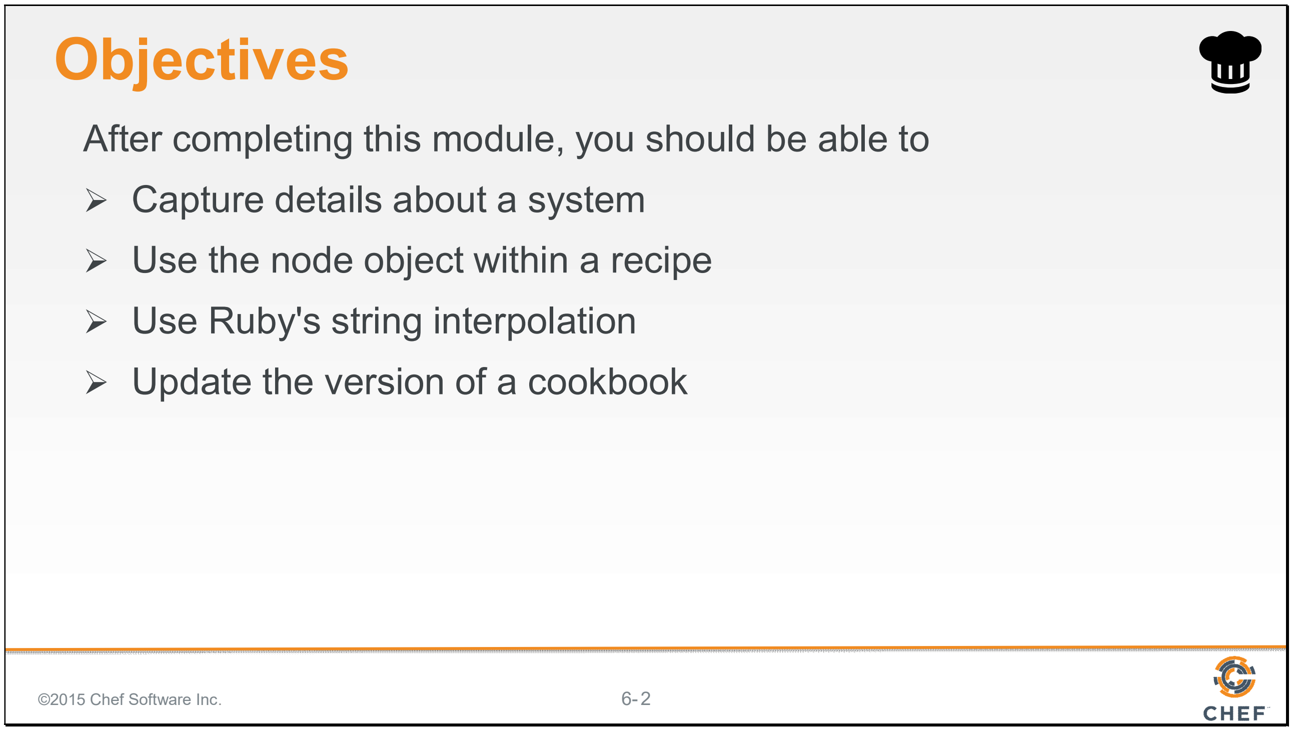
6: Details About the System

Details About the System

Finding and Displaying Information About Our System

Slide 2



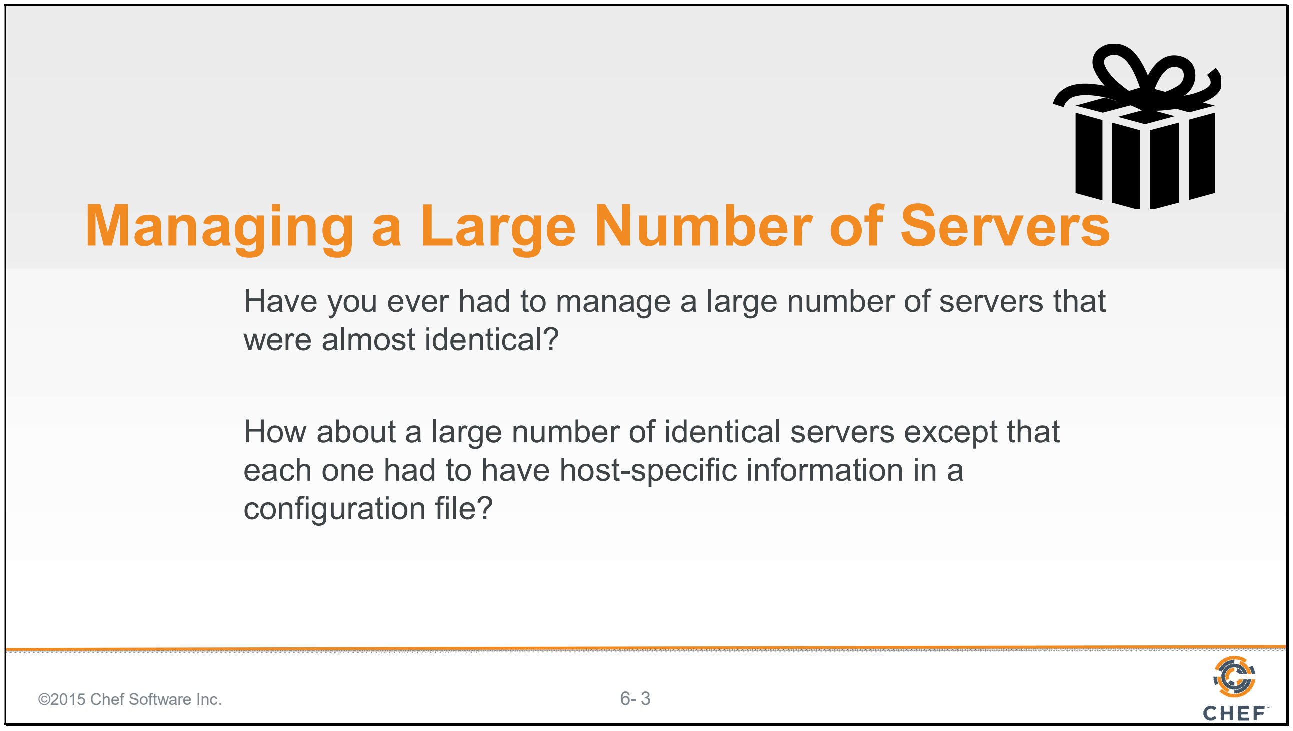
Objectives

After completing this module, you should be able to

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

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

Slide 3



Managing a Large Number of Servers

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

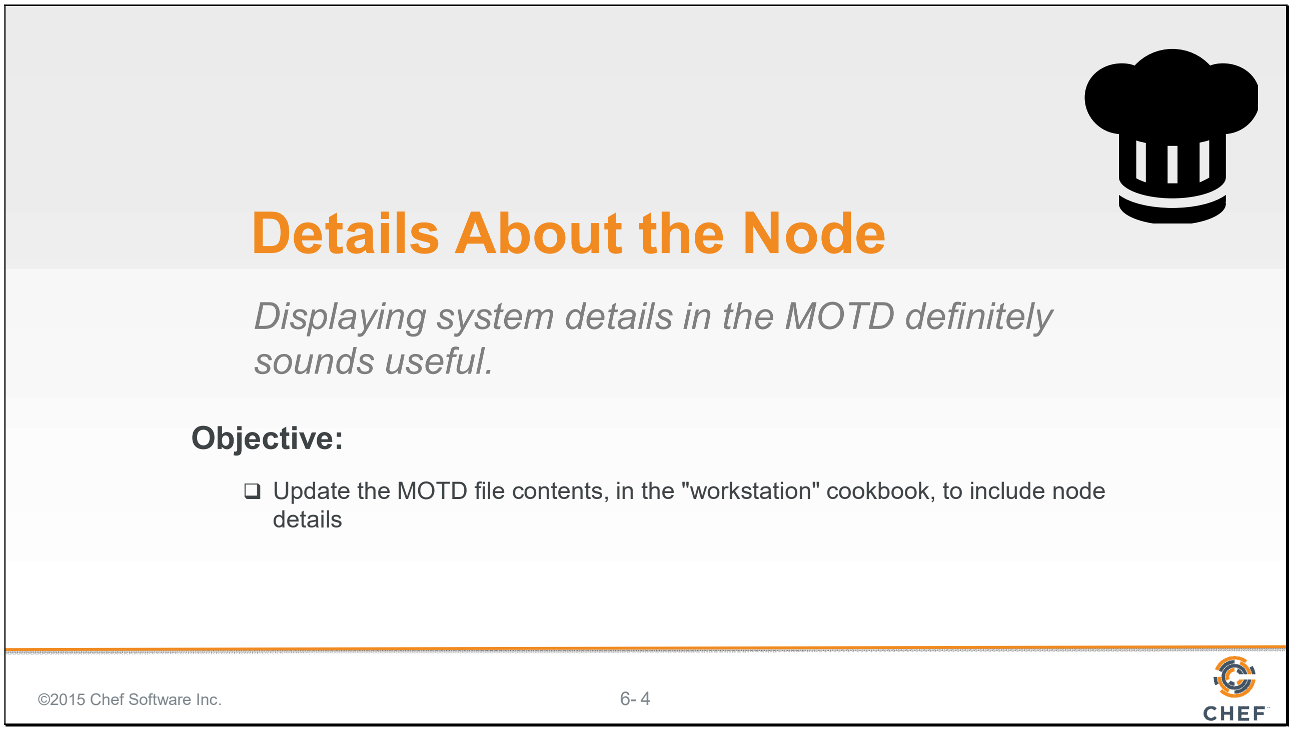
How about a large number of identical servers except that each one had to have host-specific information in a configuration file?

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

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

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

Slide 4



Details About the Node

Displaying system details in the MOTD definitely sounds useful.

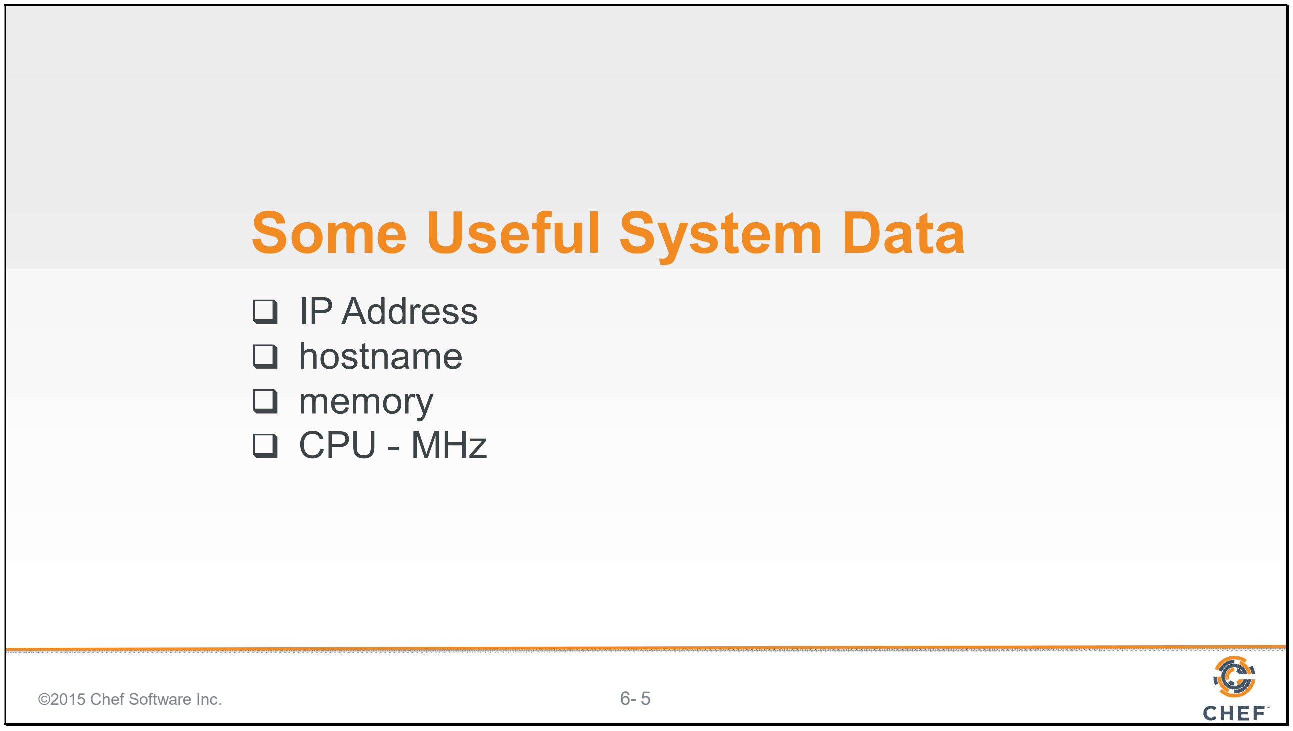
Objective:

- ❑ Update the MOTD file contents, in the "workstation" cookbook, to include node details

Here we've been given the simple request of providing some additional details about our node in both our Message of the Day and our default index page that we deploy with our web server.

We'll start first with our message of the day.

Slide 5



Some Useful System Data

- ❑ IP Address
- ❑ hostname
- ❑ memory
- ❑ CPU - MHz

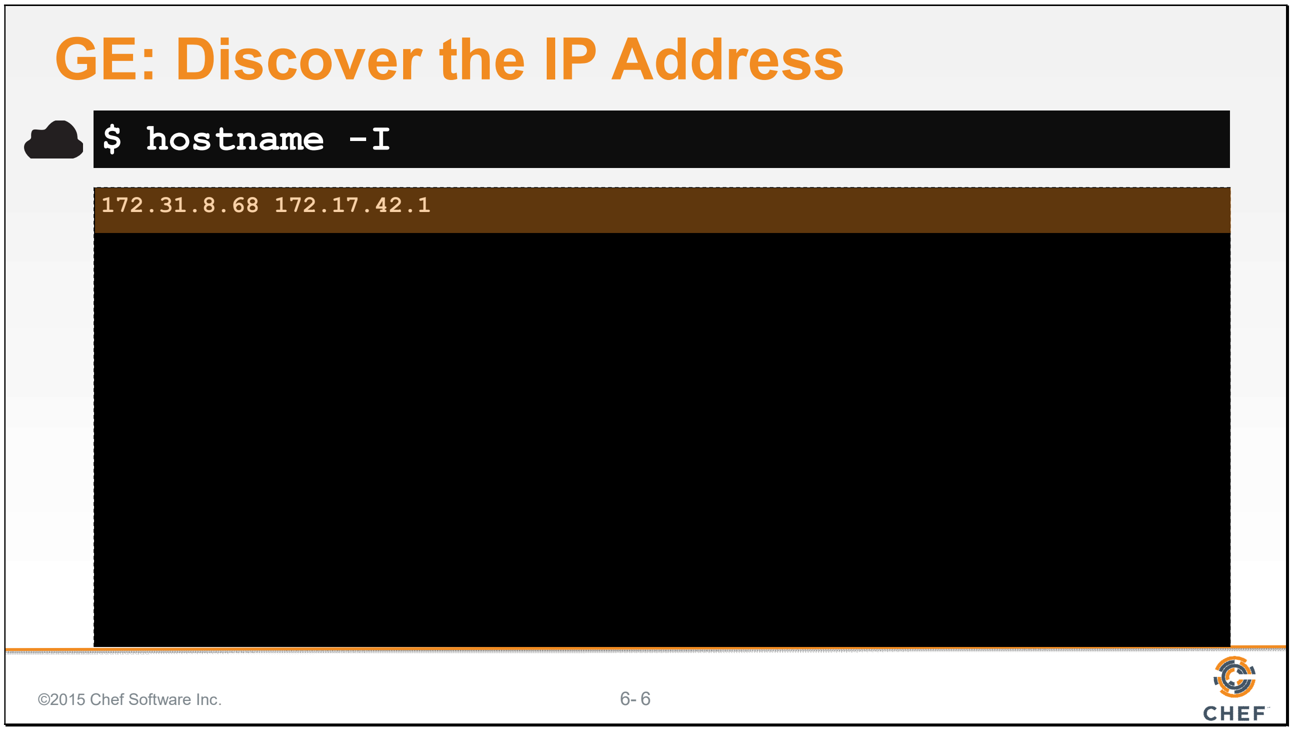
Thinking about some of the scenarios that we mentioned at the start of the session makes us think that it would be useful to capture:

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

We'll walk through capturing that information using various system commands starting with the IP address.

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

Slide 6



GE: Discover the IP Address

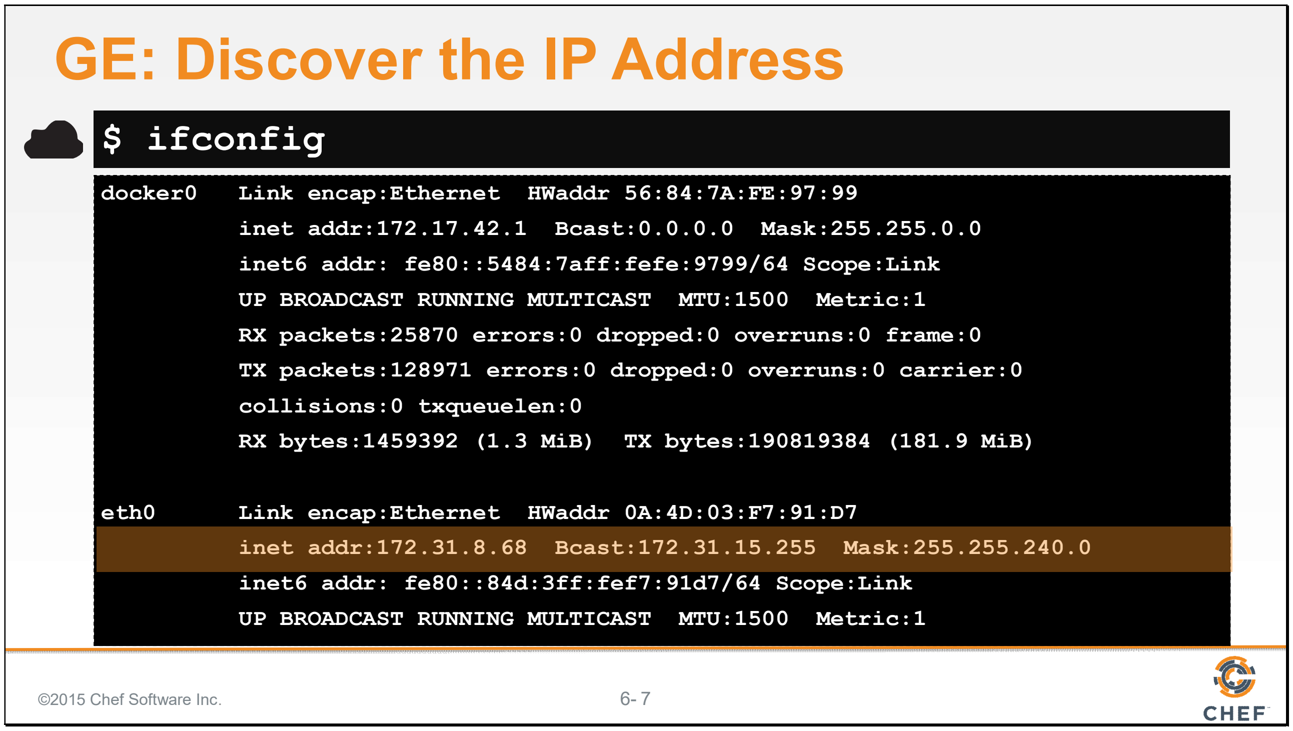
```
$ hostname -I
```

```
172.31.8.68 172.17.42.1
```

To discover the IP address of the node, we can issue the command

`hostname -I`

Slide 7



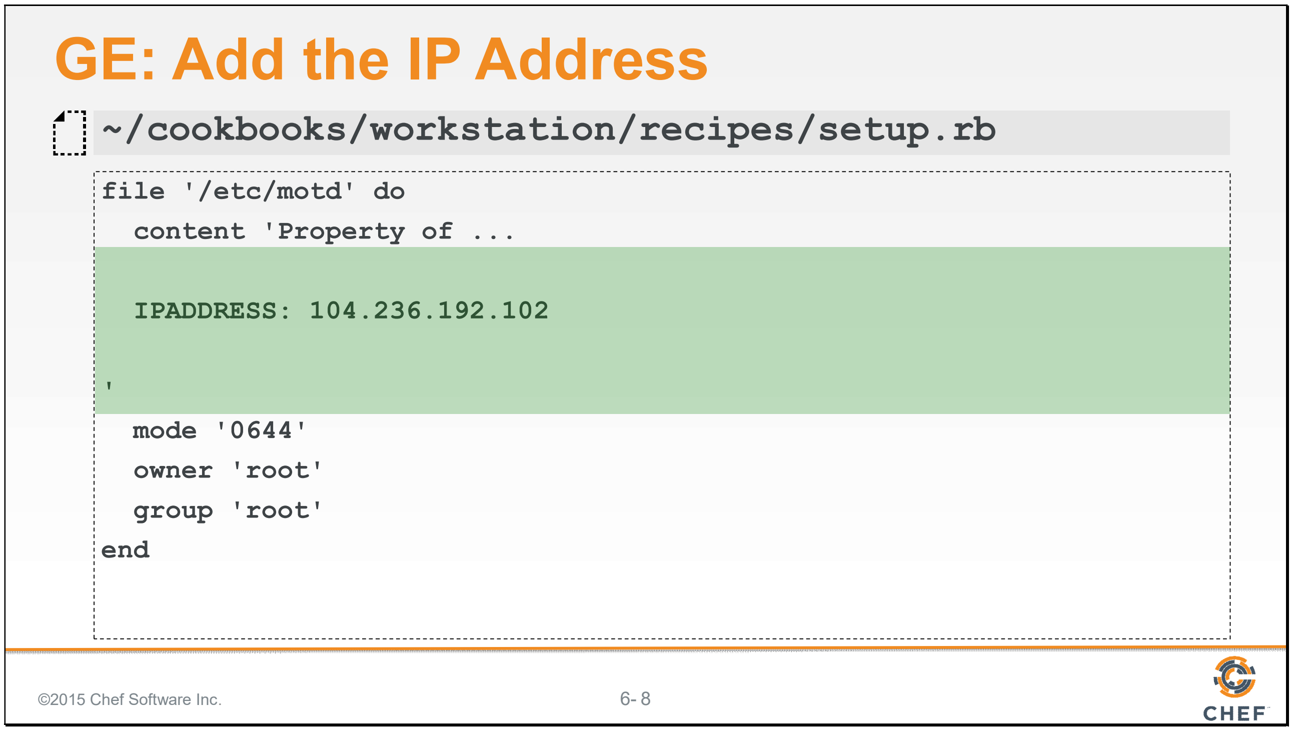
GE: Discover the IP Address

```
$ ifconfig
```

```
docker0   Link encap:Ethernet  HWaddr 56:84:7A:FE:97:99
          inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::5484:7aff:fefe:9799/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25870 errors:0 dropped:0 overruns:0 frame:0
          TX packets:128971 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1459392 (1.3 MiB)  TX bytes:190819384 (181.9 MiB)

eth0      Link encap:Ethernet  HWaddr 0A:4D:03:F7:91:D7
          inet addr:172.31.8.68  Bcast:172.31.15.255  Mask:255.255.240.0
          inet6 addr: fe80::84d:3ff:fef7:91d7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Or you can dig it out of the results of running `ifconfig`.

Slide 8



GE: Add the IP Address

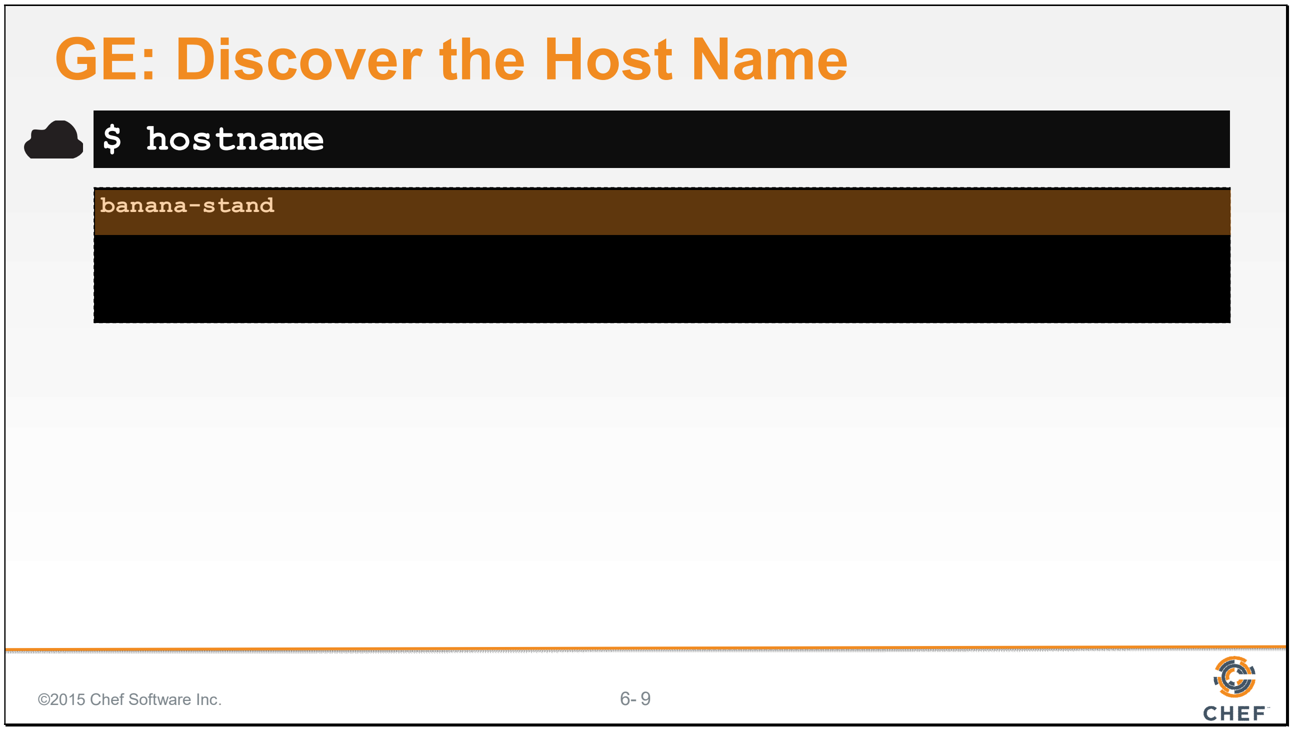
~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: 104.236.192.102

'
  mode '0644'
  owner 'root'
  group 'root'
end
```

You can include this information in our '/etc/motd by updating the contents of the file resource's content attribute.

Within the existing string value we've inserted a number of new lines for formatting and placed our IP address along with its value.

Slide 9



GE: Discover the Host Name

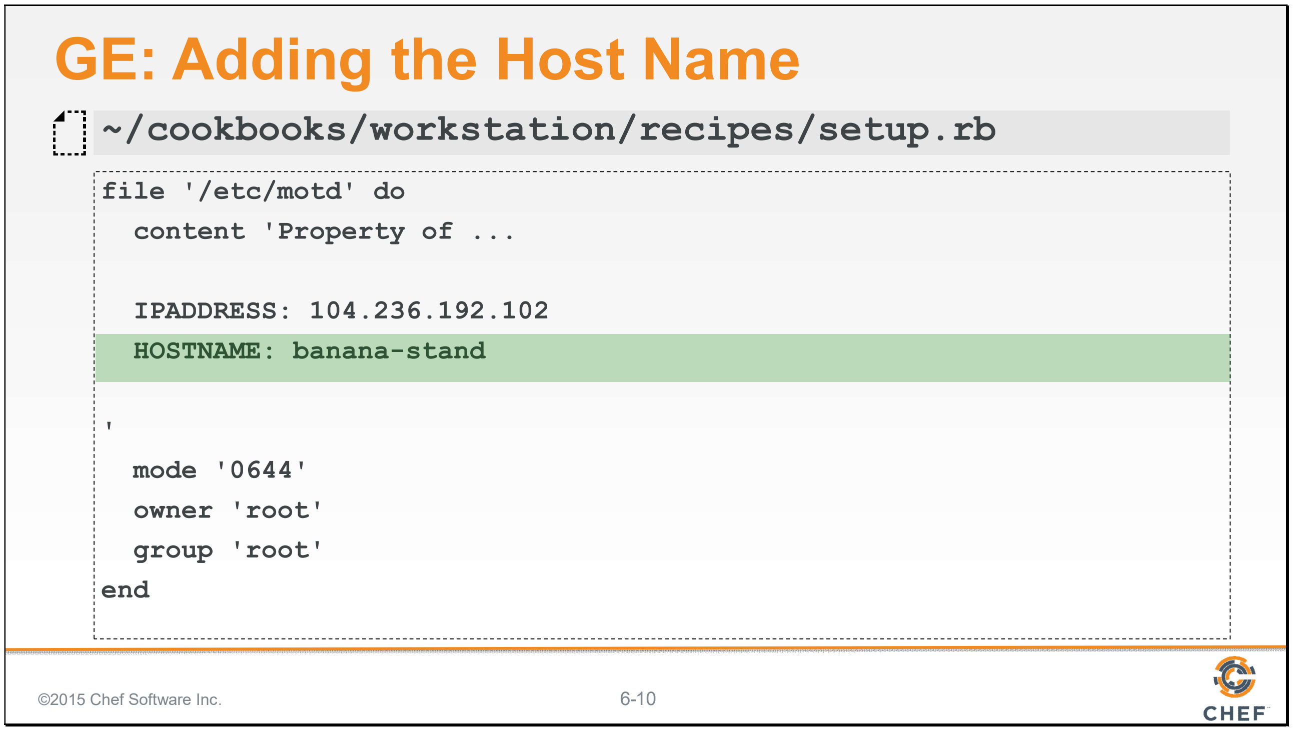
```
$ hostname
```

```
banana-stand
```

Next is the machine's hostname. This is easily retrievable with the `hostname` command.

Note: The host name of your virtual workstation may simply be an IP address. For example, 'ip-172-31-2-14x'.

Slide 10



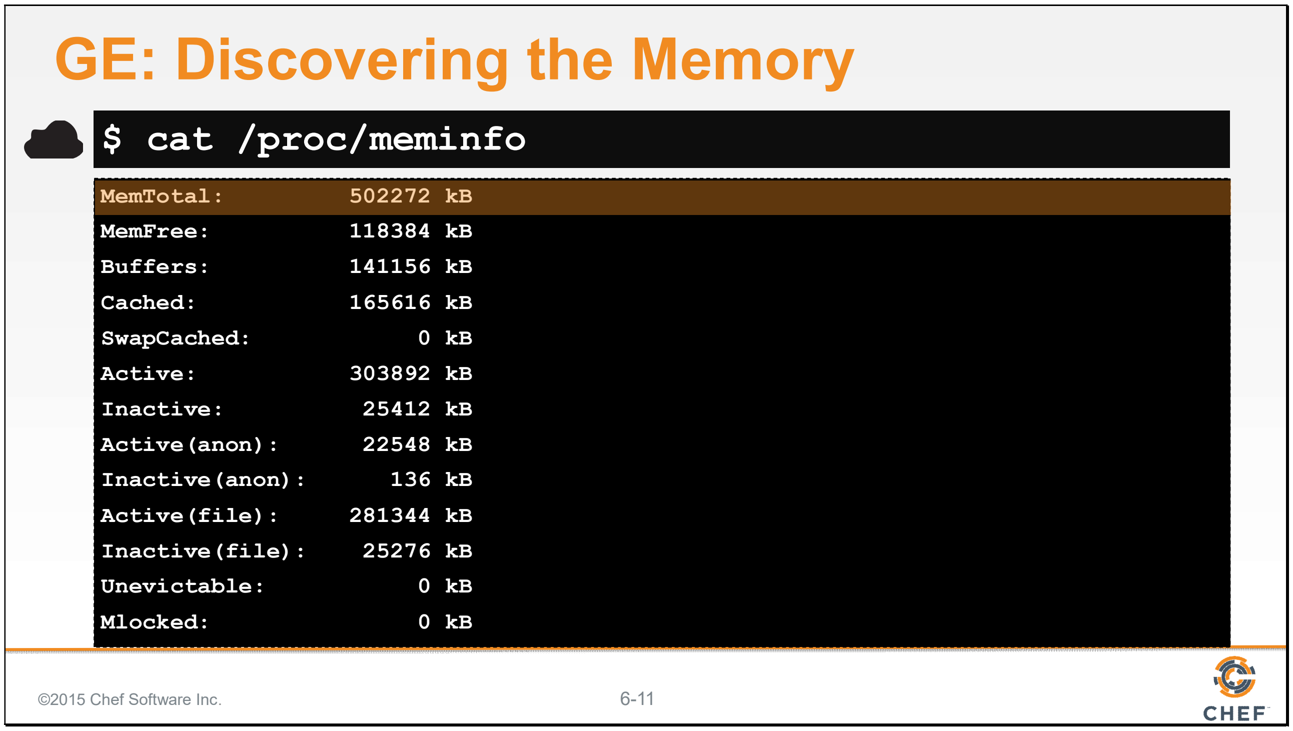
GE: Adding the Host Name

~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: 104.236.192.102
  HOSTNAME: banana-stand

'
  mode '0644'
  owner 'root'
  group 'root'
end
```

We can also include this information in the file resource's attribute on a new line below our IP address.

Slide 11



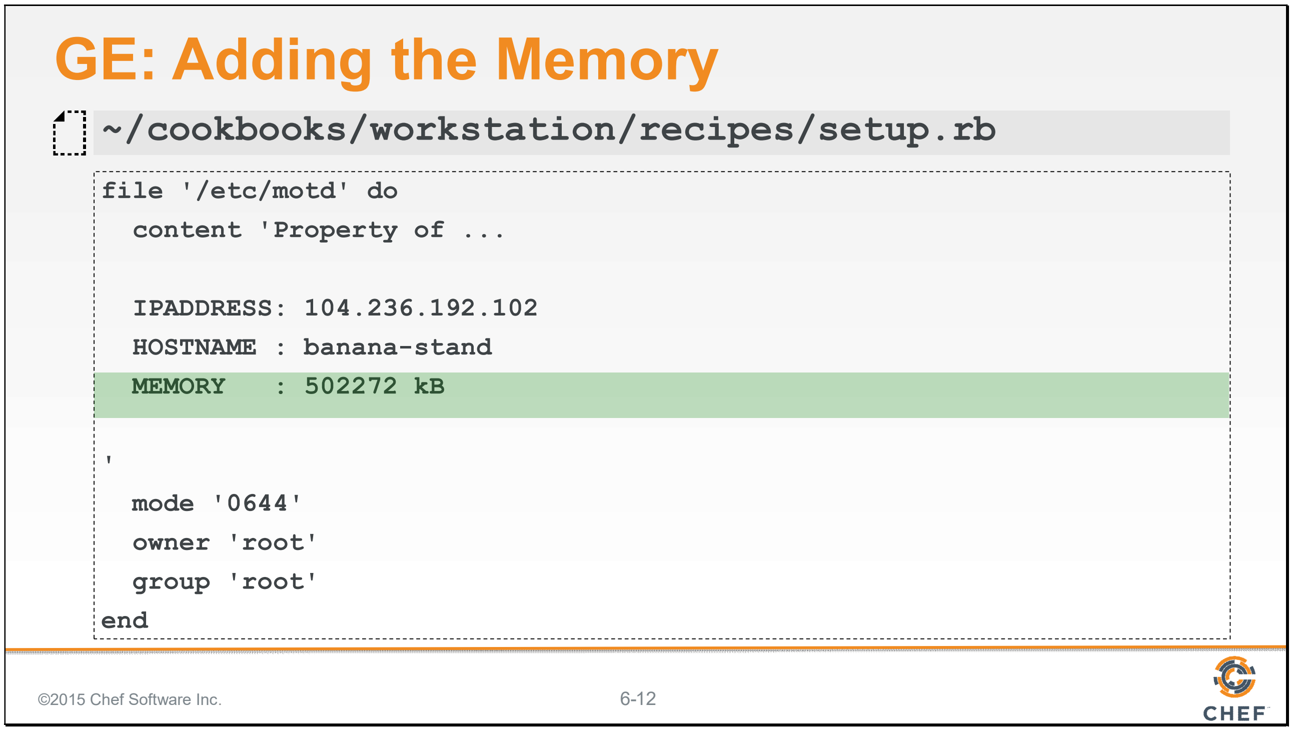
GE: Discovering the Memory

```
$ cat /proc/meminfo
```

```
MemTotal:            502272 kB
MemFree:             118384 kB
Buffers:             141156 kB
Cached:              165616 kB
SwapCached:               0 kB
Active:              303892 kB
Inactive:             25412 kB
Active(anon):         22548 kB
Inactive(anon):         136 kB
Active(file):        281344 kB
Inactive(file):       25276 kB
Unevictable:              0 kB
Mlocked:                  0 kB
```

One way to gather the memory of our system is to `cat` the contents of the /proc/meminfo. There we can select the total memory available on the system.

Slide 12



GE: Adding the Memory

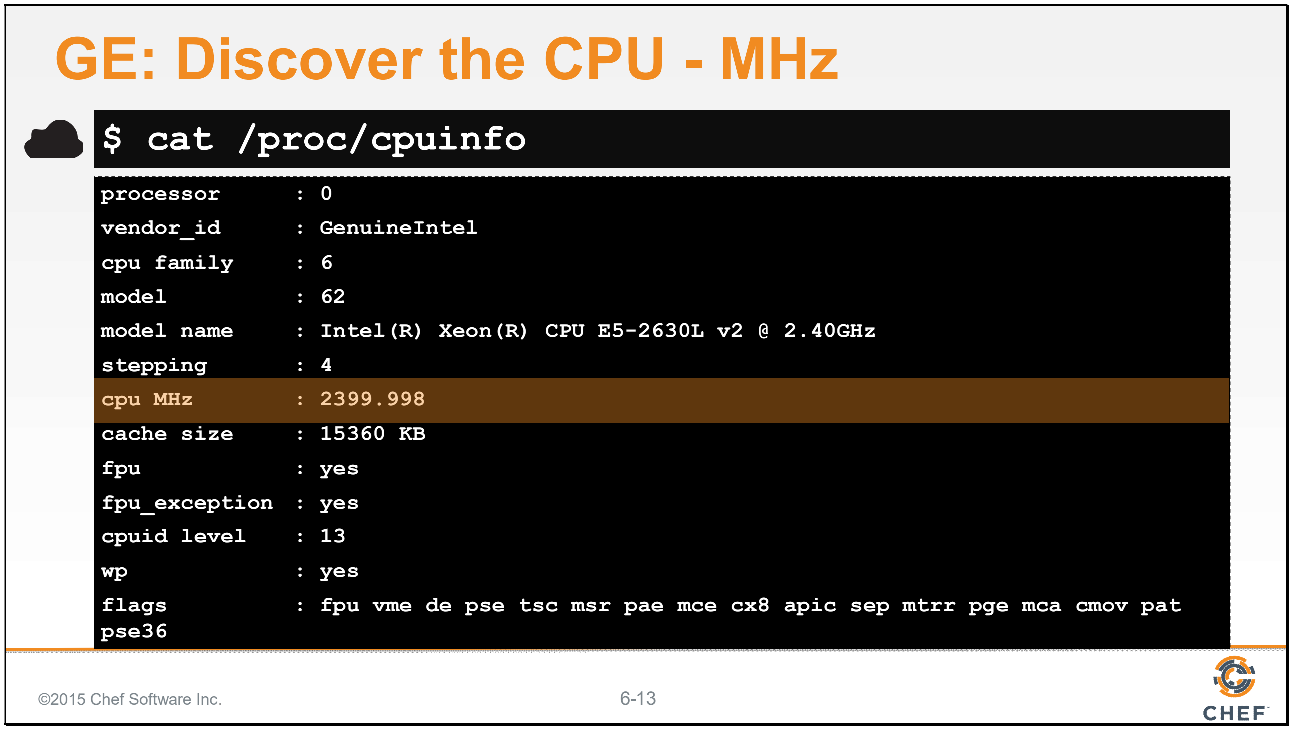
~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: 104.236.192.102
  HOSTNAME : banana-stand
  MEMORY   : 502272 kB

'
  mode '0644'
  owner 'root'
  group 'root'
end
```

©2015 Chef Software Inc. 6-12

And again, add it in the file resource's attribute below your hostname.

Slide 13



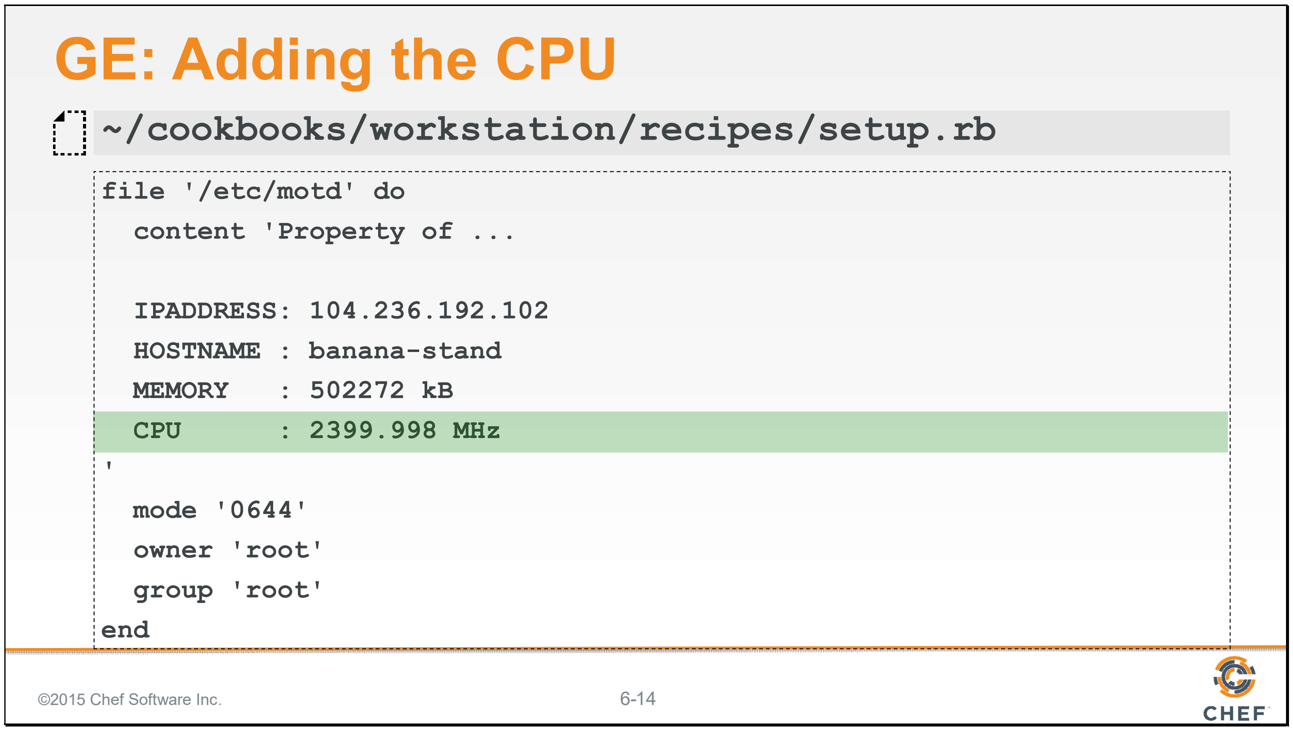
GE: Discover the CPU - MHz

```
$ cat /proc/cpuinfo
```

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40GHz
stepping        : 4
cpu MHz         : 2399.998
cache size      : 15360 KB
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat
pse36
```

Discovering information about the system's CPU is very similar. We can `cat` the contents of /proc/cpuinfo and select the CPU megahertz from the results.

Slide 14



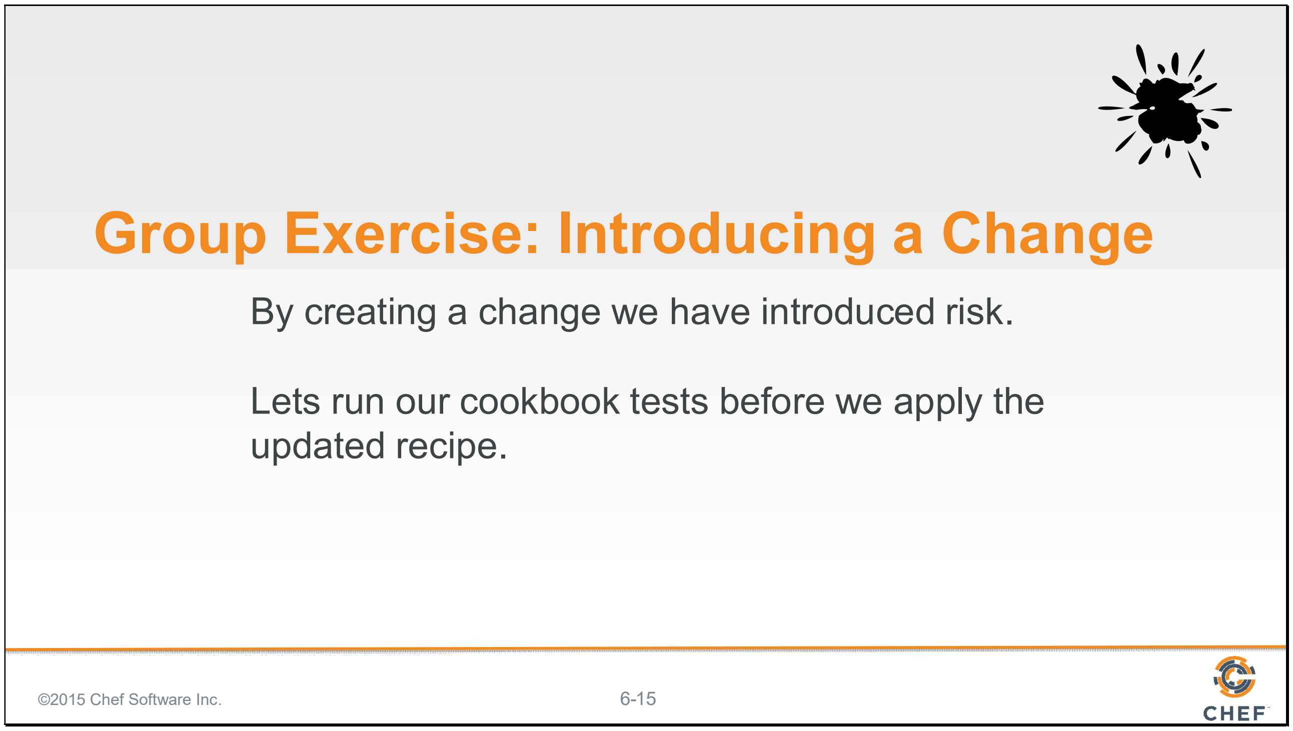
GE: Adding the CPU

~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: 104.236.192.102
  HOSTNAME : banana-stand
  MEMORY   : 502272 kB
  CPU      : 2399.998 MHz
'
  mode '0644'
  owner 'root'
  group 'root'
end
```

Add the CPU information to the file resource's content attribute.

Slide 15



Group Exercise: Introducing a Change

By creating a change we have introduced risk.

Lets run our cookbook tests before we apply the updated recipe.

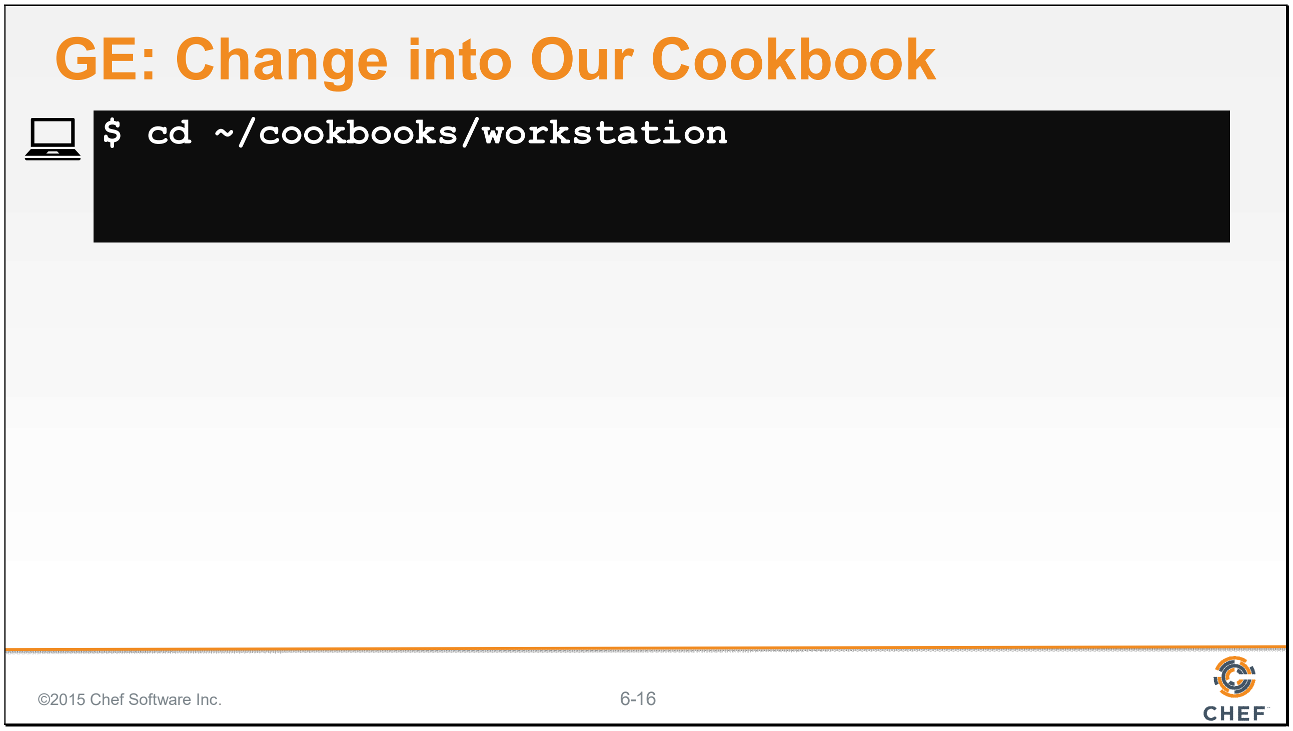
©2015 Chef Software Inc. 6-15

By updating the file resource we have introduced a change to the cookbook and introduced a risk. This change may not work. It could be a typo when transcribed from the slide, or the code that we have provided you may be out-of-date, or very possibly, incorrect.

Before we apply the updated recipe we can use testing to ensure the recipe is correctly defined.

Instructor Note: This is important to reinforce the workflow of working with recipes. Even though this seems like a small change it important to verify every change.

Slide 16

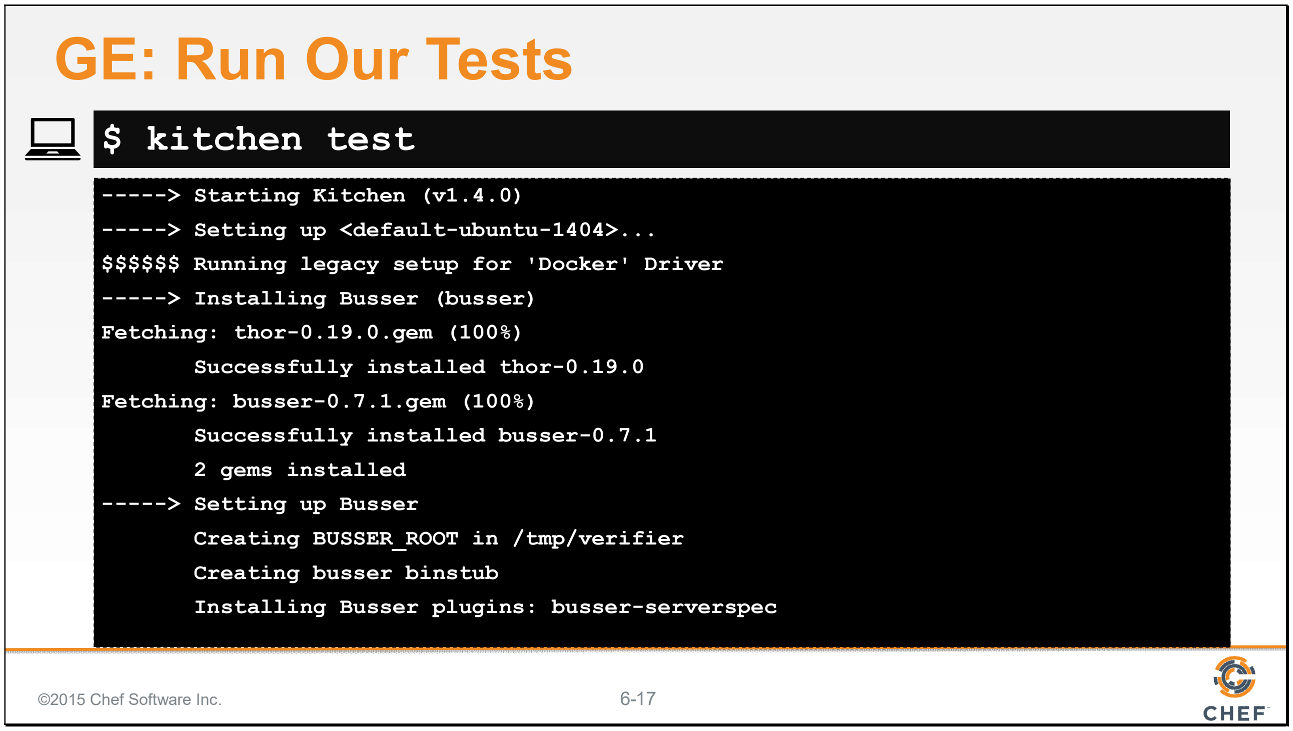


GE: Change into Our Cookbook

```
$ cd ~/cookbooks/workstation
```

Remember, we are testing a specific cookbook with kitchen so we need to be within the directory of the cookbook. So change directory into the workstation cookbook's directory.

Slide 17



GE: Run Our Tests

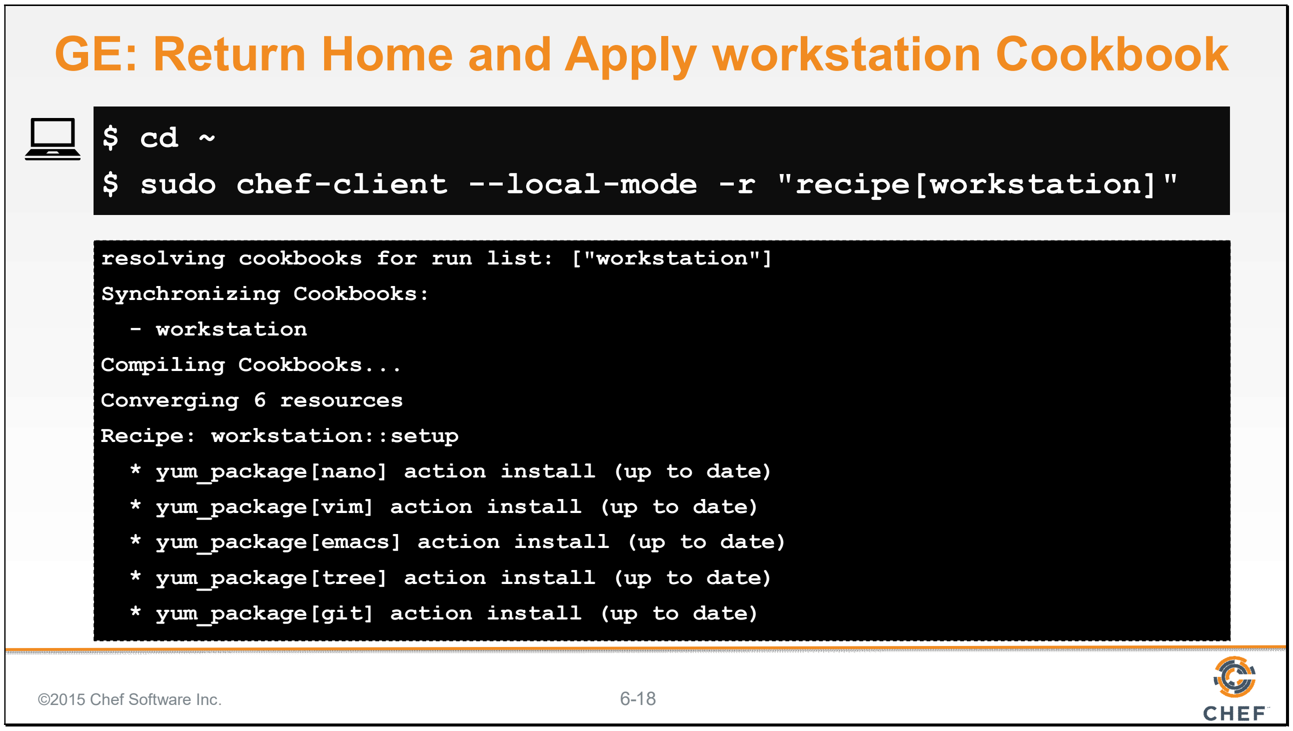
```
$ kitchen test
```

```
-----> Starting Kitchen (v1.4.0)
-----> Setting up <default-ubuntu-1404>...
$$$$$$ Running legacy setup for 'Docker' Driver
-----> Installing Busser (busser)
Fetching: thor-0.19.0.gem (100%)
       Successfully installed thor-0.19.0
Fetching: busser-0.7.1.gem (100%)
       Successfully installed busser-0.7.1
       2 gems installed
-----> Setting up Busser
       Creating BUSSER_ROOT in /tmp/verifier
       Creating busser binstub
       Installing Busser plugins: busser-serverspec
```

©2015 Chef Software Inc. 6-17

We have not defined any new tests related to the content changes of the /etc/motd. So running the tests will tell us if we have accidentally broken any of the existing functionality but there is nothing testing the new functionality that we added.

Slide 18



GE: Return Home and Apply workstation Cookbook

```
$ cd ~
$ sudo chef-client --local-mode -r "recipe[workstation]"
```

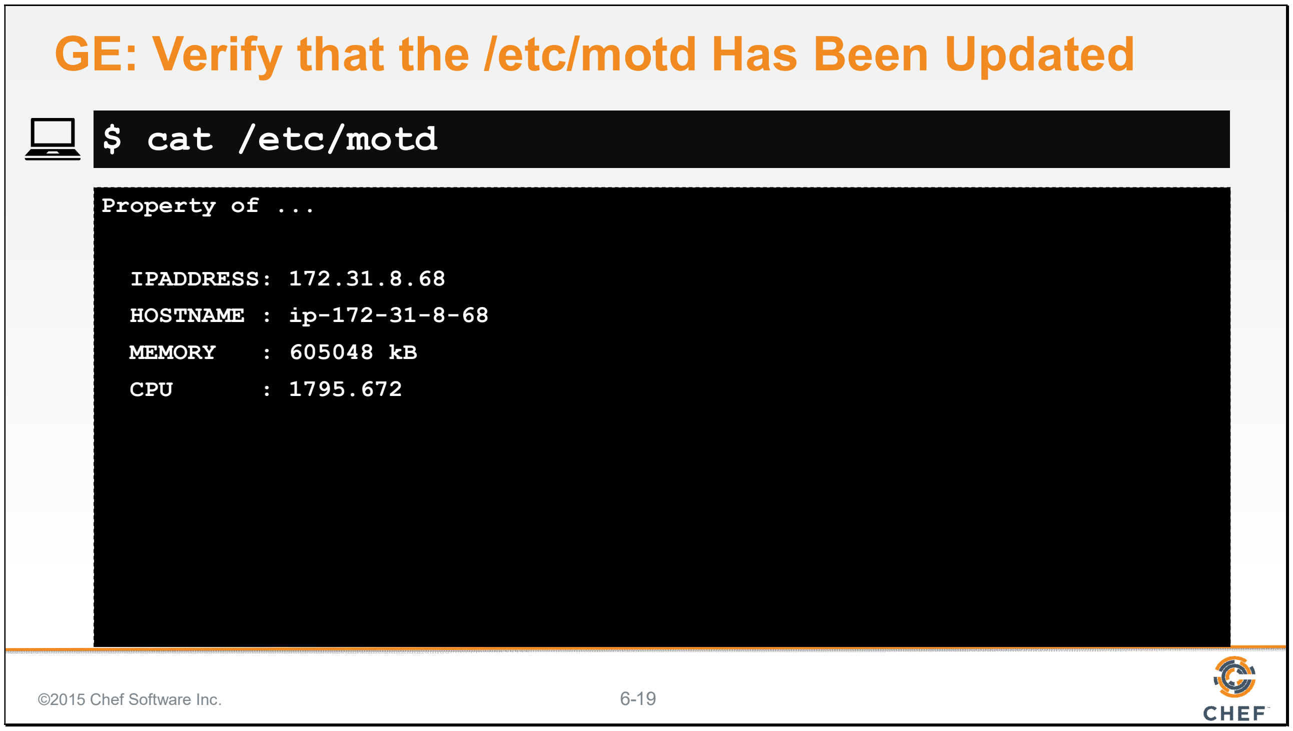
```
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
Converging 6 resources
Recipe: workstation::setup
  * yum_package[nano] action install (up to date)
  * yum_package[vim] action install (up to date)
  * yum_package[emacs] action install (up to date)
  * yum_package[tree] action install (up to date)
  * yum_package[git] action install (up to date)
```

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

1. So we need to return home to the parent directory of all our cookbooks.

2. Then use `chef-client` to locally apply the run list defined as: the workstation cookbook's default recipe.

Slide 19



GE: Verify that the /etc/motd Has Been Updated

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

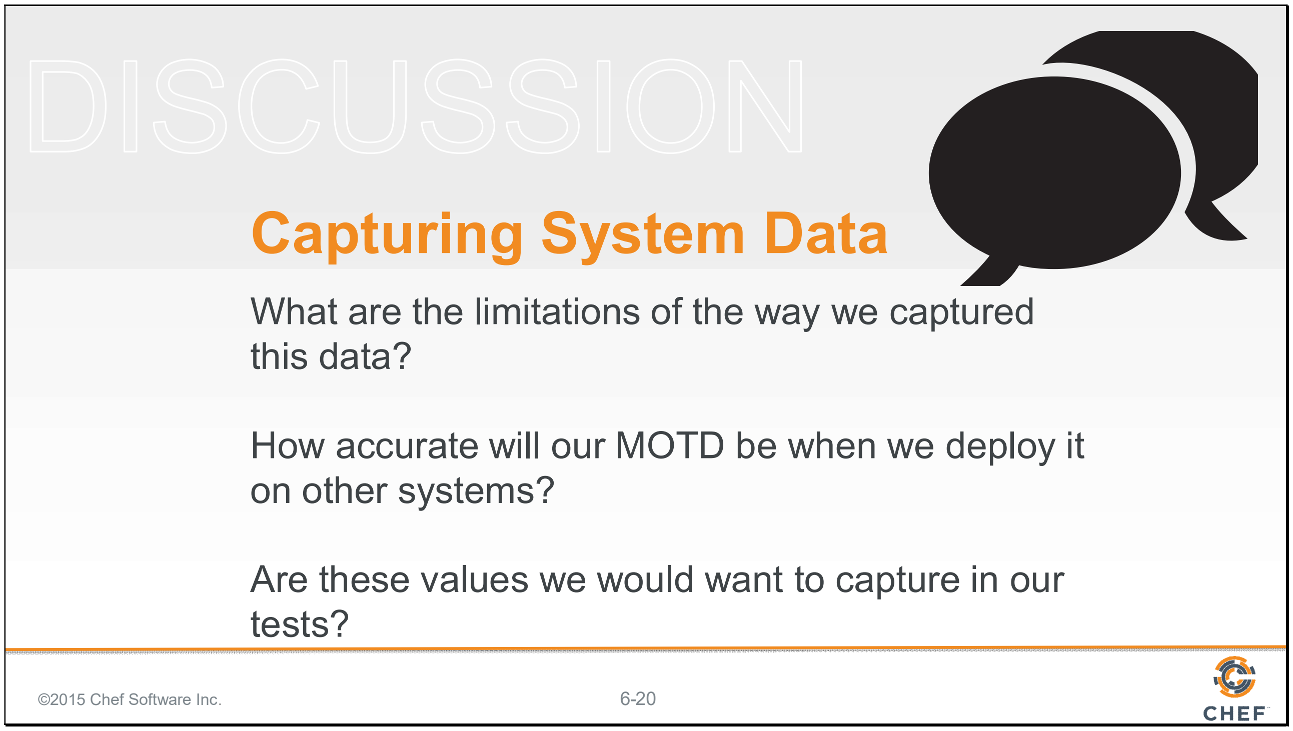
```
Property of ...

  IPADDRESS: 172.31.8.68
  HOSTNAME : ip-172-31-8-68
  MEMORY   : 605048 kB
  CPU      : 1795.672
```

©2015 Chef Software Inc.

6-19

Verify that your /etc/motd had been updated with our values.

Slide 20



DISCUSSION

Capturing System Data

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

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

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

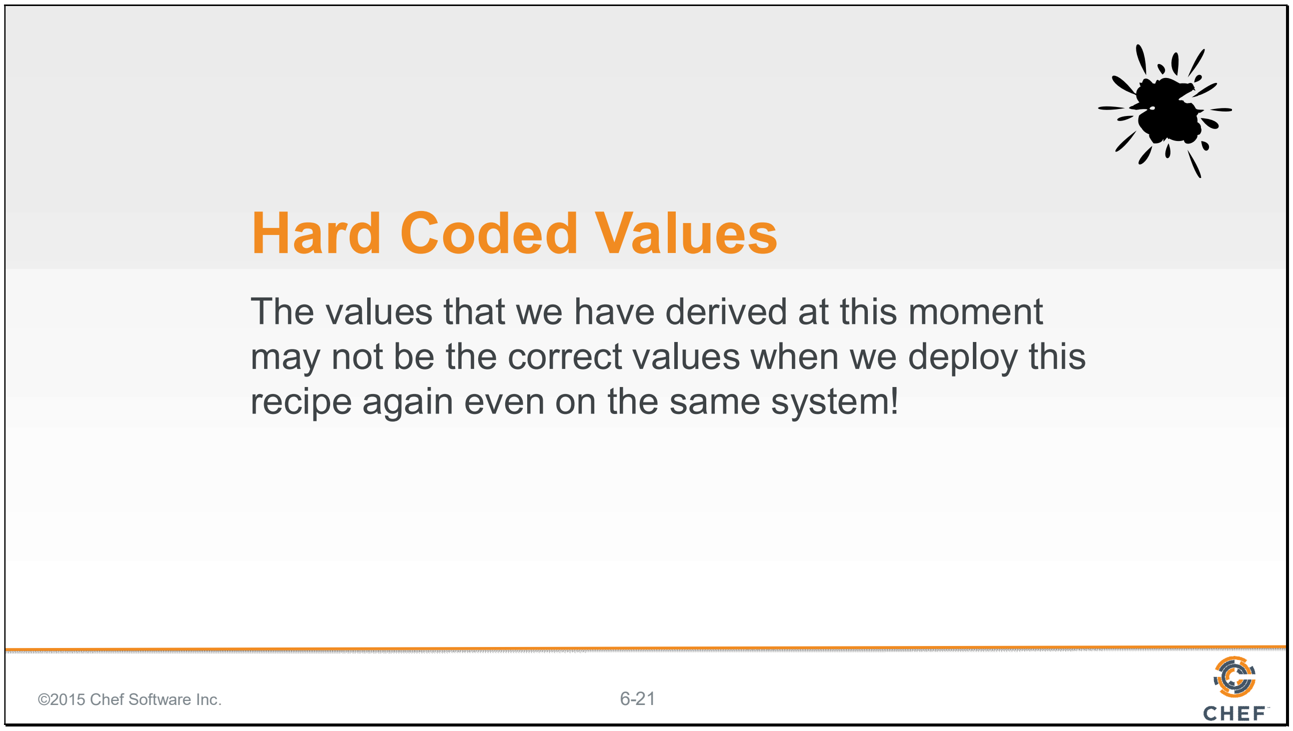
Now that we've defined these values, let’s reflect:

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

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

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

Slide 21



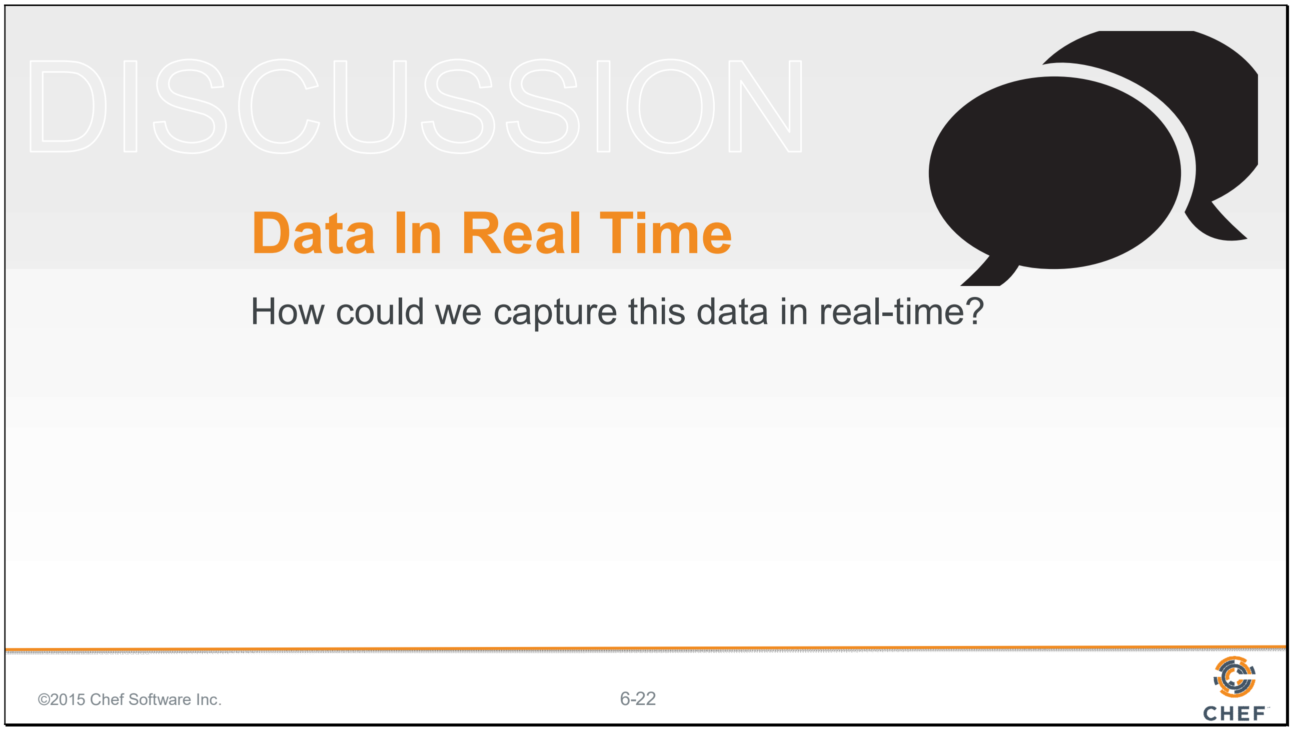
Hard Coded Values

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!

©2015 Chef Software Inc. 6-21

CHEF

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


DISCUSSION

Data In Real Time

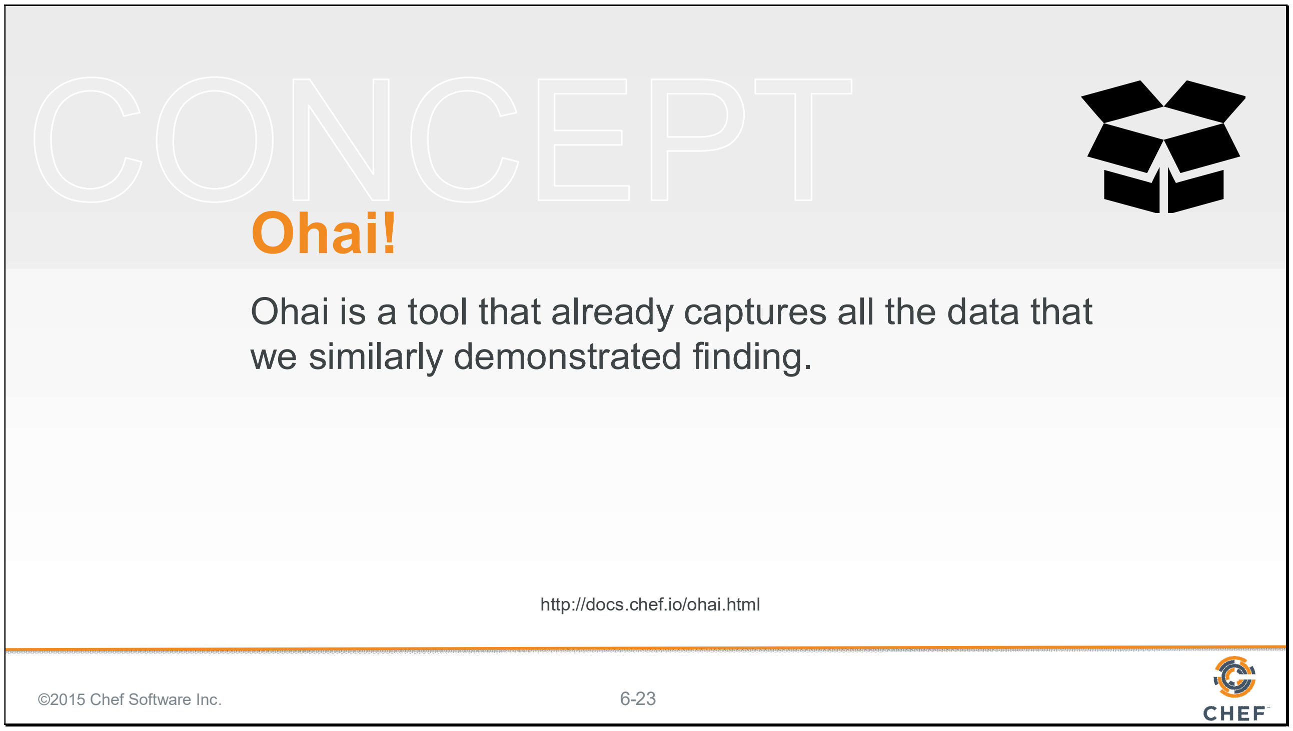
How could we capture this data in real-time?

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

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

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

Slide 23



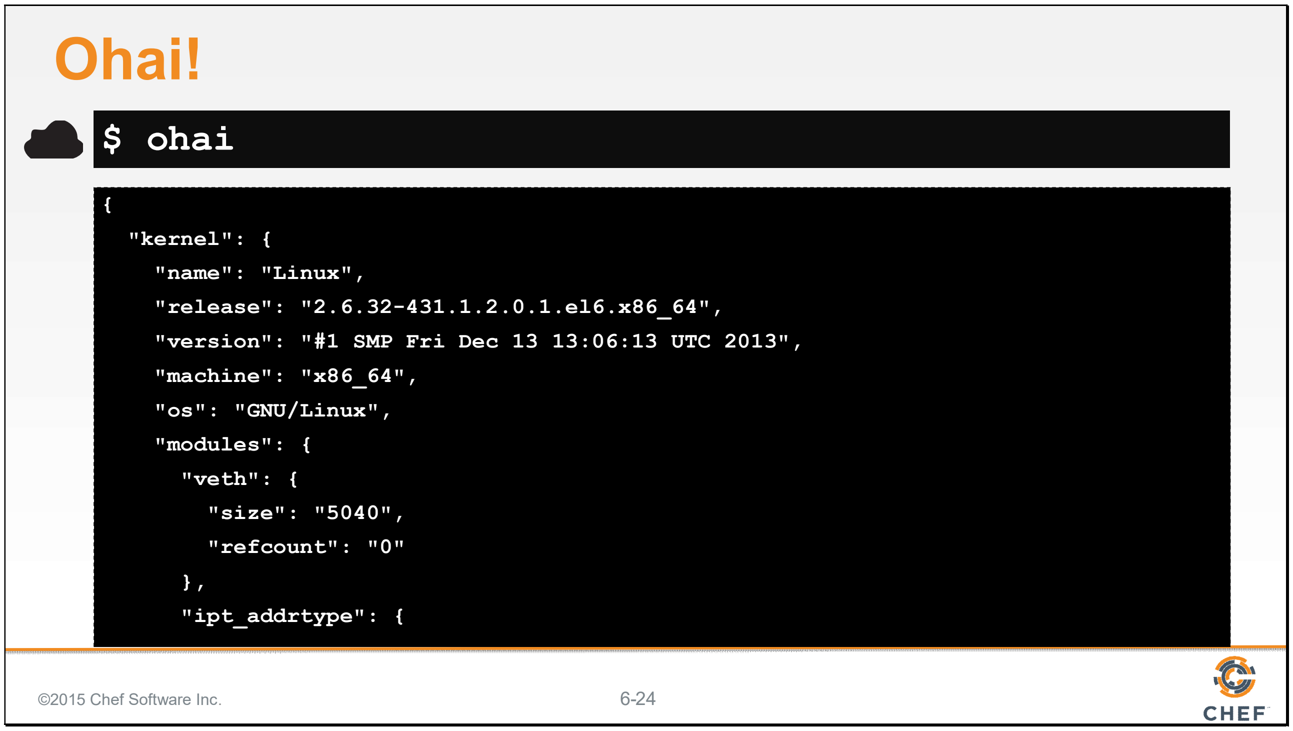
CONCEPT

Ohai!

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

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

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



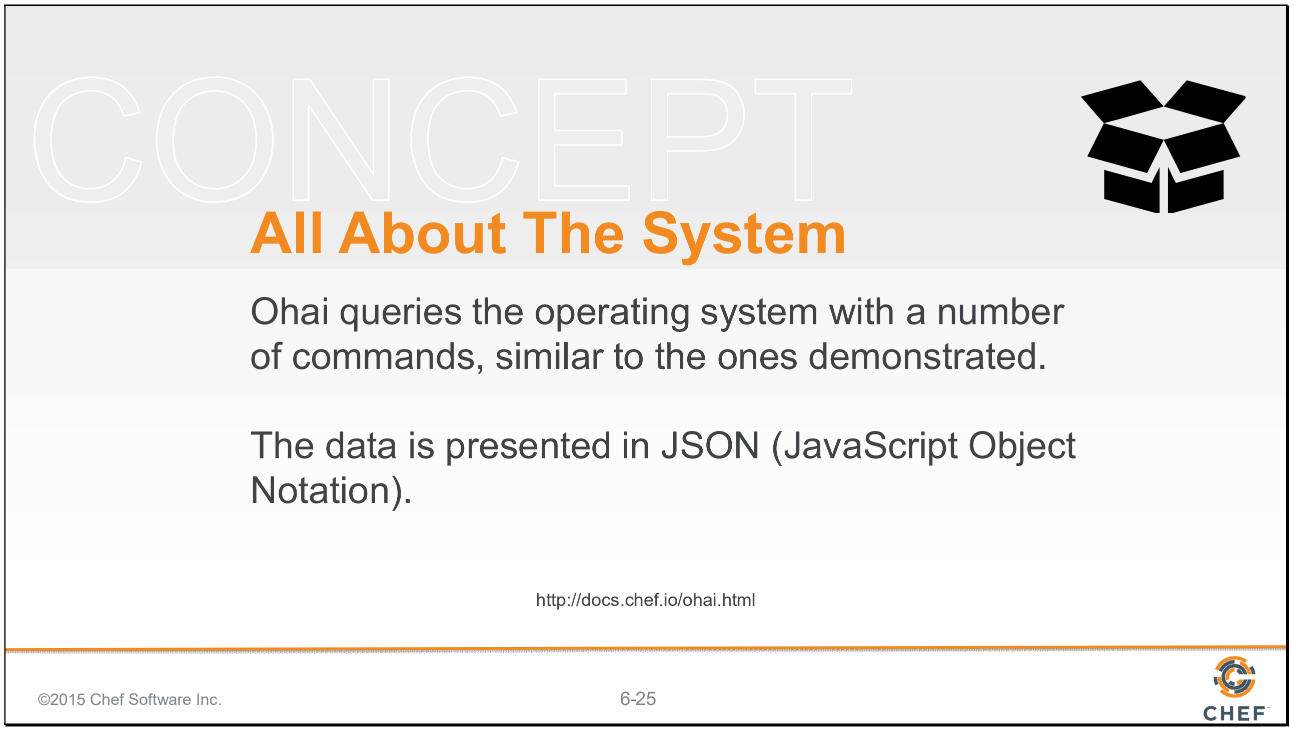
Ohai!

```
$ ohai
```

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

Ohai is also a command-line application that is part of the ChefDK.

Slide 25



CONCEPT

All About The System

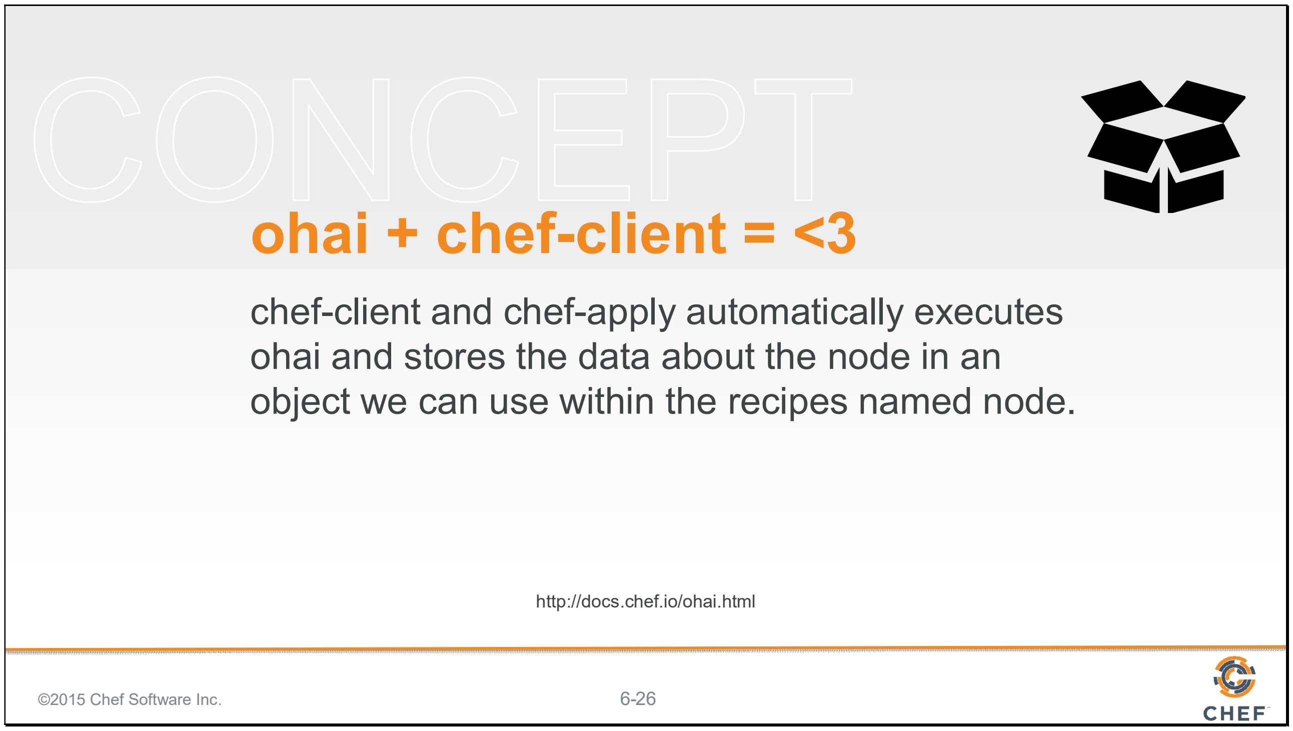
Ohai queries the operating system with a number of commands, similar to the ones demonstrated.

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

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

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

Slide 26


CONCEPT

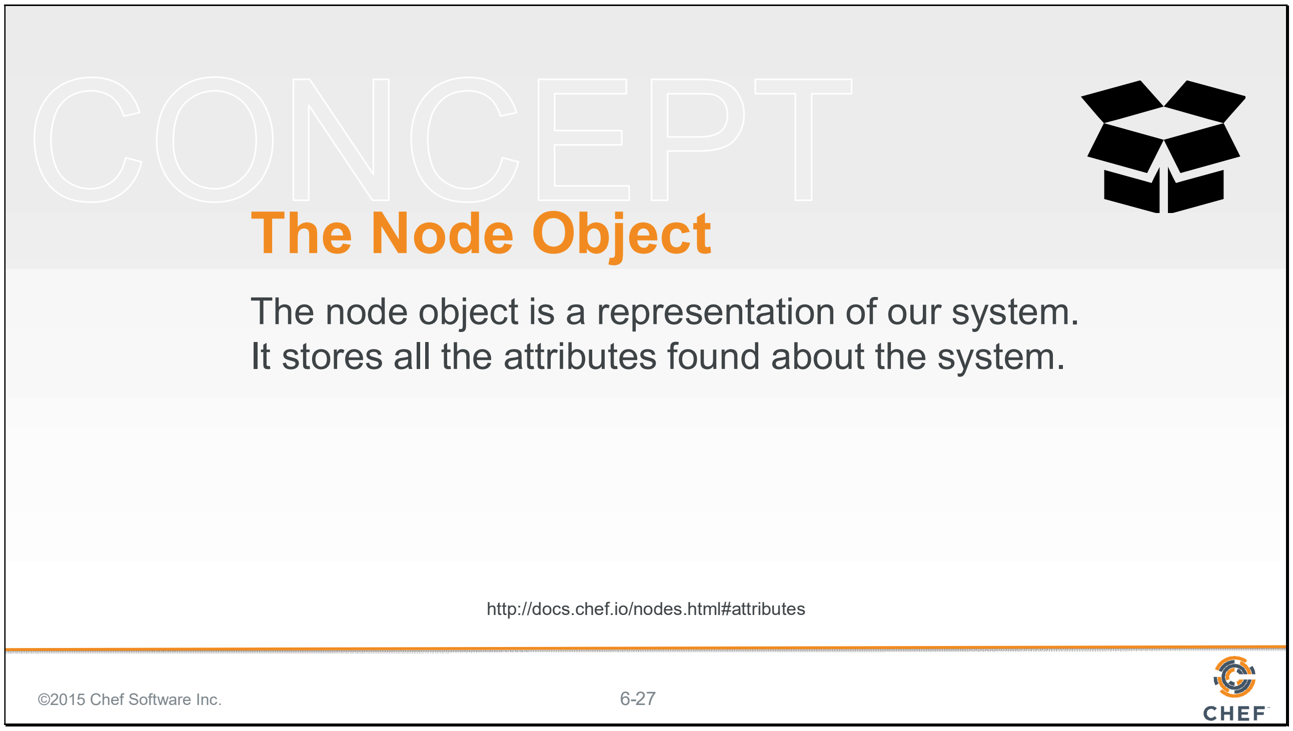
ohai + chef-client = <3

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

CHEF

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



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

6-27

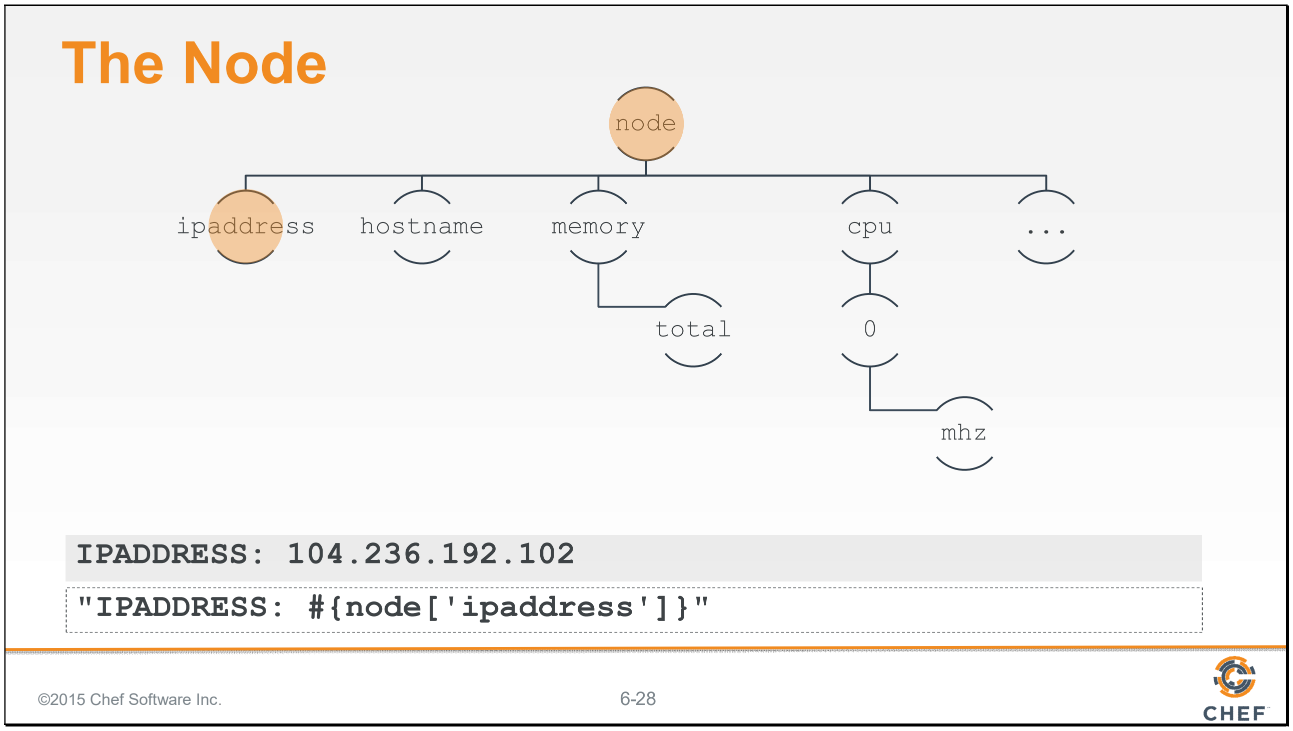
CHEF

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

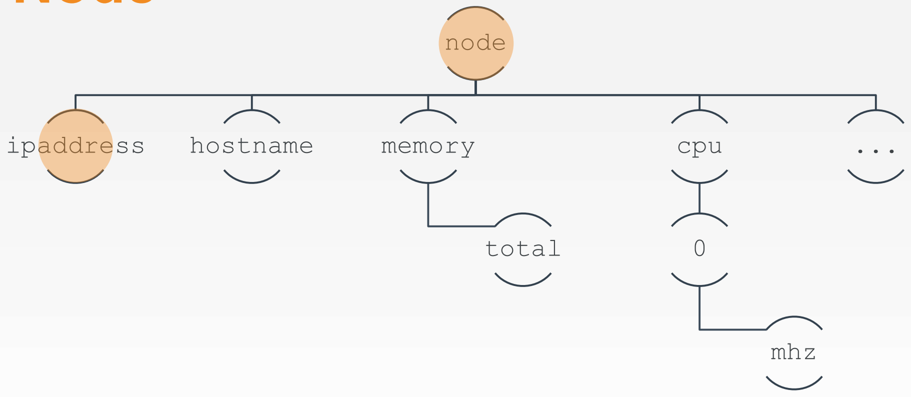
An attribute is a specific detail about a node, such as an IP address, a host name, a list of loaded kernel modules, the version(s) of available programming languages that are available, and so on.

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

Slide 28



The Node



```
IPADDRESS: 104.236.192.102
```

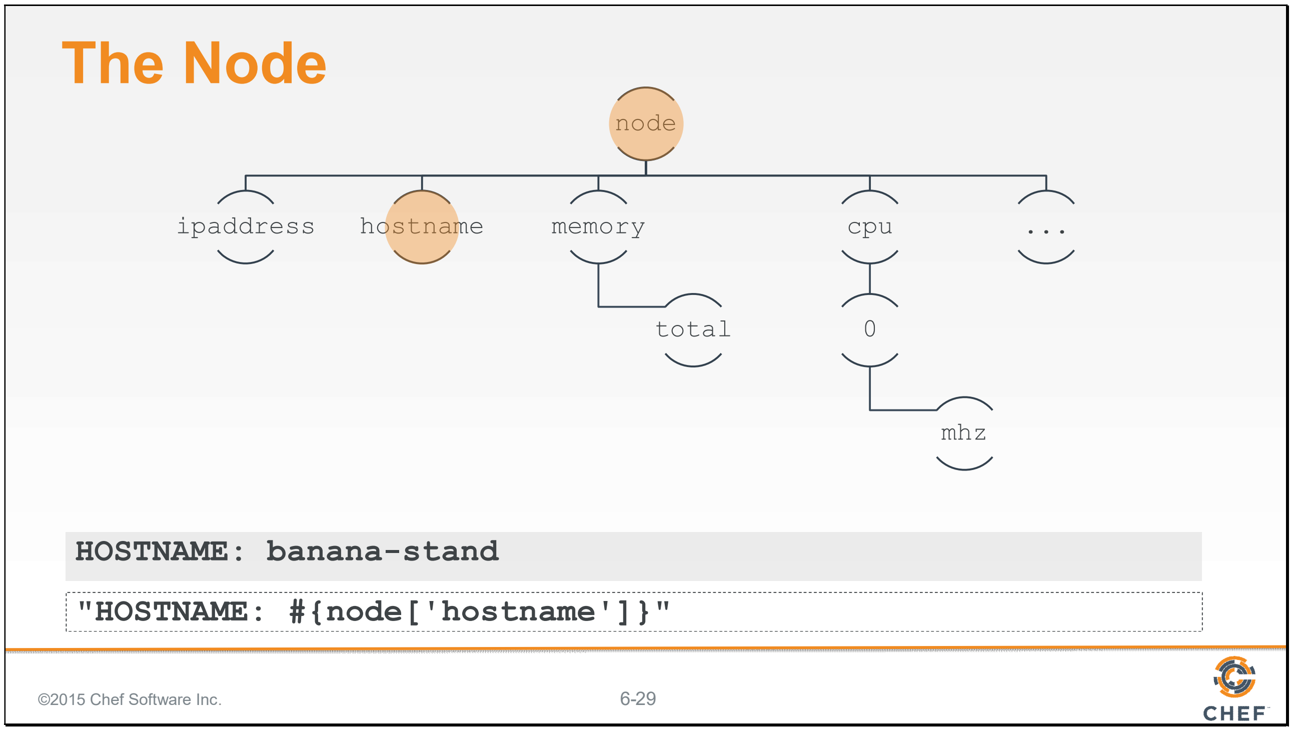
```
"IPADDRESS: #{node['ipaddress']}"
```

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

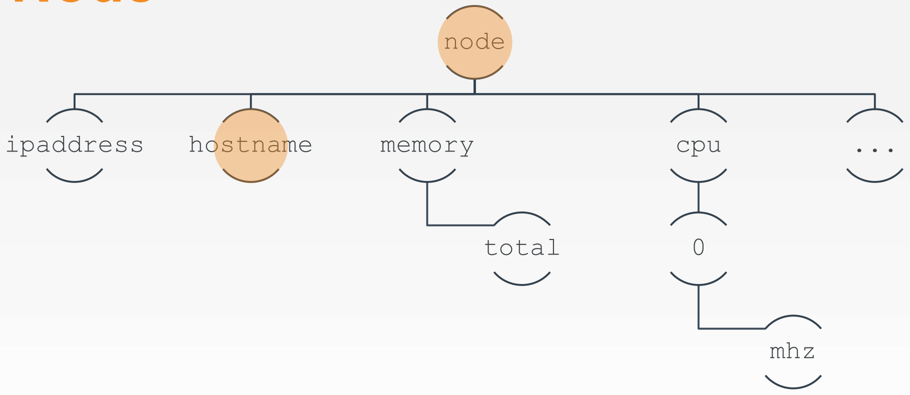
The shaded text near the bottom of this slide is the hard-coded value we currently have in the file resource's content attribute.

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

Slide 29



The Node

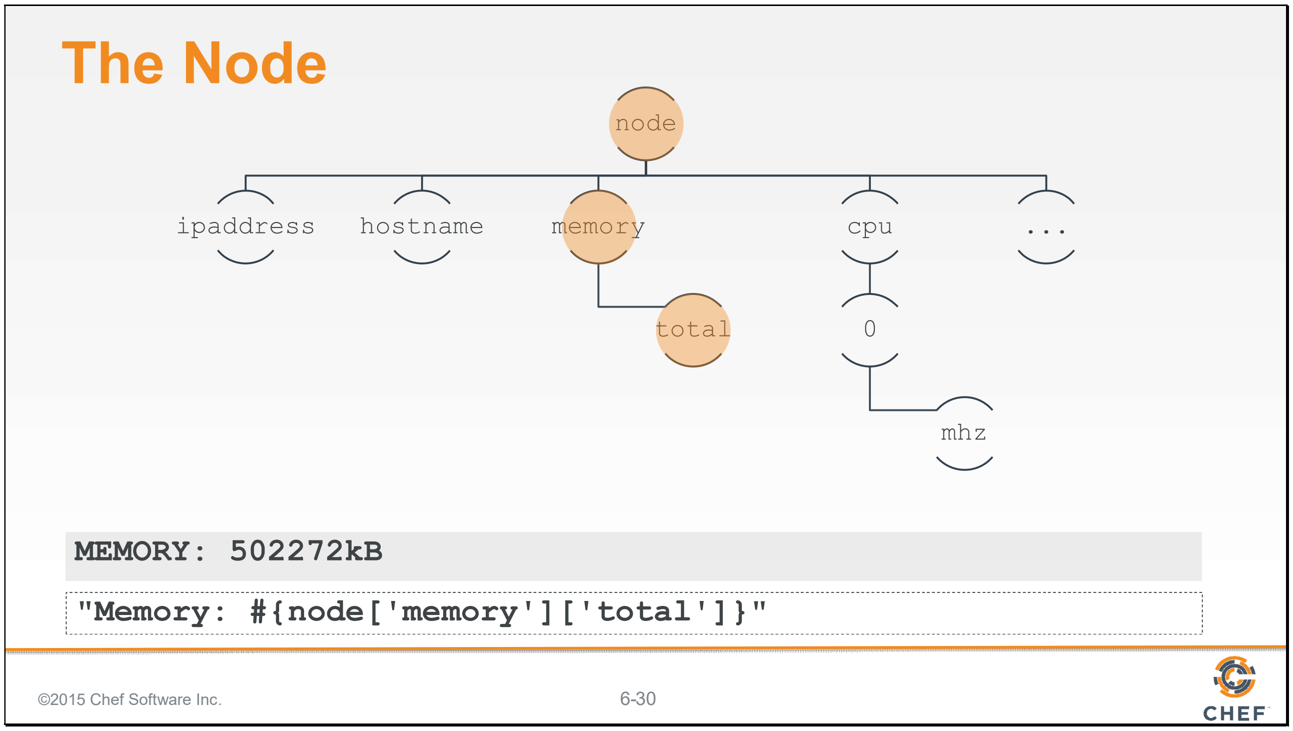


```
HOSTNAME: banana-stand
```

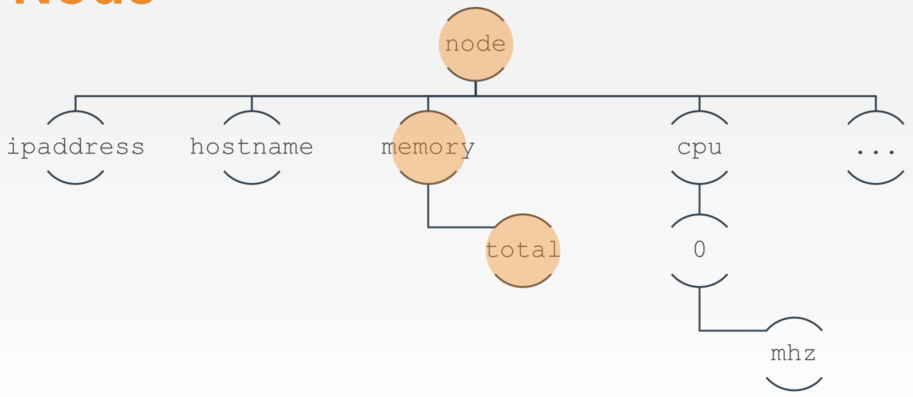
```
"HOSTNAME: #{node['hostname']}"
```

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

Slide 30



The Node



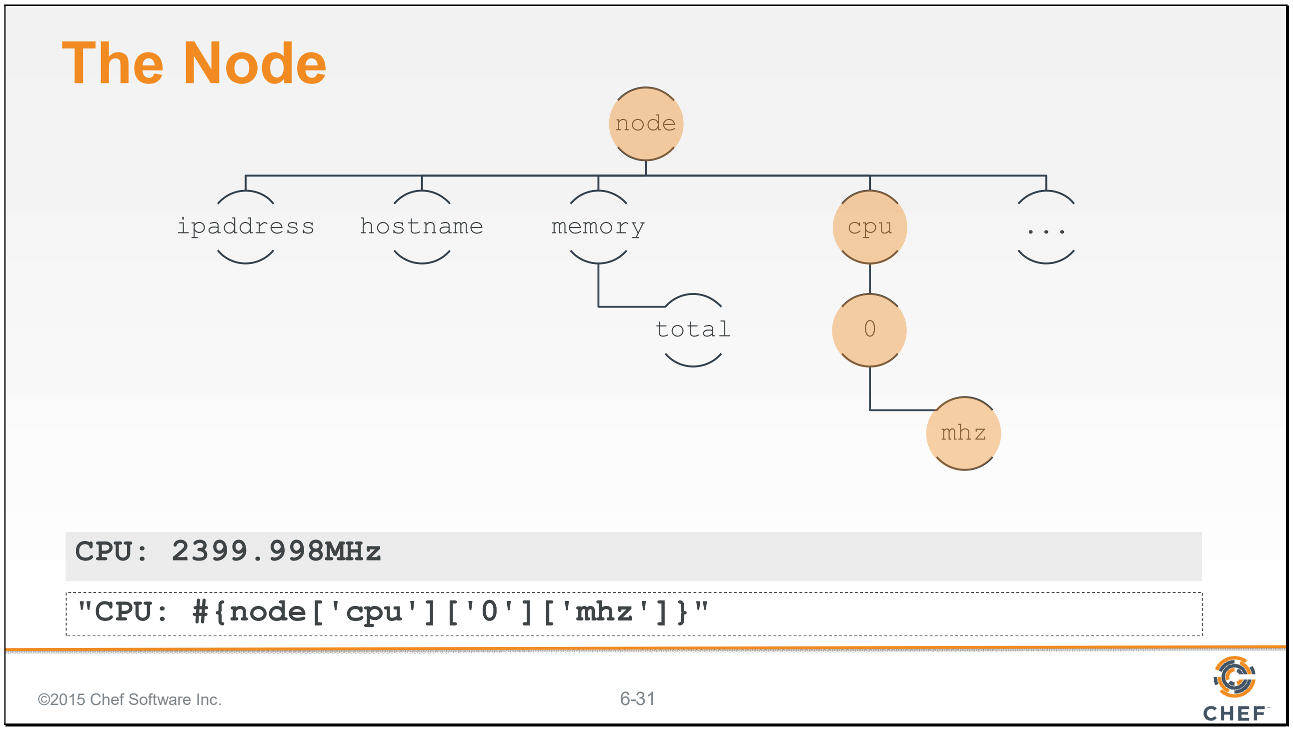
```
MEMORY: 502272kB
```

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

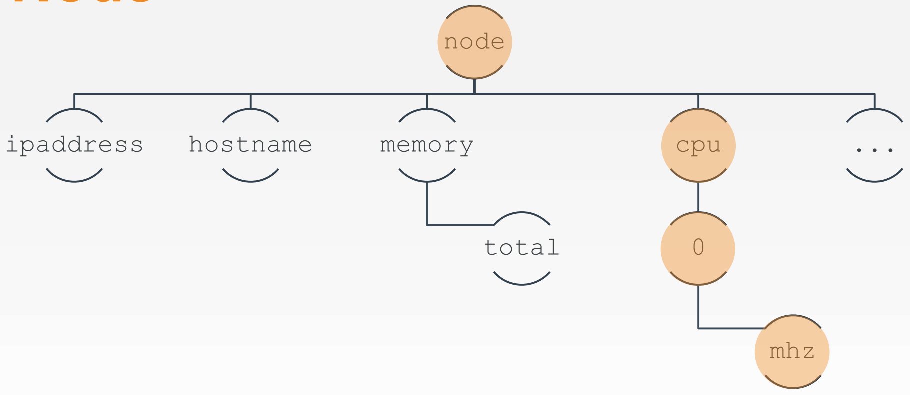
The node contains a top-level value memory which has a number of child elements. One of those child elements is the total amount of system memory.

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

Slide 31



The Node

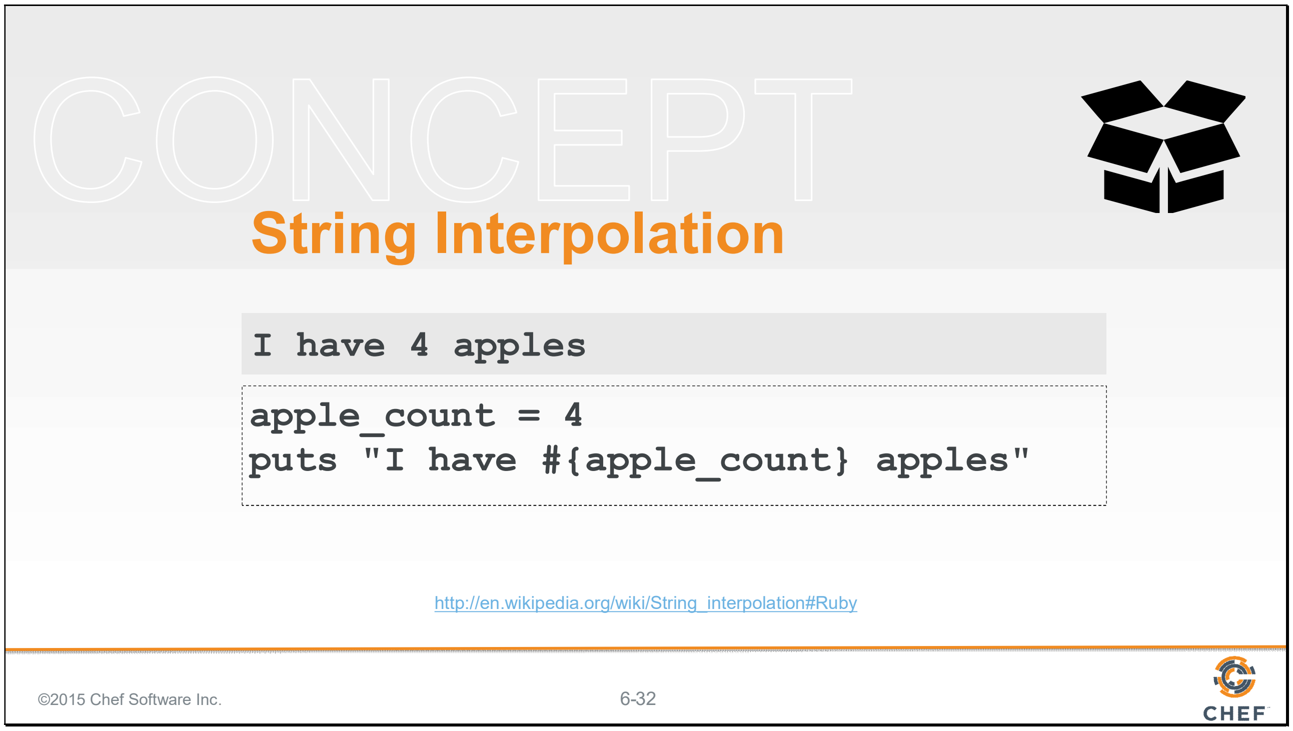


```
CPU: 2399.998MHz
```

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

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

Slide 32



CONCEPT

String Interpolation

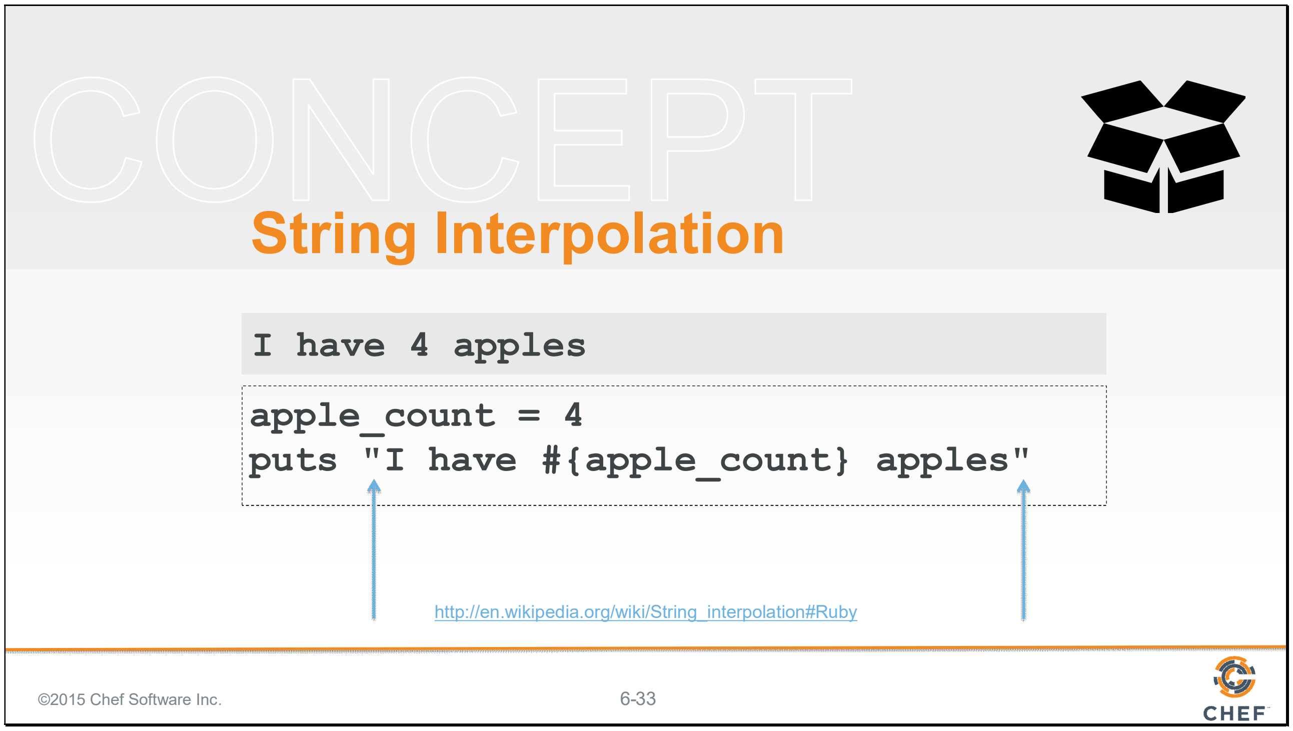
```
I have 4 apples
```

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

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

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

Slide 33



CONCEPT

String Interpolation

```
I have 4 apples
```

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

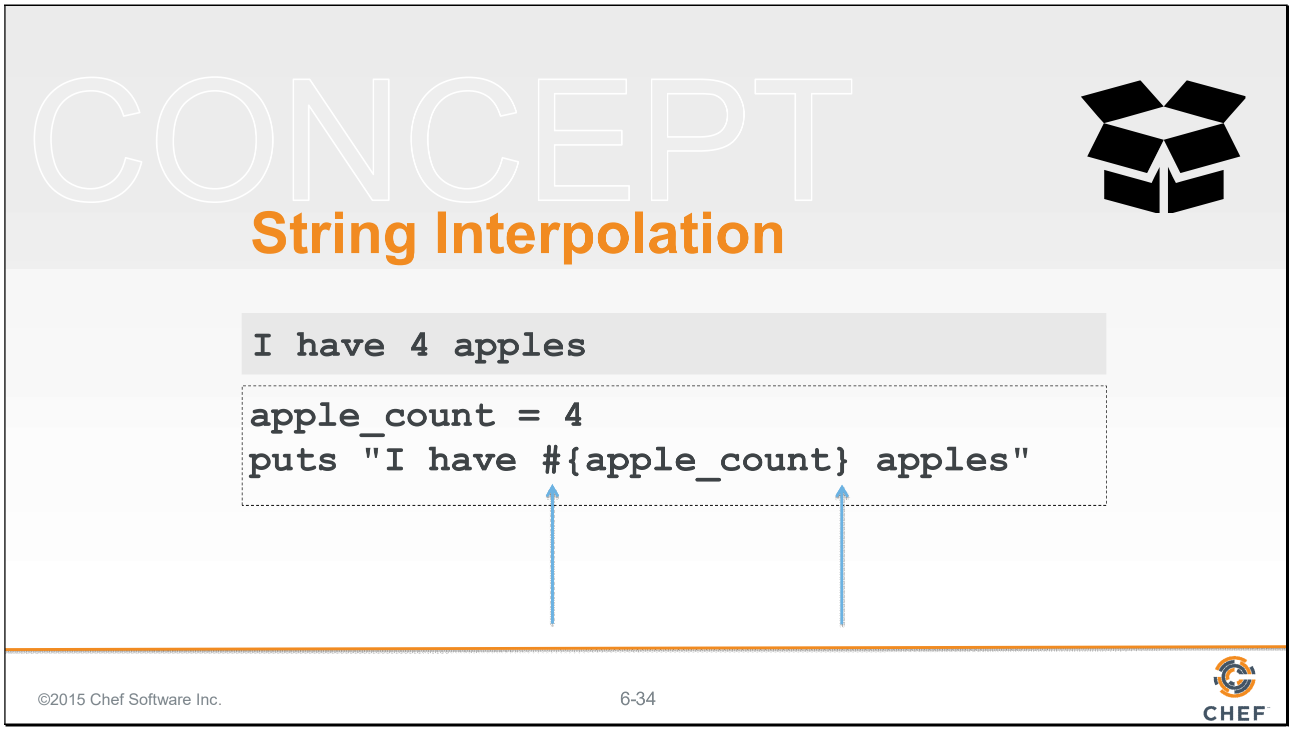
http://en.wikipedia.org/wiki/String_interpolation#Ruby

©2015 Chef Software Inc.

6-33

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

Slide 34


CONCEPT

String Interpolation

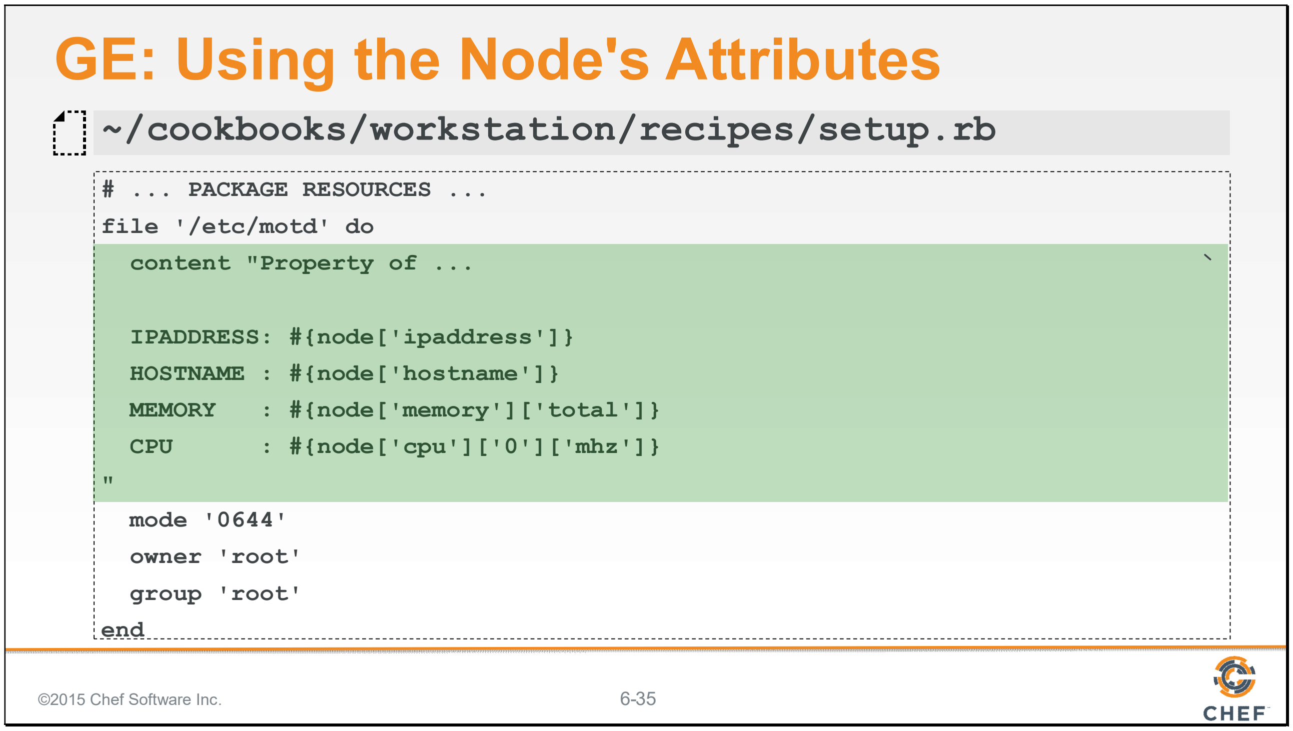
```
I have 4 apples
```

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

CHEF

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

Slide 35



GE: Using the Node's Attributes

~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: #{node['ipaddress']}
  HOSTNAME : #{node['hostname']}
  MEMORY   : #{node['memory']['total']}
  CPU      : #{node['cpu']['0']['mhz']}
"
  mode '0644'
  owner 'root'
  group 'root'
end
```

In this group exercise, instead of using hard-coded values, use string interpolation within the file resource's content attribute to allow the system to access the node object's attribute for:

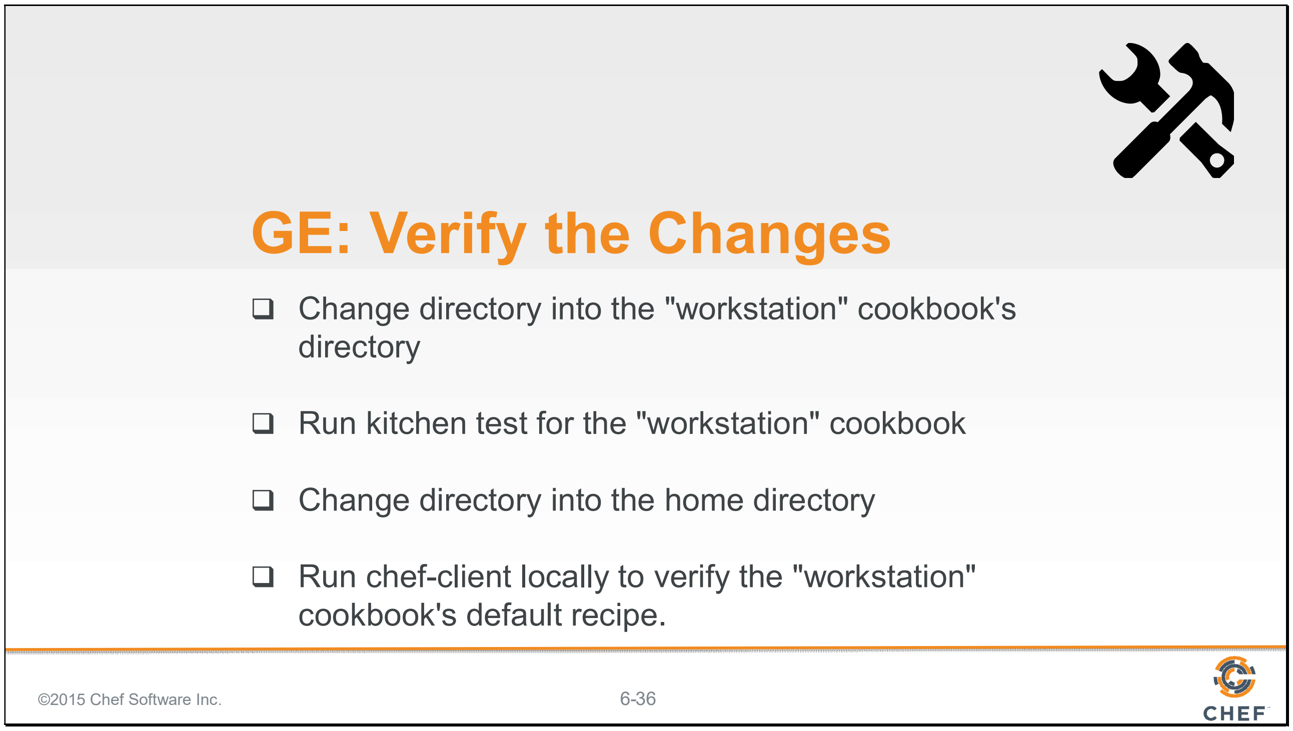
IP address

Hostname

Total memory

Megahertz of the first CPU.

Slide 36



GE: Verify the Changes

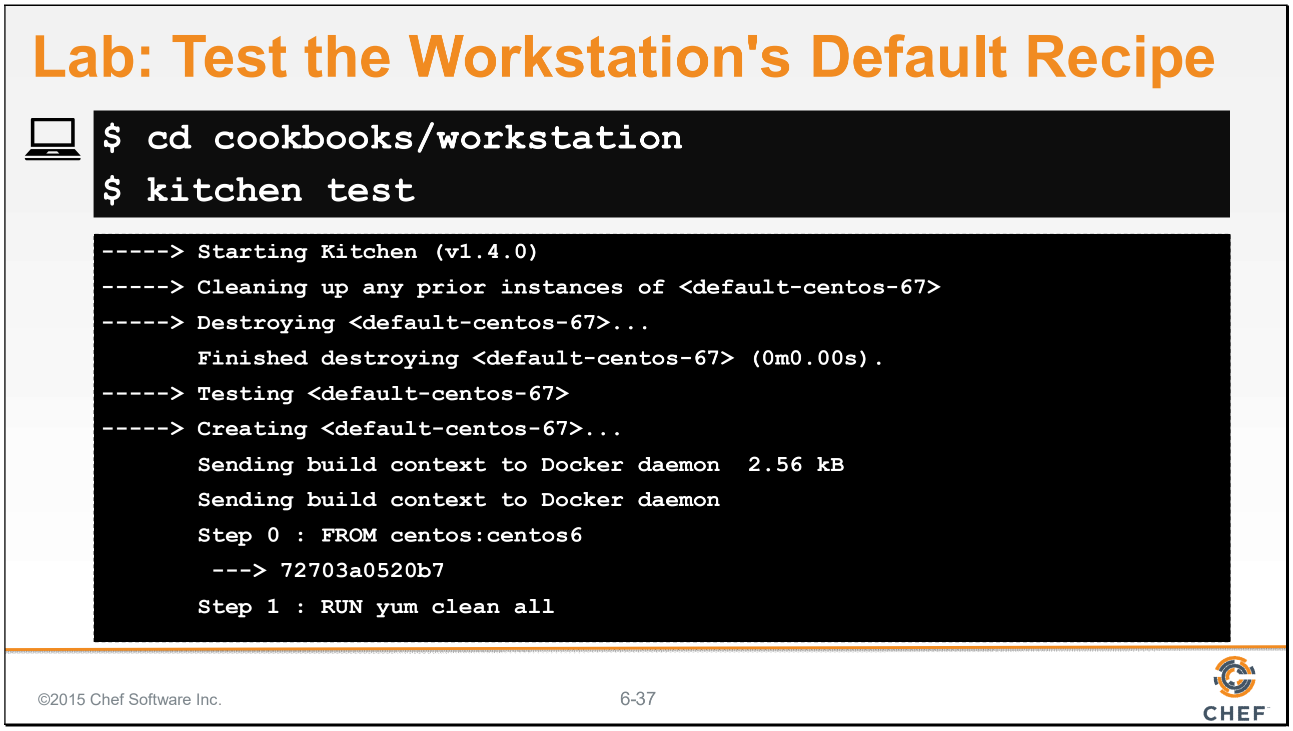
- ❑ Change directory into the "workstation" cookbook's directory
- ❑ Run kitchen test for the "workstation" cookbook
- ❑ Change directory into the home directory
- ❑ Run chef-client locally to verify the "workstation" cookbook's default recipe.

Again we have created a change.

Move into the workstation cookbook's directory. Verify the changes we made to the workstation cookbook's default recipe with kitchen. Return to the home directory. Use 'chef-client' to locally apply the workstation cookbook's default recipe.

Instructor Note: Allow 8 minutes to complete this exercise.

Slide 37



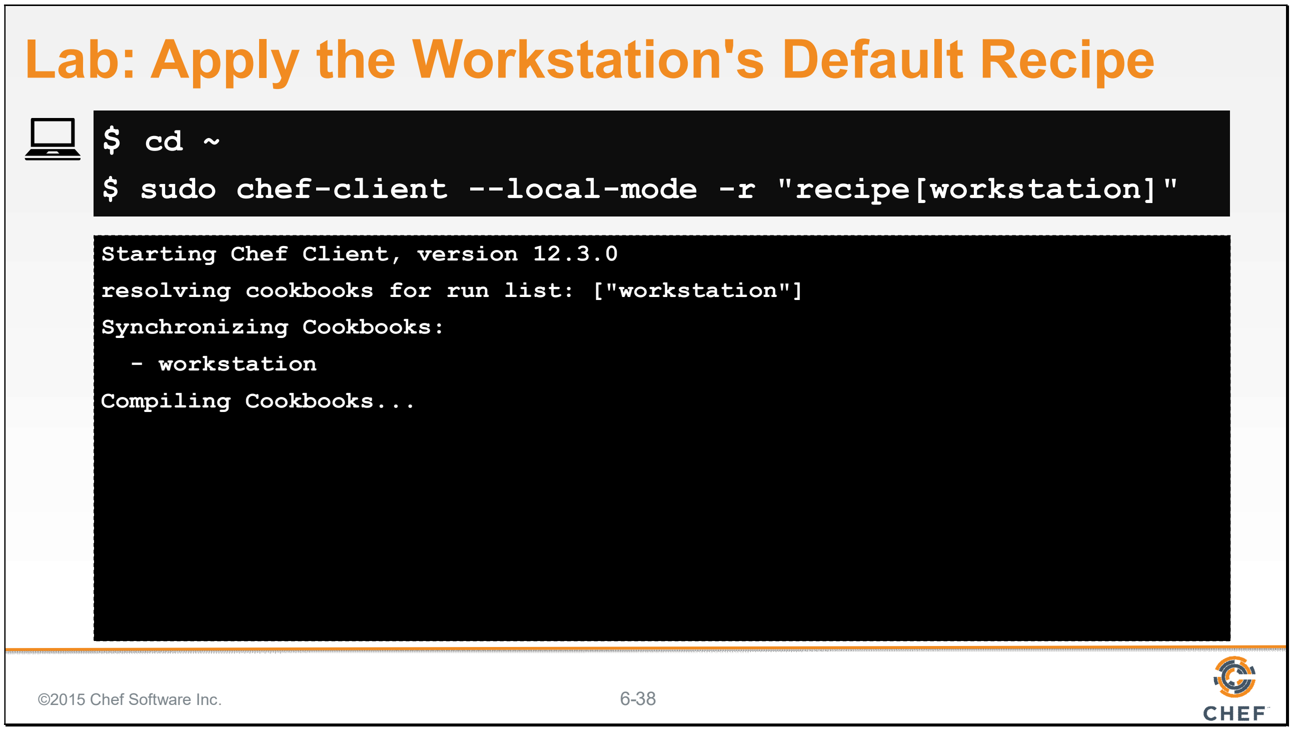
Lab: Test the Workstation's Default Recipe

```
$ cd cookbooks/workstation
$ kitchen test
```

```
-----> Starting Kitchen (v1.4.0)
-----> Cleaning up any prior instances of <default-centos-67>
-----> Destroying <default-centos-67>...
       Finished destroying <default-centos-67> (0m0.00s).
-----> Testing <default-centos-67>
-----> Creating <default-centos-67>...
       Sending build context to Docker daemon  2.56 kB
       Sending build context to Docker daemon
       Step 0 : FROM centos:centos6
        ---> 72703a0520b7
       Step 1 : RUN yum clean all
```

Change into the workstation cookbook's directory and then run `kitchen test` to verify that the changes we introduced did not cause a regression.

Slide 38



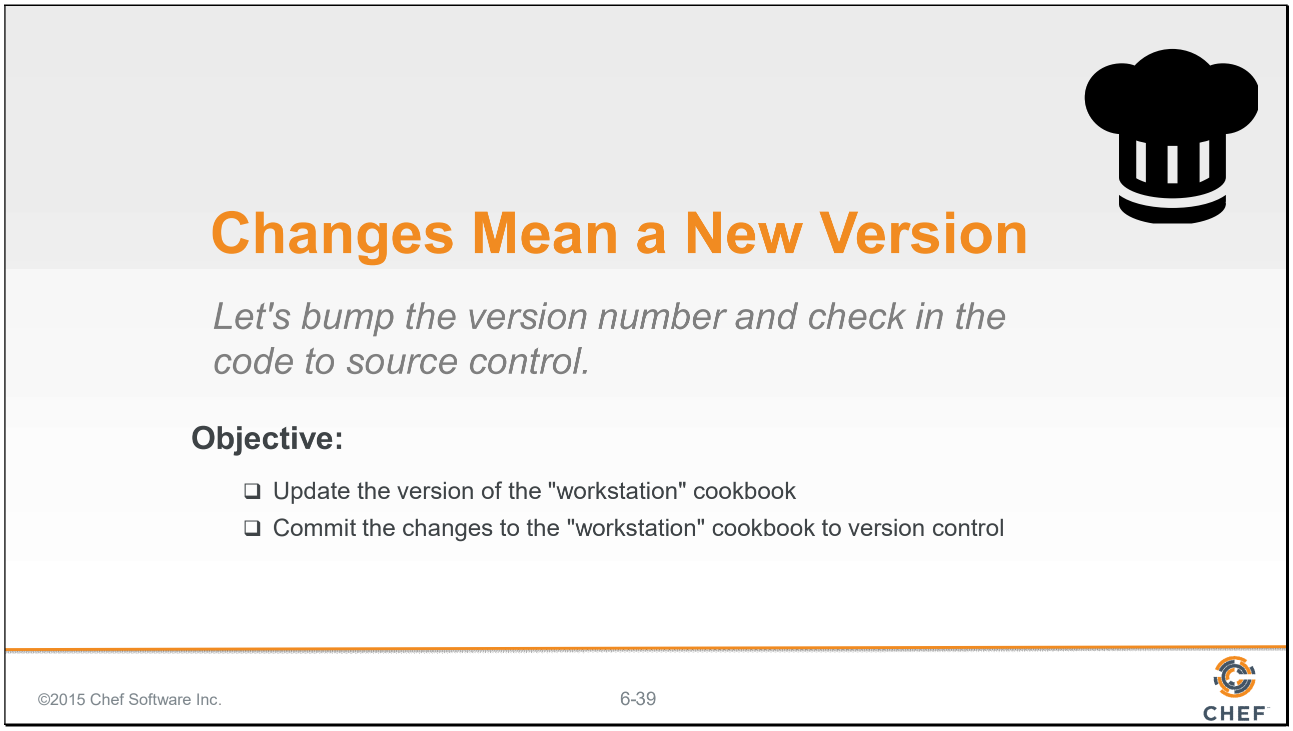
Lab: Apply the Workstation's Default Recipe

```
$ cd ~
$ sudo chef-client --local-mode -r "recipe[workstation]"
```

```
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
```

If everything passes and you feel confident that it will also work on the current workstation, change to the home directory and then run `chef-client` to apply the apache cookbook locally to the system.

Slide 39



Changes Mean a New Version

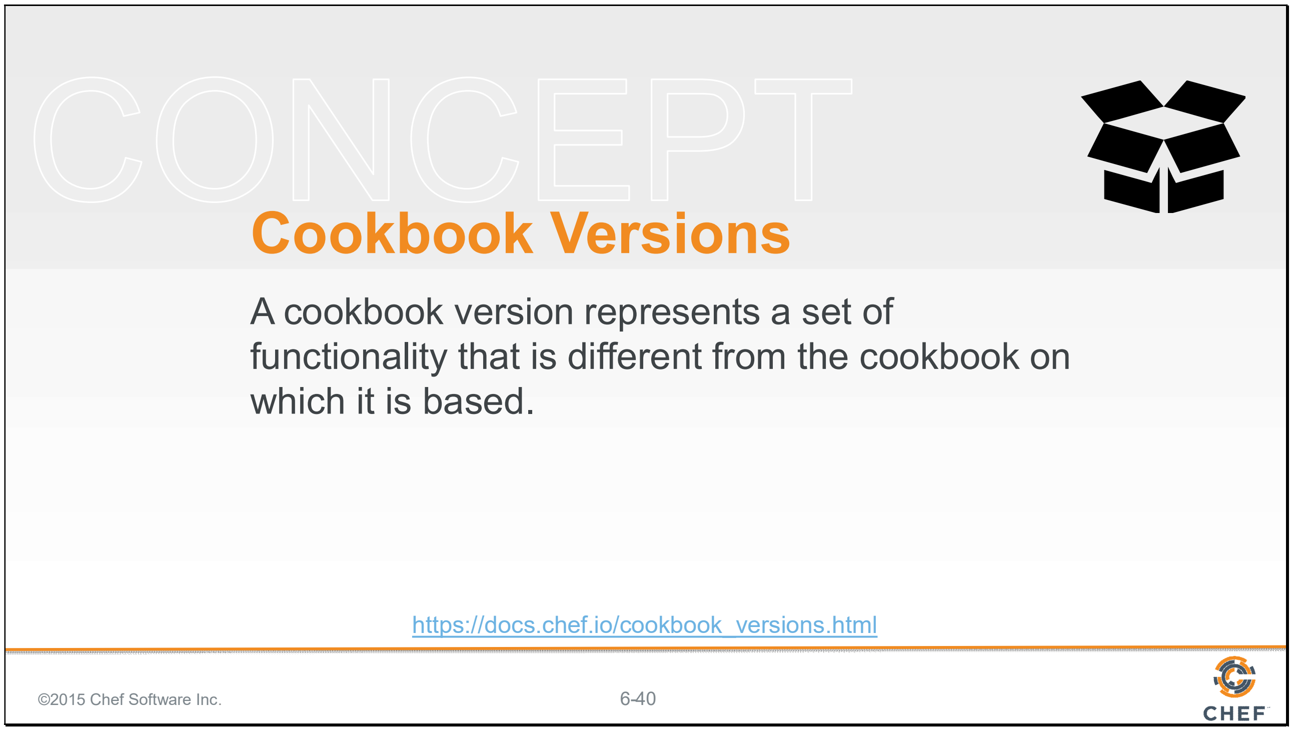
Let's bump the version number and check in the code to source control.

Objective:

- ❑ Update the version of the "workstation" cookbook
- ❑ Commit the changes to the "workstation" cookbook to version control

Now that we've made these significant changes and verified that they work, its time we bumped the version of the cookbook and commit the changes.

Slide 40



CONCEPT

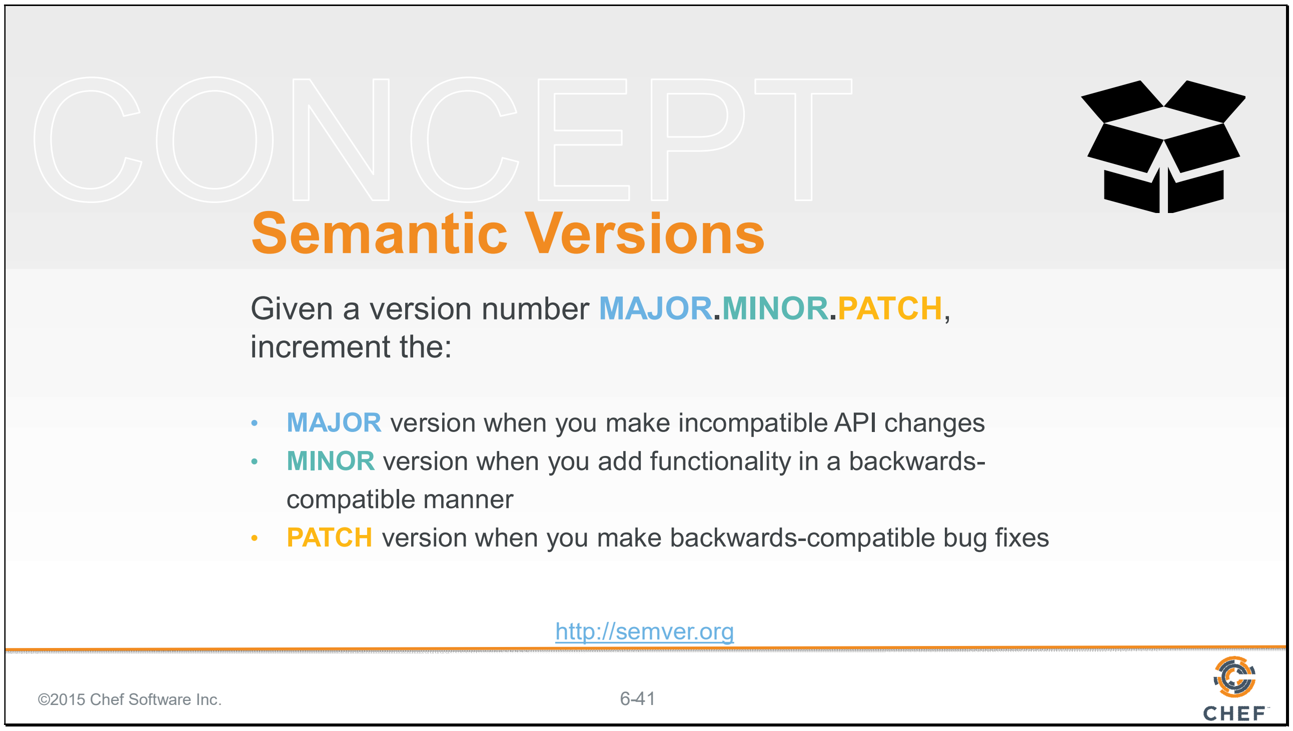
Cookbook Versions

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

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

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



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

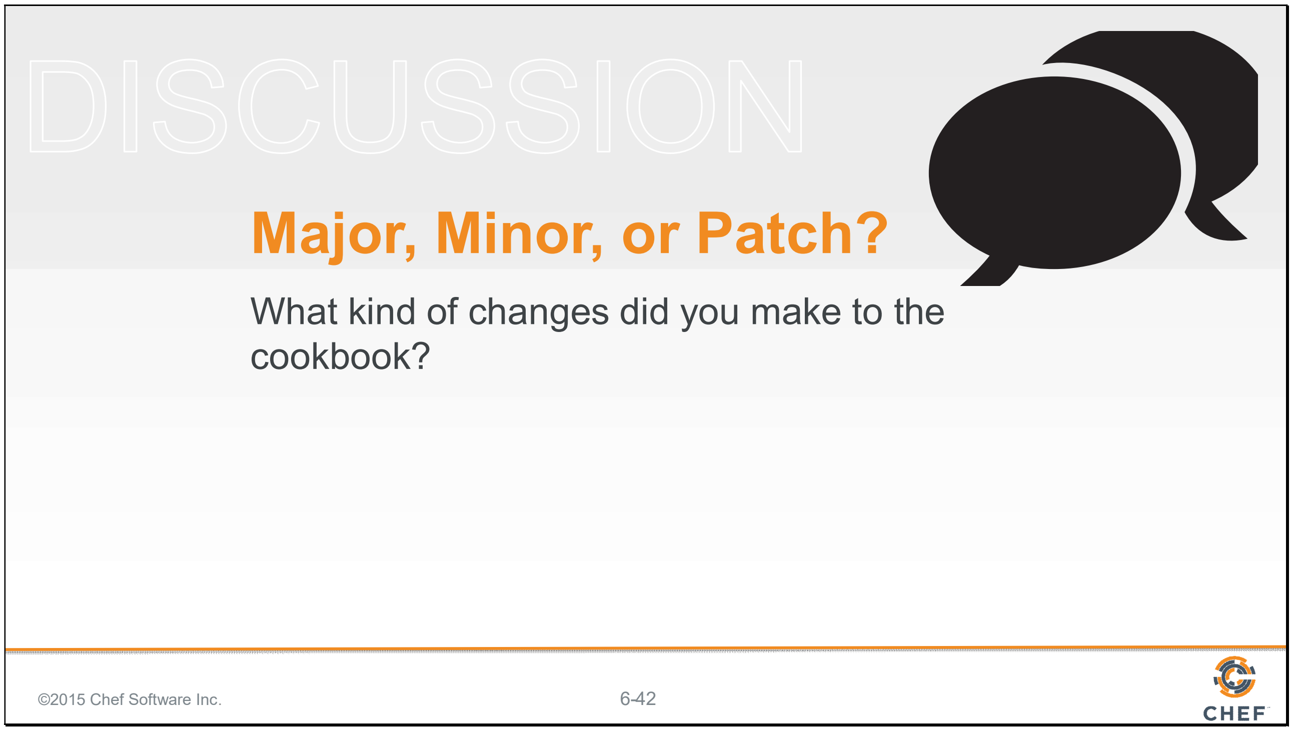
©2015 Chef Software Inc. 6-41

CHEF

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

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

Slide 42



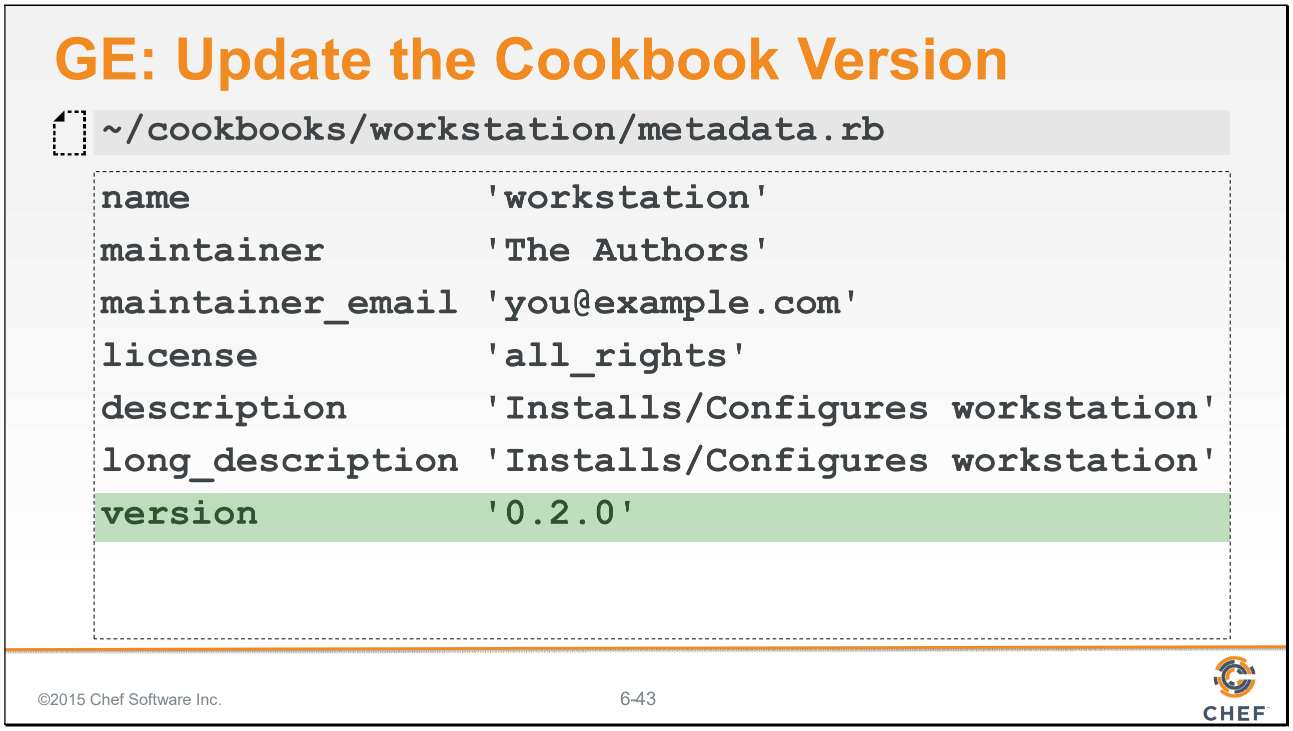
DISCUSSION

Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

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

Slide 43



GE: Update the Cookbook Version

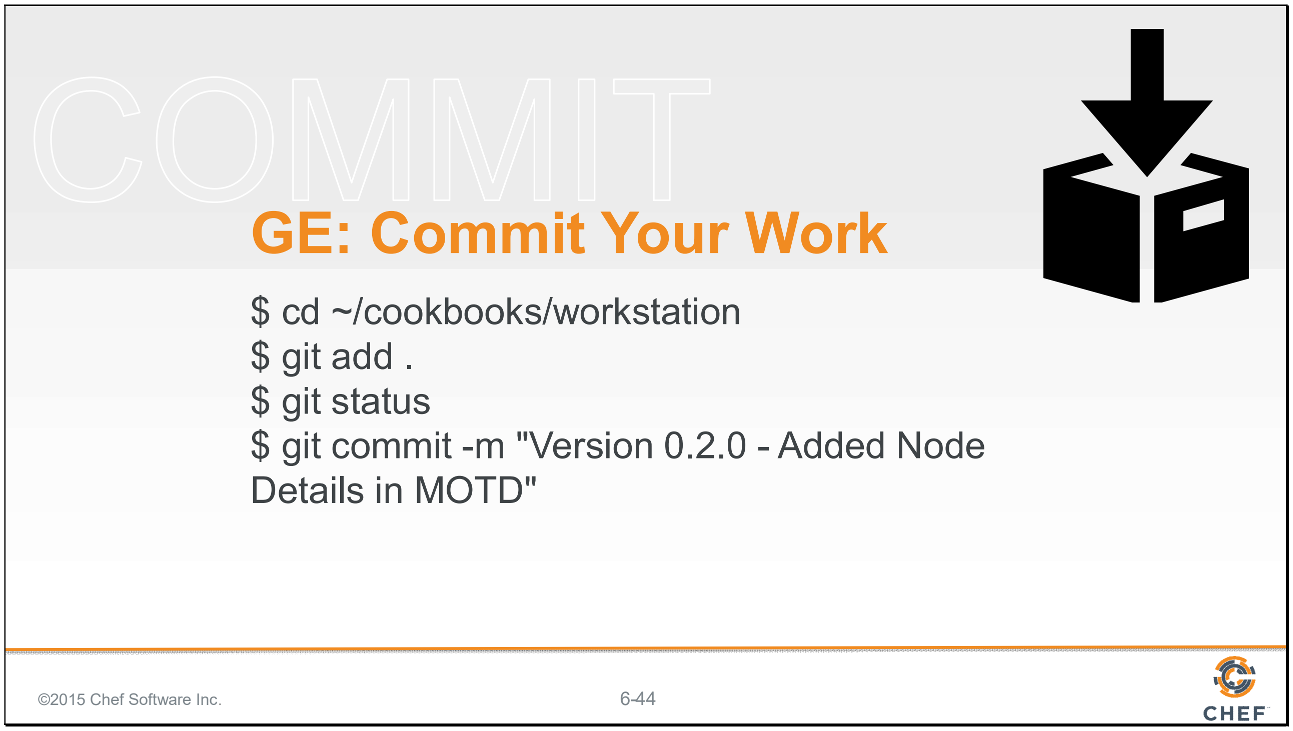
~/cookbooks/workstation/metadata.rb

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

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

Edit ~/cookbooks/workstation/metadata.rb and update the version's minor number to 0.2.0.

Slide 44



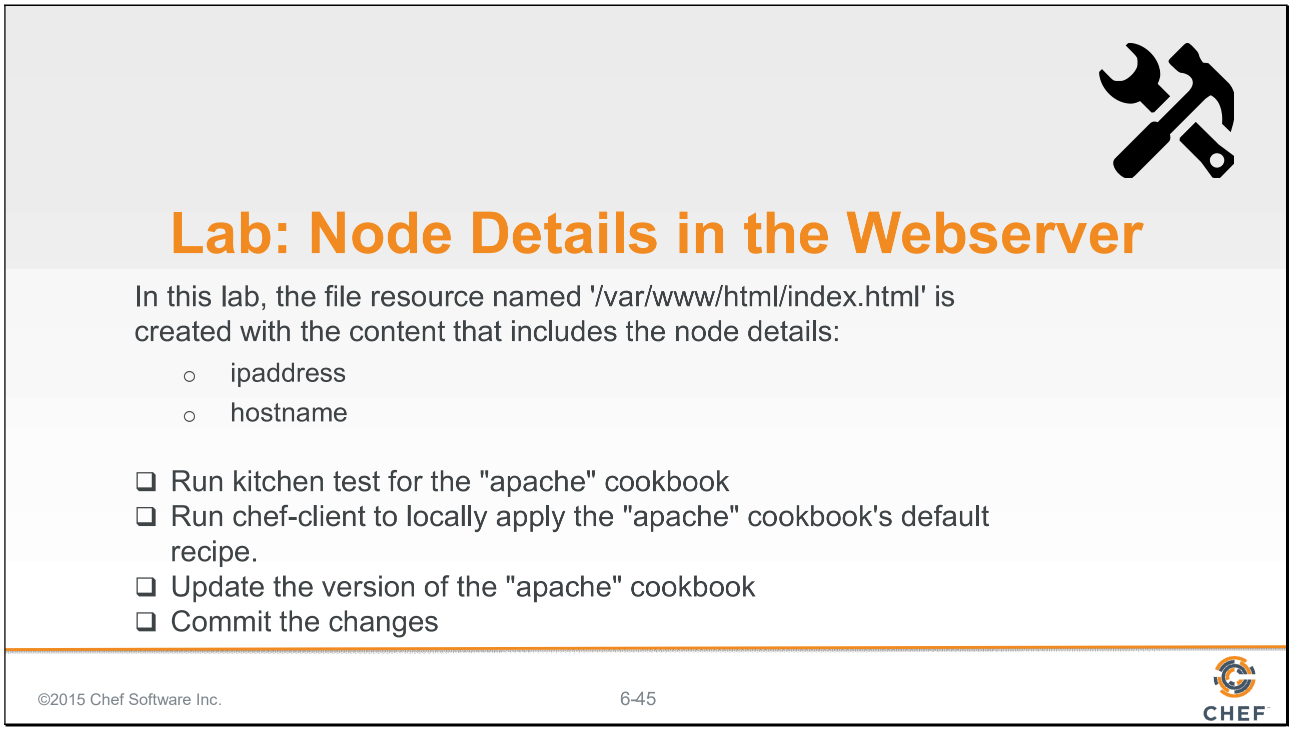
COMMIT

GE: Commit Your Work

```
$ cd ~/cookbooks/workstation
$ git add .
$ git status
$ git commit -m "Version 0.2.0 - Added Node
Details in MOTD"
```

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

Slide 45



Lab: Node Details in the Webserver

In this lab, the file resource named '/var/www/html/index.html' is created with the content that includes the node details:

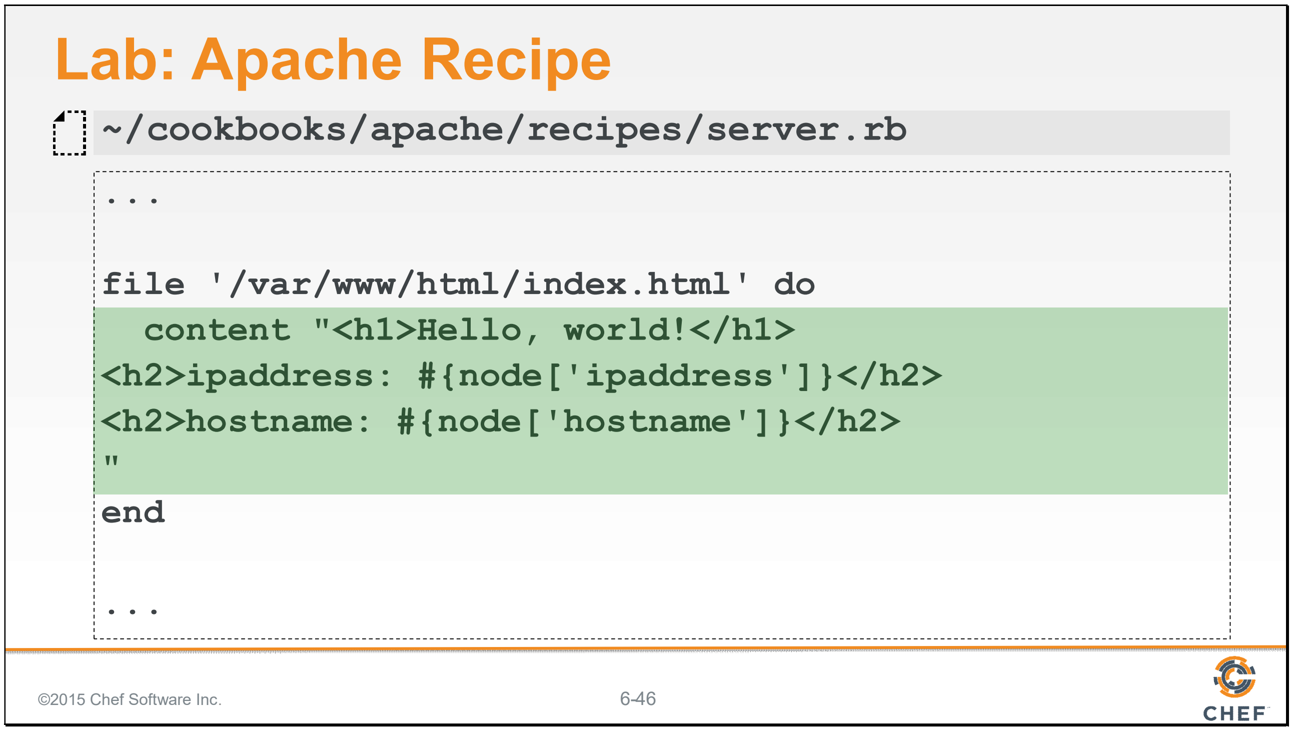
- ipaddress
- hostname

❑ Run kitchen test for the "apache" cookbook
❑ Run chef-client to locally apply the "apache" cookbook's default recipe.
❑ Update the version of the "apache" cookbook
❑ Commit the changes

Now it's time to add similar functionality to the apache cookbook. You should try to follow the high-level steps in this slide to complete this lab.

Instructor Note: Allow 15 minutes to complete this exercise.

Slide 46



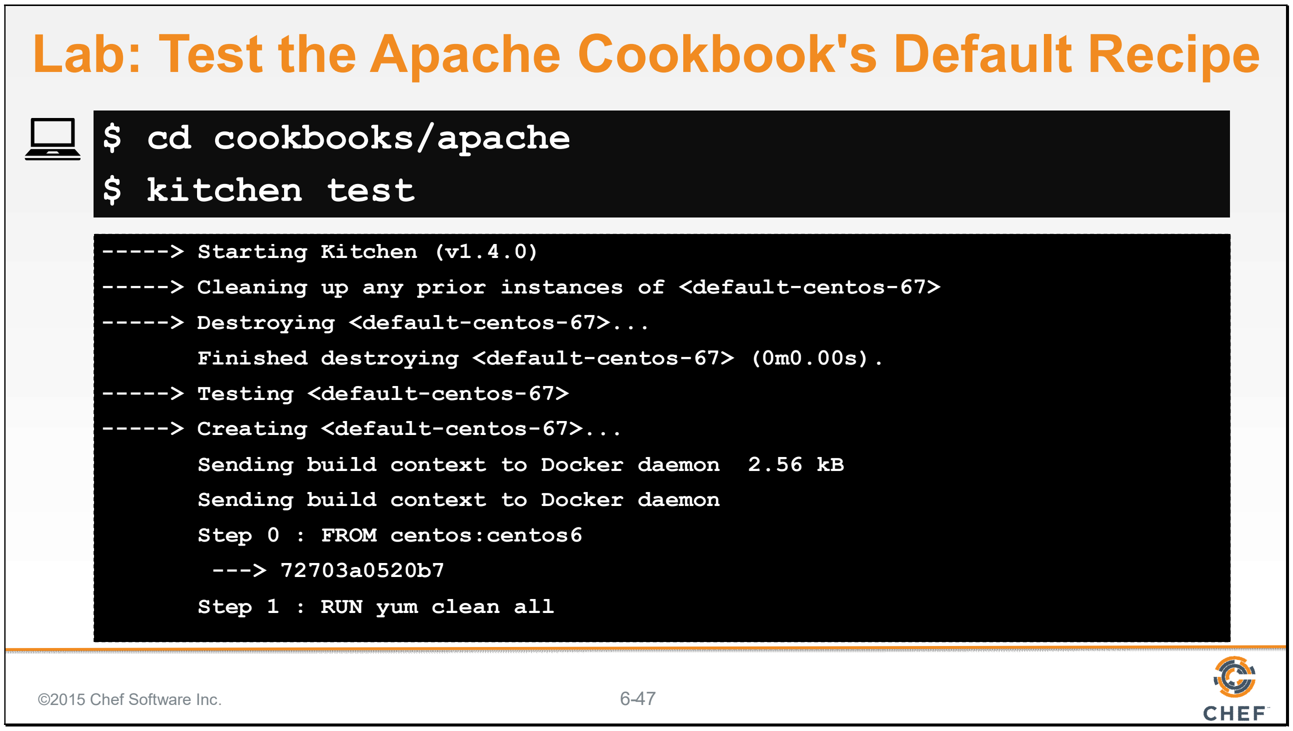
Lab: Apache Recipe

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

```
...

file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>
<h2>ipaddress: #{node['ipaddress']}</h2>
<h2>hostname: #{node['hostname']}</h2>
"
end

...
```

Update the file resource, named '/var/www/html/index.html, to be created with the content that includes the node's IP address and its host name.

Slide 47



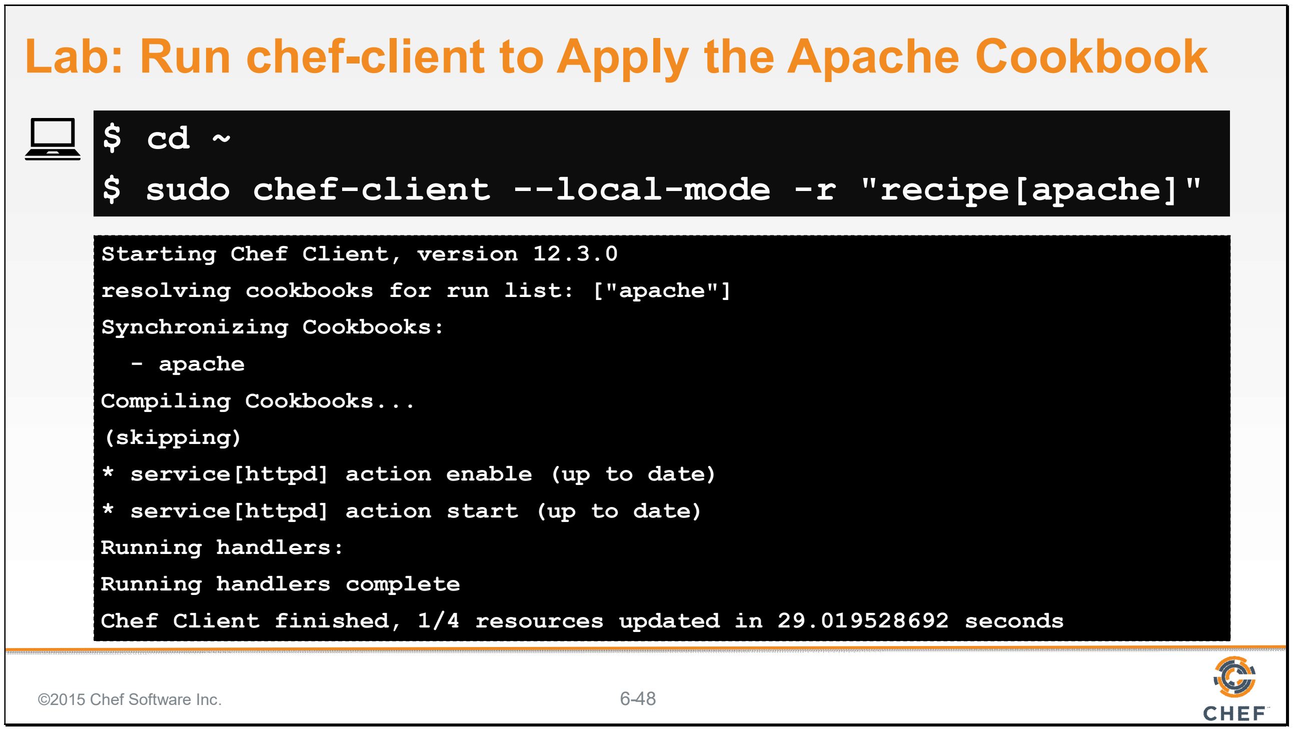
Lab: Test the Apache Cookbook's Default Recipe

```
$ cd cookbooks/apache
$ kitchen test
```

```
-----> Starting Kitchen (v1.4.0)
-----> Cleaning up any prior instances of <default-centos-67>
-----> Destroying <default-centos-67>...
       Finished destroying <default-centos-67> (0m0.00s).
-----> Testing <default-centos-67>
-----> Creating <default-centos-67>...
       Sending build context to Docker daemon  2.56 kB
       Sending build context to Docker daemon
       Step 0 : FROM centos:centos6
        ---> 72703a0520b7
       Step 1 : RUN yum clean all
```

Change into the apache cookbook's directory and then run `kitchen test` to verify that the changes we introduced did not cause a regression.

Slide 48



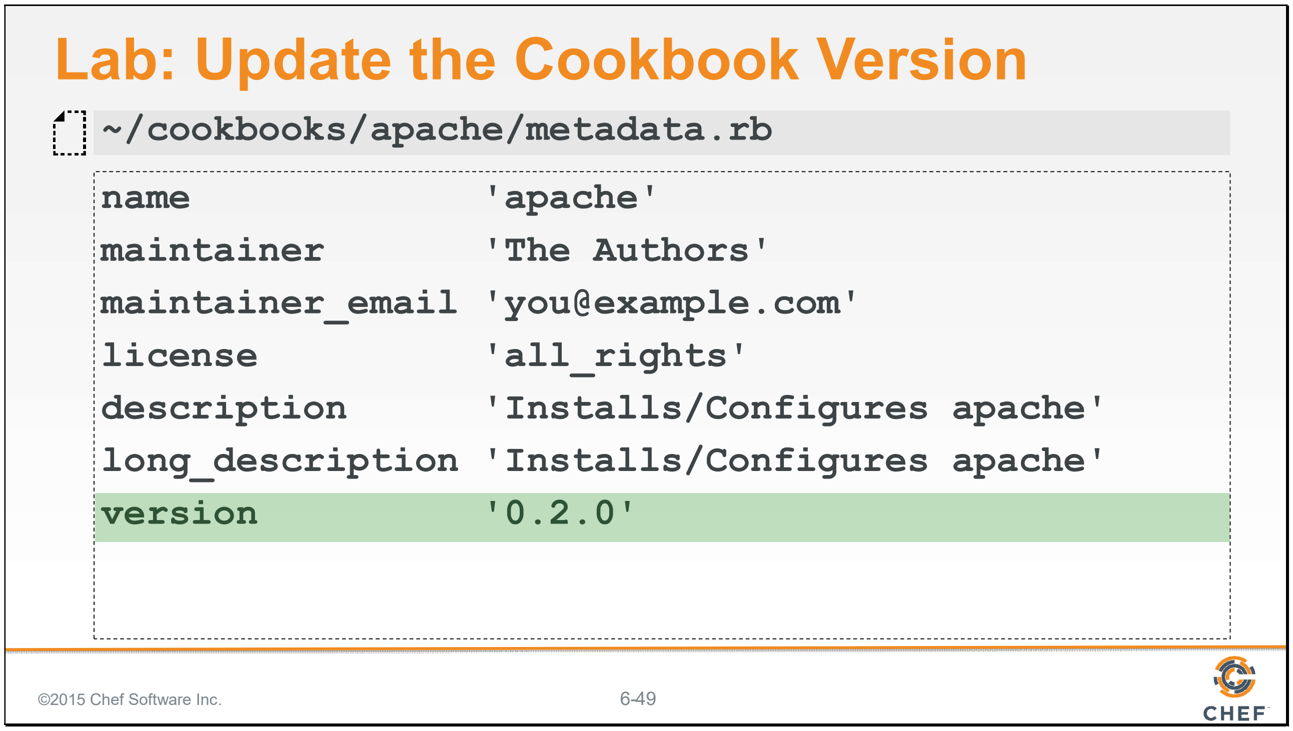
Lab: Run chef-client to Apply the Apache Cookbook

```
$ cd ~
$ sudo chef-client --local-mode -r "recipe[apache]"
```

```
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
(skipping)
* service[httpd] action enable (up to date)
* service[httpd] action start (up to date)
Running handlers:
Running handlers complete
Chef Client finished, 1/4 resources updated in 29.019528692 seconds
```

If everything passes and you feel confident that it will also work on the current workstation, change to the home directory and then run `chef-client` to apply the apache cookbook locally to the system.

Slide 49



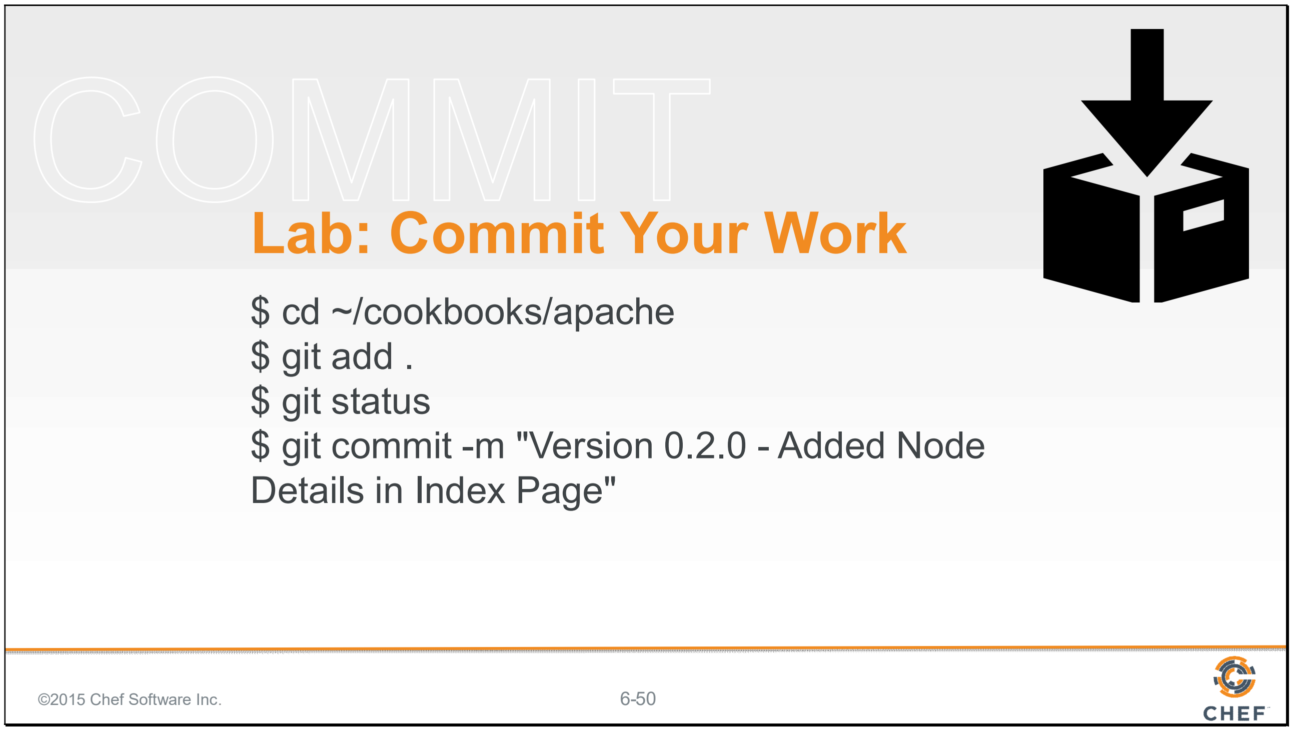
Lab: Update the Cookbook Version

~/cookbooks/apache/metadata.rb

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

Showing these two attributes in the index html page seems very similar to the feature we added for the workstation cookbook. So update the version of the apache cookbook to 0.2.0 as well.

Slide 50

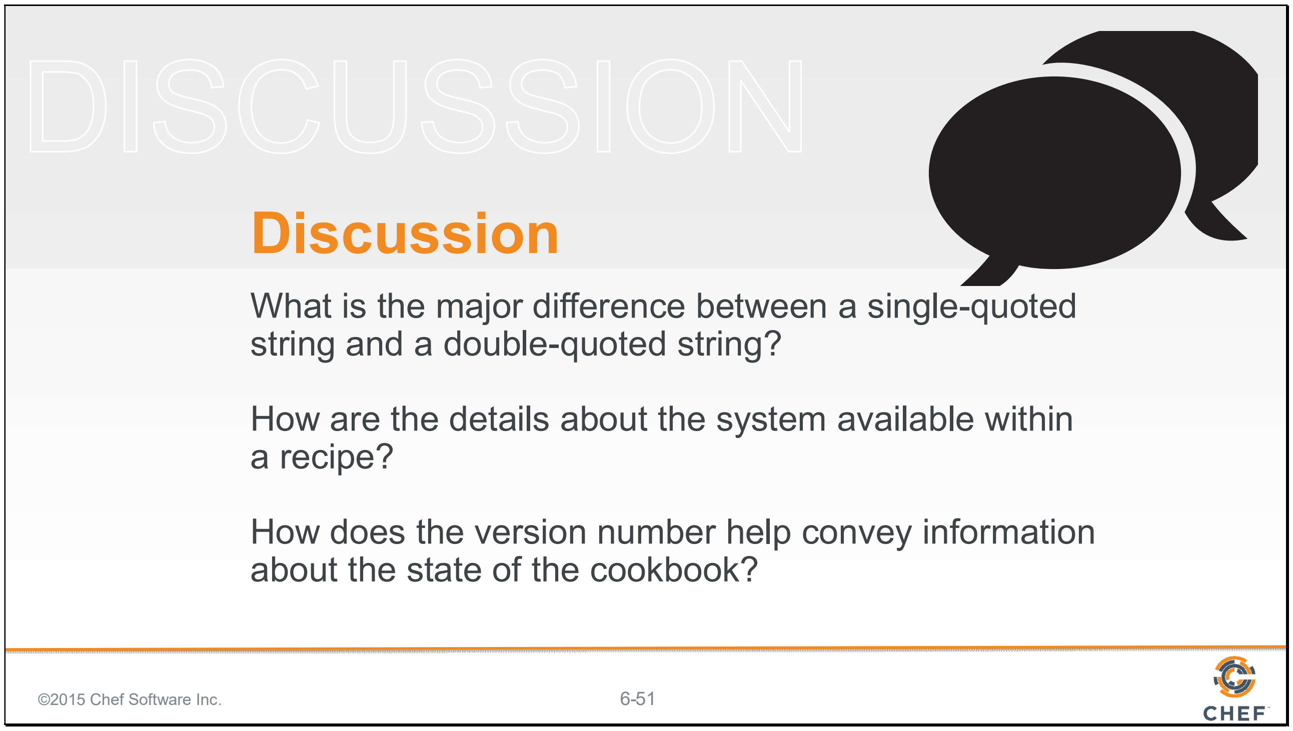


COMMIT

Lab: Commit Your Work

```
$ cd ~/cookbooks/apache
$ git add .
$ git status
$ git commit -m "Version 0.2.0 - Added Node
Details in Index Page"
```

And finally, commit your changes to git.



DISCUSSION

Discussion

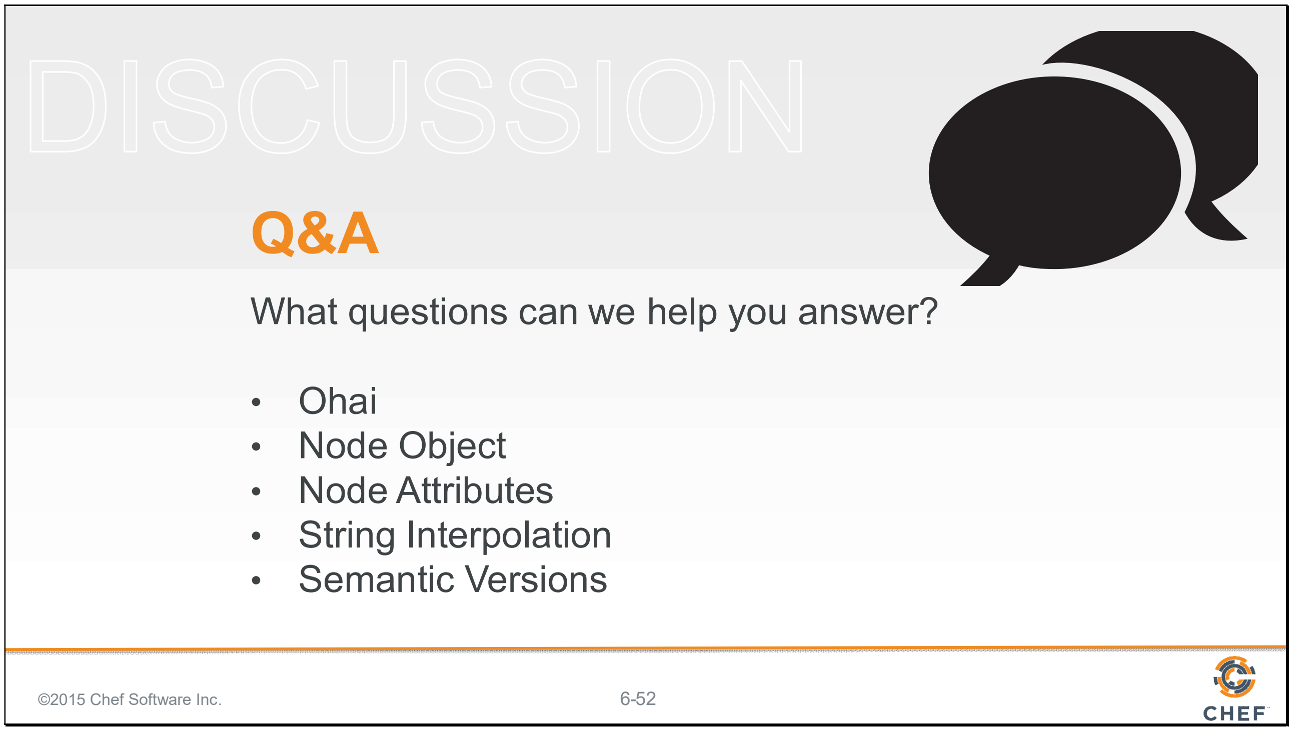
What is the major difference between a single-quoted string and a double-quoted string?

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

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

Answer these questions.

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



DISCUSSION

Q&A

What questions can we help you answer?

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

With that we have added all of the requested features.

What questions can we help you answer?

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

Slide 53

CHEF™