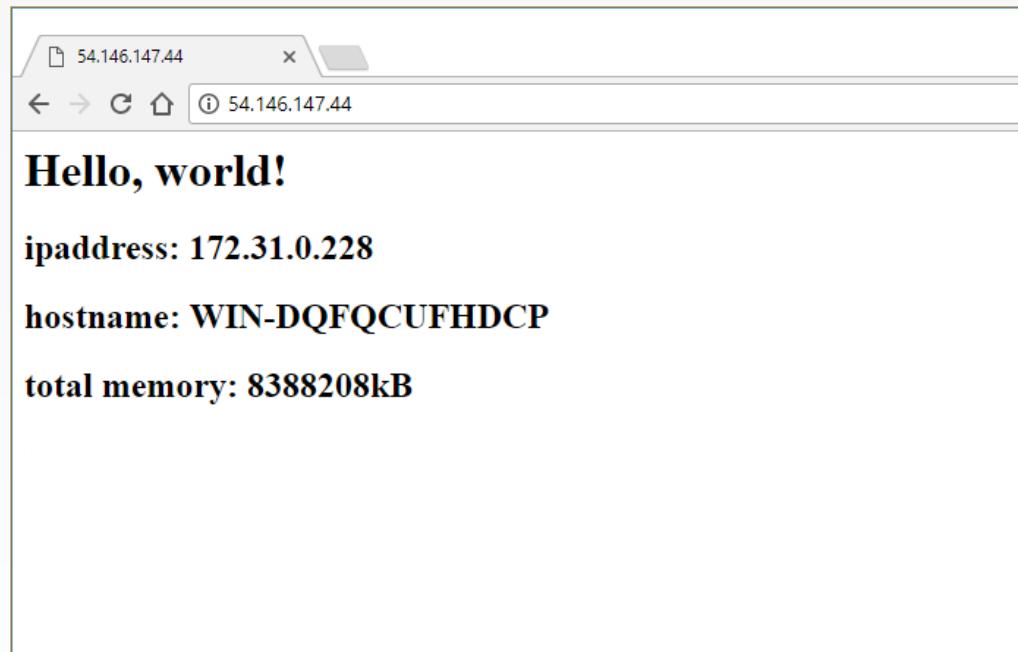
# Lab: Verify the New Node



```
> knife node show web2
```

```
Node Name:      web2
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            54.146.147.44
Run List:      recipe[myiis]
Roles:
Recipes:        myiis, myiis::default, myiis::server
Platform:       windows 6.3.9600
Tags:
```

# Verify that the New Node Serves the Page



# 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?

## Q&A

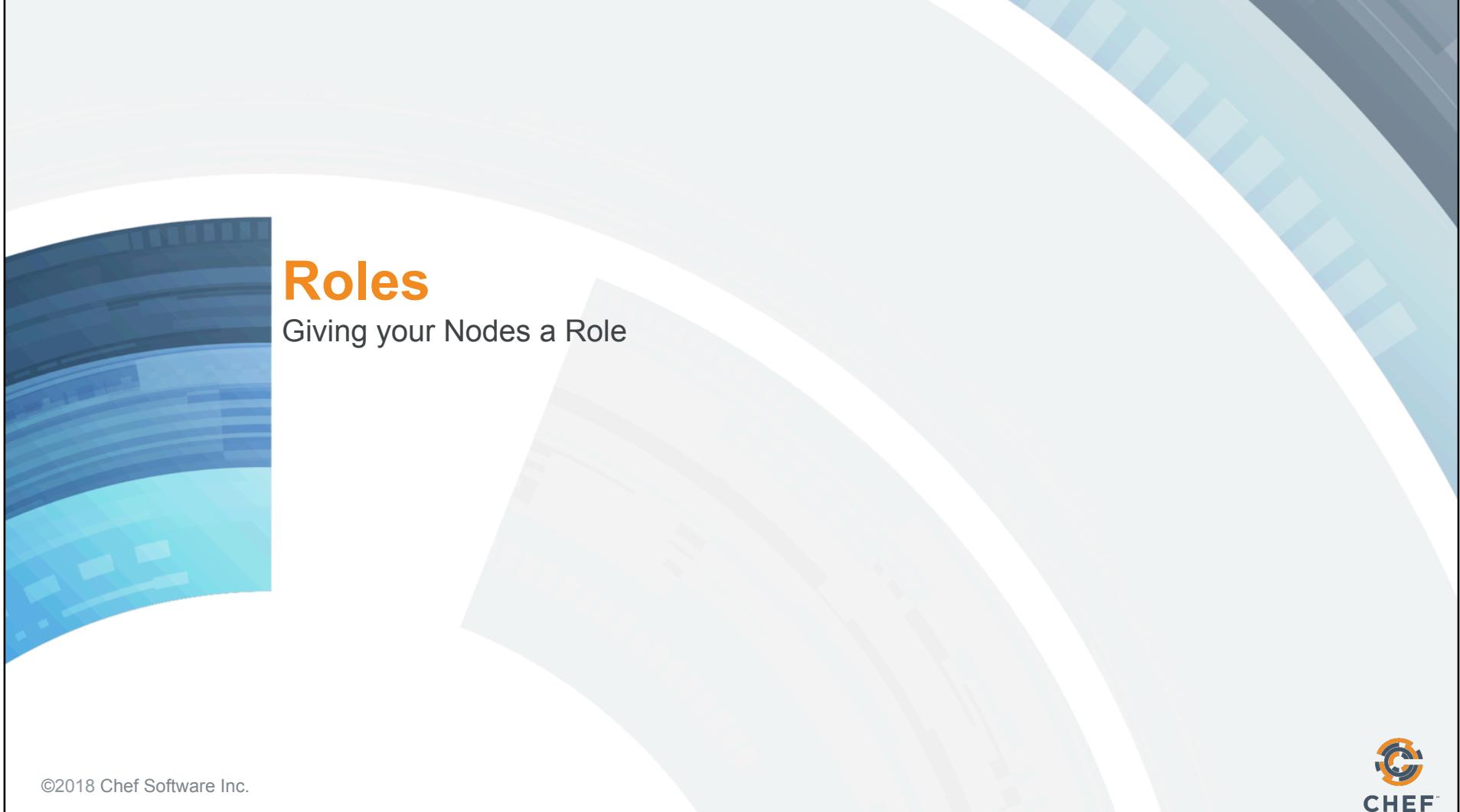


What questions can you help you answer?

- Chef Server
- Managed Chef
- Berkshelf
- Bootstrapping Nodes



CHEF™



# Roles

Giving your Nodes a Role

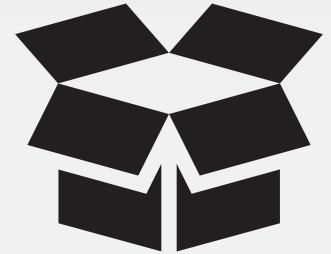
# Objectives

After completing this module, you should be able to:

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

# CONCEPT

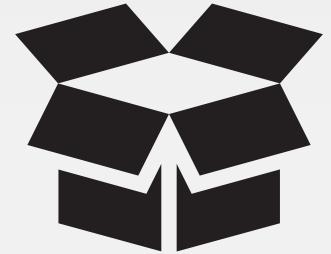
## Roles



A role describes a run list of recipes that are executed on the node. It may also define new attribute defaults or define overrides for existing attribute values.

# CONCEPT

## Roles

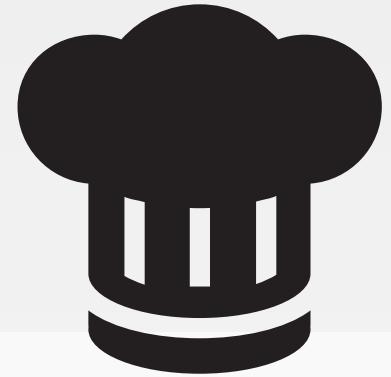


When you assign a role to a node you do so in its run list.

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

# LAB

## GL: Define a Web Role



*A role would be a better way to describe what are obviously the web servers.*

### OBJECTIVE:

- Create and upload a role named "web" that has the run list "recipe[myiis]"
- Set web1's run list to be "role[web]"
- Set web2's run list to be "role[web]"
- Converge all the nodes that have been assigned the web role

# GL: What Can 'knife role' Do?



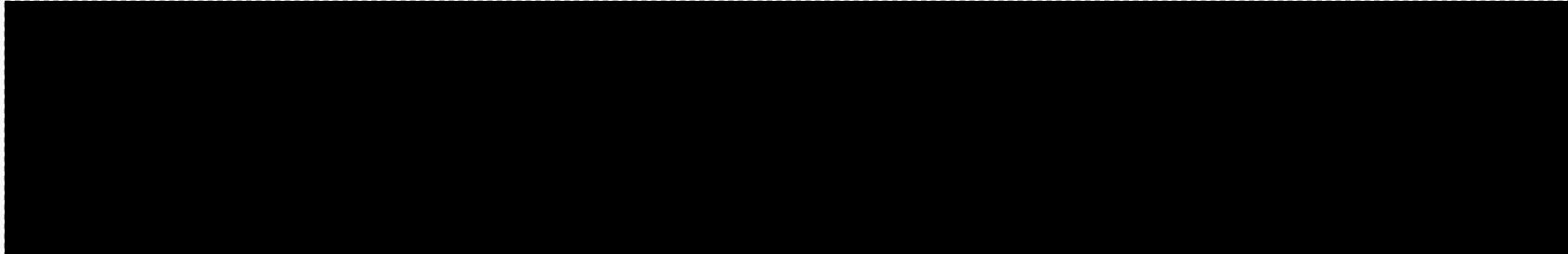
```
> cd ~\chef-repo  
> knife role --help
```

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

## GL: Run 'knife role list'



```
> knife role list
```



# GL: Create the Web Role

```
[ ] ~\chef-repo\roles\web.rb
```

```
name 'web'  
description 'Web Server'  
run_list 'recipe[myiis]'
```

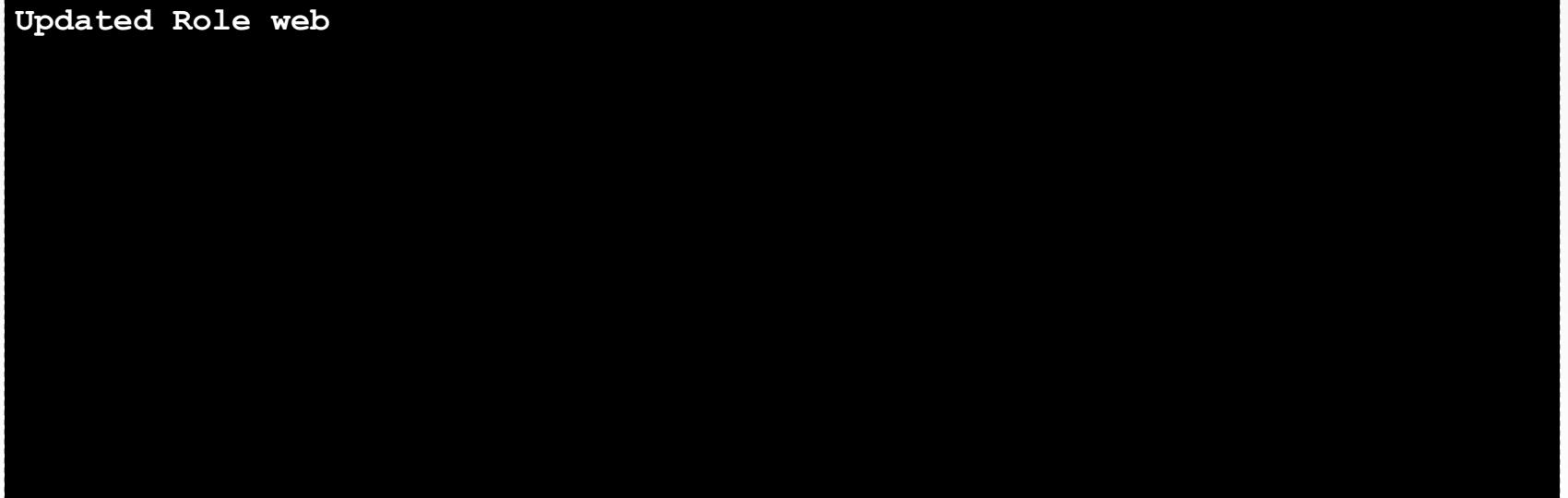
©2018 Chef Software Inc.

# GL: Upload the Role to the Chef Server



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

```
Updated Role web
```



## GL: Verify the Role on the Chef Server



```
> knife role list
```

```
web
```

## GL: Verify Specific Information About the Role

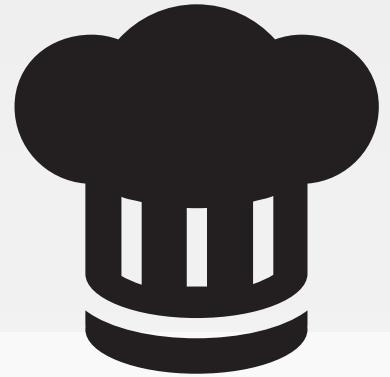


```
> knife role show web
```

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

# LAB

## GL: Define a Web Role



*A role would be a better way to describe what are obviously the web servers.*

### OBJECTIVE:

- ✓ Create and upload a role named "web" that has the run list "recipe[myiis]"
- Set web1's run list to be "role[web]"
- Set web2's run list to be "role[web]"
- Converge all the nodes that have been assigned the web role

## GL: View web1



```
> knife node show web1
```

```
Node Name:      web1
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            174.129.71.79
Run List:       recipe[myiis]
Roles:
Recipes:        myiis, myiis::default, myiis::server
Platform:       windows 6.3.9600
Tags:
```

# GL: Set web1's Run List



```
> knife node run_list set web1 "role[web]"
```

```
web1:  
  run_list: role[web]
```

## GL: View web1 again

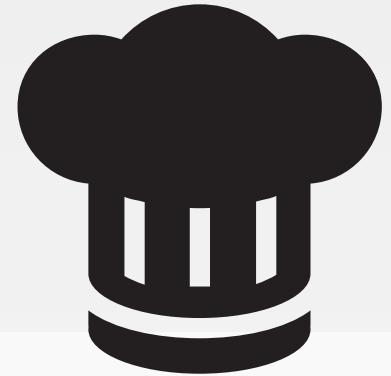


```
> knife node show web1
```

```
Node Name:      web1
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            174.129.71.79
Run List:       role[web]
Roles:
Recipes:        myiis, myiis::default, myiis::server
Platform:       windows 6.3.9600
Tags:
```

# LAB

## GL: Define a Web Role



*A role would be a better way to describe what are obviously the web servers.*

### OBJECTIVE:

- ✓ Create and upload a role named "web" that has the run list "recipe[myiis]"
- ✓ Set web1's run list to be "role[web]"
- Set web2's run list to be "role[web]"
- Converge all the nodes that have been assigned the web role

## GL: View node2



```
> knife node show web2
```

```
Node Name:      web2
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            54.146.147.44
Run List:      recipe[myiis]
Roles:
Recipes:        myiis, myiis::default, myiis::server
Platform:       windows 6.3.9600
Tags:
```

## GL: Set web2's Run List



```
> knife node run_list set web2 "role[web]"
```

```
web2:  
  run_list: role[web]
```

# GL: View web2 again

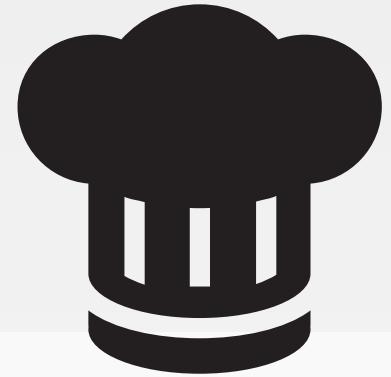


```
> knife node show web2
```

```
Node Name:      web2
Environment:    _default
FQDN:          WIN-DQFQCUFHDCP.ec2.internal
IP:            54.146.147.44
Run List:       role[web]
Roles:
Recipes:        myiis, myiis::default, myiis::server
Platform:       windows 6.3.9600
Tags:
```

# LAB

## GL: Define a Web Role



*A role would be a better way to describe what are obviously the web servers.*

### OBJECTIVE:

- ✓ Create and upload a role named "web" that has the run list "recipe[myiis]"
- ✓ Set web1's run list to be "role[web]"
- ✓ Set web2's run list to be "role[web]"
- Converge all the nodes that have been assigned the web role

## GL: Capture Node's Public Host Name and IP



```
> knife node show web1 -a cloud
```

```
web1:  
  cloud:  
    local_hostname: ip-172-31-0-169.ec2.internal  
    local_ipv4: 172.31.0.169  
    local_ipv4_addrs: 172.31.0.169  
    provider: ec2  
    public_hostname: ec2-174-129-71-79.compute-1.amazonaws.com  
    public_ipv4: 174.129.71.79  
    public_ipv4_addrs: 174.129.71.79
```

## GL: Capture Nodes' IP



```
> knife node show web1 -a cloud.public_ipv4
```

```
web1:
```

```
  cloud.public_ipv4: 174.129.71.79
```

```
> knife node show web2 -a cloud.public_ipv4
```

```
web2:
```

```
  cloud.public_ipv4: 54.146.147.44
```

**Issue:** We will use the `node['cloud']['public_ipv4']` attribute value

# GL: Converge All Web Nodes



```
> knife winrm "role:web" -a cloud.public_ipv4 -x USER -P PWD "chef-client"
```

```
54.146.147.44 Starting Chef Client, version 13.4.24
...
54.146.147.44 [2018-09-20T19:01:24+00:00] INFO: Run List is [role[web]]
54.146.147.44 [2018-09-20T19:01:24+00:00] INFO: Run List expands to [myiis]
54.146.147.44 [2018-09-20T19:01:24+00:00] INFO: Starting Chef Run for web2
...
74.129.71.79 Starting Chef Client, version 13.4.24
...
174.129.71.79 [2018-09-20T19:01:33+00:00] INFO: Run List is [role[web]]
174.129.71.79 [2018-09-20T19:01:33+00:00] INFO: Run List expands to [myiis]
174.129.71.79 [2018-09-20T19:01:33+00:00] INFO: Starting Chef Run for web1
174.129.71.79 [2018-09-20T19:01:33+00:00] INFO: Running start handlers
...

```

# 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?

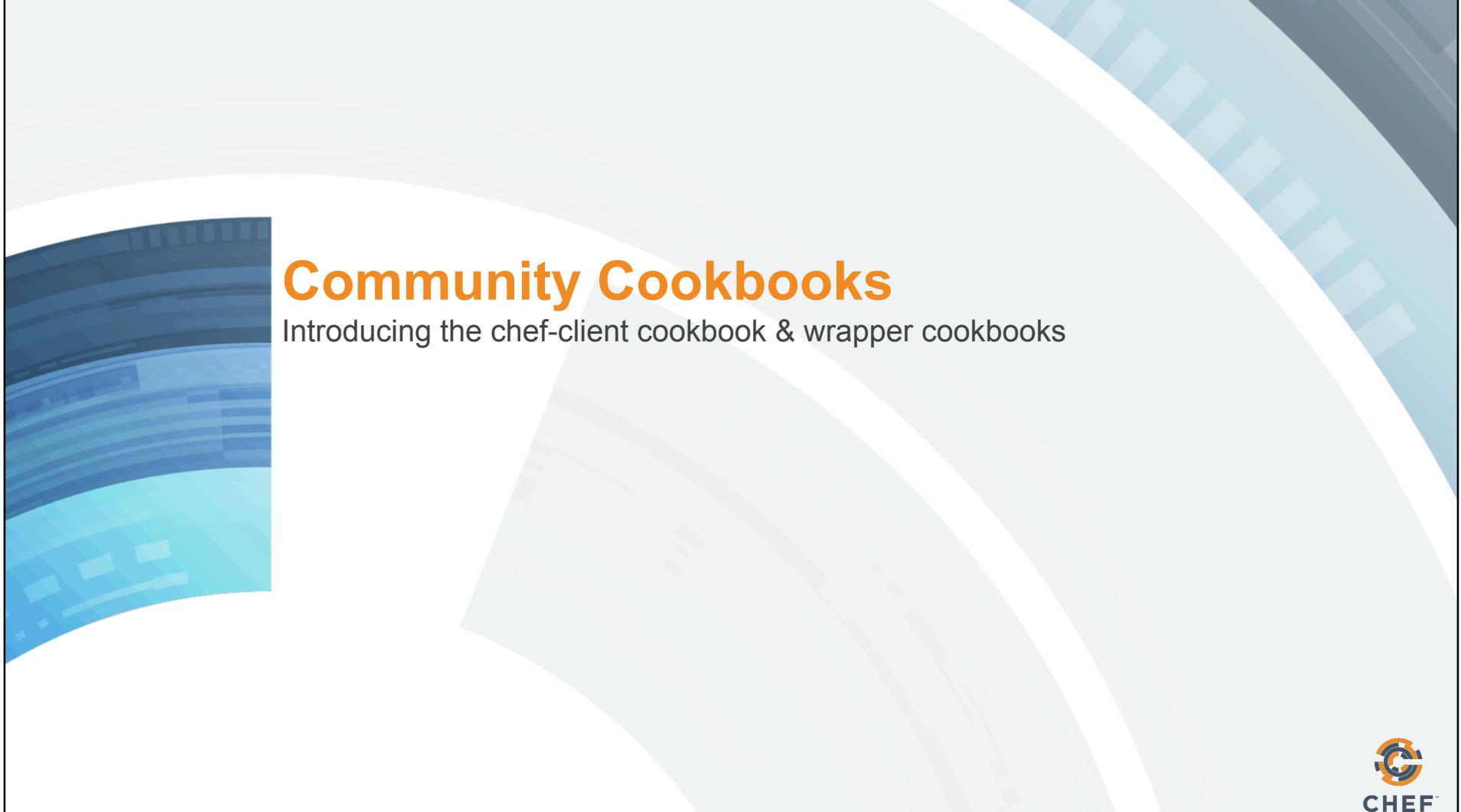
## Q&A



What questions can we help you answer?



CHEF™



# Community Cookbooks

Introducing the chef-client cookbook & wrapper cookbooks

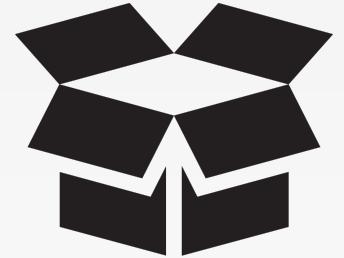


# Lesson Objectives

After completing this module, you should be able to:

- Find and preview cookbooks on the Chef Supermarket
- Override community cookbook defaults using wrapper cookbooks
- Upload community cookbooks to Chef Server
- Run chef-client as a service/task

# CONCEPT



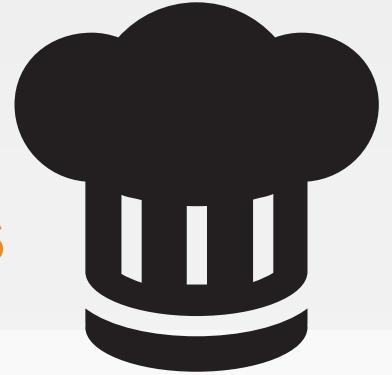
## Running chef-client

Manually executing chef-client is fine but how can we set up a node to converge automatically.

- How can I run chef-client as a service or Windows task?
- Where can I configure logging?

LAB

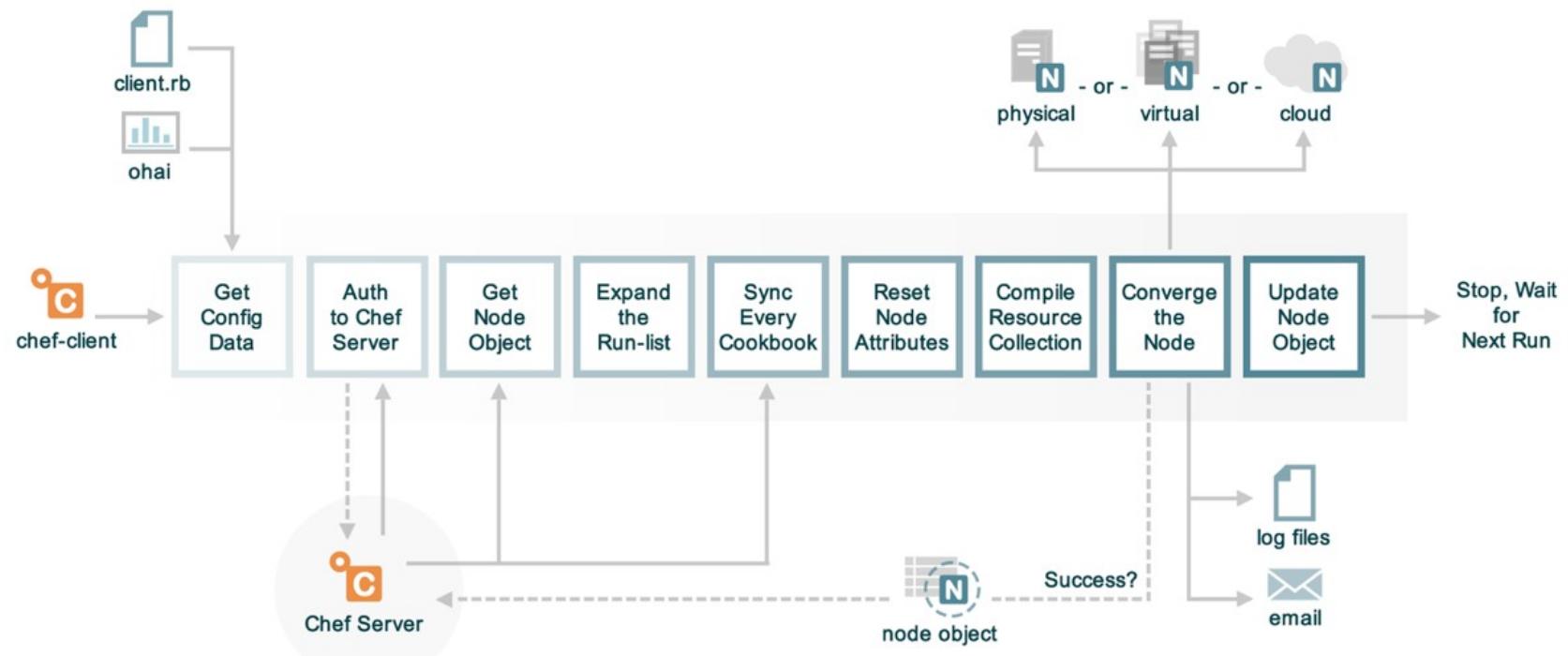
## GL: View How chef-client is Configured



### OBJECTIVE:

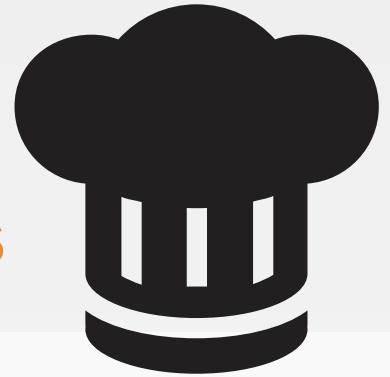
- Review the overview of the chef-client run
- Review the chef-client configuration file

# The Chef Client Run



LAB

## GL: View How chef-client is Configured



### OBJECTIVE:

- ✓ Review the overview of the chef-client run
- ❑ Review the chef-client configuration file

# GL: View chef-client config Directory



```
> knife winrm role:web -a cloud.public_ipv4 -x USER -P PWD  
"dir C:\chef\client.rb"
```

```
54.146.147.44 Volume in drive C has no label.  
54.146.147.44 Volume Serial Number is D4E5-7A7A  
54.146.147.44  
54.146.147.44 Directory of C:\chef  
54.146.147.44  
54.146.147.44 09/20/2018 06:05 PM 332 client.rb  
54.146.147.44 1 File(s) 332 bytes  
54.146.147.44 0 Dir(s) 11,205,259,264 bytes free  
174.129.71.79 Volume in drive C has no label.  
174.129.71.79 Volume Serial Number is D4E5-7A7A  
174.129.71.79  
174.129.71.79 Directory of C:\chef  
174.129.71.79  
174.129.71.79 09/20/2018 05:27 PM 332 client.rb ...
```

# GL: View chef-client Configuration

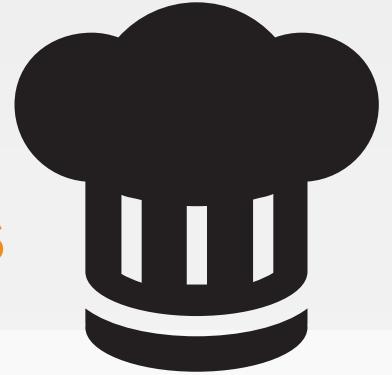


```
> knife winrm role:web -a cloud.public_ipv4 -x USER -P PWD  
"type C:\chef\client.rb"
```

```
54.146.147.44 chef_server_url  "https://api.chef.io/organizations/janesmith11org"  
54.146.147.44 validation_  
54.146.147.44 client_name "chef-validator"  
54.146.147.44 file_cache_path    "c:/chef/cache"  
54.146.147.44 file_backup_path   "c:/chef/backup"  
54.146.147.44 cache_options      ({:path => "c:/chef/cache/checksums", :skip_expires => true})  
54.146.147.44 node_name "web2"  
54.146.147.44 log_level         :info  
54.146.147.44 log_location       STDOUT  
174.129.71.79 chef_server_url  "https://api.chef.io/organizations/janesmith11org"  
174.129.71.79 validation_  
174.129.71.79 client_name "chef-validator"  
174.129.71.79 file_cache_path    "c:/chef/cache"  
174.129.71.79 file_backup_path   "c:/chef/backup" ...
```

LAB

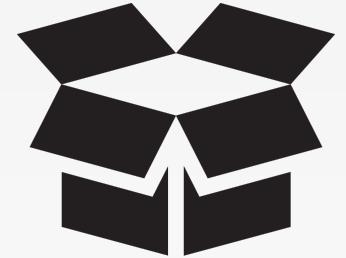
## GL: View How chef-client is Configured



### OBJECTIVE:

- ✓ Review the overview of the chef-client run
- ✓ Review the chef-client configuration file

# CONCEPT



## Managing chef-client

chef-client is a complicated application. Reviewing the conceptual flow of the application and the location of the configuration file helps make it easier to understand. However, there is still much more to learn if you wanted to manage it with a recipe defined in a cookbook.

DOCS

## Community Cookbooks



Someone already wrote that cookbook?

Available through the community site called  
Supermarket

<https://supermarket.chef.io>

# Demo: 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.

The screenshot shows the homepage of the Chef Supermarket at <https://supermarket.chef.io>. The page features a header with the Chef logo, navigation links for COOKBOOKS, CONTRIBUTORS, and TOOLS & PLUGINS, and options to CREATE ACCOUNT or SIGN IN. A search bar with a dropdown menu for 'Cookbooks' and a 'Search' button is prominently displayed. Below the search bar, a message reads 'Welcome to Supermarket. Find, explore and view Chef cookbooks for all of your ops needs.' There are two buttons: 'Sign Up For a Chef Account' and 'Sign In With Your Chef Account'. The main content area is divided into three sections: 'Explore' (with links to 'Browse Cookbooks' and 'Read the Chef Blog'), 'Learn' (with links to 'Learn Chef', 'Read the Chef Docs', 'Community Guidelines', and 'How to Contribute'), and 'Share' (with links to 'Share your cookbooks', 'Chat on IRC at #chef on irc.freenode.net', 'Join the Supermarket Mailing List', 'Contribute to Supermarket', and 'Share Your Ideas for Improving Chef'). At the bottom, the page displays 'Community Stats' with the numbers '2,433 Cookbooks' and '64,385 Chefs'.

# 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.

# LAB

## 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:

- Create a cookbook that wraps the chef-client cookbook
- Upload chef-client cookbook to Chef Server
- Configure each nodes to run chef-client as a service
- Converge each node and verify the service is running

## Wrapper Cookbooks



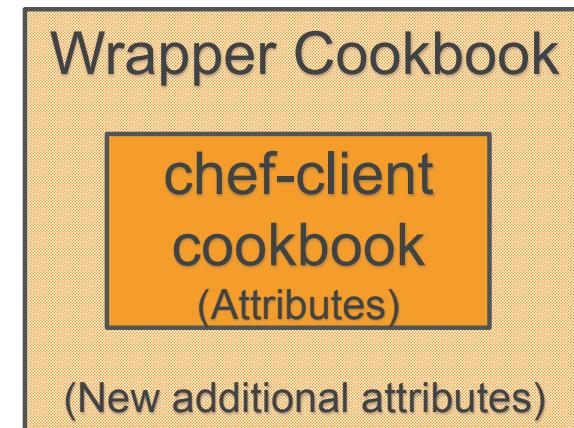
Don't use forked community cookbooks in production, or you will miss out on upstream changes, and will have to rebase

Instead use **wrapper cookbooks** to wrap upstream cookbooks and change their behavior without forking

# Supermarket Cookbooks

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

It defines new default values for the recipes.



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

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

# GL: Create my\_chef\_client Wrapper Cookbook



```
> cd chef-repo  
> chef generate cookbook cookbooks\my_chef_client
```

```
Generating cookbook my_chef_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\my_chef_client` to enter it.  
...
```

# GL: Edit my\_chef\_client Default Recipe

~\chef-repo\cookbooks\my\_chef\_client\recipes\default.rb

```
#  
# Cookbook Name:: my_chef_client  
# Recipe:: default  
#  
# Copyright (c) 2018 The Authors, All Rights Reserved.
```

```
include_recipe 'chef-client::default'
```

This recipe just calls the recipe `chef-client::default`

## GL: Update my\_chef\_client metadata.rb

```
~\chef-repo\cookbooks\my_chef_client\metadata.rb
```

```
name          'my_chef_client'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures my_chef_client'  
long_description 'Installs/Configures my_chef_client'  
version        '0.1.0'  
  
depends      'chef-client'
```

# LAB

## 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:

- ✓ Create a cookbook that wraps the chef-client cookbook
- Upload chef-client cookbook to Chef Server
- Configure each nodes to run chef-client as a service
- Converge each node and verify the service is running



# GL: Install the Dependencies



```
> cd cookbooks\my_chef_client  
> berks install
```

```
Resolving cookbook dependencies...  
Fetching 'my_chef_client' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Using chef-client (8.1.8)  
Using logrotate (2.2.0)  
Using my_chef_client (0.1.0) from source at .  
Using ohai (5.2.0)  
Installing windows (3.1.3)  
Using compat_resource (12.19.0)  
Using cron (4.1.3)
```

# GL: View Berksfile.lock



```
> gc Berksfile.lock
```

```
DEPENDENCIES

my_chef_client
  path: .
  metadata: true

GRAPH

chef-client (8.1.8)
  cron (>= 2.0.0)
  logrotate (>= 1.9.0)
  windows (>= 2.0.0)
  compat_resource (12.19.0)
  cron (4.1.3)
  compat_resource (>= 0.0.0)....
```

## GL: View the Berkshelf in Your Home Directory



```
> dir ~\berkshelf\cookbooks
```

```
drwxr-xr-x 12 YOU staff 408 2 Dec 15:46 chef_handler-1.1.6
drwxr-xr-x 12 YOU staff 408 2 Dec 15:46 cron-1.6.1
drwxr-xr-x 21 YOU staff 714 2 Dec 15:46 logrotate-1.8.0
drwxr-xr-x 15 YOU staff 510 2 Dec 15:46 windows-1.36.6
drwxr-xr-x 9 YOU staff 306 28 Nov 15:22 build-essential-2.1.3
drwxr-xr-x 25 YOU staff 850 6 Nov 16:27 build-essential-2.1.2
drwxr-xr-x 11 YOU staff 374 6 Nov 16:27 cpu-0.2.0
```

Note: Windows users can also use `ls -force ~\berkshelf\cookbooks`

# GL: Upload Cookbooks to Chef Server



```
> berks upload
```

```
Uploaded chef-client (8.1.8) to: 'https://api.chef.io:443/organizations/  
awesomestudent'  
Uploaded compat_resource (12.19.0) to: 'https://api.chef.io:443/organizations/  
awesomestudent'
```

# GL: View Cookbooks on Chef Server



```
> knife cookbook list
```

```
chef-client          8.1.8
compat_resource     12.19.0
cron                4.1.3
logrotate            2.2.0
my_chef_client      0.1.0
myiis               0.2.1
ohai                5.2.0
windows              3.1.3
workstation          0.1.0
```

# LAB

## 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:

- ✓ Create a cookbook that wraps the chef-client cookbook
- ✓ Upload chef-client cookbook to Chef Server
- ❑ Configure each nodes to run chef-client as a service
- ❑ Converge each node and verify the service is running

# 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
```

## 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[my_chef_client]'
```

## GL: Add the base Role to web Role

```
1 ~\chef-repo\roles\web.rb
```

```
name 'web'  
description 'Web Server'  
run_list 'role[base]', 'recipe[myiis]'
```

## GL: Upload the roles File



```
> cd ~\chef-repo  
> knife role from file base.rb web.rb
```

```
Updated Role base
```

```
Updated Role web
```

# LAB

## 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:

- ✓ Create a cookbook that wraps the chef-client cookbook
- ✓ Upload chef-client cookbook to Chef Server
- ✓ Configure each nodes to run chef-client as a service
- Converge each node and verify the service is running

# GL: Converge All Nodes



```
> knife winrm role:web -a cloud.public_ipv4 -x USER -P PWD  
"chef-client"
```

```
54.146.147.44 [2018-09-20T21:21:05+00:00] INFO: *** Chef 13.4.24 ***  
54.146.147.44 [2018-09-20T21:21:05+00:00] INFO: Platform: x64-mingw32  
...  
54.146.147.44 [2018-09-20T21:21:09+00:00] INFO: Starting Chef Run for node2  
54.146.147.44 [2018-09-20T21:21:09+00:00] INFO: Running start handlers  
54.146.147.44 [2018-09-20T21:21:09+00:00] INFO: Start handlers complete  
...  
174.129.71.79 [2018-09-20T21:21:20+00:00] INFO: Run List is [role[web]]  
174.129.71.79 [2018-09-20T21:21:20+00:00] INFO: Run List expands to [my_chef_client,  
myiis]  
174.129.71.79 [2018-09-20T21:21:20+00:00] INFO: Starting Chef Run for web1
```

## GL: Verify chef-client is Running



```
> knife winrm role:web -a cloud.public_ipv4 -x USER -P PWD  
"schtasks | findstr chef-client"
```

54.146.147.44 chef-client  
174.129.71.79 chef-client

9/20/2018 9:51:00 PM Ready  
9/20/2018 9:51:00 PM Ready

# LAB

## Introducing chef-client Cookbook

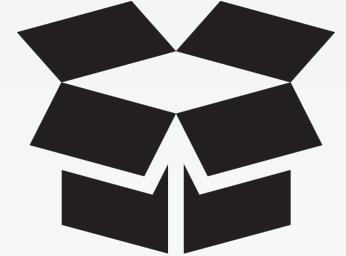


*We will use the chef-client community cookbook to configure chef-client on each of our nodes to run periodically*

### OBJECTIVE:

- ✓ Create a cookbook that wraps the chef-client cookbook
- ✓ Upload chef-client cookbook to Chef Server
- ✓ Configure each nodes to run chef-client as a service
- ✓ Converge each node and verify the service is running

# CONCEPT

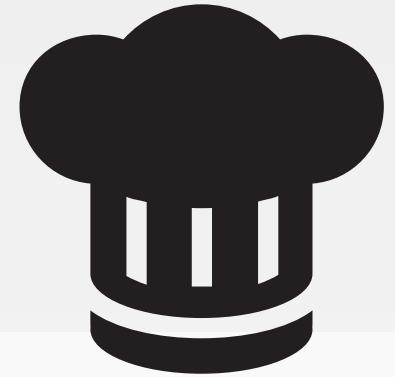


## **chef-client has Default Settings**

The chef-client cookbook provides default attributes for a lot of application settings. We can provide new values to override the defaults.

# LAB

## Converge Faster!

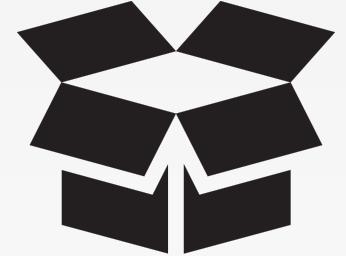


*Converging every 30 minutes is too slow. Can we make it every 5 minutes?*

### OBJECTIVE:

- Find the attribute to override and the location to override it
- Override the attribute
- Upload the cookbook and converge all nodes with the web role

# CONCEPT



## Attribute Files

The Node Object contains many automatic attributes generate by Ohai.

You can also define attributes within a cookbook. These are like variables or parameters for your cookbook and allow recipes to be data driven.

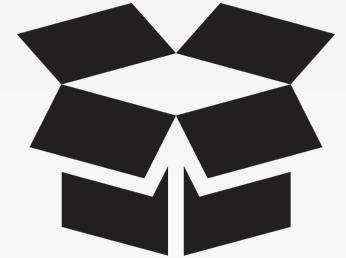
## GL: Finding the chef-client run Interval Default

```
~\.berkshelf\cookbooks\chef-client<VER>\attributes\default.rb
```

```
...
default['chef_client']['log_file']      = 'client.log'
default['chef_client']['interval']       = '1800'
default['chef_client']['splay']          = '300'
default['chef_client']['conf_dir']        = '/etc/chef'
default['chef_client']['bin']            = '/usr/bin/chef-client'
...
...
```

1800 seconds (30 minutes) is the time to wait between converges.

# CONCEPT



## Defining Attributes

precedence      attribute name      attribute value

```
default[ "attributename" ] = "value"
```

# Default Attribute Precedence



Please note this is a simplified diagram, and the precedence shown can be overridden.



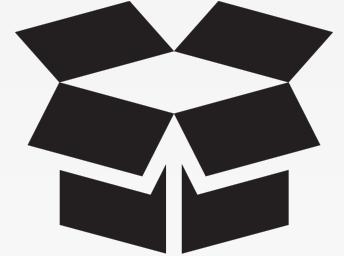
# Full Attribute Precedence

	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	



# CONCEPT

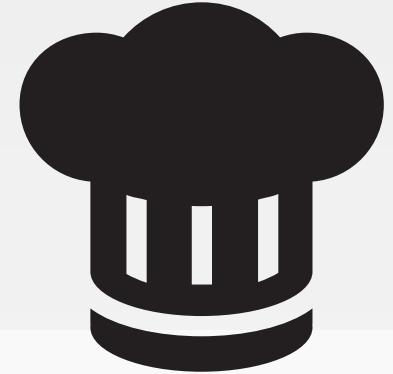
## Best Practice: Do Not Fork Community Cookbooks



- Unable to grab updated versions of the cookbook without merging
- Breaking of default expectations that people come to expect
- A small change means a new version; making it even harder to reintegrate when versions collide.

# LAB

## Converge Faster!



*Converging every 30 minutes is too slow. Can we make it every 5 minutes?*

### OBJECTIVE:

- ✓ Find the attribute to override and the location to override it
- ❑ Override the attribute
- ❑ Upload the cookbook and converge all nodes with the web role

# GL: Edit my\_chef\_client Default Recipe

~\chef-repo\cookbooks\my\_chef\_client\recipes\default.rb

```
#  
# Cookbook Name:: my_chef_client  
# Recipe:: default  
#  
# Copyright (c) 2018 The Authors, All Rights Reserved.  
  
node.default['chef_client']['interval'] = '300'  
  
include_recipe 'chef-client::default'
```

Note 'node.default' notation when setting an attribute value from within a recipe

# Full 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	



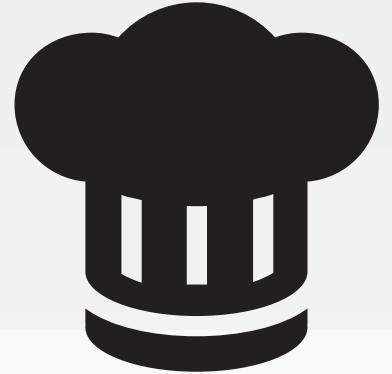
## GL: Update my\_chef\_client metadata.rb

```
~\chef-repo\cookbooks\my_chef_client\metadata.rb
```

```
name          'my_chef_client'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license        'all_rights'  
description    'Installs/Configures my_chef_client'  
long_description 'Installs/Configures my_chef_client'  
version        '0.1.1'  
  
Depends 'chef-client'
```

# LAB

## Converge Faster!



*Converging every 30 minutes is too slow. Can we make it every 5 minutes?*

### OBJECTIVE:

- ✓ Find the attribute to override and the location to override it
- ✓ Override the attribute
- Upload the cookbook and converge all nodes with the web role

## GL: Run berks install



```
> cd cookbooks\my_chef_client  
> berks install
```

```
...  
Fetching 'my_chef_client' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Using chef-client (8.1.8)  
Using logrotate (2.2.0)  
Using my_chef_client (0.1.1) from source at .  
Using ohai (5.2.0)  
Using windows (3.1.3)  
Using compat_resource (12.19.0)
```

# GL: Upload the Cookbook to the Chef Server



```
> berks upload
```

```
Skipping chef-client (8.1.8) (frozen)
Skipping compat_resource (12.19.0) (frozen)
Skipping cron (4.1.3) (frozen)
Skipping logrotate (2.2.0) (frozen)
Uploaded my_chef_client (0.1.1) to: 'https://api.chef.io:443/organizations/awesomestudent'
Skipping chai (5.2.0) (frozen)
Skipping windows (3.1.3) (frozen)
```

# GL: Converge All Nodes



```
> knife winrm role:web -a cloud.public_ipv4 -x USER -P PWD  
"chef-client"
```

```
34.196.63.231 [2018-01-24T15:17:50+00:00] INFO: *** Chef 12.13.37 ***  
34.196.63.231 [2018-01-24T15:17:50+00:00] INFO: Platform: i386-mingw32  
34.196.63.231 [2018-01-24T15:17:50+00:00] INFO: Chef-client pid: 2380  
34.196.63.231 [2018-01-24T15:18:28+00:00] INFO: Run List is [role[web]]  
34.196.63.231 [2018-01-24T15:18:28+00:00] INFO: Run List expands to [my_chef_client,  
myiis]  
34.196.63.231 [2018-01-24T15:18:28+00:00] INFO: Starting Chef Run for node1  
34.196.63.231 [2018-01-24T15:18:28+00:00] INFO: Running start handlers  
34.196.63.231 [2018-01-24T15:18:28+00:00] INFO: Start handlers complete.  
...
```

## GL: Verify chef-client is Running



```
> knife winrm role:web -a cloud.public_ipv4 -x USER -P PWD  
"schtasks | findstr chef-client"
```

54.146.147.44 chef-client  
174.129.71.79 chef-client

9/20/2018 9:51:00 PM Ready  
9/20/2018 9:51:00 PM Ready

## Q&A



What questions can we help you answer?

- Chef Supermarket
- Wrapper Cookbooks
- chef-client
- node attributes



CHEF™

# Search

Update a Cookbook to Dynamically Use Nodes with the Web Role

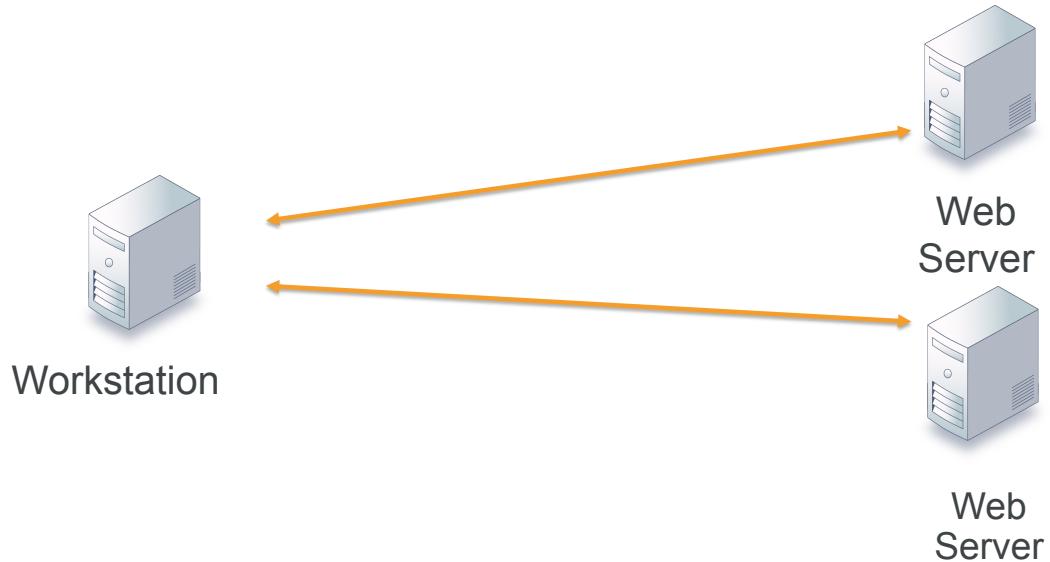
# Objectives

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
- Create load balancer cookbook that dynamically finds nodes with an assigned role

# Our current Environment

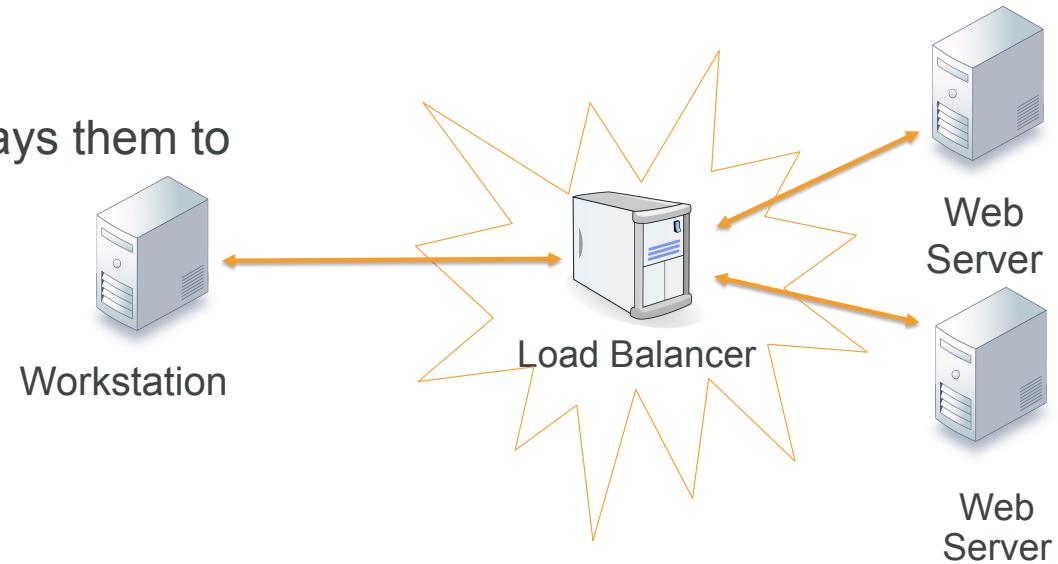
Currently we have two web servers  
that we must browse to individually



# Introducing a Load Balancer

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

Receives requests and relays them to other systems.

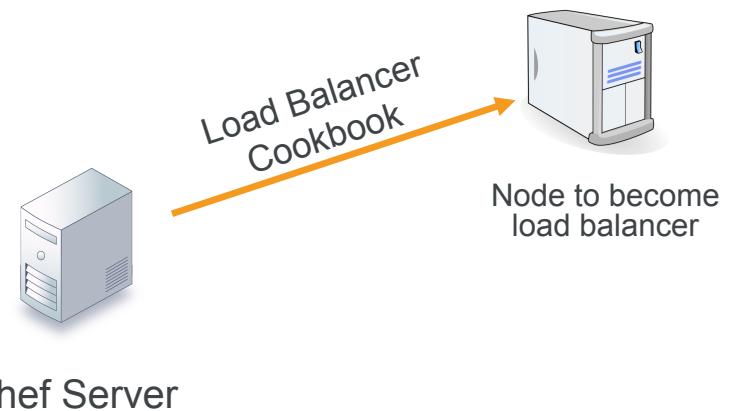


# Load Balancer

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

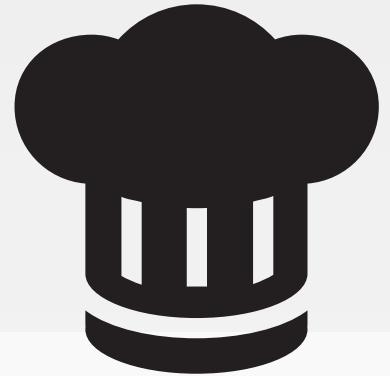
Write a load balancer cookbook.

Establish a new node within our organization to which we apply that cookbook.



# LAB

## Load Balancer



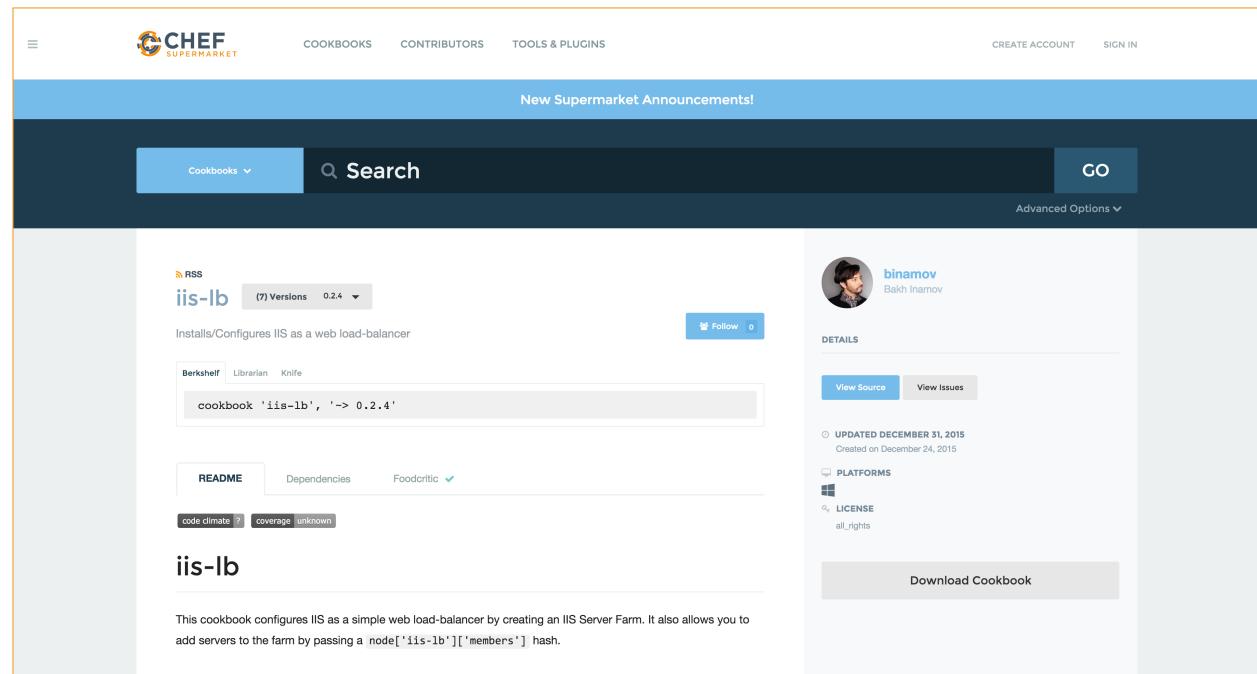
*Adding a load balancer will allow us to better grow our infrastructure.*

### OBJECTIVE:

- Find or Create a Cookbook to Manage a load balancer
- Configure the load balancer to send traffic to the web servers
- Upload cookbook to Chef Server
- Create 'load\_balancer' role that includes the 'base' role
- Bootstrap a new node (lb1) that runs the load balancer cookbook (via 'load\_balancer' role)

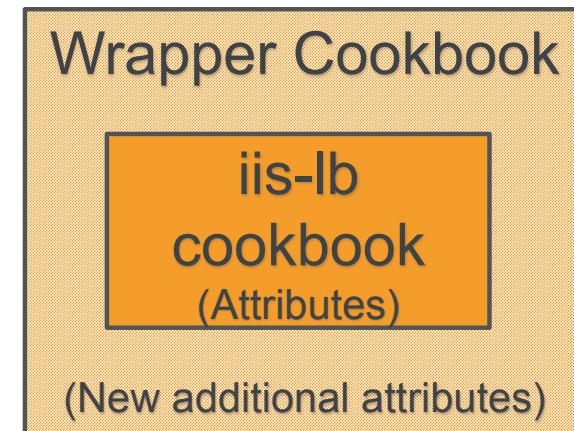
# Supermarket Cookbooks

We will use the iis-lb community cookbook



# Supermarket Cookbooks

We don't want to change the community iis-lb cookbook, so we'll wrap it in a myiis\_lb cookbook



# GL: Generate the Wrapper Cookbook



```
> cd ~\chef-repo  
> chef generate cookbook cookbooks\myiis_lb
```

```
Generating cookbook myiis_lb
```

- 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\myiis_lb` to enter it.
```

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

## GL: Create a dependency and set version

```
~\chef-repo\cookbooks\myiis_lb\metadata.rb
```

```
name          'myiis_lb'  
maintainer    'The Authors'  
maintainer_email 'you@example.com'  
license       'all_rights'  
description   'Installs/Configures myiis_lb'  
long_description 'Installs/Configures myiis_lb'  
version       '0.1.0'
```

```
depends 'iis-lb'
```

+

# Demo: Edit the myiis\_lb Default Recipe

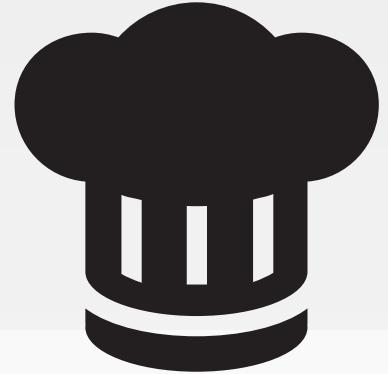
~\chef-repo\cookbooks\myiis\_lb\recipes\default.rb

```
#  
# Cookbook Name:: myiis-lb  
# Recipe:: default  
#  
# Copyright (c) 2018 The Authors, All Rights Reserved.
```

@2018 Chef Software Inc.

# LAB

## Load Balancer



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

### OBJECTIVE:

- ✓ Find or create a cookbook to manage a load balancer
- ❑ Configure the load balancer to send traffic to the web servers
- ❑ Upload cookbook to Chef Server
- ❑ Create 'load\_balancer' role that includes the 'base' role
- ❑ Bootstrap a new node (node3) that runs the load balancer cookbook (via 'load\_balancer' role)

# iis-lb Supermarket Cookbook

In the iis-lb cookbook, web servers in the load balancing pool are listed as attributes (iis-lb/attributes/default.rb)

The example in the cookbook's **README.md** illustrates how to overwrite these, but we need to add the IP Address or Hostname of our **node1** and **node2**

## Usage

Specify your servers by setting the `node['iis-lb']['members']` attribute hash in a wrapper cookbook. Then include the `'iis-lb::default'` recipe. This creates the Server Farm and adds your servers to it. For example:

```
# contents of chef-repo/cookbooks/my-wrapper-cookbook/recipes/default.rb
node.default['iis-lb']['members'] = [
  {
    'address' => 'localhost',
    'weight' => 100,
    'port' => 4000,
    'ssl_port' => 4000
  },
  {
    'address' => '127.0.0.1',
    'weight' => 100,
    'port' => 4001,
    'ssl_port' => 4001
  }
]
```

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

## Demo: myiis\_lb default Recipe Could Resemble This

```
~\chef-repo\cookbooks\myiis_lb\recipes\default.rb
```

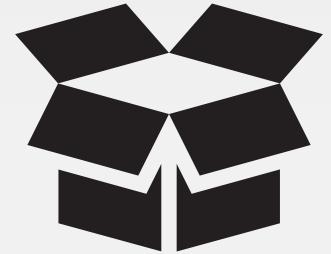
```
node.default['iis-lb']['members'] = [
  {
    'address' => '34.196.63.231',
    'weight' => 100,
    'port' => 80,
    'ssl_port' => 80
  },
  {
    'address' => '34.196.104.17',
    'weight' => 100,
    'port' => 80,
    'ssl_port' => 80
  }
]

include_recipe 'iis-lb::default'
```

©2018 Chef Software Inc.

# CONCEPT

## Using Search



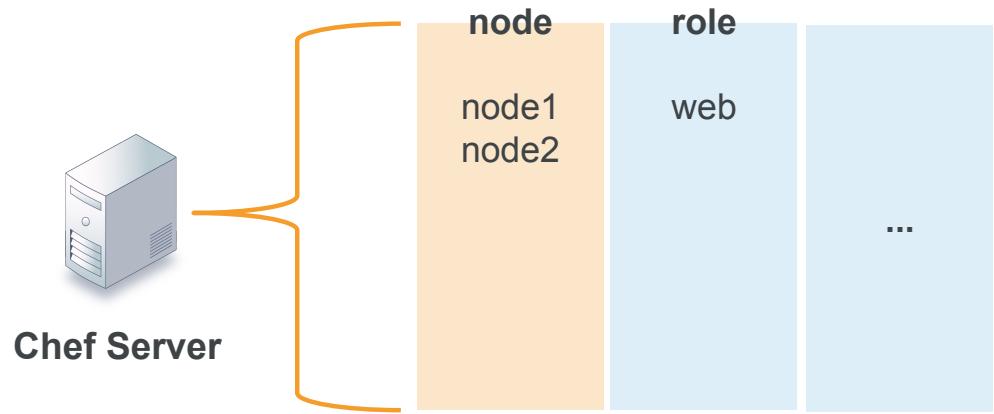
It seems inefficient to have to update a cookbook recipe each time we add another web server

To add new web servers to load balancing pool, we would need to bootstrap a new web server and then update our load balancer's myiis\_lb cookbook recipe

We'll use search to update the load balancer automatically

# The Chef Server and Search

Chef Server maintains a representation of all the nodes, roles, clients, etc. within our infrastructure. This information can be used to help our recipes discover other services within our organization.

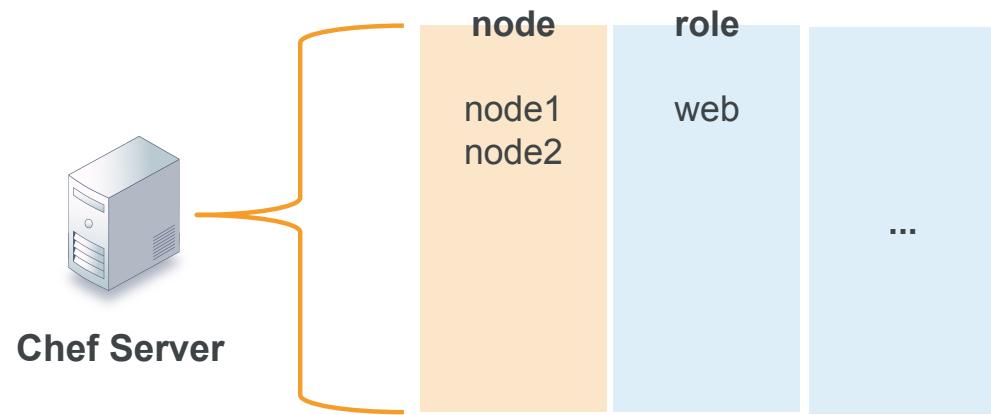


[https://docs.chef.io/chef\\_search.html](https://docs.chef.io/chef_search.html)

[https://docs.chef.io/chef\\_search.html#search-indexes](https://docs.chef.io/chef_search.html#search-indexes)

# The Chef Server 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` method.



Query	knife search command	search method	results
Find me all the <i>nodes</i> with the <i>name</i> node1	<code>knife search node name:node1</code>	<code>search('node', 'name:node1')</code>	node1
Find me all the <i>nodes</i> with the <i>role</i> web	<code>knife search node role:web</code>	<code>search('node', 'role:web')</code>	node1, node2

# 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 (node, role, environment, databag) and search\_pattern defines what will be searched for

The search criteria "`*:*`" returns every node

# GL: View Information for All Nodes



```
> knife search node "*:*"
```

```
Node Name: node1
Environment: _default
FQDN: WIN-DCK5NTVVLBH
IP: 34.196.63.231
Run List: role[web]
Roles: web, base
Recipes: my_chef_client, my_chef_client::default, myiis, myiis::default, chef-client::default, chef-client::task, myiis::server
Platform: windows 6.3.9600
Tags:

Node Name: node2
Environment: _default
FQDN: WIN-DCK5NTVVLBH
IP: 34.196.104.17
Run List: role[web]
Roles: web, base
Recipes: my_chef_client, my_chef_client::default, myiis, myiis::default, chef-client::default, chef-client::task, myiis::server
Platform: windows 6.3.9600
Tags:
```

# GL: Narrow the Search for all Web Nodes



```
> knife search node "role:web" -a cloud.public_hostname
```

```
node1:
```

```
  cloud.public_hostname: ec2-34-196-63-231.compute-1.amazonaws.com
```

```
node2:
```

```
  cloud.public_hostname: ec2-34-196-104-17.compute-1.amazonaws.com
```

# Search Syntax within a Recipe

```
all_web_nodes = search('node', 'role:web')
```

Search the Chef Server for all **nodes** that have the **role** equal to '**web**' and store the results into a local variable named '**all\_web\_nodes**'.

# Search Syntax within a Recipe

```
all_web_nodes = search('node', 'role:web')
```

creates and names a variable

assigns the value of the  
operation on the right  
into the variable on the left

invokes the search method

the index or items to search

the search criteria - key:value

# GL: Edit the myiis\_lb Default Recipe

~\chef-repo\cookbooks\myiis\_lb\recipes\default.rb

```
all_web_nodes = search('node', 'role:web')

members = []

all_web_nodes.each do |web_node|
  member = {
    'address' => web_node['cloud']['public_hostname'],
    'weight' => 100,
    'port' => 80,
    'ssl_port' => 80
  }
  members.push(member)
end

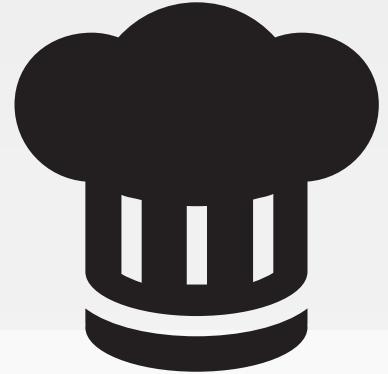
node.default['iis-lb']['members'] = members

include_recipe 'iis-lb::default'
```

©2018 Chef Software Inc.

# LAB

## Load Balancer



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

### OBJECTIVE:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the web servers
- ❑ Upload cookbook to Chef Server
- ❑ Create 'load\_balancer' that includes the 'base' role
- ❑ Bootstrap a new node (lb1) that runs the load balancer cookbook (via 'load\_balancer' role)



## Lab: Upload the Cookbook

- Upload the cookbook to the Chef Server

# Lab: Install Cookbook Dependencies



```
> cd ~\chef-repo\cookbooks\myiis_lb  
> berks install
```

```
Resolving cookbook dependencies...  
Fetching 'myiis_lb' from source at .  
Fetching cookbook index from https://supermarket.chef.io...  
Using compat_resource (12.16.3)  
Installing iis (5.0.5)  
Installing iis-lb (0.2.4)  
Using myiis_lb (0.1.0) from source at .  
Using chai (4.2.3)  
Installing webpi (3.1.0)  
Using windows (2.1.1)
```

# Lab: Upload the Cookbook to Chef Server



```
> berks upload
```

```
Skipping compat_resource (12.16.3) (frozen)
Uploaded iis (5.0.5) to: 'https://api.chef.io:443/organizations/awesomestudent'
Uploaded iis-lb (0.2.4) to: 'https://api.chef.io:443/organizations/awesomestudent'
Uploaded myiis_lb (0.1.0) to: 'https://api.chef.io:443/organizations/awesomestudent'
Skipping ohai (4.2.3) (frozen)
Uploaded webpi (3.1.0) to: 'https://api.chef.io:443/organizations/awesomestudent'
Skipping windows (2.1.1) (frozen)
```

# Lab: Verify the Cookbook Upload



```
> knife cookbook list
```

```
chef-client      7.1.0
compat_resource 12.16.3
cron            3.0.0
iis              5.0.5
iis-lb           0.2.4
logrotate        2.1.0
my_chef_client   0.1.1
myiis            0.2.1
myiis_lb          0.1.0
ohai             4.2.3
webpi            3.1.0
windows          2.1.1
workstation       0.1.0
```

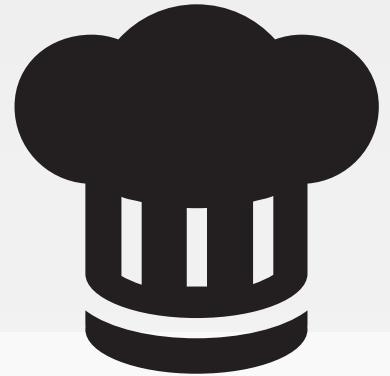


## Lab: Upload the Cookbook

- ✓ Upload the cookbook to the Chef Server

# LAB

## Load Balancer



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

### OBJECTIVE:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the web servers
- ✓ Upload cookbook to Chef Server
- Create 'load\_balancer' role that includes the 'base' role
- Bootstrap a new node (lb1) that runs the load balancer cookbook (via 'load\_balancer' role)

## GL: Create the Load Balancer Role

```
[ ] ~\chef-repo\roles\load_balancer.rb
```

```
name 'load_balancer'  
description 'Load Balancer'  
run_list 'role[base]', 'recipe[myiis_lb]'
```

©2018 Chef Software Inc.

## GL: Upload the Role to the Chef Server



```
> cd ~\chef-repo  
> knife role from file load_balancer.rb
```

```
Updated Role load_balancer
```

# GL: Validate Chef Server Received It

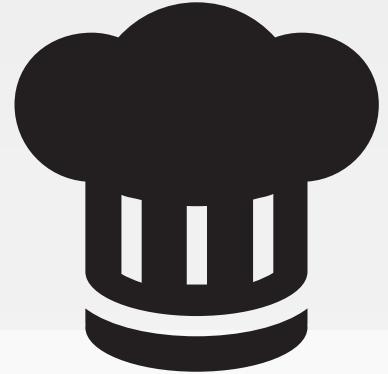


```
> knife role list
```

```
base
load_balancer
web
```

# LAB

## Load Balancer



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

### OBJECTIVE:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the web servers
- ✓ Upload cookbook to Chef Server
- ✓ Create 'load\_balancer' role that includes the 'base' role
- Bootstrap a new node (lb1) that runs the load balancer cookbook (via 'load\_balancer' role)

# Lab: Bootstrap a New Node



```
> knife bootstrap windows winrm IP_OF_NODE3 -x USER -P PWD  
-N lb1
```

```
Creating new client for lb1  
Creating new node for lb1  
  
Waiting for remote response before bootstrap.34.196.106.169 .  
34.196.106.169 Response received.  
Remote node responded after 0.02 minutes.  
Bootstrapping Chef on 34.196.106.169  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 1  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 2  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 3  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 4  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 5  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 6  
34.196.106.169 Rendering "C:\Users\ADMINI~1\AppData\Local\Temp\bootstrap-3664-1485280632.bat" chunk 7  
...
```

# Lab: Add a Recipe to a Run List

 > knife node run\_list set lb1  
"role[load\_balancer]"

```
lb1:  
  run_list: role[load_balancer]
```

# Lab: Validate the Run List



```
> knife node show lb1
```

```
Node Name:      lb1
Environment:    _default
FQDN:          WIN-DCK5NTVVLBH
IP:            34.196.106.169
Run List:       role[load_balancer]
Roles:          load_balancer, base
Recipes:        my_chef_client, my_chef_client::default, myiis_lb,
                myiis_lb::default, chef-client::default, chef-client::task, iis-lb::default,
                iis-lb::arr, webpi::default, webpi::install-msi
Platform:       windows 6.3.9600
Tags:
```

# Lab: Change the web server cache time



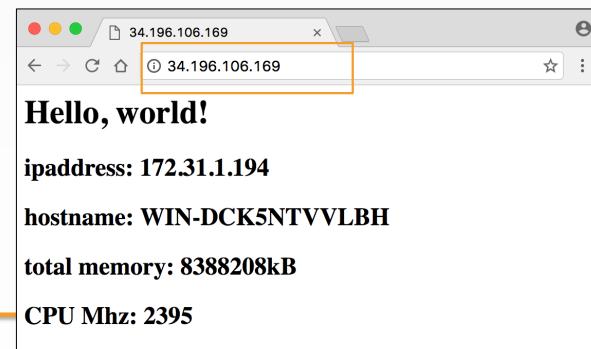
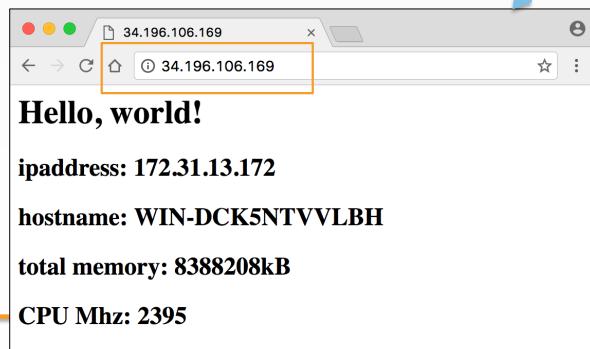
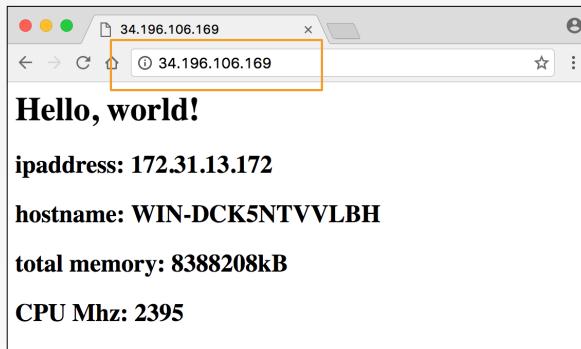
IIS Server has 60 second server side cache turned on by default.  
To switch this off, RDP into lb1 (load balancer node) and:

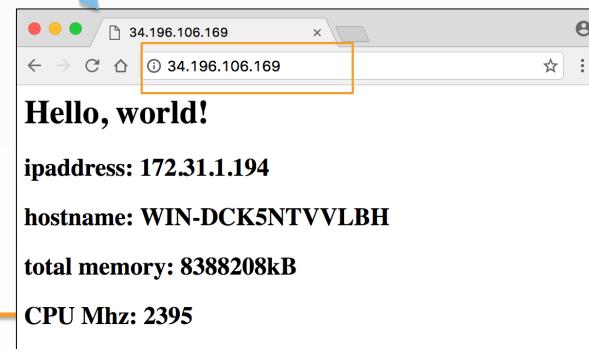
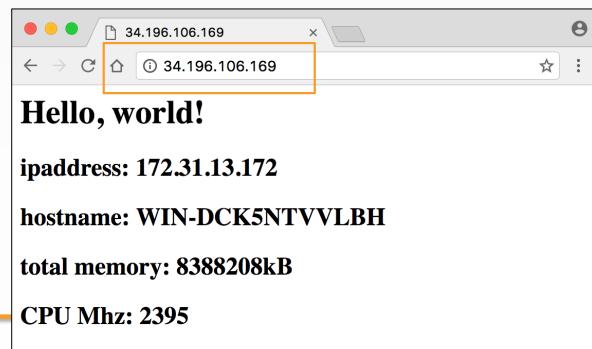
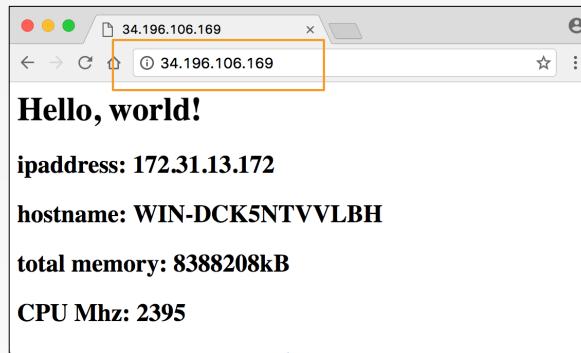
1. Start menu > Administrative Tools >
2. Internet Information Services (IIS) Manager.
3. Expand your node on left side > Server Farms > myServerFarm
4. Double click 'caching', and set to '0'
5. Close & save

Point a web browser to the IP address of your load balancer node.

Refresh the page a few times to see the load balancer serve up each web server.

Please see notes below this slide.





## 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?

## Q&A



What questions can we help you answer?



CHEF™



# Testing Cookbooks

Eliminating Risk

©2018 Chef Software Inc.



# Agenda

## Day 1

---

Getting a Workstation  
Using Resources  
Building Cookbooks  
chef-client  
Ohai and the Node Object  
Connecting to Chef Server  
Community Cookbooks  
Search

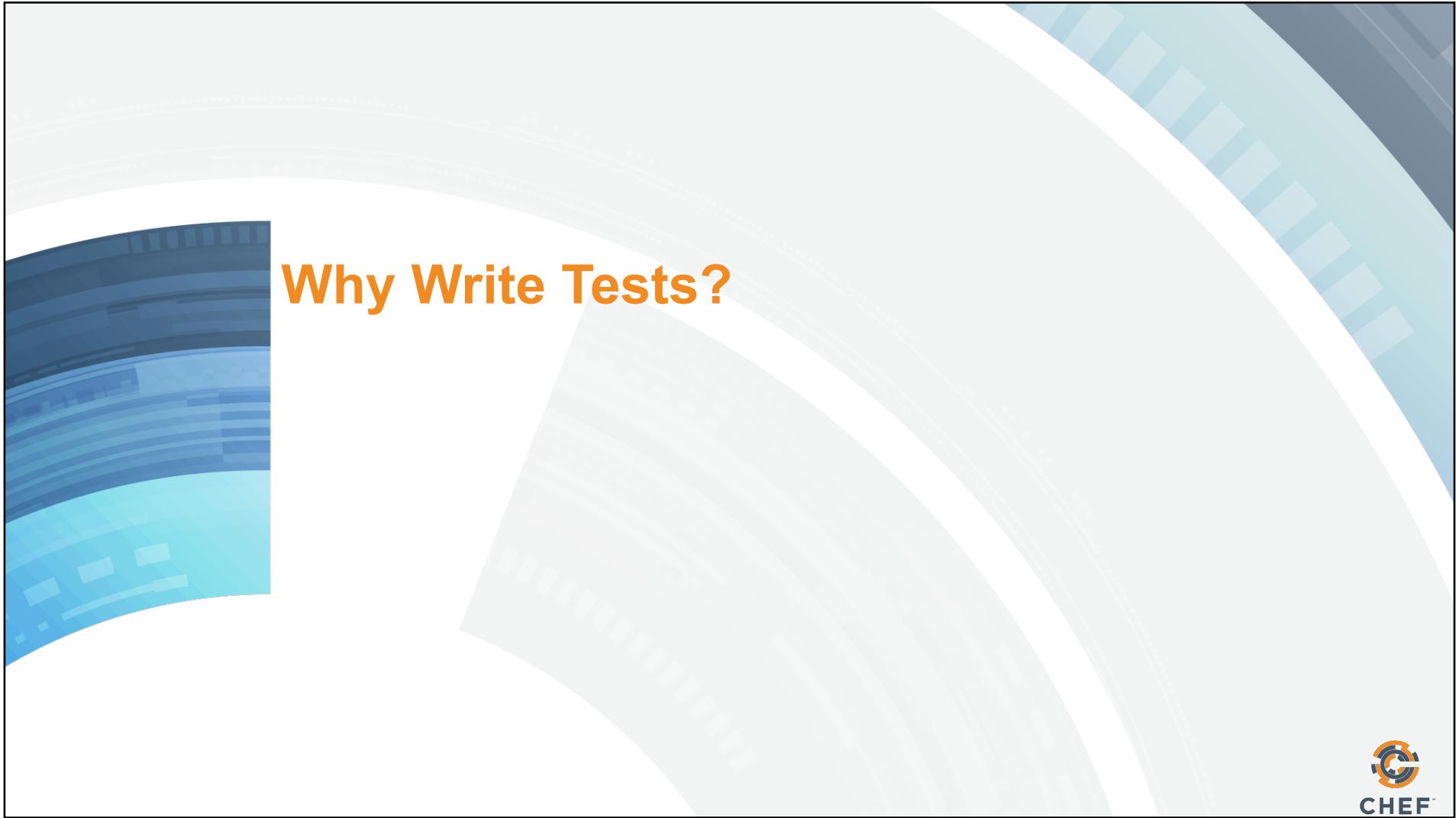
## Day 2

---

Why Write Tests?  
Writing a Test First  
Refactoring Cookbooks with Tests  
Faster Feedback with Unit Testing  
Testing Resources in Recipes  
Refactoring to Attributes  
Refactoring to Multiple Platforms



CHEF™



## Why Write Tests?



# EXERCISE



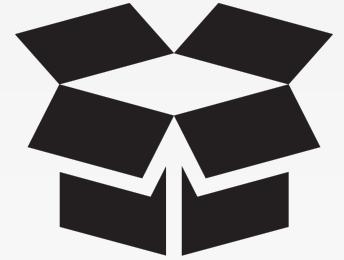
## Why Write Tests?

*Should I write a test? Perhaps the answer to that question lies in: why write tests?*

### **Objective:**

- Discussion about Writing Tests
- Discussion about Why Writing Tests is Hard

# CONCEPT



## Current Cookbook Development Workflow



1. Write some cookbook code
2. Perform ad-hoc verification
3. Upload the cookbook to the Chef Server



# CONCEPT

## Current Cookbook Development Workflow



1. Chef-client run retrieves the updated cookbook
2. Perform ad-hoc verification
3. Promote the cookbook to the next environment

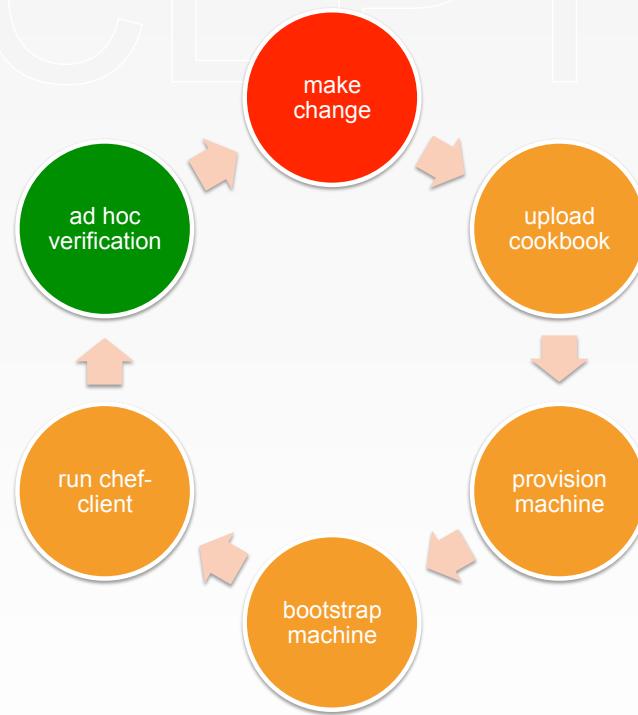
# PROBLEM

## Risk

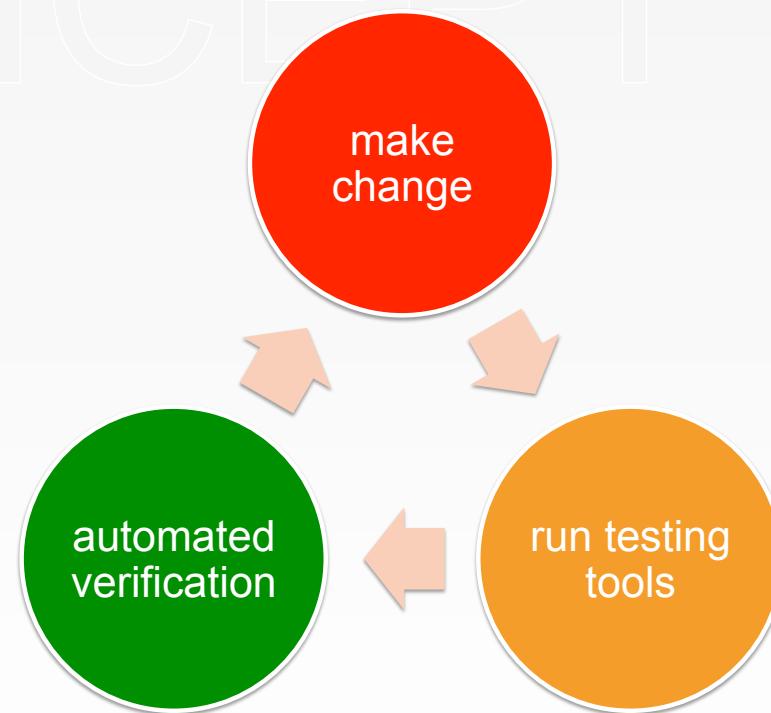
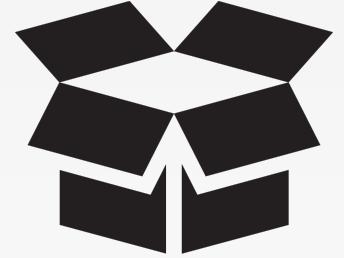


Every change to our cookbooks introduce risk. Validating every change would take too long in this system. To alleviate that we often batch these changes up. Batching up the changes make it harder to discover when we introduced an error.

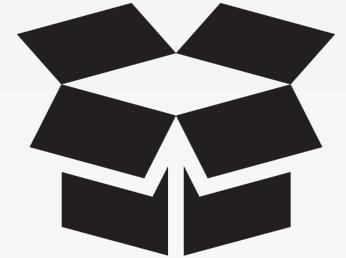
# CONCEPT



# CONCEPT



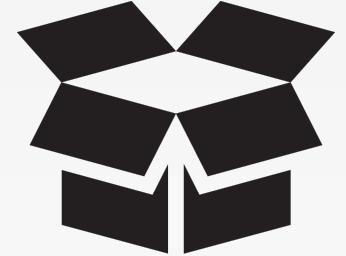
# CONCEPT



## Another Language

Learning to write tests require you to learn a whole new language that you must understand grammatically.

# CONCEPT



## Another Workflow

Executing tests requires learning new tools, commands, flags and configurations with entirely new mechanisms that provide you feedback.



## Discussion



What are reasons you see for writing tests?

What are reasons you see for not writing tests?

## Q&A



What questions can we answer for you?



CHEF™



# Testing Cookbooks

Validating Our Recipes in Virtual Environments

©2019 Chef Software Inc.



# Objectives

After completing this module, you should be able to:

- Use Test Kitchen to verify your recipes converge on a virtual instance
- Refer to the InSpec documentation
- Write and execute tests



## We Should Test Cookbooks

As we start to define our infrastructure as code we also need to start thinking about testing it.

We have an automated way to ensure code accomplishes the intended goal and help the team understand its intent.  
It's called Test Kitchen.

# Steps to Verify Cookbooks

Create Virtual Machine

Install Chef Tools

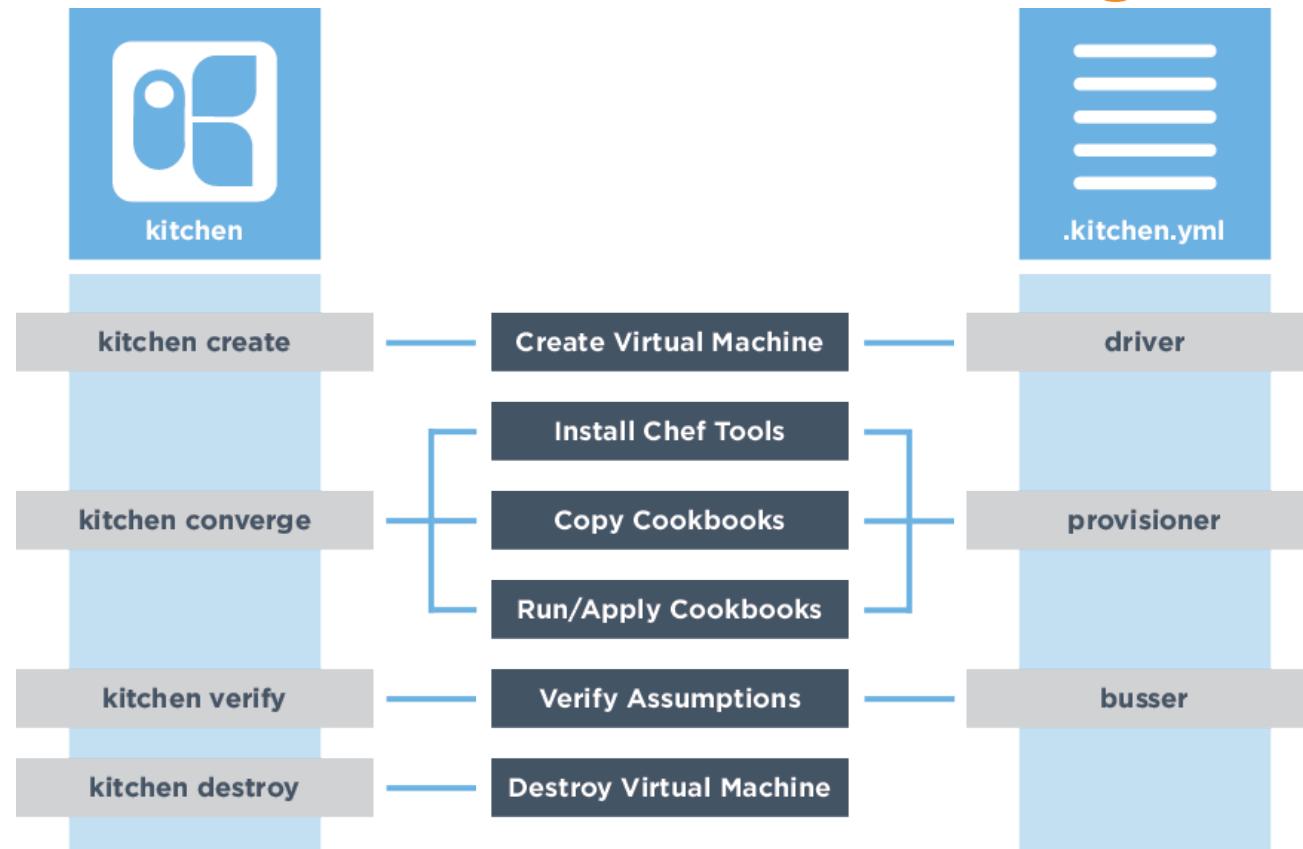
Copy Cookbooks

Run/Apply Cookbooks

Verify Assumptions

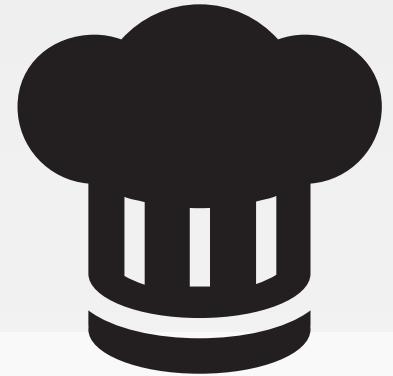
Destroy Virtual Machine

# Test Kitchen Commands and Configuration



# LAB

## Test Configuration



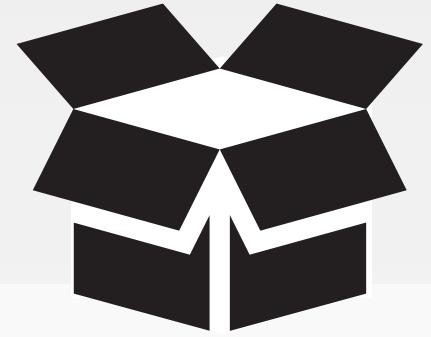
*What are we running in production? Maybe I could test the cookbook against a virtual machine.*

### OBJECTIVE:

- Configure the "workstation" cookbook to test against a Windows 2012R2 platform
- Apply the "workstation" cookbook's default recipe to that virtual machine

# CONCEPT

## .kitchen.yml



When chef generates a cookbook, a default .kitchen.yml is created. It contains kitchen configuration for the driver, provisioner, platform, and suites.

<http://kitchen.ci/docs/getting-started/creating-cookbook>

# Demo: The kitchen Driver

```
~\chef-repo\cookbooks\workstation\.kitchen.yml
```

```
---
```

```
driver:
```

```
  name: vagrant
```

```
provisioner:
```

```
  name: chef_zero
```

```
# Uncomment the following verifier...
```

```
# default verifier)
```

```
# verifier:
```

```
#   name: inspec
```

```
# KITCHEN CONFIGURATION CONTINUES....
```

The driver is responsible for creating a machine that we'll use to test our cookbook.

Today we will use an EC2 driver that will spin up an instance in AWS that we can run Test Kitchen against.

# Demo: The kitchen Provisioner

~\chef-repo\cookbooks\workstation\.kitchen.yml

```
---
```

```
driver:
```

```
  name: vagrant
```

```
provisioner:
```

```
  name: chef_zero
```

```
# Uncomment the following verifier...
```

```
# default verifier)
```

```
# verifier:
```

```
#   name: inspec
```

```
# .KITCHEN CONFIGURATION CONTINUES....
```

This tells Test Kitchen how to run Chef, to apply the code in our cookbook to the machine under test.

The default and simplest approach is to use `chef_zero`.

# Demo: The kitchen Platforms



~\chef-repo\cookbooks\workstation\.kitchen.yml

```
# TOP PART OF KITCHEN CONFIGURATION

platforms:
  - name: ubuntu-14.04
  - name: centos-7.1

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

This is a list of operation systems on which we want to run our code.

# Demo: The kitchen Suites

~\chef-repo\cookbooks\workstation\.kitchen.yml

```
# TOP PART OF KITCHEN CONFIGURATION

platforms:
  - name: ubuntu-14.04
  - name: centos-7.1

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

This section defines what we want to test. It includes the Chef run-list of recipes that we want to test.

We define a single suite named "default".

# Demo: The kitchen Suites

~\chef-repo\cookbooks\workstation\.kitchen.yml

```
# TOP PART OF KITCHEN CONFIGURATION

platforms:
  - name: ubuntu-14.04
  - name: centos-7.1

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

The suite named "default" defines a run\_list.

Run the "workstation" cookbook's "default" recipe file.

# CONCEPT

## Kitchen Test Matrix



Kitchen defines a list of instances, or test matrix, based on the platforms multiplied by the suites.

PLATFORMS x SUITES

Running kitchen list will show that matrix.

# Example: View the Kitchen Test Matrix

```
> kitchen list
```

Instance	Driver	Provisioner	Last Action
default-ubuntu-1204	Vagrant	ChefZero	<Not Created>
default-centos-65	Vagrant	ChefZero	<Not Created>

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```

# Example: View the Kitchen Test Matrix

```
> kitchen list

Instance          Driver   Provisioner  Last Action
default-ubuntu-1204 Vagrant  ChefZero    <Not Created>
default-centos-65   Vagrant  ChefZero    <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
  attributes:
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```

# PROBLEM

## Virtualization



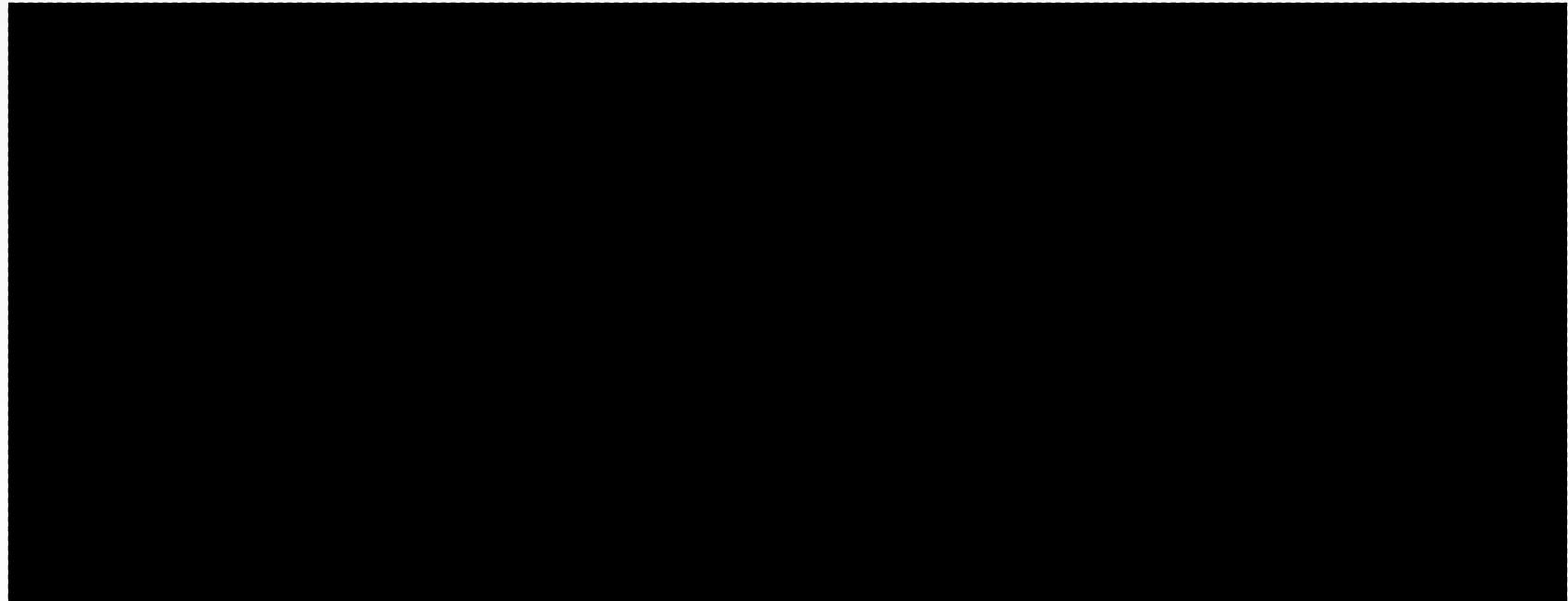
Vagrant is great for local development but when building cookbooks on a virtual machine in the cloud we will not need to use a local virtualization tool.

Instead, we will ask Amazon EC2 to provision nodes for us to test against.

# Move into the Cookbook Directory



```
> cd cookbooks\workstation
```



# Add Settings to the Kitchen Configuration

```
1 ~/chef-repo/cookbook/workstation/.kitchen.yml
```

```
---
driver:
  name: ec2
  aws_ssh_key_id: visa-tk1
  region: us-east-1
  availability_zone: b
  subnet_id: subnet-0028e52b
  instance_type: t2.2xlarge
  image_id: ami-0a37c3940b6d29915
```

## Add Settings to the Kitchen Configuration

```
1 ~/chef-repo/cookbook/workstation/.kitchen.yml
```

```
instance_type: t2.2xlarge
image_id: ami-0a37c3940b6d29915
security_group_ids: ["sg-3aa7a341"]
retryable_tries: 120
tags:
  # Replace YOURNAME
  Name: "YOURNAME HERE"
  created-by: "test-kitchen"
  user: <%= ENV['USER'] %>
```

## Add Settings to the Kitchen Configuration

```
1 ~/chef-repo/cookbook/workstation/.kitchen.yml
```

```
created-by: "test-kitchen"
user: <%= ENV['USER'] %>

provisioner:
  name: chef_zero

verifier:
  name: inspec
```

# Add Settings to the Kitchen

Configuration  
~/chef-repo/cookbook/workstation/.kitchen.yml

```
verifier:  
  name: inspec
```

```
transport:  
  name: winrm  
  elevated: true  
  username: Administrator  
  password: Cod3Can!
```

# Add Settings to the Kitchen

Configuration  
~/chef-repo/cookbook/workstation/.kitchen.yml

```
username: Administrator
```

```
password: Cod3Can!
```

```
platforms:
```

```
  - name: windows-2012R2
```

```
    driver:
```

```
      customize:
```

```
        memory: 4096
```

```
suites:
```

```
  - name: default
```

## GL: Add Your Name, Company & Cookbook

```
[ ] ~/chef-repo/cookbook/workstation/.kitchen.yml
```

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    verifier:
      inspec_tests:
        - test/integration/default
```

Line 40 – Replace ‘smoke’ with ‘integration’

# Complete File Can Be Downloaded



**<https://tinyurl.com/ybd7qq8w>**

©2019 Chef Software Inc.

# Create the AWS Credentials file



```
> cd ~
```

```
> mkdir .aws
```

# Create the AWS Credentials file



Download from <https://tinyurl.com/y824euq7>

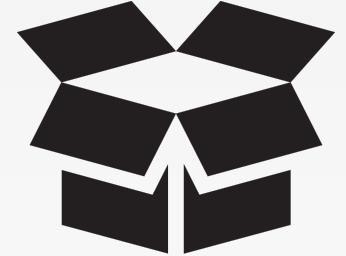
Place in `~/.aws/credentials`

[default]

`aws_access_key_id = AKIAI6Y5IS67EBHWTK3A`

`aws_secret_access_key = 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`

# CONCEPT



## Kitchen List

Kitchen defines a list of instances, or test matrix, based on the **platforms** multiplied by the **suites**.

PLATFORMS x SUITES

Running `kitchen list` will show that matrix.

# View the Test Matrix for Test Kitchen



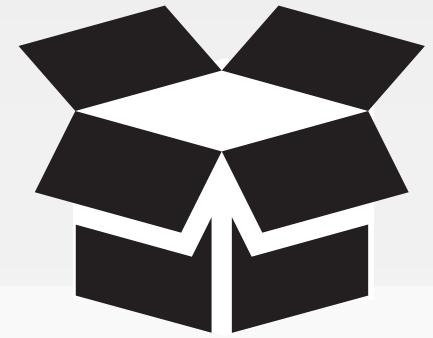
```
> kitchen list
```

Instance	Driver	Provisioner	Verifier	Transport	Last Action
default-windows-2012R2	EC2	ChefZero	InSpec	Winrm	<Not Created>



# CONCEPT

## Kitchen Create



kitchen  
create

kitchen  
converge

kitchen  
verify

> **kitchen create [INSTANCE|REGEXP|all]**

**Create one or more instances.**

# GL: Converge the Cookbook



```
> cd ~\chef-repo\cookbooks\workstation  
> kitchen create
```

```
-----> Starting Kitchen (v1.11.1)  
-----> Creating <default-windows-2012r2>...  
    Instance <i-1be58ae2> requested.  
    EC2 instance <i-1be58ae2> created.  
    Waited 0/300s for instance <i-1be58ae2> to become ready.      Waited  
5/300s for instance <i-1be58ae2> to become ready.
```

```
...
```

```
Waited 280/300s for instance <i-1be58ae2> to become ready.
```

```
EC2 instance <i-1be58ae2> ready.
```

```
[WinRM] Established
```

```
...
```

# CONCEPT

## Group Exercise: Kitchen Converge



kitchen  
create

kitchen  
converge

kitchen  
verify

> **kitchen converge [INSTANCE|REGEXP|all]**

Create the instance (if necessary) and then apply the run list to one or more instances.

# GL: Converge the Cookbook



```
> kitchen converge
```

```
-----> Starting Kitchen (v1.22.0)
-----> Converging <default-windows-2012r2>...
    Preparing files for transfer
    Preparing dna.json
    Resolving cookbook dependencies with Berkshelf 7.0.4...
    ...
Running handlers:
    Running handlers complete
    Chef Client finished, 2/2 resources updated in 05 seconds
    Downloading files from <default-windows-2012r2>
Finished converging <default-windows-2012r2> (1m15.86s).

-----> Kitchen is finished. (1m26.13s)
```

# Test Kitchen



What does this test when kitchen converges a recipe?

And what does it NOT test when kitchen converges a recipe?

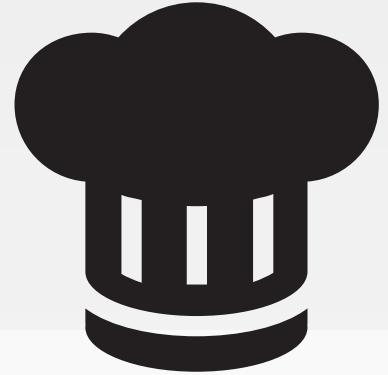
# Test Kitchen



What is left to validate to ensure that the cookbook successfully applied the policy defined in the recipe?

# LAB

## The First Test



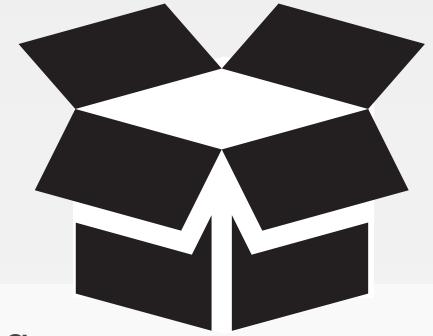
*Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?*

### OBJECTIVE:

- Write a test that asserts that we have disabled UAC when the "workstation" cookbook's default recipe is applied
- Execute the test that we have written

# CONCEPT

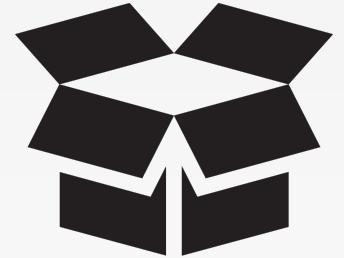
## InSpec



InSpec tests your servers' actual state by executing commands locally or via SSH, via WinRM, via Docker API and so on against a test instance.

[https://docs.chef.io/inspec\\_reference.html](https://docs.chef.io/inspec_reference.html)

# CONCEPT



## RSpec and InSpec

RSpec is a Domain Specific Language (DSL) that allows you to express and execute expectations. These expectations are expressed in examples that are asserted in different example groups.

InSpec provides helpers and tools that allow you to express expectations about the state of infrastructure.

InSpec

RSpec

Chef

Ruby



# Auto-generated Spec File in Cookbook

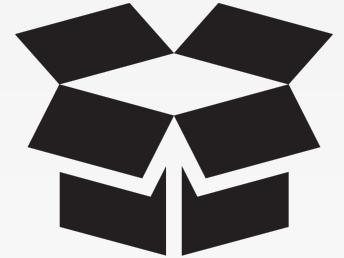
```
cookbooks/workstation/test/integration/default/default_test.rb
```

```
unless os.windows?

  # This is an example test, replace with your own test.
  describe user('root'), :skip do
    it { should exist }
    end
  end

  # This is an example test, replace it with your own test.
  describe port(80), :skip do
    it { should_not be_listening }
  end
```

# CONCEPT



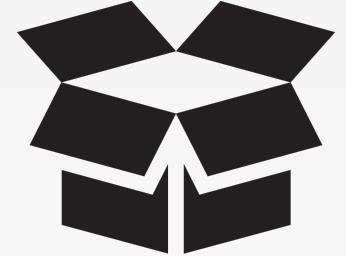
## Where do Tests Live?

```
cookbooks/workstation/test/integration/default/default_test.rb
```

Test Kitchen will look for tests to run under this directory. This corresponds to the value specified in the Test Kitchen configuration file (.kitchen.yml) in the suites section.



# CONCEPT



## Where do Tests Live?

`workstation/test/integration/default/default_test.rb`

The default\_test.rb file is a Ruby file that contains the tests that we want to run when we spin up a test instance.

# Components of a InSpec Example

```
unless os.windows?
```

OS conditional

```
  describe user('root'), :skip do
```

InSpec resource

```
    it { should exist }
```

expectation

```
  end
```

```
end
```

When not on Windows, I expect the user named 'root', to exist.

# Components of a InSpec Example

```
describe port(80) , :skip do  
  it { should_not be_listening }  
end
```

InSpec resource

expectation

When on any platform, I expect the port 80 **not** to be listening for incoming connections.

# Remove the Test for the root User

```
workstation/test/integration/default/
default_test.rb

unless os.windows?

  # This is an example test, replace with your own test.
  describe user('root'), :skip do
    it { should exist }
    end
  end

  # This is an example test, replace it with your own test.
  describe port(80), :skip do
    it { should_not be_listening }
  end
```

## Remove the Test for port 80

workstation/test/integration/default/  
default\_test.rb

```
# This is an example test, replace it with your  
own test.  
  
describe port(80), :skip do  
  it { should_not be_listening }  
end
```

# GL: Is the Registry Key Set Correctly?

`workstation/test/integration/default/default_test.rb`

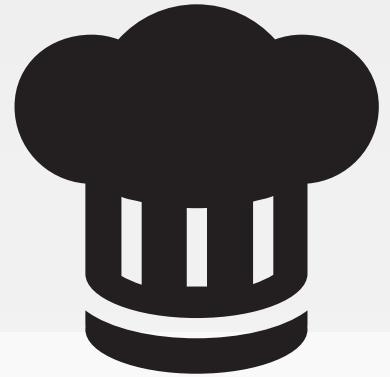
```
system_policies =
'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System'

describe registry_key('System Policies',
system_policies) do
  its('EnableLUA') { should eq 0 }
end
```

[https://docs.chef.io/inspec\\_reference.html#registry-key](https://docs.chef.io/inspec_reference.html#registry-key)

# LAB

## The First Test



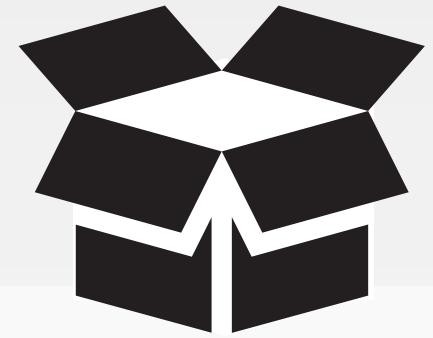
*Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?*

### OBJECTIVE:

- ✓ Write a test that asserts that we have disabled UAC when the "workstation" cookbook's default recipe is applied
- ❑ Execute the test that we have written

# CONCEPT

## Kitchen Verify



kitchen  
create

kitchen  
converge

kitchen  
verify

```
> kitchen verify [INSTANCE|REGEXP|all]
```

Create, converge, and verify one or more instances.

## GL: Ensure you are in the Cookbook



```
> cd ~\chef-repo\cookbooks\workstation
```

# GL: Running the Test



```
> kitchen verify
```

```
----> Starting Kitchen (v1.22.0)
----> Setting up <default-windows-2012r2>...
...
Registry Key System Policies
  [PASS]  EnableLUA should eq 0

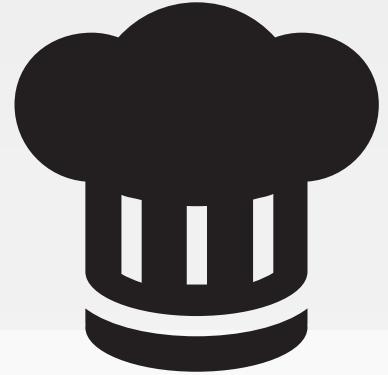
Test Summary: 1 successful, 0 failures, 0 skipped

Finished verifying <default-windows-2012r2> (0m4.54s).

----> Kitchen is finished. (0m14.63s)
```

# LAB

## The First Test



*Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?*

### OBJECTIVE:

- ✓ Write a test that asserts that we have disabled UAC when the "workstation" cookbook's default recipe is applied
- ✓ Execute the test that we have written



## Lab: Additional Test

- ❑ Add tests that validate that the registry key 'ConsentPromptBehaviorAdmin' has been set to the value to 0. HINT: You can have multiple 'its' lines within a describe block.
  
- ❑ Run kitchen verify to validate the test meets the expectations that you defined

# Lab: Add This to the Existing Test File

```
cookbooks\workstation\test\integration\default\default_test.rb
```

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

describe registry_key('System Policies', system_policies) do
  its('EnableLUA') { should eq 0 }
  its('ConsentPromptBehaviorAdmin') { should eq 0 }
end
```

©2019 Chef Software Inc.

# Lab: Running the Test



```
> kitchen verify
```

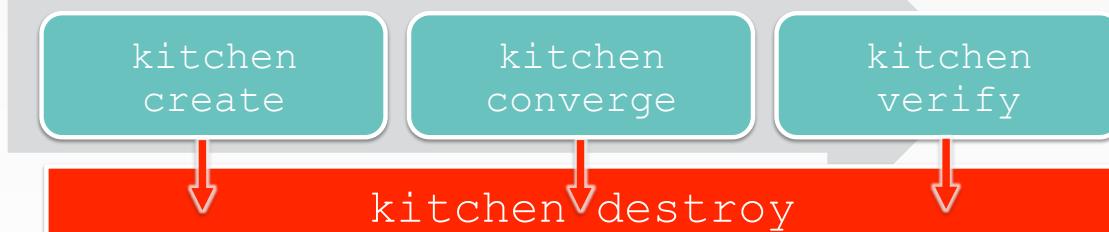
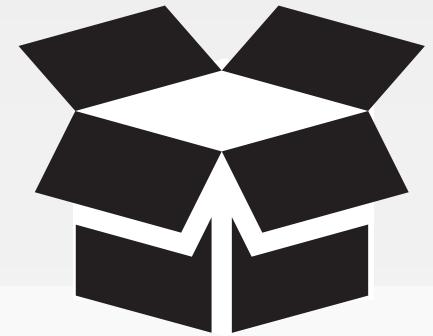
```
-----> Starting Kitchen (v1.22.0)
-----> Verifying <default-windows-2012r2>...
      Loaded tests from {:path=>"C:
\Users\Administrator\cookbooks\workstation\test\integration\default"}
...
Registry Key System Policies
  [PASS]  EnableLUA should eq 0
  [PASS]  ConsentPromptBehaviorAdmin should eq 0

Test Summary: 2 successful, 0 failures, 0 skipped
  Finished verifying <default-windows-2012r2> (0m4.05s).

-----> Kitchen is finished. (0m14.20s)
```

# CONCEPT

## Kitchen Destroy

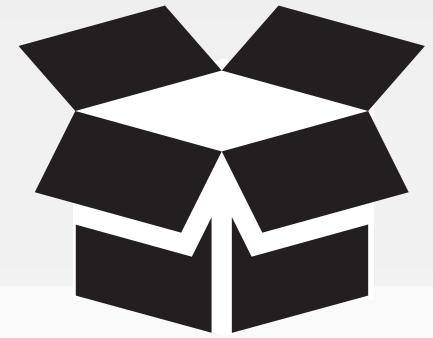


```
> kitchen destroy [INSTANCE|REGEXP|all]
```

**Destroys one or more instances.**

# CONCEPT

## Kitchen Test



kitchen  
destroy

kitchen  
create

kitchen  
converge

kitchen  
verify

kitchen  
destroy

> `kitchen test [INSTANCE|REGEXP|all]`

Destroys (for clean-up), creates, converges, verifies and then destroys one or more instances.



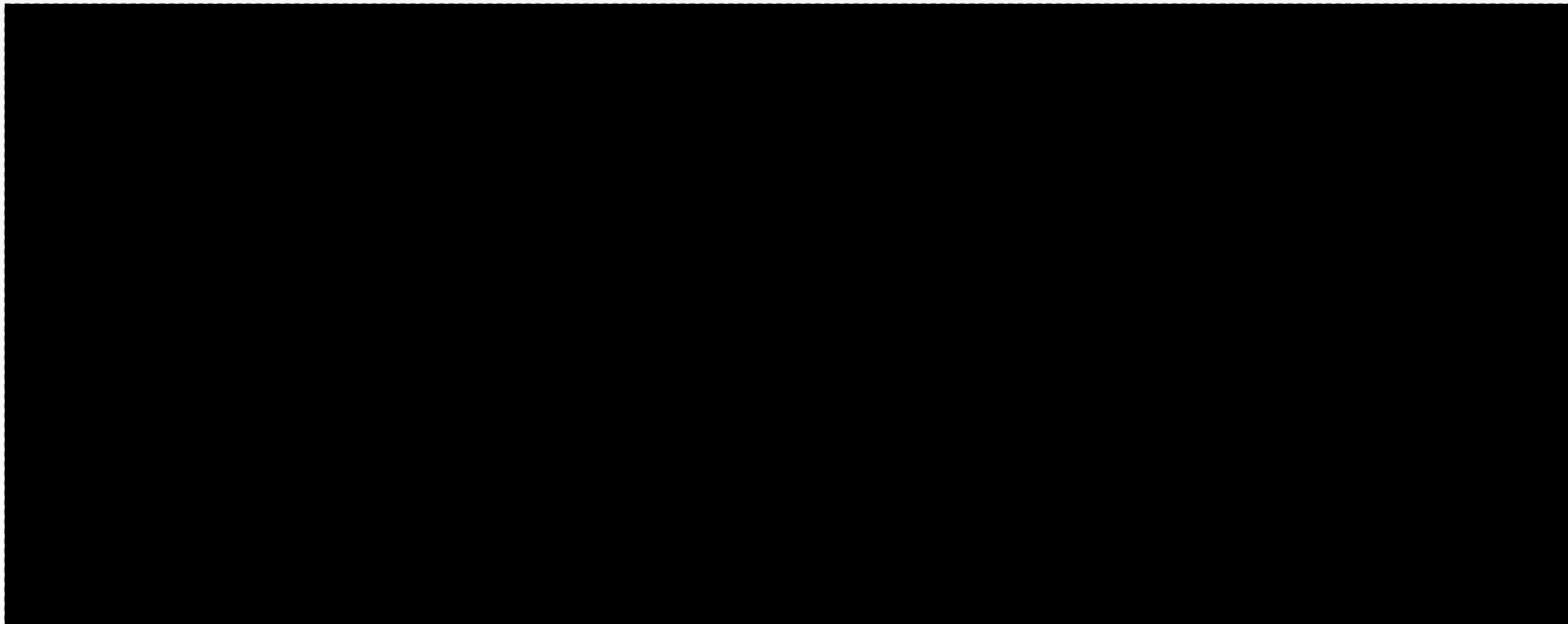
## Lab: kitchen destroy

We are done with Test Kitchen so please execute `kitchen destroy` to terminate the EC2 instance used in this module.

## Lab: Run `kitchen destroy` Clean Everything Up



```
> kitchen destroy
```



# Discussion



Why do you have to run kitchen within the directory of the cookbook?

Where would you define additional platforms?

Why would you define a new test suite?

What are the limitations of using Test Kitchen to validate recipes?

## Q&A

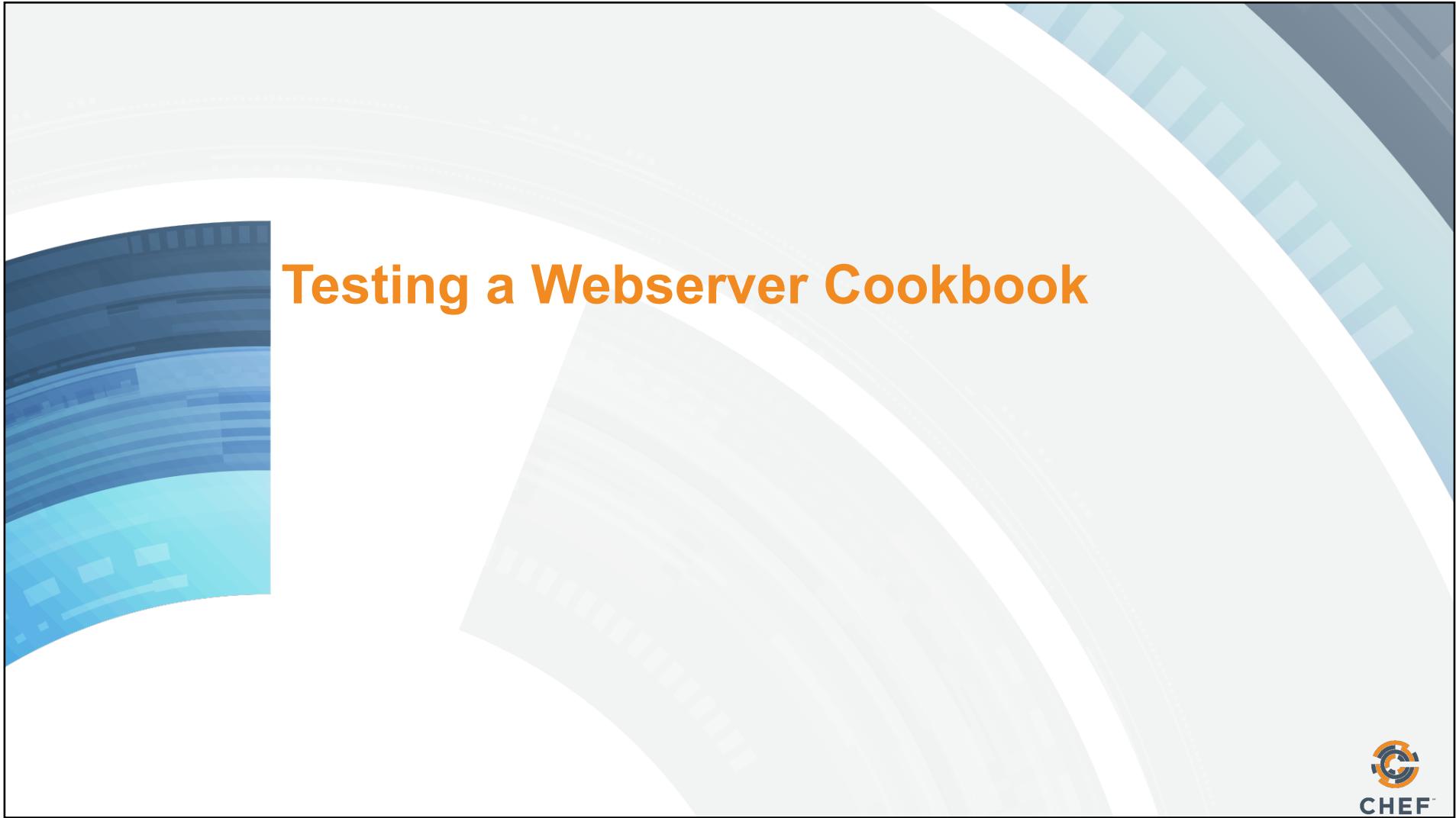


What questions can we help you answer?

- Test Kitchen
- kitchen commands
- kitchen configuration
- InSpec



CHEF™



# Testing a Webserver Cookbook



# Objectives

After completing this module, you should be able to:

- Write an integration test
- Use Test Kitchen to create, converge, and verify a recipe
- Develop a cookbook with a test-driven approach



# Building a Web Server

1. Install the IIS web server
2. Write out a test page
3. Start and enable the w3svc service



## **Scenario: Potential User Visits Website**

**Given that I am a potential user**

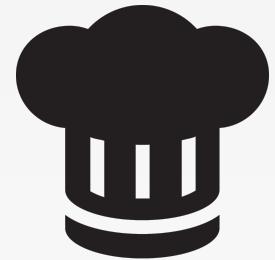
**When I visit the company website in my browser**

**Then I should see a welcome message**



# EXERCISE

## Build a Reliable Cookbook



*This time it will be different.*

### Objective:

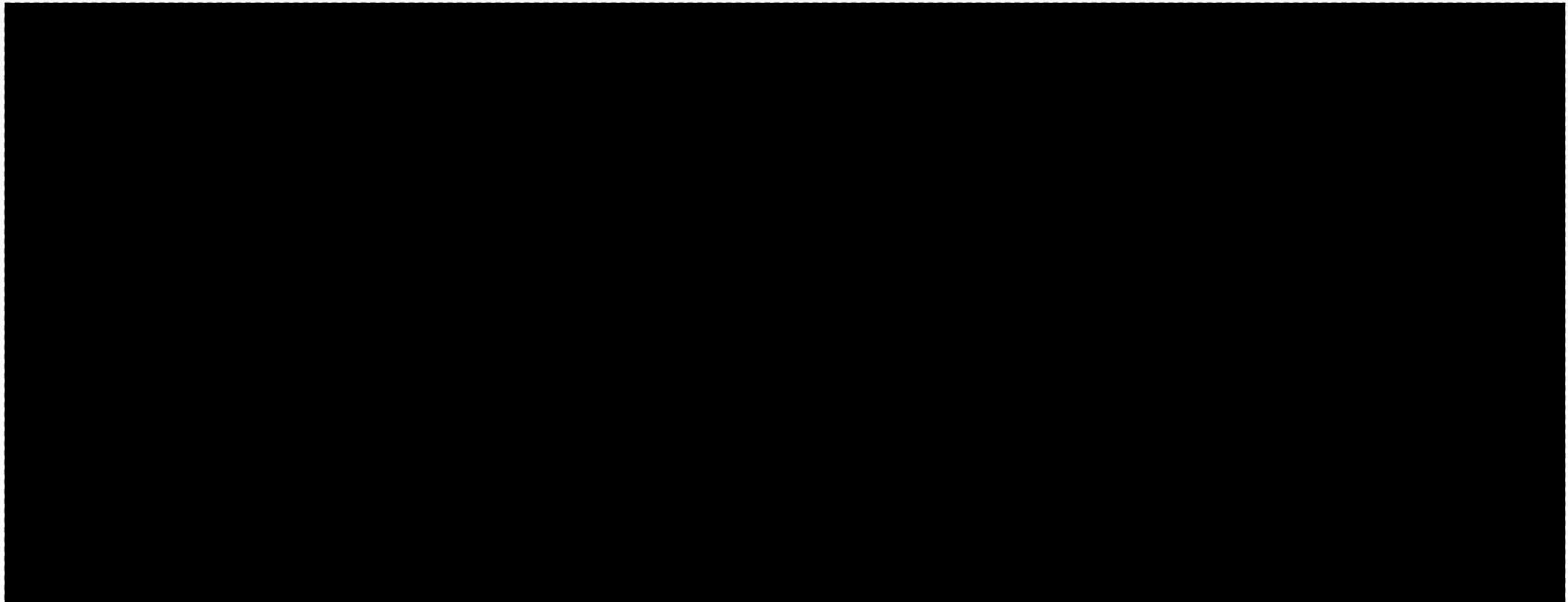
- Examine the cookbook
- Write tests that verifies the cookbook does what we want it to do
- Execute the tests and see failure
- Write the recipe to make the test pass
- Execute the tests and see success



# Start in the Cookbooks Directory



```
> cd ~/chef-repo/cookbooks
```



## View the Tests in the Generated Cookbook



```
> tree myiis
```

```
C:\USERS\ADMINISTRATOR\CHEF-REPO\COOKBOOKS\MYIIS
```

```
  ├── recipes
  └── spec
      └── unit
          └── recipes
  └── test
      └── integration
          └── default
```

# Remove the Test for the root User

```
chef-repo/cookbooks/myiis/test/integration/default/default_test.rb
```

```
unless os.windows?

  # This is an example test, replace with your own test.
  describe user('root'), :skip do
    it { should exist }
    end
  end

  # This is an example test, replace it with your own test.
  describe port(80), :skip do
    it { should_not be_listening }
  end
```

# Update the Test for Port 80

```
[chef-repo/cookbooks/myiis/test/integration/default/default_test.rb]
```

```
# ... FIRST EXAMPLE DELETED ...
```

```
# This is an example test, replace it with your  
own test.
```

```
describe port(80), :skip do  
  it { should be_listening }  
end
```

## Add a Test to Validate a Working Website

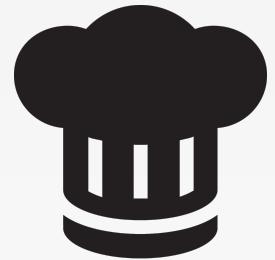
```
chef-repo/cookbooks/myiis/test/integration/
```

```
default/default_test.rb
```

```
describe port(80) do
  it { should be_listening }
end
```

```
+-----+
describe command('curl http://localhost') do
  its(:stdout) { should match(/Hello, world/) }
end
```

# EXERCISE



## Build a Reliable Cookbook

*This time it will be different.*

### Objective:

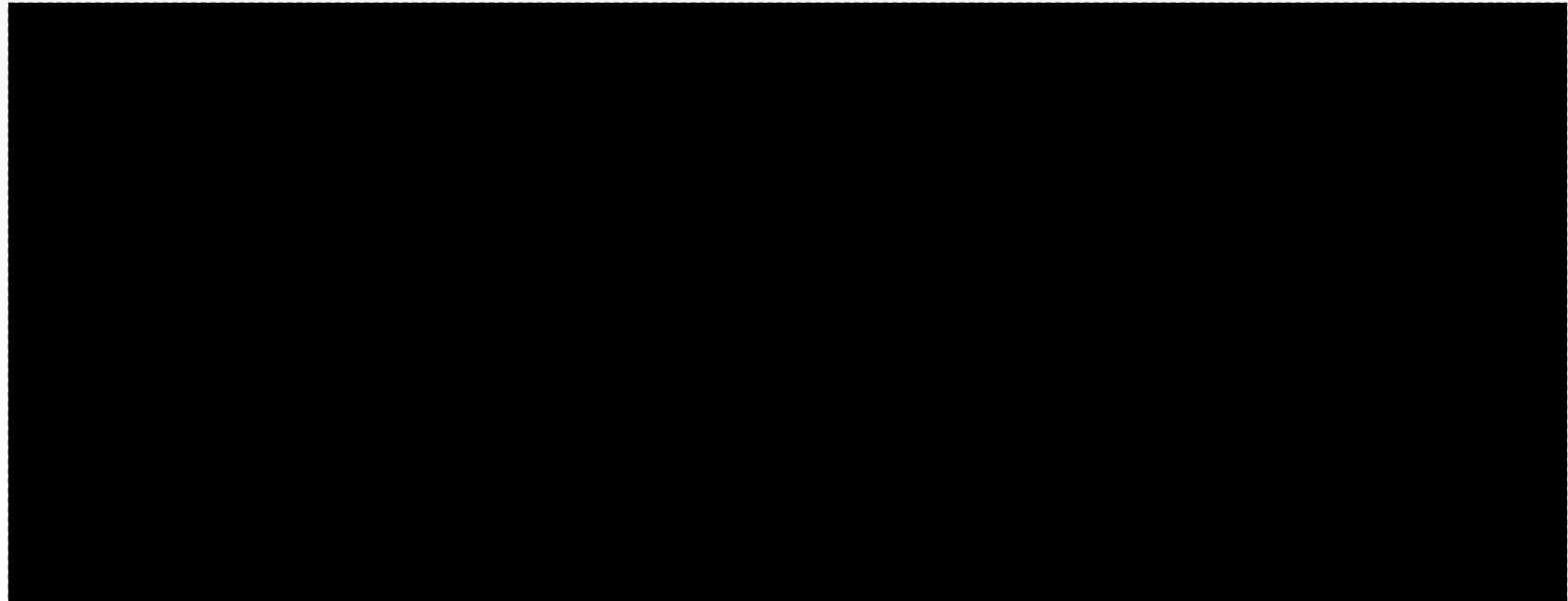
- ✓ Examine the cookbook
- ✓ Write tests that verifies the cookbook does what we want it to do
- Execute the tests and see failure
- Write the recipe to make the test pass
- Execute the tests and see success



# Move into the Cookbook Directory



```
> cd myiis
```



## Copy the existing kitchen.yml file



```
➤ cd ~/chef-repo/cookbooks  
➤ cp workstation/.kitchen.yml myiis
```

# GL: Change the Test Suite

```
1 ~/chef-repo/cookbook/myiis/.kitchen.yml
```

```
suites:
  - name: default
    run_list:
      - recipe[myiis::default]
  verifier:
    inspec_tests:
      - test/integration/default
```

# View the Test Matrix for Test Kitchen



```
> kitchen list
```

Instance	Driver	Provisioner	Verifier	Transport	Last Action
default-windows-2012R2	EC2	ChefZero	InSpec	Winrm	<Not Created>



# Create the Virtual Instance



```
> kitchen create
```

```
-----> Starting Kitchen (v1.19.2)
-----> Creating <default-windows-2012R2>...
-----> Creating <default-windows-2012R2>...
      Detected platform: windows version 2012rtm on x86_64.
Instance Type: t2.2xlarge. Default username: administrator
(default).

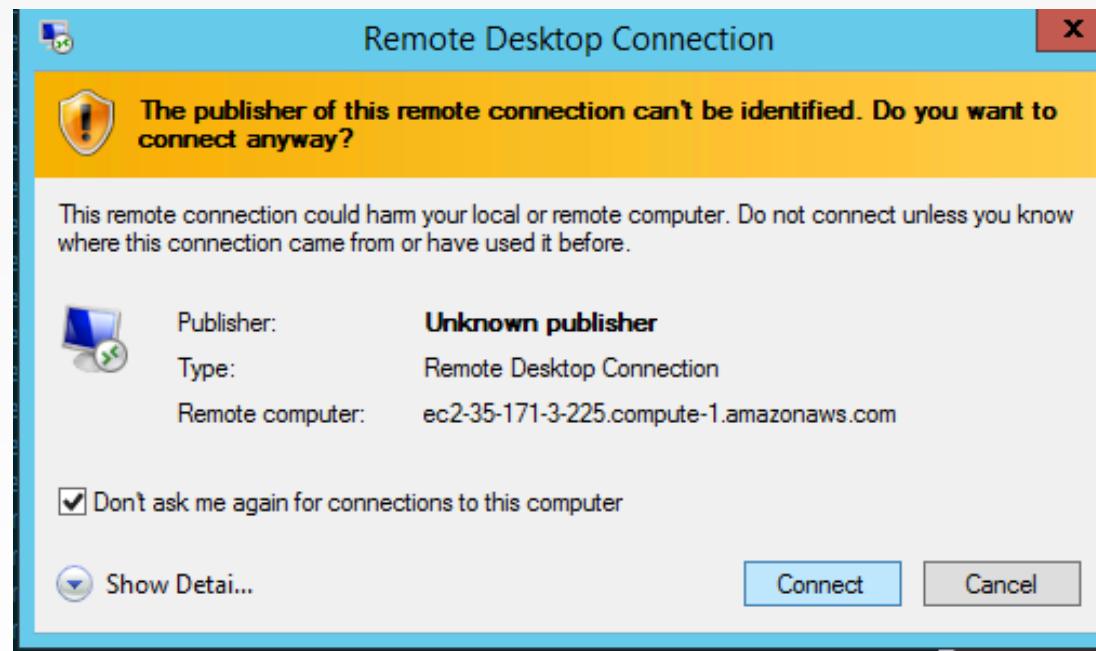
. . . . .

      Finished creating <default-windows-2012R2> (3m10.37s).
-----> Kitchen is finished. (3m21.15s)
PS C:\Users\Administrator\cookbooks\myiis>
```

# Inspect the Virtual Instance



> kitchen login



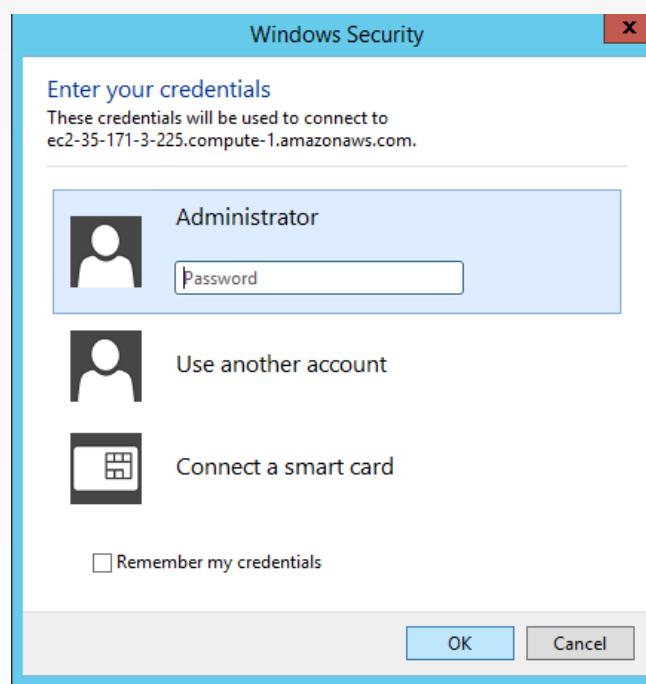
# Inspect the Virtual Instance



> kitchen login

Password for  
our instances:

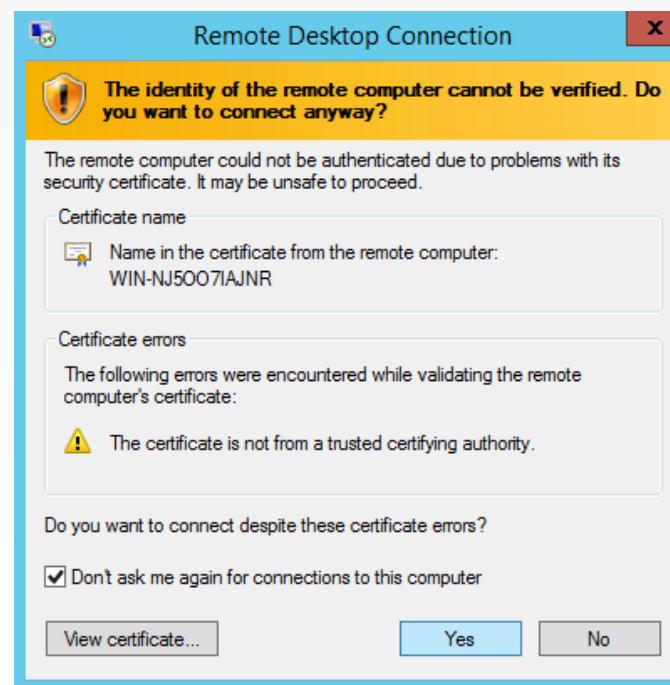
Cod3Can!



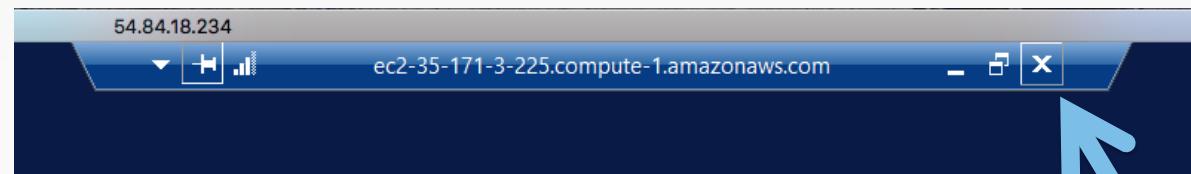
# Inspect the Virtual Instance



> kitchen login



# Exit the Virtual Instance



# Converge the Virtual Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.19.2)
-----> Converging <default-windows2012r2>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
-----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["apache::default"]
...
-----> Finished converging <default-centos-69> (0m27.64s).
-----> Kitchen is finished. (0m28.58s)
```



# Re-Verify the Virtual Instance



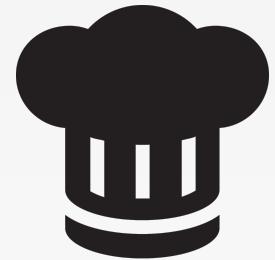
```
> kitchen verify
```

```
----> Starting Kitchen (v1.19.2)
----> Verifying <default-windows-2012r2>...
      Use `/Users/Administrator/myiis/test/smoke/default` for testing
...
Target: winrm://kitchen@localhost:32768
Port 80
  [PASS] should be listening
Command: `Invoke-WebRequest localhost`
  [PASS] stdout should match /Hello, world/
Test Summary: 2 successful, 0 failures, 0 skipped
```



# EXERCISE

## Build a Reliable Cookbook



*This time it will be different.*

### Objective:

- ✓ Examine the cookbook
- ✓ Write tests that verifies the cookbook does what we want it to do
- ✓ Execute the tests and see failure
- ✓ Write the recipe to make the test pass
- ✓ Execute the tests and see success



## Q&A



What questions can we answer for you?

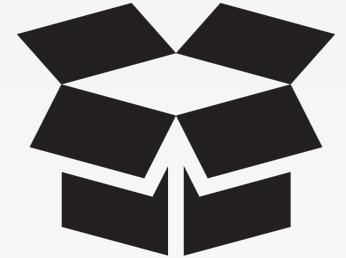


CHEF™

# Refactoring Cookbooks with Tests



# CONCEPT



## Test Driven Development

1. Define a test set for the unit first
2. Then implement the unit
3. Finally verify that the implementation of the unit makes the tests succeed.
4. **Refactor**

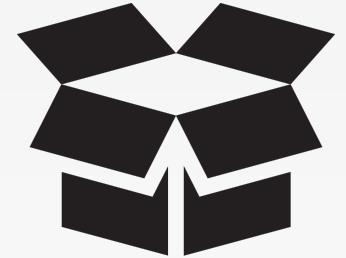
# Objectives

After completing this module, you should be able to:

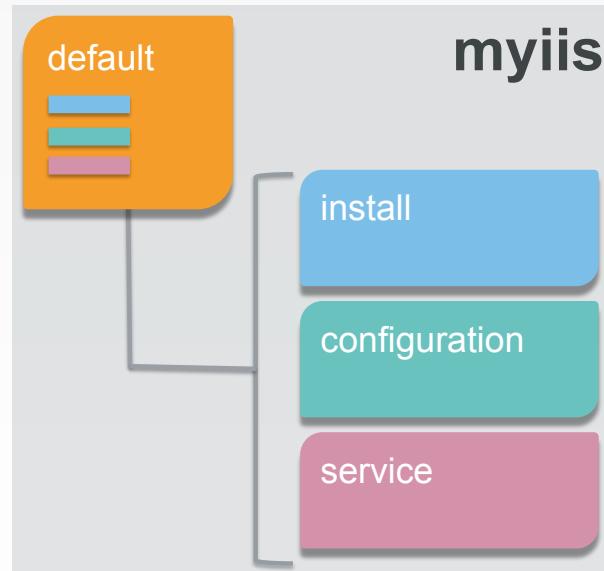
- Refactor a recipe using `include_recipe`
- Use Test Kitchen to validate the code you refactored



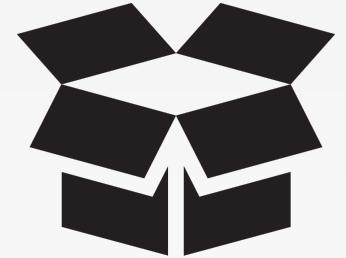
# CONCEPT



## Modular Cookbook Recipes



# CONCEPT



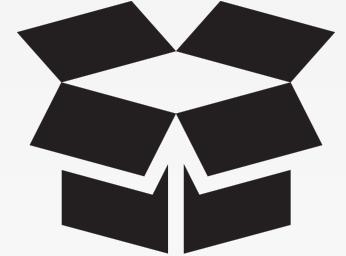
## **include\_recipe**

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.

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



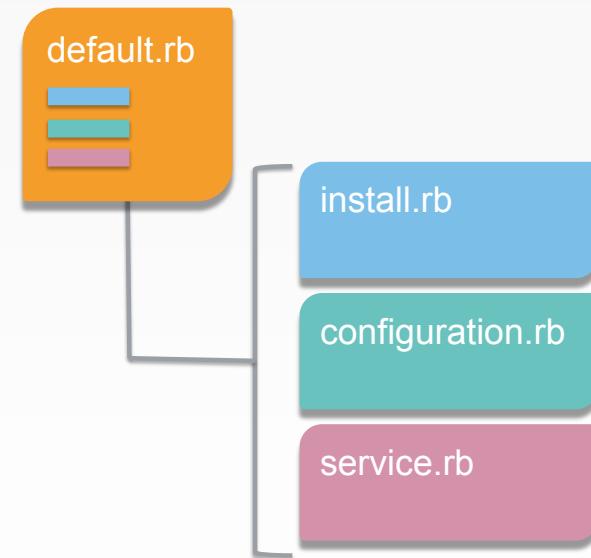
# CONCEPT



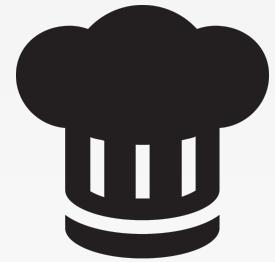
## Recipe Organization

`recipes/default.rb`

```
include_recipe 'cookbook::install'  
include_recipe 'cookbook::configuration'  
include_recipe 'cookbook::service'
```



# EXERCISE

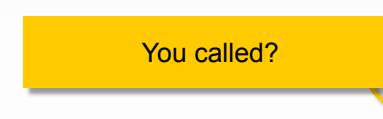


## Refactor to Modular Recipes

*This is why we **can** have nice things!*

### Objective:

- Refactor the installation into a separate recipe
- Converge the cookbook and execute the tests



# Ask Chef About Generating a Recipe



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

Usage: chef generate recipe [path/to/cookbook] NAME [options]

-C, --copyright COPYRIGHT	Name of the copyright hol...
-m, --email EMAIL	Email address of the auth...
-a, --generator-arg KEY=VALUE	Use to set arbitrary ...
-I, --license LICENSE	all_rights, apache2, mit,...
-g GENERATOR_COOKBOOK_PATH, --generator-cookbook	Use GENERATOR_COOKBOOK_PA...

# Generate an Install Recipe



```
> chef generate recipe install
```

```
Recipe: code_generator::recipe
  * directory[/Users/Administrator/chef-repo/cookbooks/myiis/spec/unit/recipes] action create (up to date)
  * cookbook_file[/Users/Administrator/chef-repo/cookbooks/myiis/spec/spec_helper.rb] action create_if_missing (up to date)
  * template[/Users/Administrator/chef-repo/cookbooks/myiis/spec/unit/recipes/install_spec.rb] action create_if_missing
    - create new file /Users/Administrator/chef-repo/cookbooks/myiis/spec/unit/recipes/install_spec.rb
    - update content in file /Users/Administrator/chef-repo/cookbooks/myiis/spec/unit/recipes/install_spec.rb from none to 187413
```



# Removing the Generated Test File



```
> rm myiis/test/integration/default/install_test.rb
```



# Write the Install Recipe

myiis/recipes/install.rb

```
#  
# Cookbook::myiis  
# Recipe:: install  
#  
# Copyright:: 2018, The Authors, All Rights Reserved.  
powershell_script 'Install IIS' do  
  code 'Add-WindowsFeature Web-Server'  
  action :run  
End
```

# Remove the Resource from the Default Recipe

myiis/recipes/default.rb

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

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

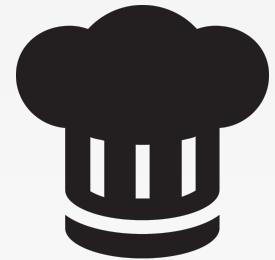
service 'w3svc' do
  action [ :enable, :start ]
end
```

# Include the Install Recipe

## myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
  
#  
# Copyright:: 2018, The Authors, All Rights Reserved.  
include_recipe 'myiis::install'  
  
file 'C:\inetpub\wwwroot\Default.htm' do  
  content 'Hello, world!'  
  rights :read, 'Everyone'  
end  
  
service 'w3svc' do  
  action [:enable, :start]  
end
```

# EXERCISE



## Refactor to Modular Recipes

*This is why we **can** have nice things!*

### Objective:

- ✓ Refactor the installation into a separate recipe
- ❑ Converge the cookbook and execute the tests

I see what you did there.



# Re-Converge the Test Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.19.2)
----> Converging <default-windows-2012r2>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["myiis::default"]
...
      Finished converging <default-windows2012r2> (0m27.64s).
----> Kitchen is finished. (0m28.58s)
```

# Re-Verify the Virtual Instance



```
> kitchen verify
```

```
----> Starting Kitchen (v1.19.2)
----> Verifying <default-windows-2012r2>...
      Use `/Users/Administrator/myiis/test/smoke/default` for testing
...
Target: winrm://kitchen@localhost:32768
Port 80
  [PASS] should be listening
Command: `Invoke-WebRequest localhost`
  [PASS] stdout should match /Hello, world/
Test Summary: 2 successful, 0 failures, 0 skipped
```



# LAB



## The Configuration

- Create a configuration recipe that defines the policy:

**The file named C:**

**\inetpub\wwwroot\Default.htm' contains  
the content '<h1>Welcome Home!</h1>'**

- Delete the automatically generated InSpec test
- Within the default recipe replace the file resource with an include recipe
- Converge and verify the test instance to ensure there are no failures

# Generate a Service Recipe



```
> chef generate recipe configuration
```

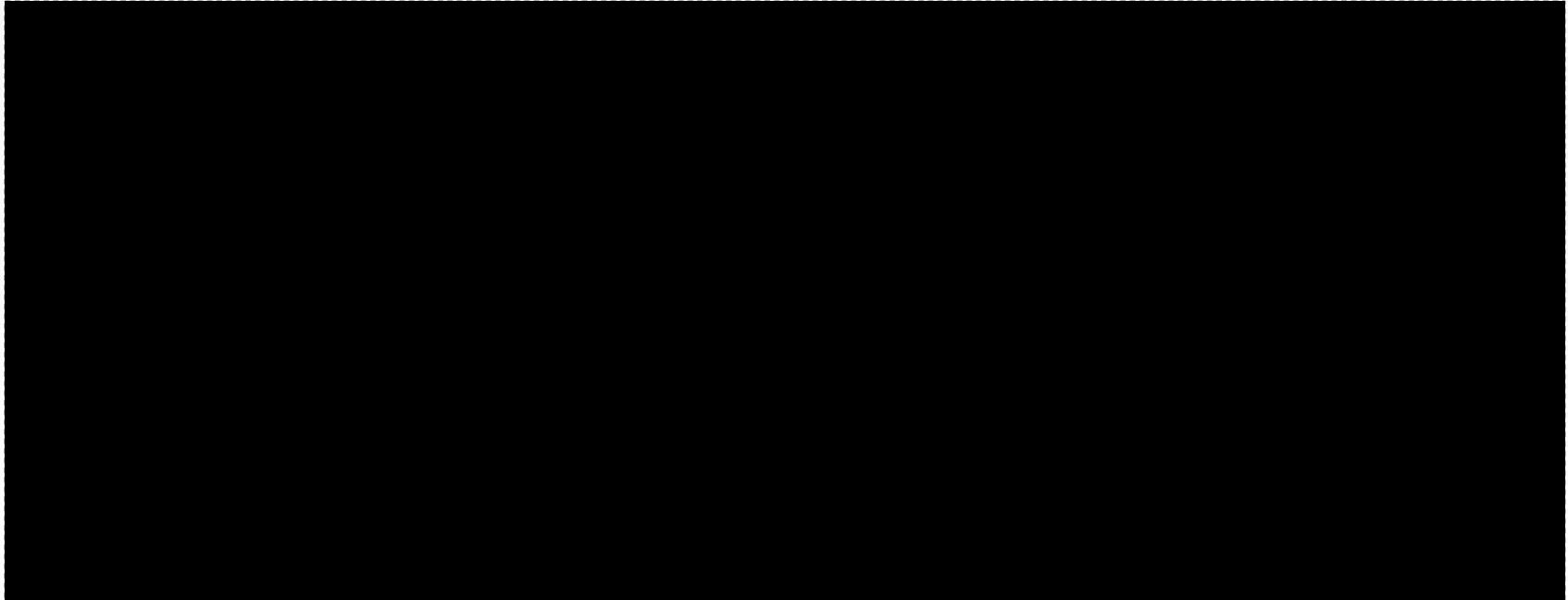
```
Recipe: code_generator::recipe
  * directory[/Users/Administrator/chef-repo/cookbooks/myiis/test/
spec/unit/recipes] action create (up to date)
  * cookbook_file[/Users/Administrator/chef-repo/cookbooks/myiis/
test/spec/spec_helper.rb] action create_if_missing (up to date)
  * template[/Users/Administrator/chef-repo/cookbooks/myiis/test/
spec/unit/recipes/configuration_spec.rb] action create_if_missing
    - create new file /Users/Administrator/chef-repo/cookbooks/
myiis/test/spec/unit/recipes/configuration_spec.rb
    - update content in file /Users/Administrator/chef-repo/
cookbooks/myiis/test/spec/unit/recipes/configuration_spec.rb from
none
```



# Remove the Generated Test File



```
> rm test/integration/default/configuration_test.rb
```



# Write the Configuration Recipe

```
myiis/recipes/configuration.rb
```

```
#  
# Cookbook:: myiis  
# Recipe:: configuration  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
file 'C:\inetpub\wwwroot\Default.htm' do  
  content 'Hello, world!'  
  rights :read, 'Everyone'  
end
```

# Remove the Default Recipe Resource

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights Reserved.  
include_recipe 'myiis::install'  
  
file 'C:\inetpub\wwwroot\Default.htm' do  
  content 'Hello, world!'  
  rights :read, 'Everyone'  
end  
  
service 'w3svc' do  
  action [:enable, :start]  
end
```

# Replace the Resource

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
include_recipe 'myiis::install'  
  
include_recipe 'myiis::configuration'  
  
service 'w3svc' do  
  action [ :enable, :start ]
```



# Re-Converge the Test Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.19.2)
----> Converging <default-windows-2012r2>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["myiis::default"]
...
      Finished converging <default-windows-2012r2> (0m27.64s) .
----> Kitchen is finished. (0m28.58s)
```

# Re-Verify the Virtual Instance



```
> kitchen verify
```

```
----> Starting Kitchen (v1.19.2)
----> Verifying <default-windows-2012r2>...
      Use `/Users/Administrator/myiis/test/smoke/default` for testing
...
Target: winrm://kitchen@localhost:32768
Port 80
  [PASS] should be listening
Command: `Invoke-WebRequest localhost`
  [PASS] stdout should match /Hello, world/
Test Summary: 2 successful, 0 failures, 0 skipped
```



# LAB



## The Service

- Create a service recipe that defines the policy:

**The service named 'w3svc' is started and enabled**

- Delete the automatically generated InSpec test
- Within the default recipe replace the service resource with an include recipe
- Converge and verify the test instance to ensure the service is running

One last time!



# Generate a Service Recipe



```
> chef generate recipe service
```

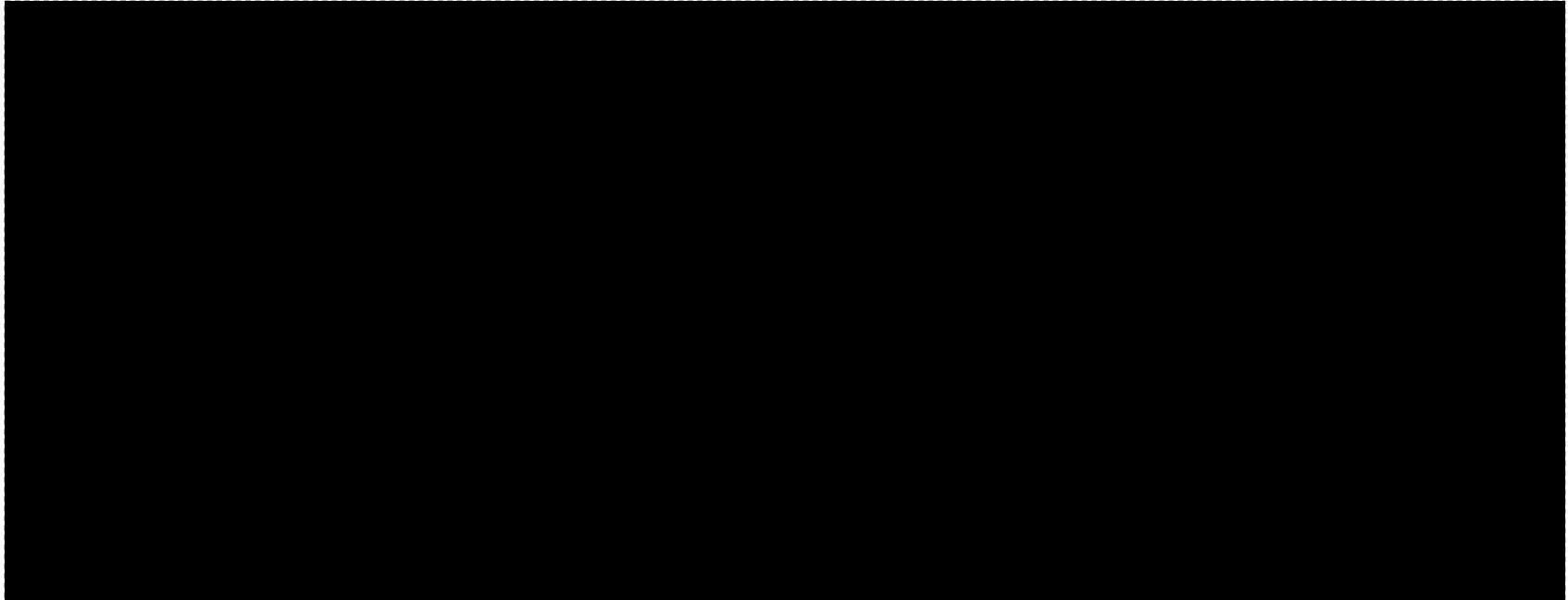
```
Recipe: code_generator::recipe
  * directory[/Users/Administrator/chef-repo/cookbooks/myiis/test/
spec/unit/recipes] action create (up to date)
  * cookbook_file[/Users/Administrator/chef-repo/cookbooks/myiis/
test/spec/spec_helper.rb] action create_if_missing (up to date)
  * template[/Users/Administrator/chef-repo/cookbooks/myiis/test/
spec/unit/recipes/service_spec.rb] action create_if_missing
    - create new file /Users/Administrator/chef-repo/cookbooks/
myiis/test/spec/unit/recipes/service_spec.rb
    - update content in file /Users/Administrator/chef-repo/
cookbooks/myiis/test/spec/unit/recipes/service_spec.rb from none
```



# Remove the Generated Test File



```
> rm test/integration/default/service_test.rb
```



# Write the Service Recipe

```
[myiis/recipes/service.rb]
```

```
#  
# Cookbook:: myiis  
# Recipe:: service  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
service 'w3svc' do  
  action [ :enable, :start ]  
end
```

# Remove the Default Recipe Resource

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
include_recipe 'myiis::install'  
include_recipe 'myiis::configuration'
```

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



# Replace the Resource

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
include_recipe 'myiis::install'  
include_recipe 'myiis::install'  
include_recipe 'myiis::install'
```

# Re-Converge the Test Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.19.2)
----> Converging <default-windows-2012r2>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["myiis::default"]
...
      Finished converging <default-windows-2012r2> (0m27.64s) .
----> Kitchen is finished. (0m28.58s)
```

# Re-Verify the Virtual Instance



```
> kitchen verify
```

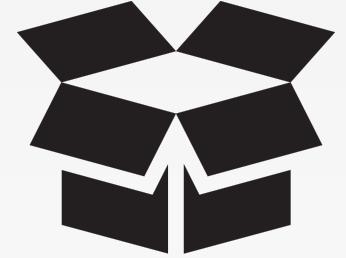
```
----> Starting Kitchen (v1.19.2)
----> Verifying <default-windows-2012r2>...
      Use `/Users/Administrator/myiis/test/smoke/default` for testing
...
Target: winrm://kitchen@localhost:32768
Port 80
  [PASS] should be listening
Command: `Invoke-WebRequest localhost`
  [PASS] stdout should match /Hello, world/
Test Summary: 2 successful, 0 failures, 0 skipped
```



# Do Our Tests Really Work?

What if we removed code from within the recipes  
and ran the tests?

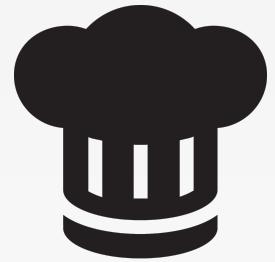
# CONCEPT



## Heckling Your Code

Mutation testing is used to design new software tests and evaluate the quality of existing software tests. Mutation testing involves modifying a program in small ways.

# EXERCISE



## Heckle That Code

*It could be a game show. Maybe on Twitch?*

### Objective:

- Remove / Comment source code
- Converge the cookbook and execute the tests

# Heckle that Code

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
#include_recipe 'myiis::install'  
include_recipe 'myiis::install'  
include_recipe 'myiis::install'
```

# Re-Converge the Test Instance



```
> kitchen converge
```

```
-----> Converging <default-windows-2012r2>...
      Synchronizing Cookbooks:
        - myiis (0.1.0)
      Compiling Cookbooks...
      Converging 3 resources
      Recipe: myiis::configuration
        (up to date)
      Recipe: myiis::service
        (up to date)
      * service[myiis] action enable (up to date)
```



# Re-Verify the Virtual Instance

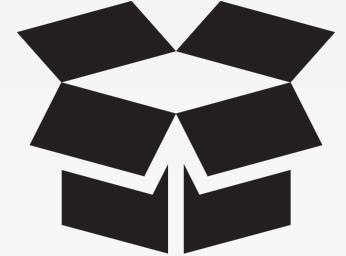


```
> kitchen verify
```

```
----> Starting Kitchen (v1.19.2)
----> Verifying <default-windows-2012r2>...
      Use `/Users/Administrator/myiis/test/smoke/default` for testing
...
Target: winrm://kitchen@localhost:32768
Port 80
  [PASS] should be listening
Command: `Invoke-WebRequest localhost`
  [PASS] stdout should match /Hello, world/
Test Summary: 2 successful, 0 failures, 0 skipped
```



# CONCEPT

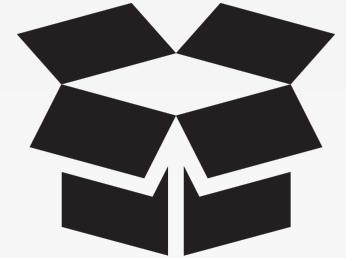


## Kitchen Converge & Verify

Running converge or verify will create a new instance the first time it is run.  
The same instance is used for each additional converge or verify.

The test instance policy changed, but no resource explicitly removed or  
uninstalled the resources defined in the install recipe.

# CONCEPT



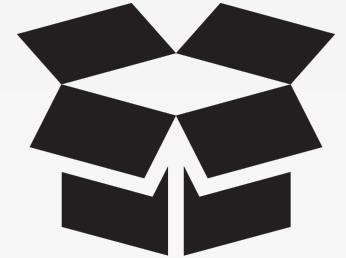
## Kitchen Destroy

```
$ kitchen destroy [INSTANCE | REGEXP | all]
```

Destroys one or more instances.



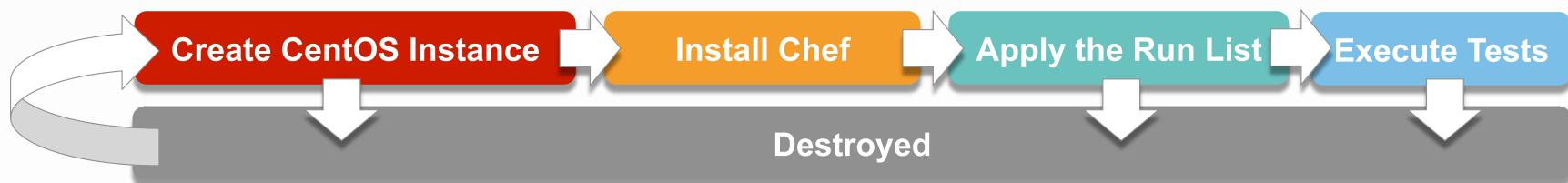
# CONCEPT



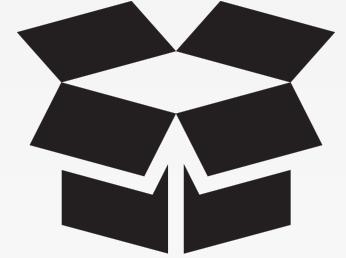
## Kitchen Test

```
$ kitchen test [INSTANCE|REGEXP|all]
```

Destroys (for clean-up), creates, converges, verifies  
and then destroys one or more instances.



# CONCEPT



## Kitchen Test

Destroying the instance ensures that the policy is being applied to a new instance.

The test instance is re-created and then the updated policy is applied to the new instance. The new policy is incomplete causing an error.

# Test the Cookbook Against a New Instance



```
> kitchen test
```

```
----> Starting Kitchen (v1.19.2)
----> Cleaning up any prior instances of <default-windows-2012r2>
----> Destroying <default-windows-2012r2>...
...
[TIMESTAMP] FATAL: Chef::Exceptions::ChildConvergeError:
Chef run process exited unsuccessfully (exit code 1)
>>>>> -----Exception-----
>>>>> Class: Kitchen::ActionFailed
>>>>> Message: 1 actions failed.
>>>>>     Converge failed on instance <default-windows-2012r2>.
```



## **Converge & Verify**

---

**Faster execution time**

Running converge twice will ensure your policy applies without error to **existing instances**

## **Test**

---

**Slower execution time**

Running test will ensure your policy applies without error to any **new instances**



## Discussion

What is happening when running kitchen test?

What types of bugs would kitchen converge & kitchen verify find when running?

What is the difference between kitchen test and running both kitchen converge & kitchen verify together?

How long do each of these approaches take?



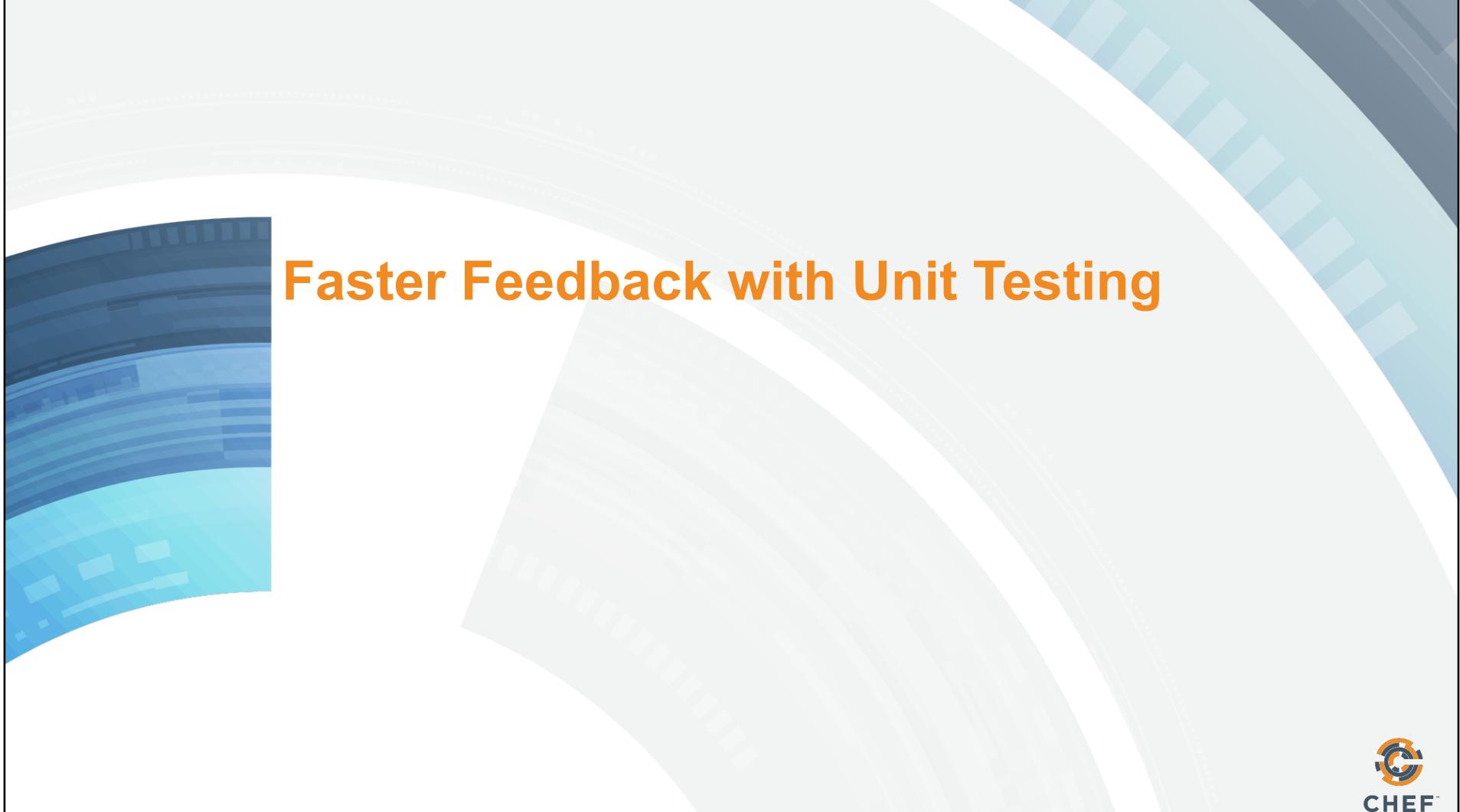
## Q&A



What questions can we answer for you?



CHEF™



## Faster Feedback with Unit Testing



# PROBLEM

## Slower Feedback Cycle



The slower the feedback loop the less value it provides to you while developing your cookbooks. You are less inclined to run the test suite. Which means you will likely miss issues as they happen.

# Objectives

After completing this module, you should be able to:

- Explain the importance and limitations of unit testing
- Write and execute a unit test

# CONCEPT

## External Dependencies

The speed of the test suite is affected by the external dependency on the creation of the test instance, installing chef, and applying the run list.

Create CentOS Instance

Install Chef

Apply the Run List

Build Node (ohai)

Synchronize Cookbooks

Build Resource Collection

Converge

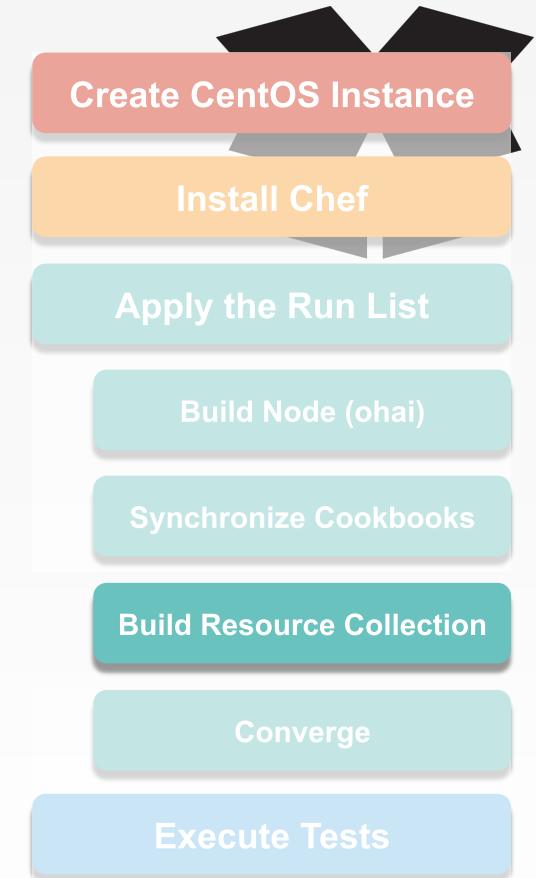
Execute Tests



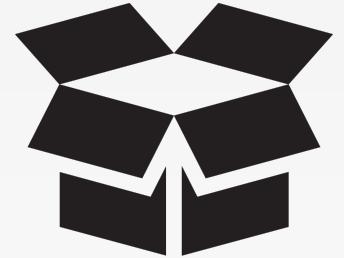
# CONCEPT

## Build Resource Collection

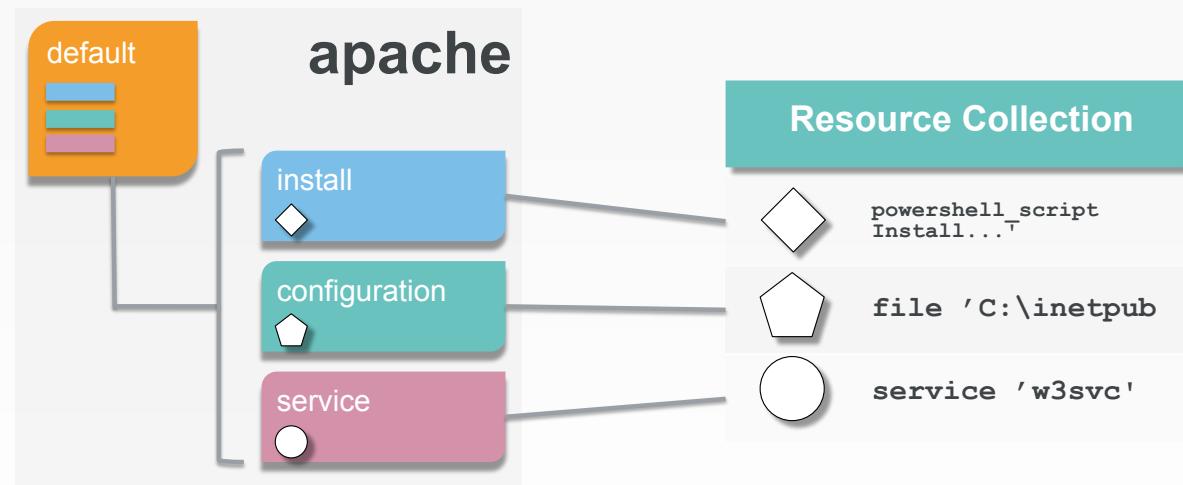
The resource collection is a list of all the resources and recipes loaded across all the recipes within the run list.



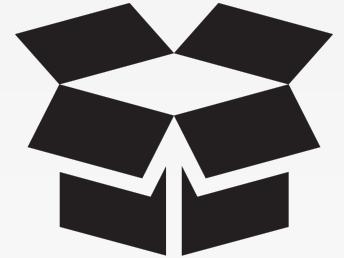
# CONCEPT



## Resource Collection



# CONCEPT



## RSpec and ChefSpec

RSpec is a Domain Specific Language (DSL) that allows you to express and execute expectations.

These expectations are expressed in examples that are asserted in different example groups.

ChefSpec provides helpers and tools that allow you to express expectations about the state of **resource collection**.

ChefSpec

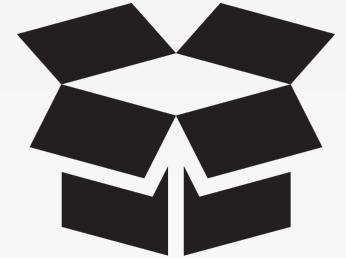
RSpec

Chef

Ruby



# CONCEPT



## Test Kitchen versus ChefSpec

ChefSpec

Build Node (Fauxhai) → Build Resource Collection → Execute Tests

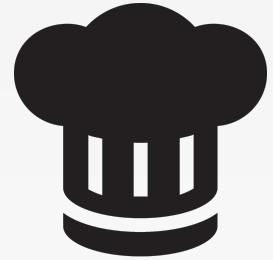
Test Kitchen using InSpec

Create CentOS Instance → Install Chef → Apply the Run List → Execute Tests



# EXERCISE

## Faster Feedback While Developing Cookbooks



*The faster the feedback from our tests, the more likely we are to run them.  
The more likely we are to run them means they will catch more issues.*

### Objective:

- Review and run the existing tests
- Identify the tests that we need to write
- Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

# View the Spec Directory



```
> tree spec
```

```
spec
└── spec_helper.rb
└── unit
    └── recipes
        ├── configuration_spec.rb
        ├── default_spec.rb
        ├── install_spec.rb
        └── service_spec.rb
```

```
2 directories, 5 files
```

# View the Test for the Default Recipe

myiis/spec/unit/recipes/default\_spec.rb

```
require 'spec_helper'

describe 'myiis::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      expect { chef_run }.to_not raise_error
    end
  end
end
```

# View the Test for the Default Recipe

~/apache/spec/unit/recipes/default\_spec.rb

```
require 'spec_helper'

describe 'myis::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      expect(chef_run).to_not raise_error
    end
  end
end
```

example groups

cookbook name::recipe name

# View the Test for the Default Recipe

~/apache/spec/unit/recipes/default\_spec.rb

```
require 'spec_helper'

describe 'myiis::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end
    it 'converges successfully' do
      expect(chef_run).to_not raise_error
    end
  end
end
```

The diagram illustrates the structure of a ChefSpec test. It shows a blue horizontal line connecting the 'runner.converge' call to a blue rectangular box labeled 'expectation'. Simultaneously, an orange horizontal line connects the same 'runner.converge' call to an orange rectangular box labeled 'example'.

# View the Test for the Default Recipe

~/.spec/unit/recipes/default\_spec.rb

```
require 'spec_helper'

describe 'myis::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new('ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end
  end
end

# This file contains tests for the myis::default recipe in
# spec/unit/recipes/default_spec.rb. To learn more about ChefSpec
# and how to write tests, see the documentation at
# https://github.com/10gen/chef-spec.

# This test file is part of the default test suite for the myis
# cookbook. It is run whenever you run `rake spec` or `bundle exec
# rspec spec` from the root of the repository.
```

described recipe

Ruby Class

chef\_run helper

# Execute the Test for the Default Recipe



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

```
.
```

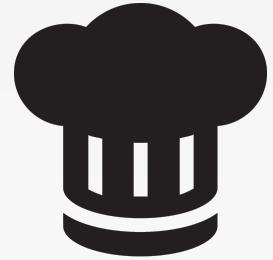
```
Finished in 0.8208 seconds (files took 2.5 seconds to load)
```

```
1 example, 0 failures
```

passing example

# EXERCISE

## Faster Feedback While Developing Cookbooks



*The faster the feedback from our tests, the more likely we are to run them.  
The more likely we are to run them means they will catch more issues.*

### Objective:

- ✓ Review and run the existing tests
- Identify the tests that we need to write
- Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

# These are the Three Things to Test

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
#include_recipe 'myiis::install'  
include_recipe 'myiis::install'  
include_recipe 'myiis::install'
```

# Update the ChefSpec Platform

myiis/spec/unit/recipes/default\_spec.rb

```
require 'spec_helper'

describe 'myiis::default' do
  context 'When all attributes are default, on Windows 2012R2' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'windows', version: '2012r2')
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      expect(chef_run).to_not raise_error
    end
  end
end
```

# Create a Pending Test

```
myis/spec/unit/recipes/default_spec.rb
```

```
# ... START OF THE SPEC FILE ...
it 'converges successfully' do
  expect { chef_run }.to_not raise_error
end

it 'includes the install recipe'

end
end
```

# Execute the Tests to See the Pending Tests



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

```
.*
```

pending example

```
Pending: (Failures listed here are expected and do not affect your suite's status)
```

```
1) myis::default When all attributes are default, on Windows 2012R2 includes the install recipe
```

```
# Not yet implemented
```

```
# ./spec/unit/recipes/default_spec.rb:22
```

```
# ... OUTPUT CONTINUES ON NEXT SLIDE ...
```

# Execute the Tests to See the Pending Tests



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

```
.*
```

```
Pending: (Failures listed here are expected and do not affect your  
suite's status)
```

summary

```
1) myisis::default When all attributes are default, on Windows  
2012R2 includes the install recipe
```

```
# Not yet implemented
```

```
# ./spec/unit/recipes/default_spec.rb:22
```

```
# ... OUTPUT CONTINUES ON NEXT SLIDE ...
```

spec file : line number

# View the Results to See the Pending Tests



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

```
# ... OUTPUT CONTINUED FROM PREVIOUS SLIDE ...
```

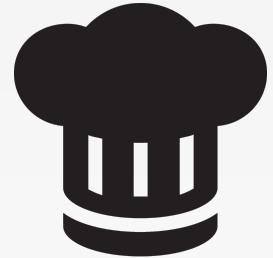
```
Finished in 0.97525 seconds (files took 1.81 seconds to load)
```

```
2 examples, 0 failures, 1 pending
```



# EXERCISE

## Faster Feedback While Developing Cookbooks



*The faster the feedback from our tests, the more likely we are to run them.  
The more likely we are to run them means they will catch more issues.*

### Objective:

- ✓ Review and run the existing tests
- ✓ Identify the tests that we need to write
- Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

# REFERENCE



## ChefSpec Documentation

Find within the documentation examples of testing for `include_recipe`.

- Search the README
- Search through the 'examples' directory

<https://github.com/chefspec/chefspec>



## Write the Test that Verifies the Include Recipe

```
myiis/spec/unit/recipes/default_spec.rb
```

```
# ... START OF THE SPEC FILE ...
it 'converges successfully' do
  expect { chef_run }.to_not raise_error
end
```

```
+-----+
it 'includes the install recipe' do
  expect(chef_run).to
include_recipe('myiis::install')
end
```

```
end
```

# Execute the Tests to See the Failure



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

```
. F
```

failing example

Failures:

```
1) myiis::default When all attributes are default, on CentOS 6.9
includes the install recipe
```

```
Failure/Error: expect(chef_run).to
include_recipe('myiis::install')

expected ["myiis::default", "myiis::configuration",
"myiis::service"] to include "myiis::install"

# ./spec/unit/recipes/default_spec.rb:21:in `block (3 levels)
in <top (required)>'
```



# EXERCISE

## Faster Feedback While Developing Cookbooks



*The faster the feedback from our tests, the more likely we are to run them.  
The more likely we are to run them means they will catch more issues.*

### Objective:

- ✓ Review and run the existing tests
- ✓ Identify the tests that we need to write
- ✓ Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

# Uncomment the Include Recipe

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
include_recipe 'myiis::install'  
include_recipe 'myiis::configuration'  
include_recipe 'myiis::service'
```

# Execute the Tests to See it Pass



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

```
..
```

```
Finished in 0.97525 seconds (files took 1.81 seconds to load)
2 examples, 0 failures
```

# LAB



## Continue with Mutation Testing

- Comment out the next line in the apache cookbook's default recipe
- Write the example with expectation that will generate a failure
- Verify that one example generates a failure
- Restore the code in the recipe
- Verify that all examples pass

❖ Repeat this series of steps for each line within the default recipe

# Comment the Include Recipe

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
  
include_recipe 'myiis::install'  
#include_recipe 'myiis::configuration'  
include_recipe 'myiis::service'
```

## Write the Test that Verifies the Include Recipe

```
myiis/spec/unit/recipes/default_spec.rb
```

```
# .... START OF THE SPEC FILE ....  
  
it 'includes the install recipe' do  
  expect(chef_run).to  
include_recipe('myiis::install')  
end  
  
it 'includes the configuration recipe' do  
  expect(chef_run).to  
include_recipe('myiis::configuration')  
end
```

# Execute the Tests to See it Fail



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

```
..F
```

**Failures:**

1) myiis::default When all attributes are default, on Windows 2012R2 includes the configuration recipe

```
Failure/Error: expect(chef_run).to
include_recipe('myiis::configuration')

expected ["myiis::default", "myiis::configuration",
"myiis::install", "myiis::service"] to include
"myiis::configuration"

# ./spec/unit/recipes/default_spec.rb:27:in `block (3 levels)
in
```



# Uncomment the Include Recipe

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
  
include_recipe 'myiis::install'  
include_recipe 'myiis::configuration'  
include_recipe 'myiis::service'
```

# Execute the Tests to See it Pass



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

```
...
```

```
Finished in 0.97525 seconds (files took 1.81 seconds to load)  
3 examples, 0 failures
```

# Comment the Service Recipe

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
include_recipe 'myiis::install'  
include_recipe 'myiis::configuration'  
#include_recipe 'myiis::service'
```

## Write the Test that Verifies the Include Recipe

myis/spec/unit/recipes/default\_spec.rb

```
# ... START OF THE SPEC FILE ...
it 'includes the install recipe' do
  expect(chef_run).to
include_recipe('myis::install')
end

it 'includes the configuration recipe' do
  expect(chef_run).to
include_recipe('myis::configuration')
end

it 'includes the service recipe' do
  expect(chef_run).to
include_recipe('myis::service')
```



# Execute the Tests to See it Fail



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

```
...F
```

**Failures:**

1) myiis::default When all attributes are default, on Windows 2012R2 includes the service recipe

```
Failure/Error: expect(chef_run).to
include_recipe('myiis::service')

expected ["myiis::default", "myiis::configuration",
"myiis::install", "myiis::service"] to include "myiis::service"
# ./spec/unit/recipes/default_spec.rb:27:in `block (3 levels)
in
```



# Uncomment the Include Recipe

myiis/recipes/default.rb

```
#  
# Cookbook:: myiis  
# Recipe:: default  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
include_recipe 'myiis::install'  
include_recipe 'myiis::configuration'  
include_recipe 'myiis::service'
```

# Execute the Tests to See it Pass



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

```
...
```

```
Finished in 0.97525 seconds (files took 1.81 seconds to load)  
4 examples, 0 failures
```

## Discussion



What functionality did you test in the integration tests?

What functionality did you test in these unit tests?

What do you see as the scope of unit testing versus integration testing?

What are the differences between a ChefSpec test and a InSpec test?

## Q&A



What questions can we answer for you?



CHEF™



# Testing Resources in Recipes



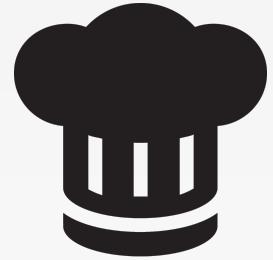
# Objectives

After completing this module, you should be able to:

- Test resources within a recipe using ChefSpec



# EXERCISE



## Testing Remaining Resources

*No resources left behind!*

### Objective:

- Write and execute tests for the Install recipe
- Verify the test validates the recipe

# Generated Recipes Also Generate Specs



```
> tree /f spec
```

```
spec
└── spec_helper.rb
└── unit
    └── recipes
        ├── configuration_spec.rb
        ├── default_spec.rb
        ├── install_spec.rb
        └── service_spec.rb
```

# Update the ChefSpec Platform

myiis/spec/unit/recipes/install\_spec.rb

```
require 'spec_helper'

describe 'myiis::configuration' do
  context 'When all attributes are default, on Windows 2012R2' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'windows', version: '2012r2')
      runner.converge(described_recipe)
    end

    it 'converges successfully' do
      expect { chef_run }.to_not raise_error
    end
  end
end
```

# Execute the Configuration Specification



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

```
.
```

```
Finished in 0.97525 seconds (files took 1.81 seconds to load)
```

```
1 examples, 0 failures
```



## Add a Pending Test to Verify the Package

myis/spec/unit/recipes/configuration\_spec.rb

```
# .... START OF THE SPEC FILE ....  
  
it 'converges successfully' do  
  expect { chef_run }.to_not raise_error  
end  
  
+  
it 'configures the necessary file'  
end  
end
```

# REFERENCE



## ChefSpec Documentation

Find within the documentation examples of testing packages

<https://github.com/chefspec/chefspec/tree/master/examples/package>



# Write the Test to Verify the Package

```
[ ] ~/apache/spec/unit/recipes/configure_spec.rb
```

```
# ... START OF THE SPEC FILE ...

it 'converges successfully' do
  expect { chef_run }.to_not raise_error
end

it 'configures the necessary file' do
  expect(chef_run).to render_file('C:
\inetpub\wwwroot\Default.htm').with_content('Hello, world')
end
end
end
```

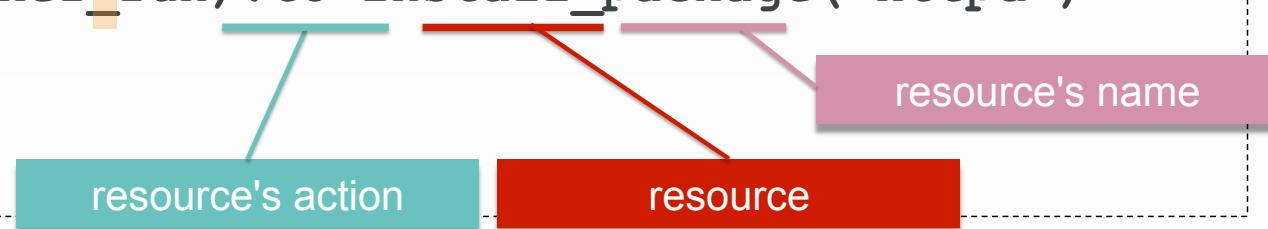
# Write the Test to Verify the Package

```
1 ~/apache/spec/unit/recipes/install_spec.rb
```

```
# ... START OF THE SPEC FILE ...

it 'converges successfully' do
  expect { chef_run }.to_not raise_error
end

it 'installs the necessary package' do
  expect(chef_run).to install_package('httpd')
end
end
end
```



## Execute the Test to See it Pass



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

```
..
```

```
Finished in 0.62738 seconds (files took 1.84 seconds to load)
```

```
2 examples, 0 failures
```



# EXERCISE



## Testing Remaining Resources

*No resources left behind!*

### Objective:

- ✓ Write and execute tests for the Install recipe
- Verify the test validates the recipe

# PROBLEM

**It's Quiet. Too Quiet.**



When a test passes immediately without having to write code (or if the code has already been written) it is time to be concerned. This is one of those moments we should ensure that the tests are working by mutating that code.

# Comment Out the Resource

myiis/recipes/service.rb

```
#  
# Cookbook::myiis  
# Recipe:: service  
#  
# Copyright:: 2018,The Authors, All Rights  
Reserved.  
#service 'w3svc' do  
#  action [:enable, :start]  
#end
```

# Execute the Test to See it Fail



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

```
. F
```

```
Failures:
```

```
  1) myis::service When all attributes are default, on Windows  
2012R2 installs the appropriate package
```

```
    Failure/Error: expect(chef_run).to start_package('w3svc')  
      expected "service[w3svc]" with action :start to be in Chef  
run.
```

# Uncomment Out the Resource

myiis/recipes/service.rb

```
#  
# Cookbook::myiis  
# Recipe:: service  
#  
# Copyright:: 2018,The Authors, All Rights  
Reserved.  
service 'w3svc' do  
  action [:enable, :start]  
end
```

## Execute the Test to See it Pass



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

```
..
```

```
Finished in 0.73662 seconds (files took 4.4 seconds to load)
```

```
2 examples, 0 failures
```



# Write the Tests to Verify the Service

```
[ ] myiis/spec/unit/recipes/service_spec.rb
```

```
# ... START OF THE SPEC FILE ...

it 'starts the necessary service' do
  expect(chef_run).to start_service('w3svc')
end

it 'enables the necessary service' do
  expect(chef_run).to enable_service('w3svc')
end
end
end
```

# Execute the Tests to See it Pass



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

```
...
```

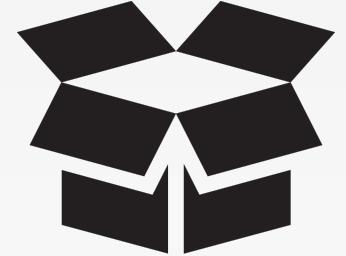
```
Finished in 0.6329 seconds (files took 1.85 seconds to load)
```

```
3 examples, 0 failures
```



# CONCEPT

## rspec



When you run `rspec` without any paths it will automatically find and execute all the `"_spec.rb"` files within the `'spec'` directory.

# Execute All the Tests in the Spec Directory



```
> chef exec rspec
```

```
.....
```

```
Finished in 2.27 seconds (files took 1.82 seconds to load)
```

```
11 examples, 0 failures
```

## Discussion



What value does it bring to validate that the resources take the appropriate action?

## Q&A



What questions can we answer for you?



CHEF™

# Testing While Refactoring to Attributes



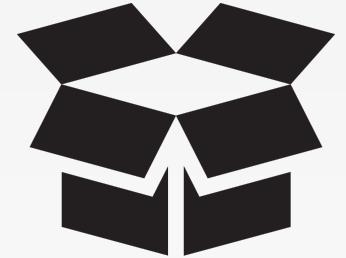
# Objectives

After completing this module, you should be able to:

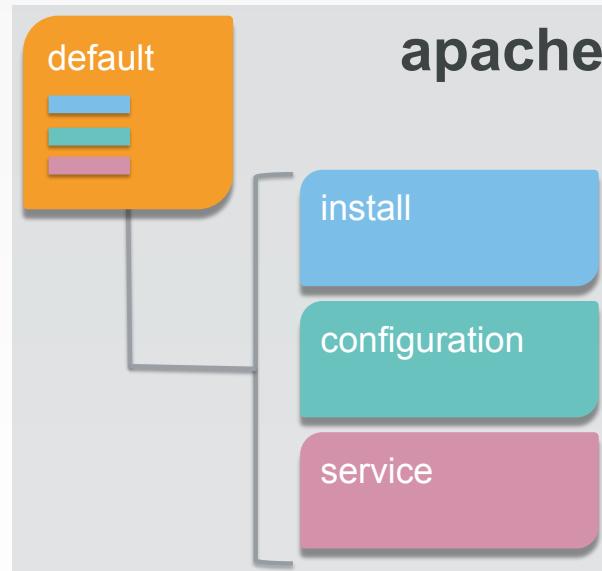
- Refactor resources to use attributes
- Use Pry to explore the current state of execution
- Make changes to your recipes with confidence



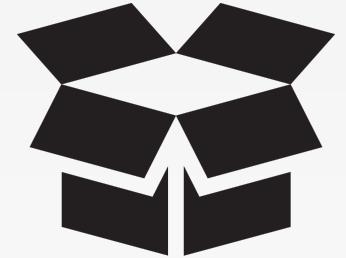
# CONCEPT



## Modular Cookbook Recipes



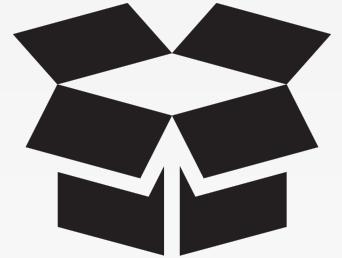
# CONCEPT



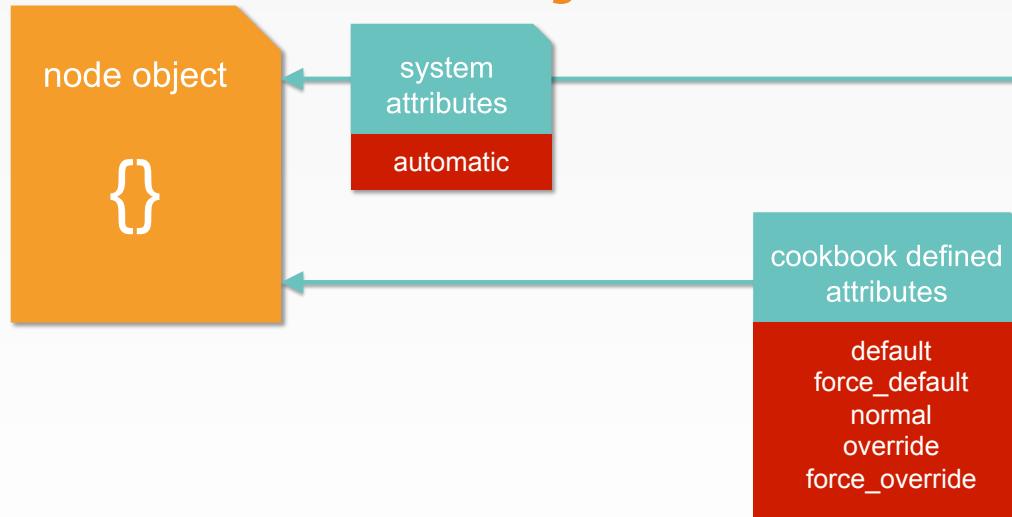
## Modular Cookbook Recipes



# CONCEPT



## The Node Object



Apply the Run List

Build Node (ohai)

Synchronize Cookbooks

Load Cookbooks

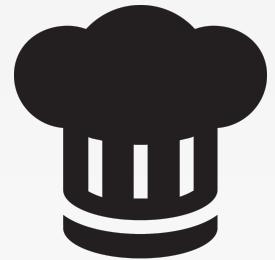
Build Resource Collection

Converge

<https://docs.chef.io/attributes.html - attribute-precedence>



# EXERCISE



## Refactor to Use Attributes

*Time to remove all the hard-coded values and make them attributes.*

### Objective:

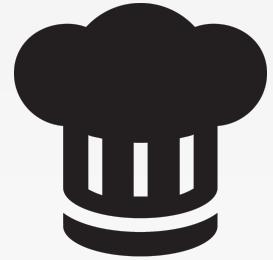
- Refactor the Install recipe to use a Node attribute
- Execute the tests and verify the tests fail
- Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

# Replace the Value with a Node Attribute

myiis/recipes/install.rb

```
#  
# Cookbook:: myiis  
# Recipe:: install  
  
#  
# Copyright:: 2018, The Authors, All Rights Reserved.  
powershell_script 'Install IIS' do  
  code node['myiis']['package_name']  
  action :run  
  
end
```

# EXERCISE



## Refactor to Use Attributes

*A change means a chance for us to run the tests!*

### Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- Execute the tests and verify the tests fail
- Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

# Execute the Tests to See it Fail



```
> chef exec rspec
```

```
..FFFFF...
```

```
Failures:
```

```
  1) apache::default When all attributes are default, on an Centos  
6.9 converges successfully
```

```
    Failure/Error: expect { chef_run }.to_not raise_error
```

```
      expected no Exception, got #<NoMethodError: undefined  
method `[]' for nil:NilClass> with backtrace:
```

```
      # /tmp/chefspec20180313-21260-taxe6tfile_cache_path/
```

COOKBOOKS



# EXERCISE



## Refactor to Use Attributes

*We definitely broke it! Now, let's fix it.*

### Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

# Ask Chef How to Generate an Attributes File



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

```
Usage: chef generate attribute [path/to/cookbook] NAME [options]
```

-C, --copyright COPYRIGHT	Name of the copyright hol...
-m, --email EMAIL	Email address of the auth...
-a, --generator-arg KEY=VALUE	Use to set arbitrary ...
-l, --license LICENSE	all_rights, apache2, mit,...
-g GENERATOR_COOKBOOK_PATH, --generator-cookbook	Use GENERATOR_COOKBOOK_PA...



# Use Chef to Generate a Default Attributes File



```
> chef generate attribute default
```

```
Compiling Cookbooks...
```

```
Recipe: code_generator::attribute
  * directory[/Users/Administrator/cookbooks/myiis/attributes]
action create
  - create new directory /Users/Administrator/cookbooks/myiis/
attributes
  * template[/Users/Administrator/cookbooks/myiis/attributes/
default.rb] action create
  - create new file /Users/Administrator/cookbooks/myiis/
attributes/default.rb
  - update content in file /Users/Administrator/cookbooks/myiis/
attributes/default.rb from none to e3b0c4
```



# View the Attributes File Generated



```
> tree attributes
```

```
attributes
```

```
└── default.rb
```

```
0 directories, 1 file
```

## Add the Default Node Attribute

myiis/attributes/default.rb

```
default['myiis']['package_name'] = 'Add-  
WindowsFeature Web-Server'
```

# EXERCISE



## Refactor to Use Attributes

*The work is done. Let's hope it's the right work. Run the tests!*

### Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- ✓ Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

# Execute the Tests to See it Pass



```
> chef exec rspec
```

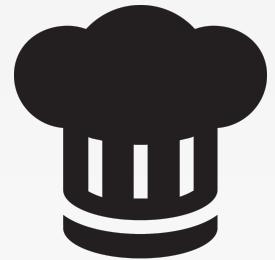
```
.....
```

```
Finished in 4.07 seconds (files took 3.93 seconds to load)
```

```
11 examples, 0 failures
```



# EXERCISE



## Refactor to Use Attributes

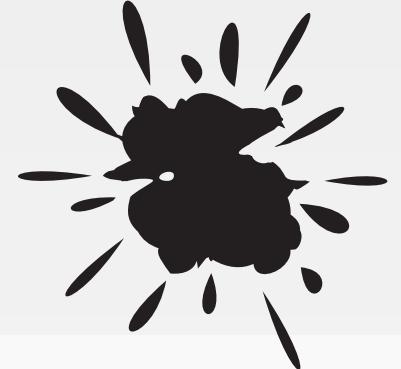
*We made a change and we know it works!*

### Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- ✓ Create the attributes file and add the Node attribute
- ✓ Execute the tests and verify the tests pass

# PROBLEM

**What if We Made a Typo?**



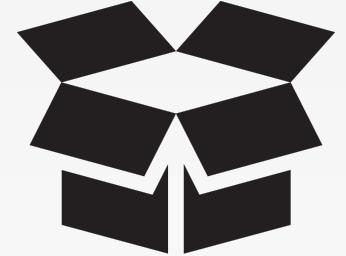
While implementing the node attribute what if made a mistake?

# Typos Like This One Will Waste Time

myis/attributes/default.rb

```
default['myis']['package_name'] = 'Add-  
WindowsFeature Web-Server'
```

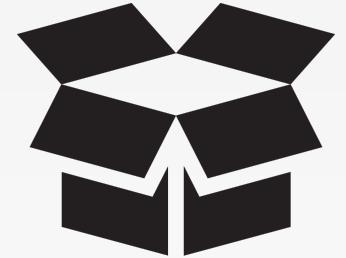
# CONCEPT



## Mental Model vs Actual Model

Faster feedback helps us build a greater mental model of the actual execution model. Tests that we define help strengthen it. However, tests are not very interactive as they are more like experiments. What we want is the ability to pause execution and look around.

# CONCEPT



## Pry a Debugger

Pry is a Ruby debugger that allows you to define break points. These breakpoints allow you to pause operation and interact with the current process being able to interrogate the current state of the system.

<http://pryrepl.org>



# EXERCISE



## Setup a Break Point

*Time to make trouble for ourselves.*

### Objective:

- Mutate the code and add the breakpoint
- Execute the tests to cause the breakpoint to trigger
- Remove the breakpoint and restore the code

## Create a Typo in the Defined Attribute

myis/attributes/default.rb

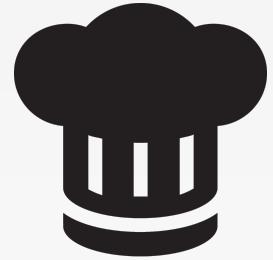
```
default['myis']['package_name'] = 'Add-  
WindowsFeature Web-Server'
```

# Add a Break Point in the Recipe

myiis/recipes/install.rb

```
#  
# Cookbook:: myiis  
# Recipe:: install  
  
#  
# Copyright:: 2018, The Authors, All Rights Reserved.  
+  
require 'pry'  
binding.pry  
  
powershell_script 'Install IIS' do  
  code node['myiis']['package_name']  
  action :run
```

# EXERCISE



## Setup a Break Point

*Time to pry into the code and see what it is going on.*

### Objective:

- ✓ Mutate the code and add the breakpoint
- Execute the tests to cause the breakpoint to trigger
- Remove the breakpoint and restore the code

# Execute the Test to Initiate Pry



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

```
From: C:/Users/Administrator/AppData/Local/Temp/2/d20181203-10224-48zz61/
cookbooks/myiis/recipes/install.rb @ line 9 Chef::Mixin::FromFile#from_file:
  4: #
  5: # Copyright:: 2018, The Authors, All Rights Reserved.
  6: require 'pry'
  7: binding.pry
  8:
=> 9: powershell_script 'Install IIS' do
 10:   code node['myiis']['package_name']
  # ... CONTINUED ON THE NEXT SLIDE ...
```

# Pry Provides an Interactive Prompt



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

```
=> 9: powershell_script 'Install IIS' do
  10:   code node['myiis']['package_name']
  11:   action :run
  12: end
[1] pry(<Chef::Recipe>)>
```

# Ask Pry for Help



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

## Help

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

## Context

`cd` Move into a new context (object or scope).

`find-method` Recursively search for a method within a class/module or the curr...

`ls` Show the list of vars and methods in the current scope.

`pry-backtrace` Show the backtrace for the pry session.

`raise-up` Raise an exception out of the current pry instance.

`reset` Reset the repl to a clean state.

To escape the help menu, type in q

## Execute Any Code As You Would in a Recipe



```
[2] pry(#<Chef::Recipe>) > node['myis']
```

```
=> nil
```



# Explore the Different Node Attributes



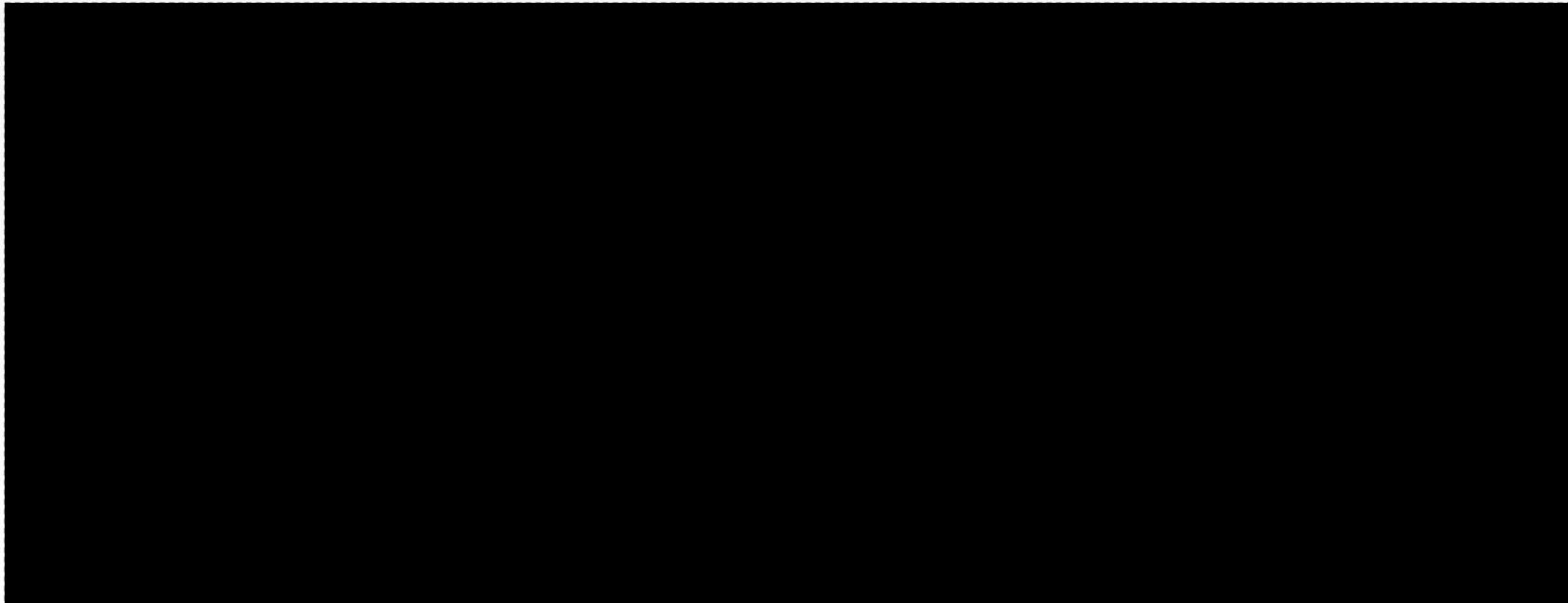
```
[3] pry(#<Chef::Recipe>) > node['myis']
```

```
=> { "package_name"=>"Add-WindowsFeature Web-Server"}
```

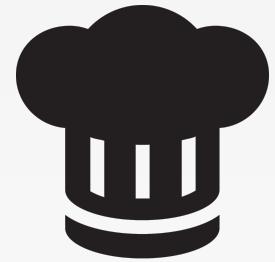
# Halt the Execution of the Test Immediately



```
[4] pry(#<Chef::Recipe>) > exit!
```



# EXERCISE



## Setup a Break Point

*Time to pry into the code and see what it is going on.*

### Objective:

- ✓ Mutate the code and add the breakpoint
- ✓ Execute the tests to cause the breakpoint to trigger
- Remove the breakpoint and restore the code

# Remove the Break Point from the Recipe

myiis/recipes/install.rb

```
#  
# Cookbook:: myiis  
# Recipe:: install  
#  
# Copyright:: 2018, The Authors, All Rights  
Reserved.  
require 'pry'  
binding.pry  
  
powershell_script 'Install IIS' do  
  code node['myiis']['package_name']  
  action :run
```

## Fix the Change in the Attributes

myiis/attributes/default.rb

```
default['myiis']['package_name'] = 'Add-  
WindowsFeature Web-Server'
```

# LAB



## Refactor Remaining Resources

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

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

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



## Update the Recipe to use the Node Attribute

myiis/recipes/service.rb

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

# Execute the Tests to See it Fail



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

```
FFF
```

```
Failures:
```

```
1) myiis::service When all attributes are default, on an Windows 2012R2  
converges successfully
```

```
Failure/Error: expect { chef_run }.to_not raise_error
```

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

```
# /tmp/chefspec20180313-17746-14gwtwpfile_cache_path/cookbooks/  
myiis/recipes/service.rb:6:in `from_file'
```



## Fix the Change in the Attributes

myiis/attributes/default.rb

```
default['myiis']['package_name'] = 'Add-  
WindowsFeature Web-Server'  
+  
default['myiis']['service_name'] = 'w3svc'
```

# Execute the Tests to See it Pass



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

```
...
```

```
Finished in 1.06 seconds (files took 4.33 seconds to load)
```

```
3 examples, 0 failures
```



## Update the Recipe to use the Node Attribute

myiis/recipes/configuration.rb

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

# Execute the Tests to See it Fail



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

```
FF
```

```
Failures:
```

```
1) myiis::configuration When all attributes are default, on an Windows  
2012R2 conve
```

```
Failure/Error: expect { chef_run }.to_not raise_error
```

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



## Fix the Change in the Attributes

myiis/attributes/default.rb

```
default['myiis']['package_name'] = 'Add-WindowsFeature  
Web-Server'  
default['myiis']['service_name'] = 'w3svc'  
default['myiis']['default_index_htm'] = 'C:  
\inetpub\wwwroot\Default.htm'
```

# Execute the Tests to See it Pass



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

```
..
```

```
Finished in 2.27 seconds (files took 1.82 seconds to load)
```

```
2 examples, 0 failures
```



# LAB



## Refactor Remaining Resources

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

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

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

## Discussion



What are the benefits of providing the package name and service name as node attributes?

What value does Pry provide to you as a Cookbook Developer?

## Q&A



What questions can we answer for you?



CHEF™