6: Details About the System

Slide 2

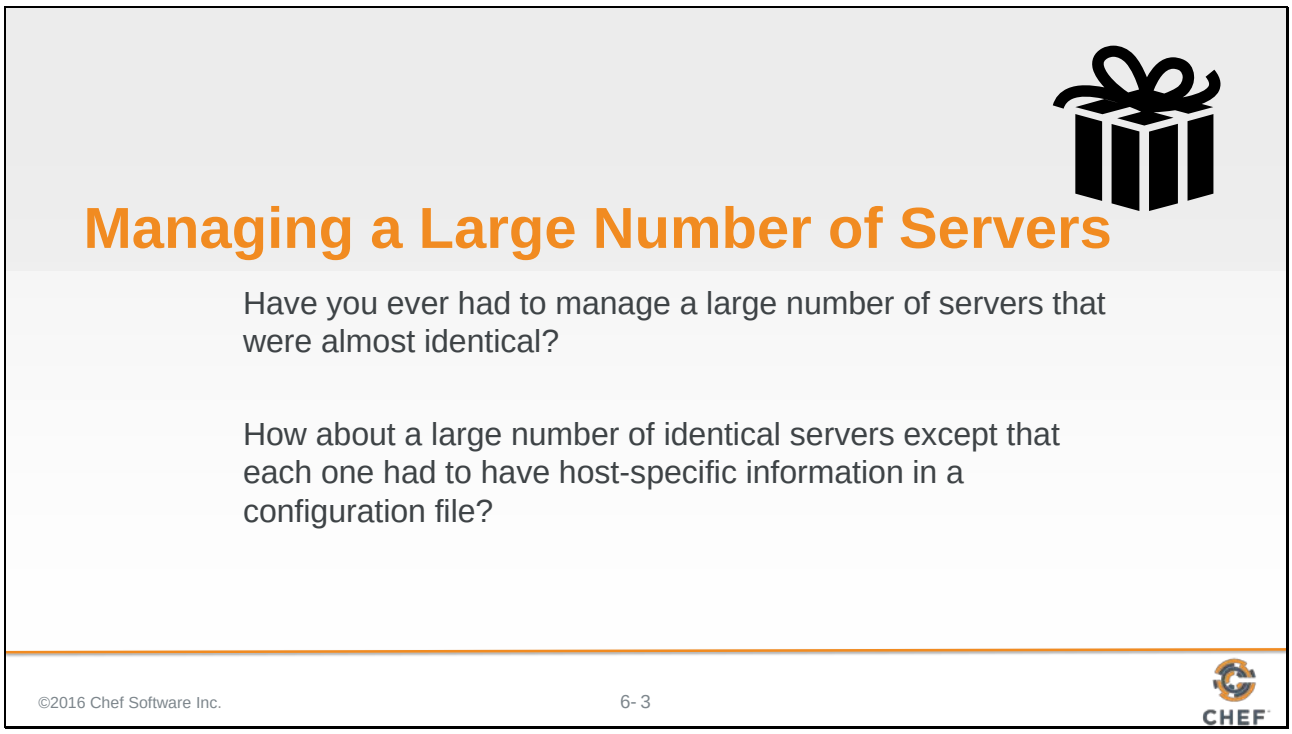
Objectives

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

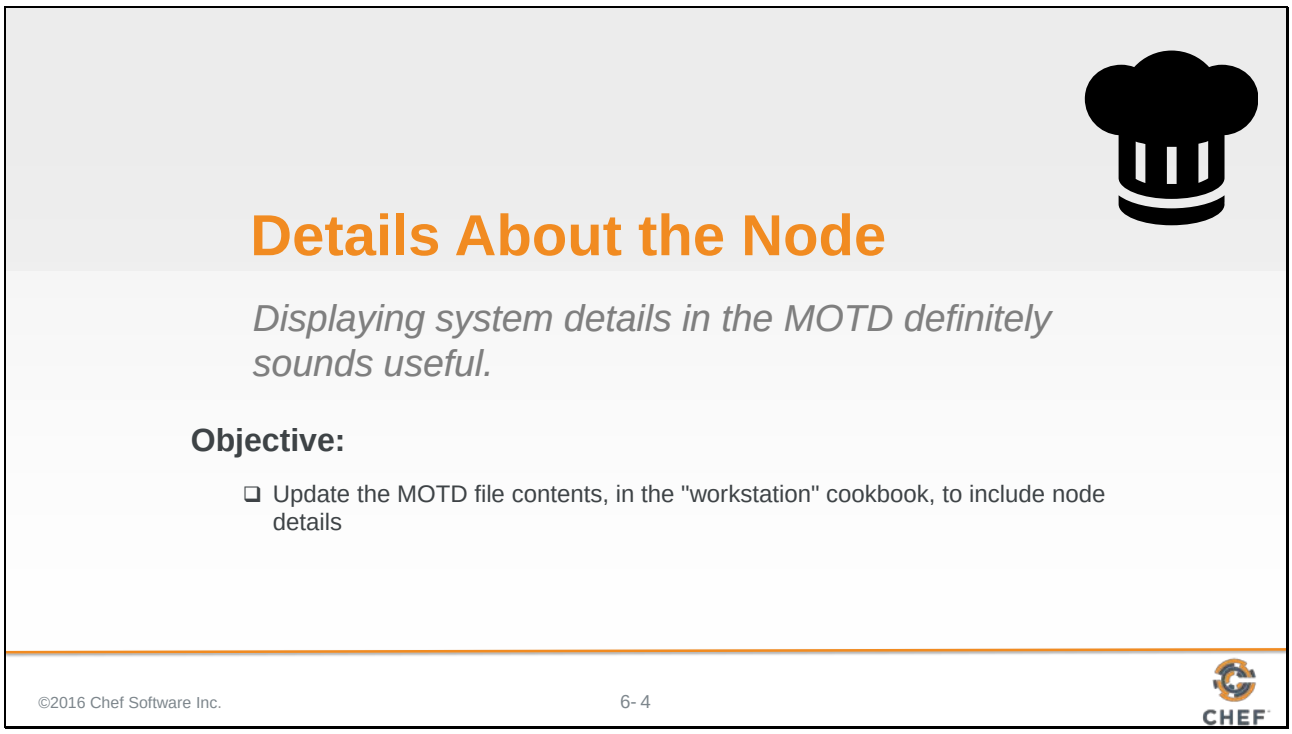


Have you ever had to manage a large number of servers that were almost identical?

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.

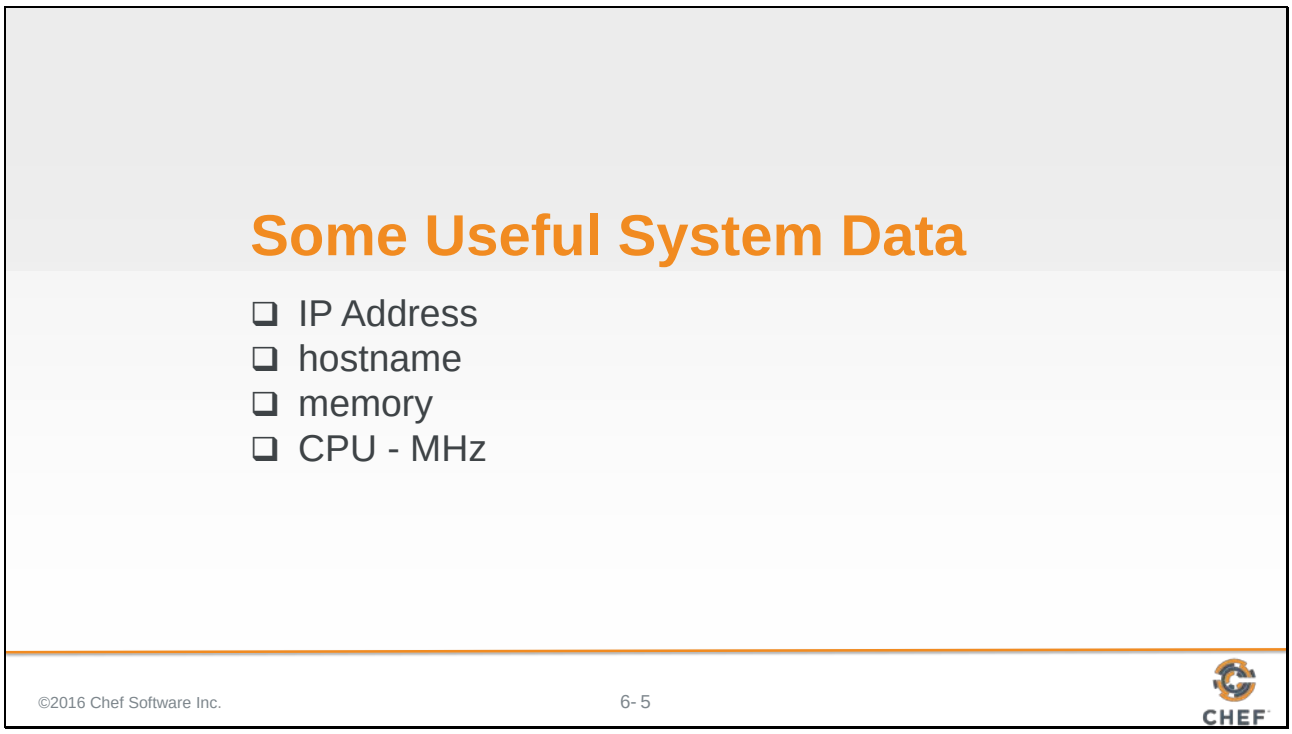
Slide 4



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

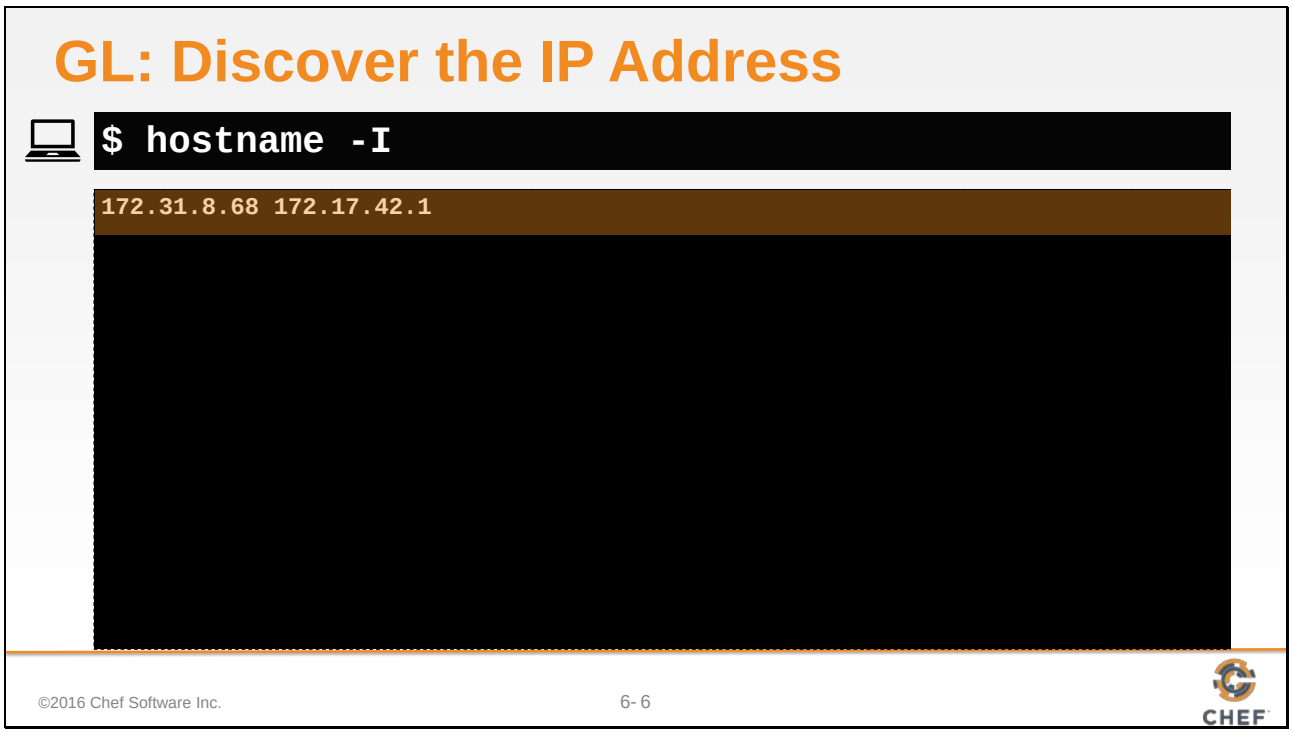


Thinking about some of the scenarios that we mentioned at the start of the session makes us think that it would be useful to capture:

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.

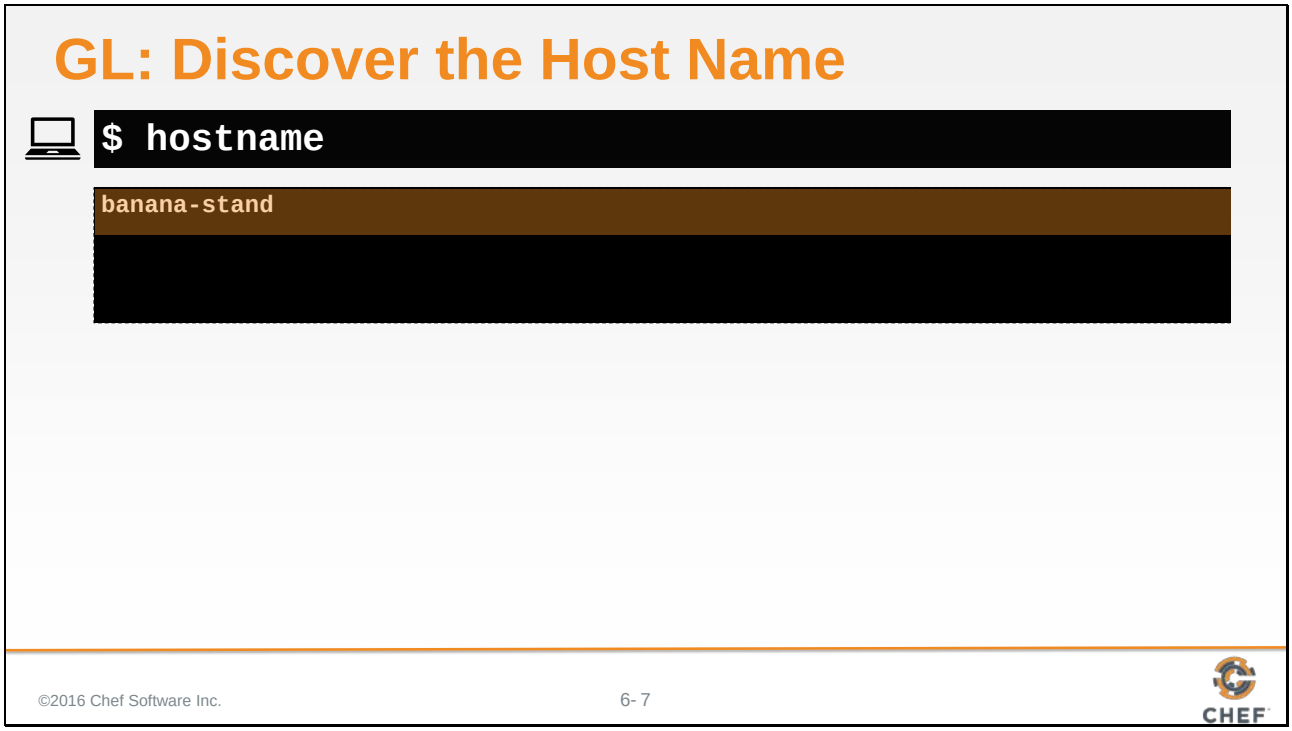
Slide 6



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

`hostname -I`

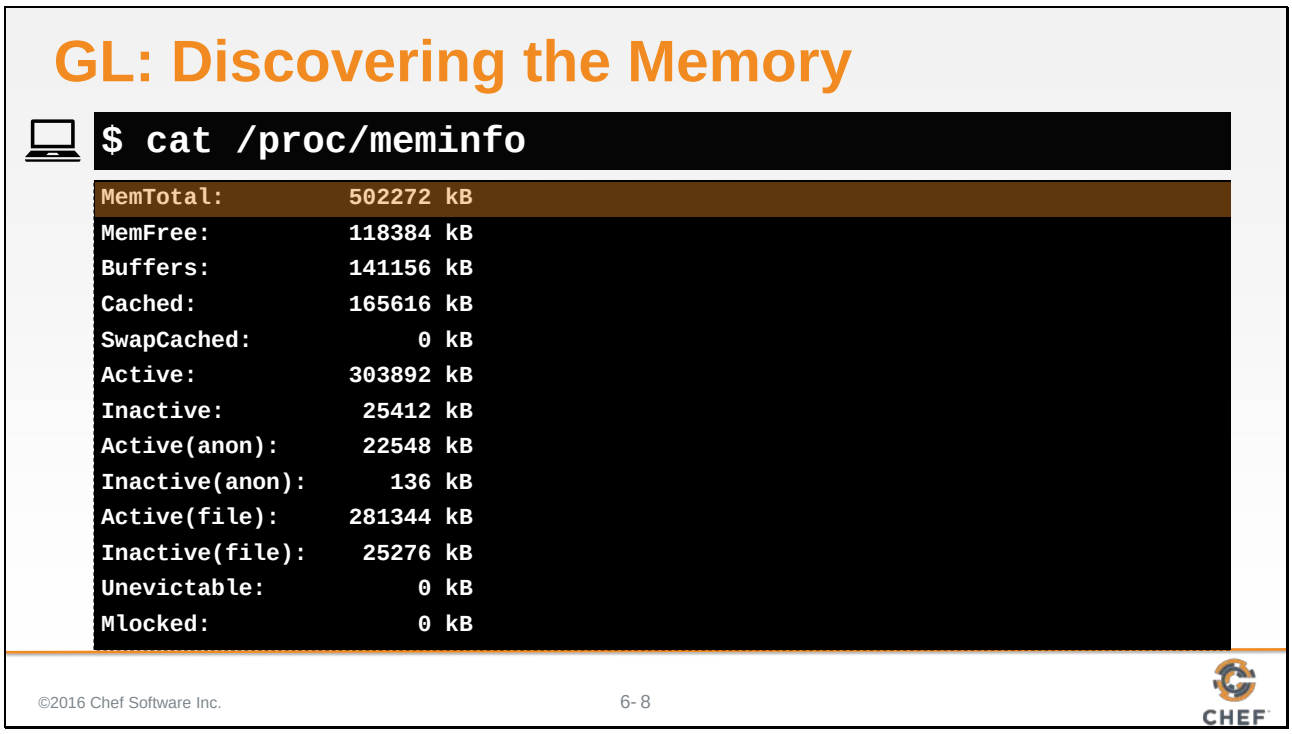
Slide 7



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

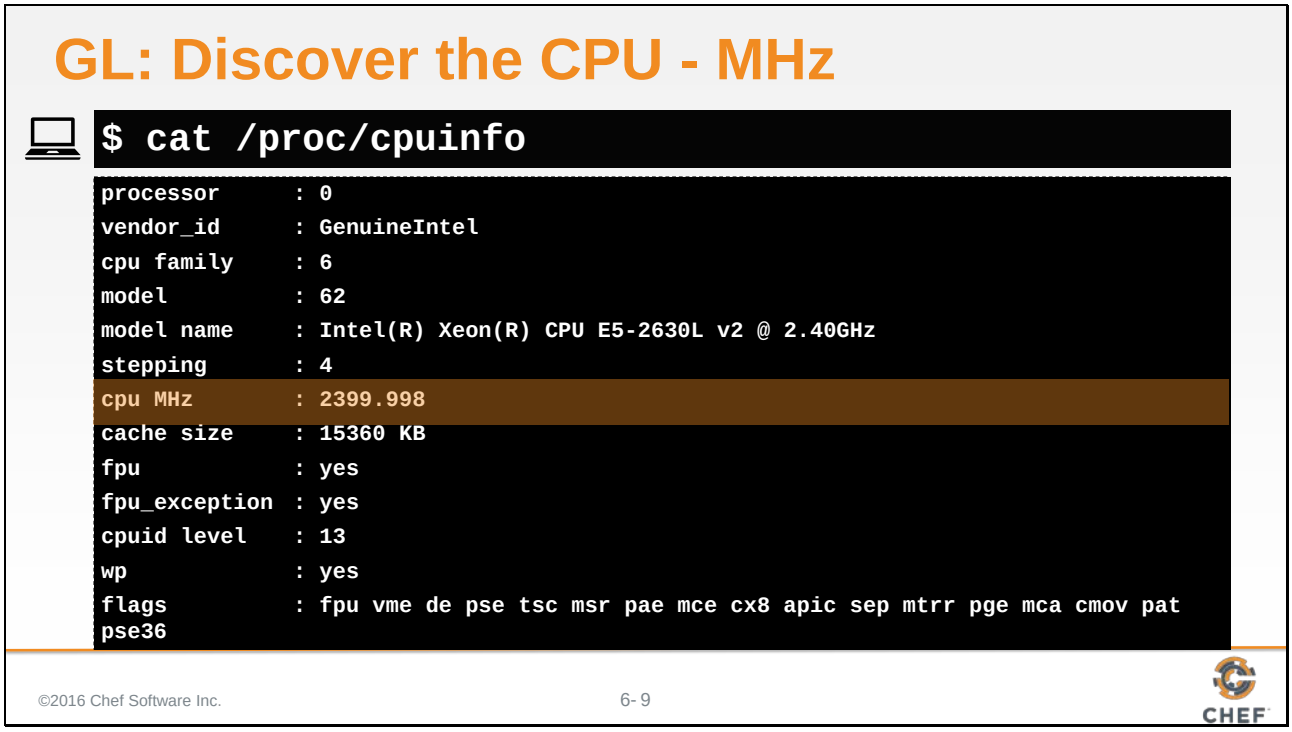
Note: The host name of your virtual workstation may simply be an IP address. For example, 'ip-172-31-2-14x'.

Slide 8



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 9



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 10

GL: Adding the CPU

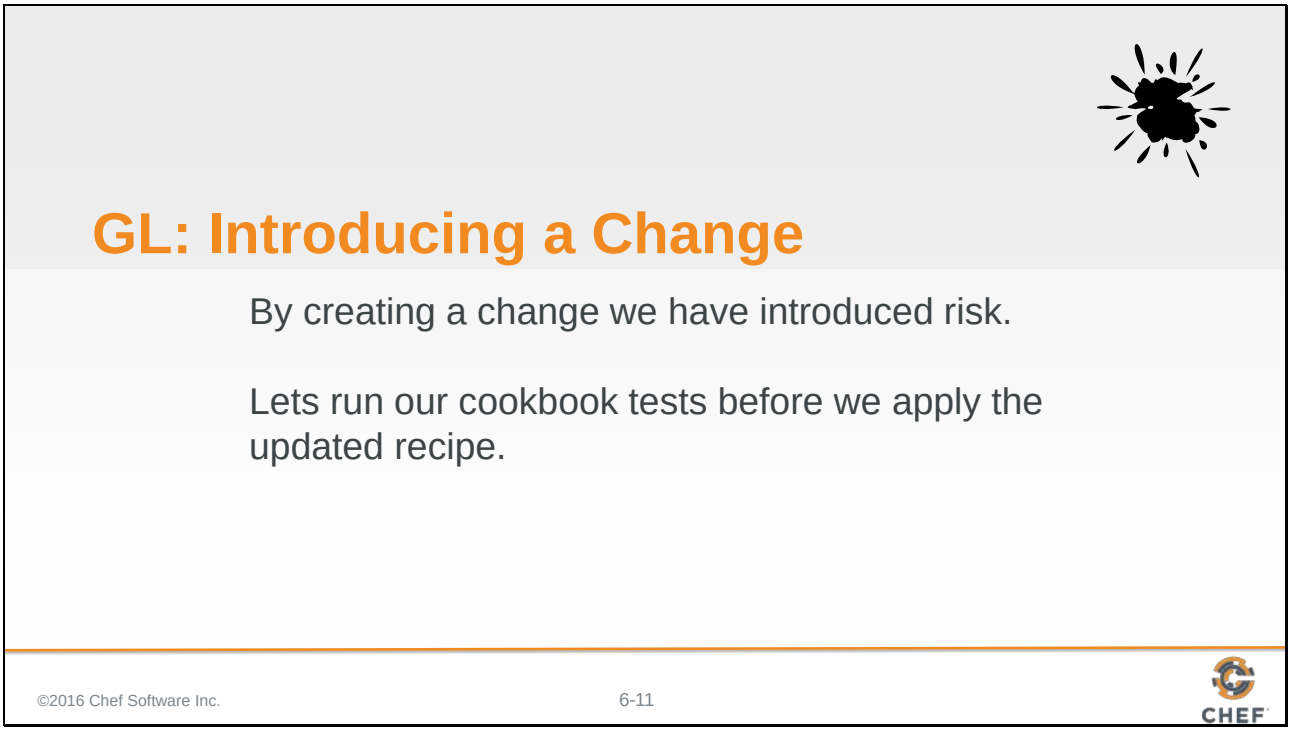
~/cookbooks/workstation/recipes/setup.rb

```
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
```


We can now add the ipaddress, hostname, memory, and cpu to the file resource's content attribute.

Slide 11



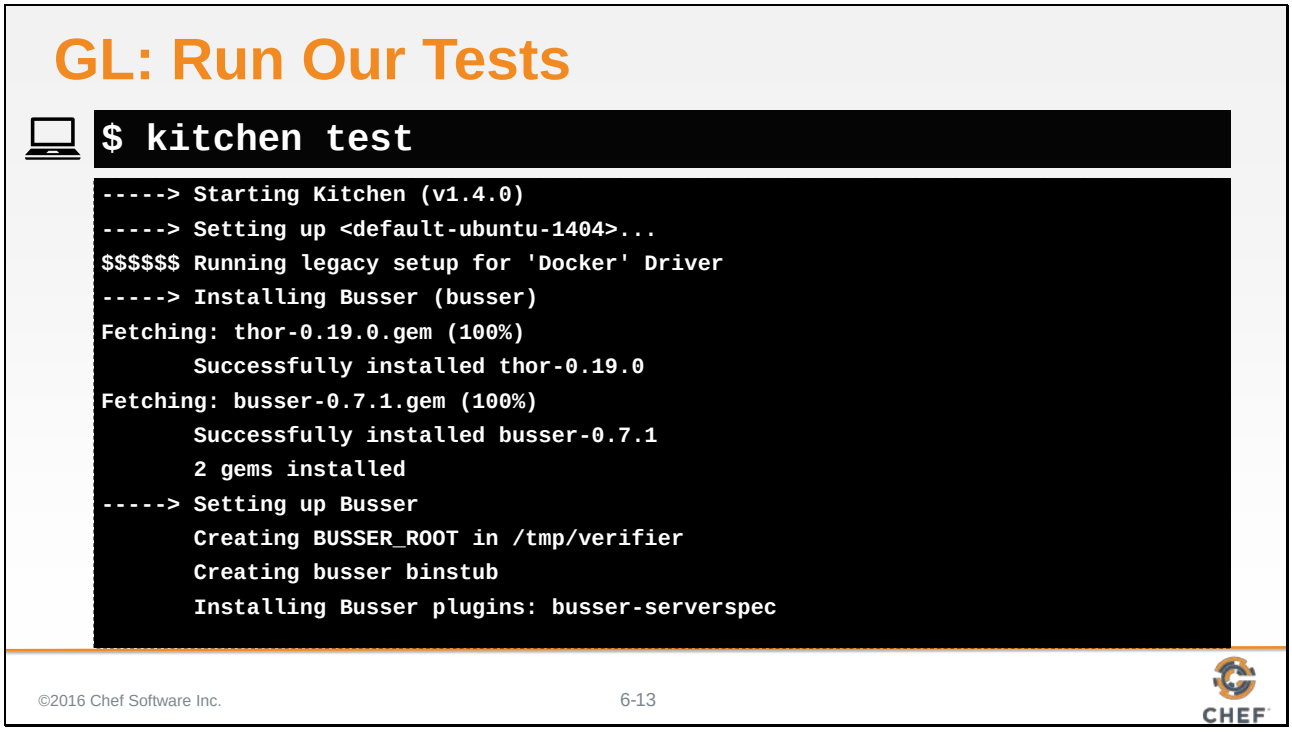
By updating the file resource we have introduced a change to the cookbook and introduced a risk. This change may not work. It could be a typo when transcribed from the slide, or the code that we have provided you may be out-of-date, or very possibly, incorrect.

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

Slide 12

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 13



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 14

GL: Return Home and Apply workstation Cookbook

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

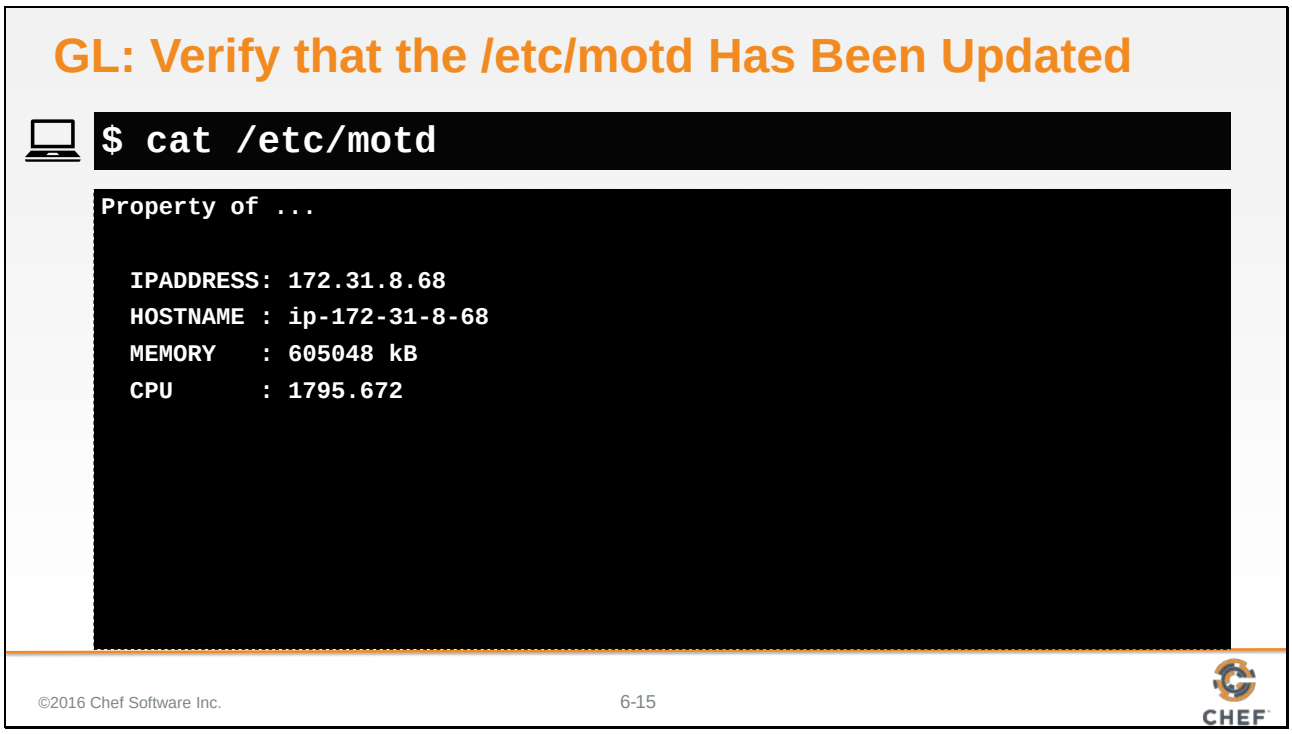
```
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
Converging 6 resources
Recipe: workstation::setup
  * yum_package[nano] action install (up to date)
  * yum_package[vim] action install (up to date)
  * yum_package[emacs] action install (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 15



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

Slide 16

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?

CHEF

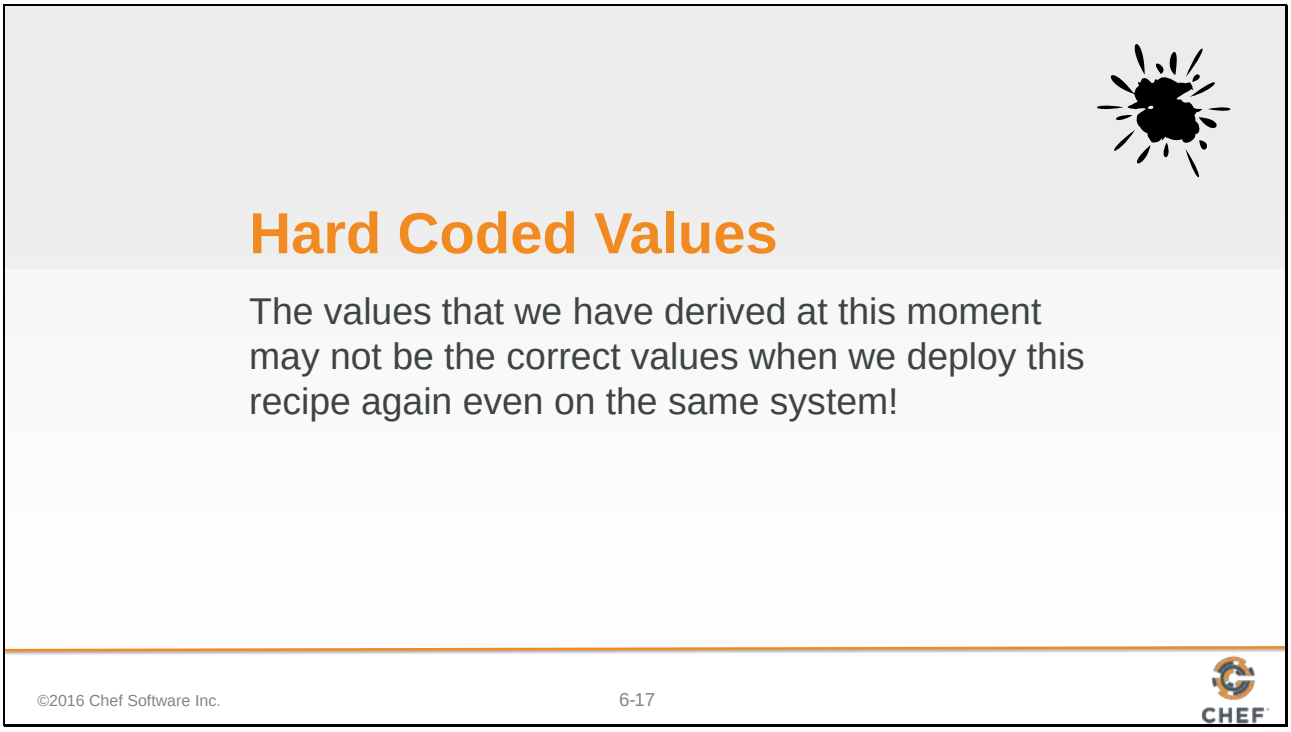
Now that we've defined these values, let's reflect:

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 17



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.

Slide 18

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.

Slide 19

CONCEPT

Ohai!

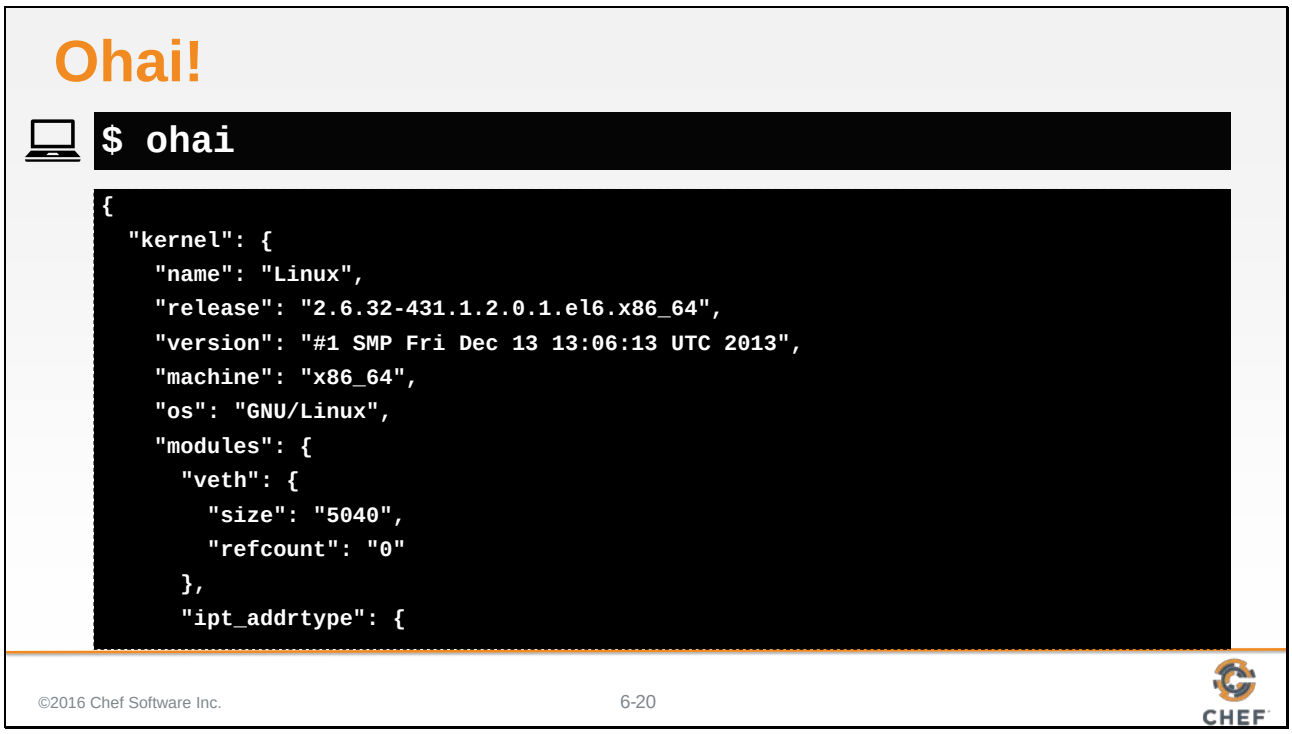
Ohai is a tool that already captures all the data that we similarly demonstrated finding.

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

CHEF

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

Slide 20



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

Slide 21

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

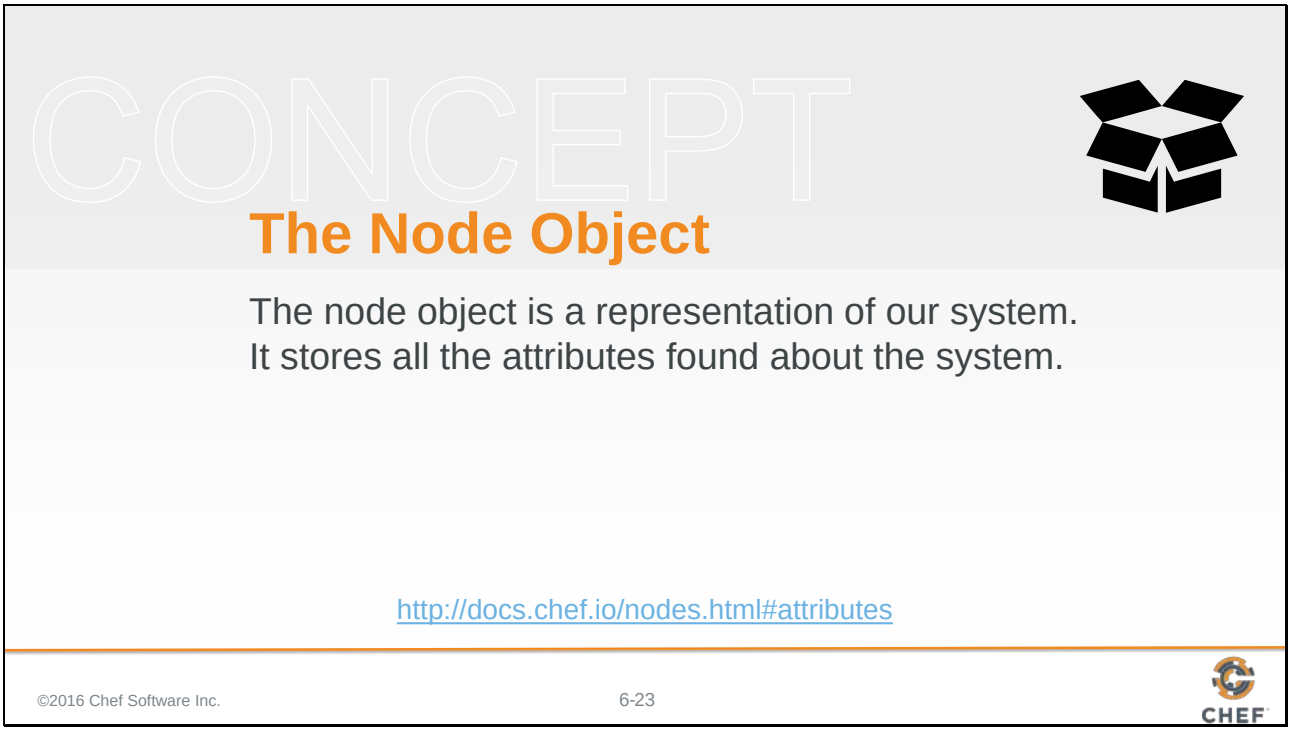
CHEF

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

Slide 22

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'

Slide 23

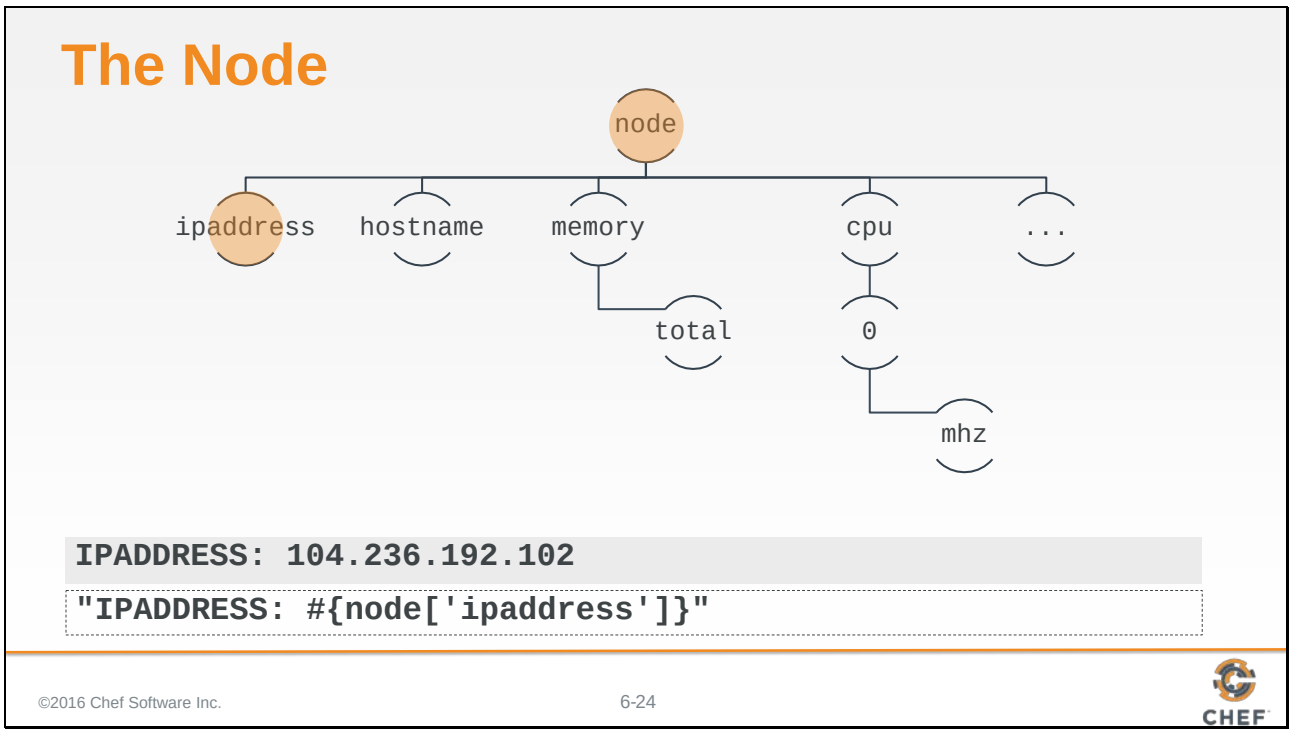


The node object is a representation of our system. It stores all these attributes found about the system. It is available within all the recipes that we write to assist us with solving the similar problems we outlined at the start.

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

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

Slide 24



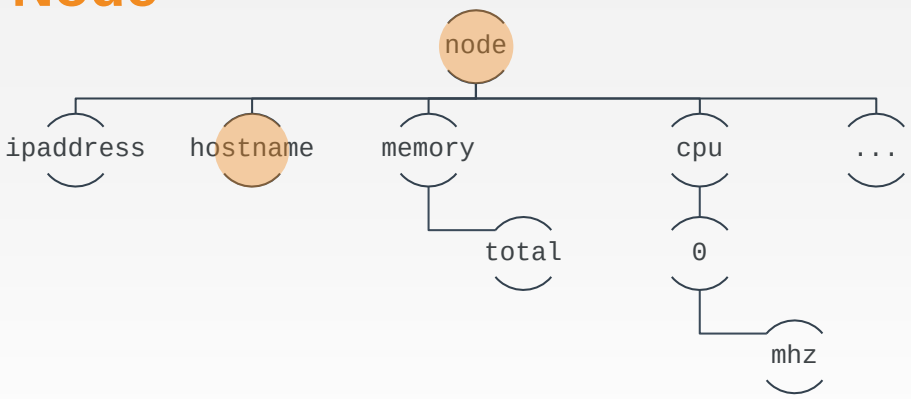
This is the visualization of the node attributes as a tree. That is done here to illustrate that the node maintains a tree of attributes that we can request from it.

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 25

The Node



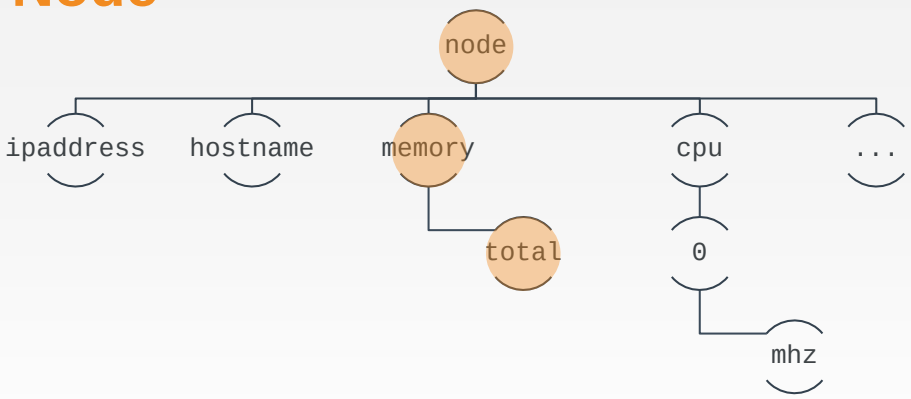

```
HOSTNAME: banana-stand
```

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


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

Slide 26

The Node




```
MEMORY: 502272kB
```

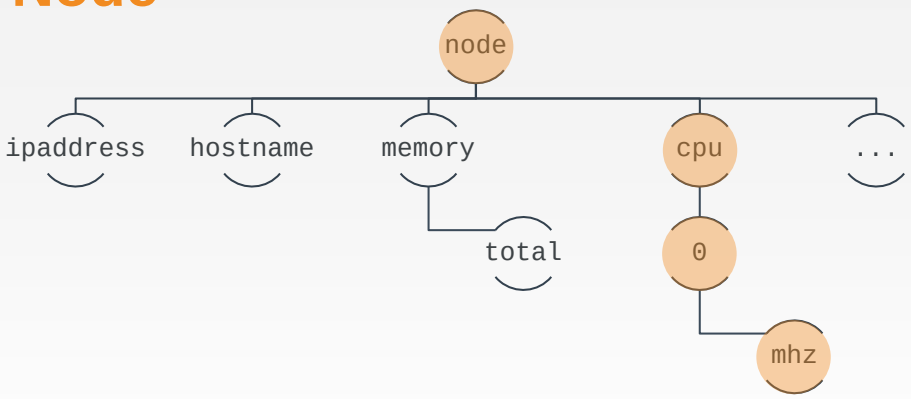
```
"Memory: #{node['memory']['total']}"
```


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 27

The Node




```
CPU: 2399.998MHz
```

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


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

Slide 28

CONCEPT

String Interpolation

```
I have 4 apples
```

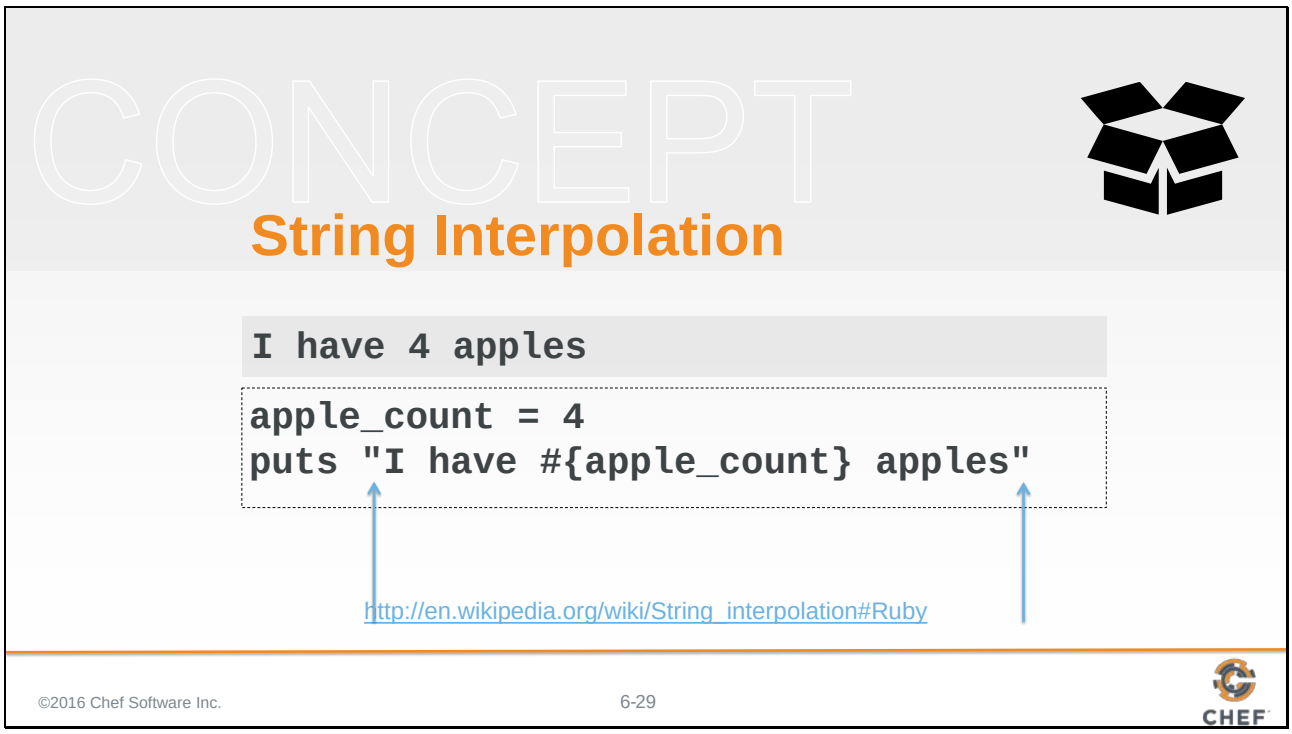
```
apple_count = 4
puts "I have #{apple_count} apples"
```

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

CHEF

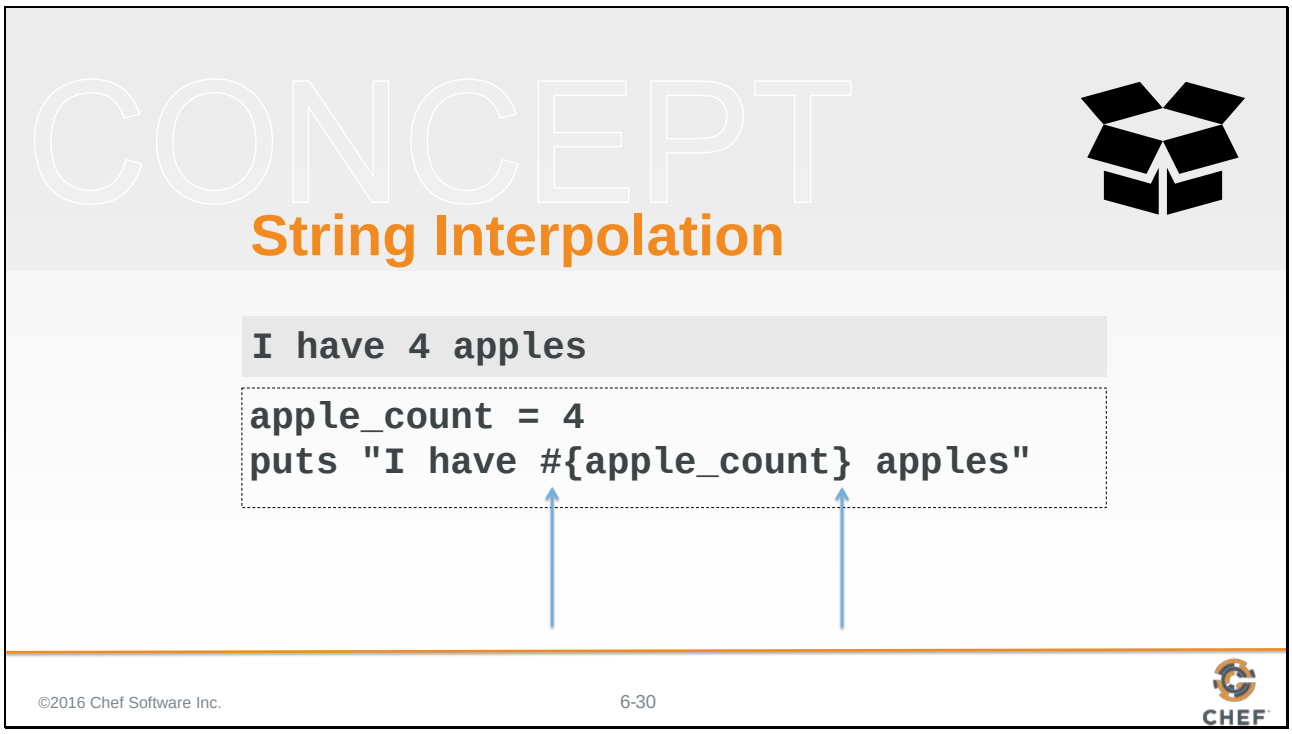
In all of the previous examples we demonstrated retrieving the values and displaying them within a string using a ruby language convention called string interpolation.

Slide 29



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

Slide 30



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 31

GL: 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 32

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.

Slide 33

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 34

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.

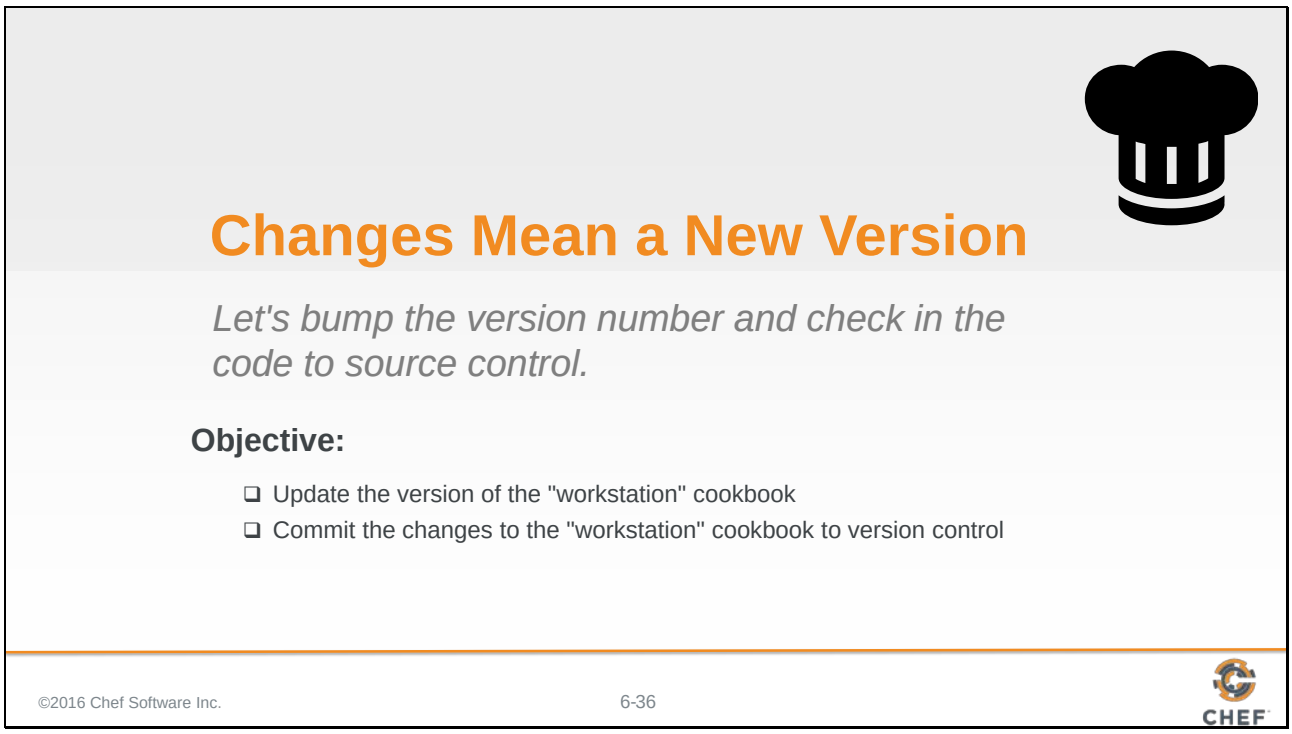
Slide 35

GL: 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.

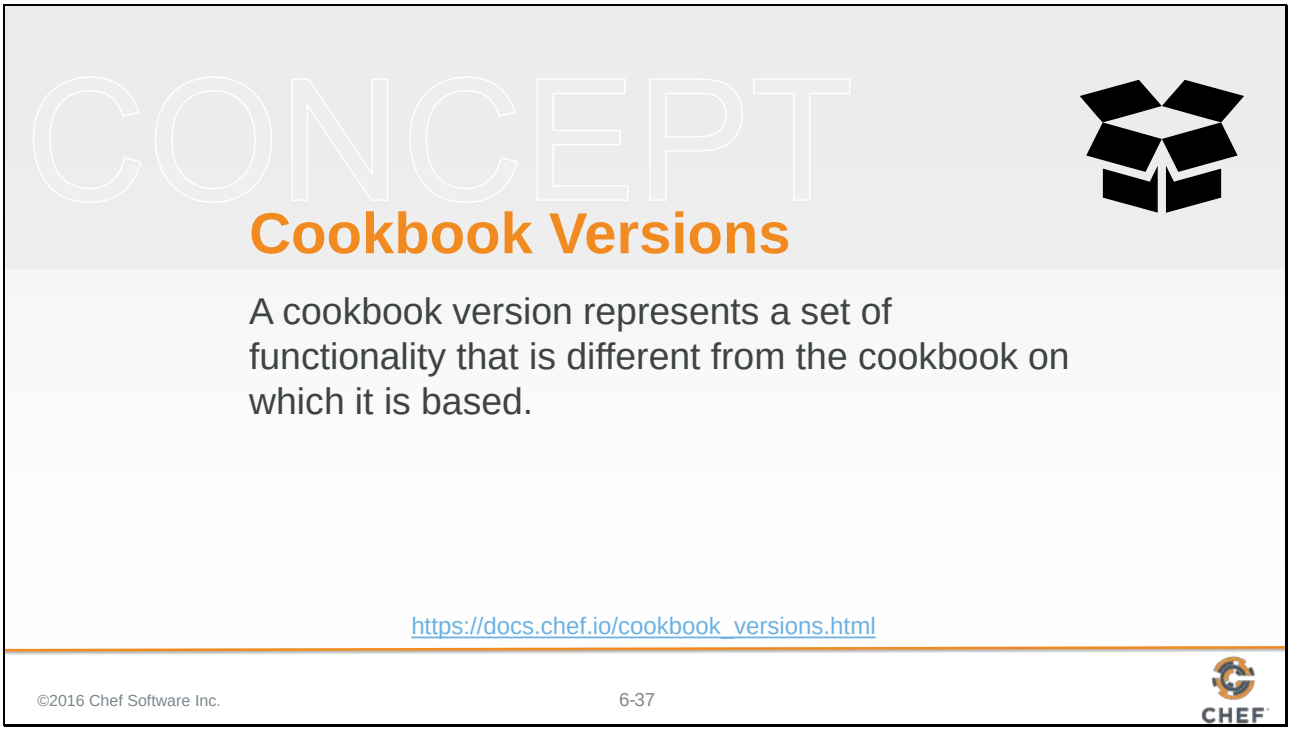
Great! Verifying the cookbook before you apply it ensures that you don't bad changes to this workstation.

Slide 36



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.

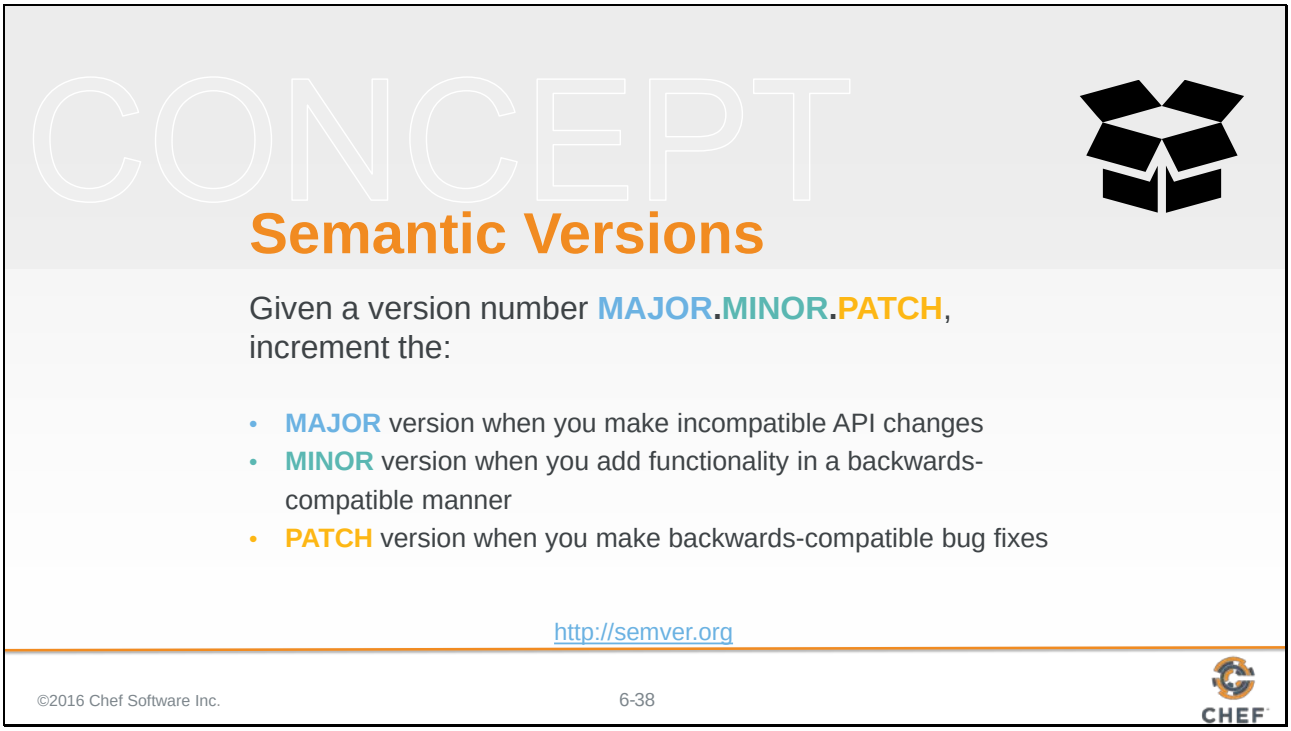
Slide 37



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

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

Slide 38



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 39

DISCUSSION

Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

CHEF

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

Slide 40

GL: Update the Cookbook Version

~/cookbooks/workstation/metadata.rb

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


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 41

COMMIT

GL: Commit Your Work

```
$ cd ~/cookbooks/workstation
$ git add .
$ git status
$ git commit -m "Release version 0.2.0"
```


CHEF

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 42

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.

Slide 43

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 44

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 45

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 46

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 47

COMMIT

Lab: Commit Your Work

```
$ cd ~/cookbooks/apache
$ git add .
$ git status
$ git commit -m "Release version 0.2.0"
```


And finally, commit your changes to git.

Slide 48

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

Congratulations. You have successfully demonstrated the entire development workflow.

Slide 49

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?

CHEF

Answer these questions.

Slide 50

DISCUSSION

Q&A

What questions can we help you answer?

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

CHEF

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 51