

## 7: Desired State and Data



## Objectives



After completing this module, you should be able to

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

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



## Cleaner Recipes

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

We added this content to the file resource in their respective recipes.

## Apache Recipe

 ~/cookbooks/apache/recipes/server.rb

```
package 'httpd'

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

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

What if new changes are given to us for the website splash page? For each new addition we would need to return to this recipe and carefully paste the contents of the new HTML into the string value of the content attribute.

## Double Quotes Close Double Quotes



Double quoted strings are terminated by double quotes.

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



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

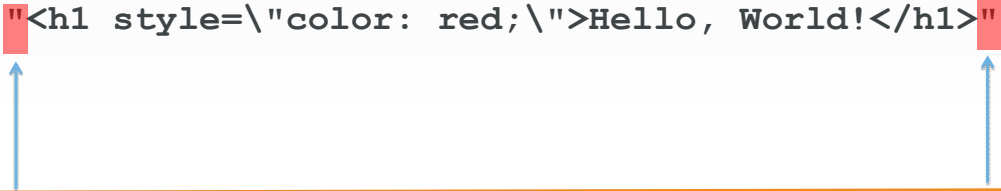
Double-quoted strings are terminated by double-quotes so if any of the text that we paste into this content field has double quotes it is going to have to be escaped.

# CONCEPT

## Backslash

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


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



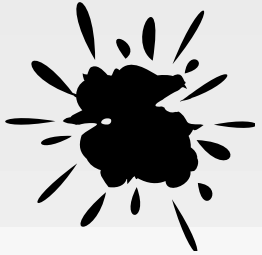
The diagram illustrates the concept of escaping a double quote character within a string. It shows a code snippet: `"<h1 style=\"color: red;\">Hello, World!</h1>"`. Two red boxes highlight the double quote characters at the beginning and end of the string. Blue arrows point from these boxes to the backslash characters that precede them, indicating that the backslashes are used to escape the double quotes so they can be included in the string.

©2015 Chef Software Inc.

7-6




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




## Backslash

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

**"**Root Path: \**"**




©2015 Chef Software Inc. 7-7 

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

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

# CONCEPT



## Backslash


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

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

---

©2015 Chef Software Inc.

7-8



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



## Unexpected Formatting

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

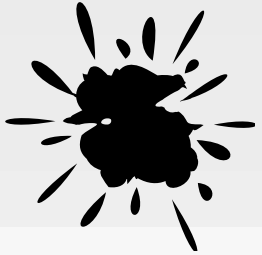
Don't even think about pasting ASCII ART in here!
'
end
```

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

```
Don't even think about pasting ASCII ART in here!
```

It is important to note that the file content may have some important formatting that might be easily overlooked when working with the content in a recipe file.


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



## Copy Paste

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

---

©2015 Chef Software Inc. 7-10 

This could sound like a bug waiting to happen.


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

# CONCEPT

## What We Need


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

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



©2015 Chef Software Inc.

7-11

 CHEF

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

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



## GE: Cleaner Recipes


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

### Objective:

- ☐ Decide which resource will help us address this issue

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

# DOCS



## GE: Let's Check the Docs...

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


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

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

---

©2015 Chef Software Inc.

7-13




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

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

Let's start with the `remote_file` resource.

# DOCS




## GE: remote\_file

Use the **remote\_file** resource to transfer a file from a remote location using file specificity. This resource is similar to the file resource.

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

©2015 Chef Software Inc.


7-14



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

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

# DOCS



## GE: cookbook\_file


Use the **cookbook\_file** resource to transfer files from a sub-directory of COOKBOOK\_NAME/files/ to a specified path located on a host that is running the chef-client.

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

---

©2015 Chef Software Inc.

7-15




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

## Demo: cookbook\_file's Source Match Up

```
$ tree cookbooks/apache/files/default
files/default
└── index.html
```

```
0 directories, 1 file
```

```
cookbook_file '/var/www/index.html' do
  source 'index.html'
end
```




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

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



# DOCS

## Template




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

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

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

©2015 Chef Software Inc.

7-17



Let's explore templates.


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

## Demo: Template File's Source Matches Up

```
$ tree cookbooks/apache/templates/default
templates/default
└── index.html.erb
```

```
0 directories, 1 file
```

```
template '/var/www/index.html' do
  source 'index.html.erb'
end
```




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

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

# DOCS

## Template



To use a template, two things must happen:


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

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

---

©2015 Chef Software Inc.

7-19



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



## GE: Cleaner Apache Recipe

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

**Objective:**

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

---

©2015 Chef Software Inc. 7-20 

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

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

## GE: 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)

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

Templates and files (for `cookbook_files`) are a few of the other things it can generate for us.

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

## GE: What Can chef generate template Do?



```
$ chef generate template --help
```

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

```
-C, --copyright COPYRIGHT      Name of the copyright holder - defaults
to 'The Authors'
-m, --email EMAIL              Email address of the author - defaults to
...
-a, --generator-arg KEY=VALUE  Use to set arbitrary attribute KEY to
VALUE in the
-I, --license LICENSE          all_rights, apache2, mit, gplv2, gplv3 -
defaults to
-s, --source SOURCE_FILE       Copy content from SOURCE_FILE
-g GENERATOR_COOKBOOK_PATH,    Use GENERATOR_COOKBOOK_PATH for the
code_generator
--generator-cookbook
```

Finally let's ask for help for generating templates.

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

## GE: Use chef to Generate a Template



```
$ cd ~  
$ chef generate template cookbooks/apache index.html
```

```
Compiling Cookbooks...
```

```
Recipe: code_generator::template
```

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

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

## GE: Lets Look at the Template File



```
$ tree cookbooks/apache/templates
```

```
cookbooks/apache/templates/
```

```
├── default
```

```
└── index.html.erb
```

```
1 directory, 1 file
```

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





## Cleaner Recipes

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


### Objective:

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

Now we need to understand what ERB means.

# CONCEPT

## ERB




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

Ruby code can be embedded using expressions and statements.

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

---

©2015 Chef Software Inc. 7-26  CHEF

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

## Text Within an ERB Template

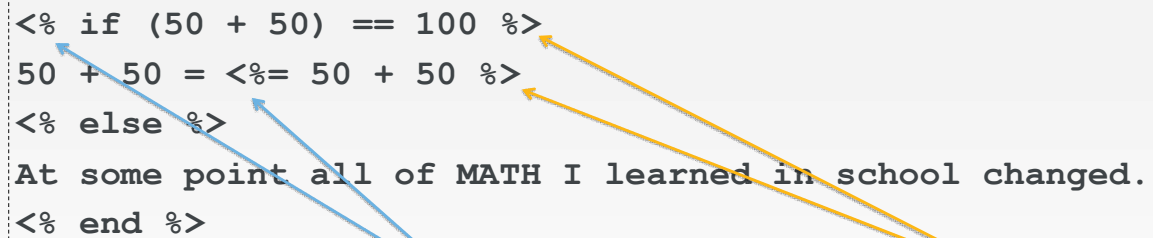
```
<% if (50 + 50) == 100 %>  
50 + 50 = <%= 50 + 50 %>  
<% else %>  
At some point all of MATH I learned in school changed.  
<% end %>
```

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

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

## Text Within an ERB Template

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

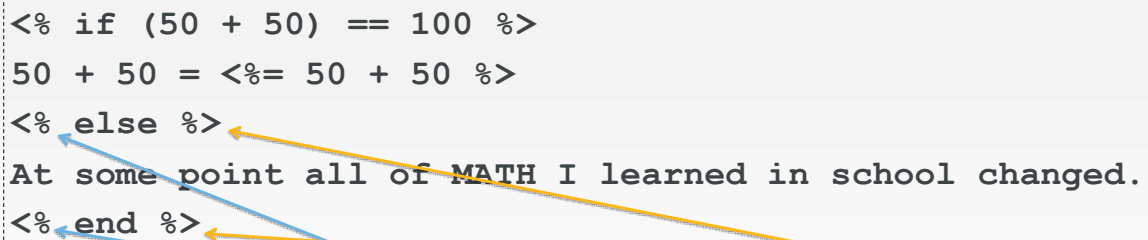
A diagram showing an ERB template snippet. The snippet is enclosed in a dashed box. It contains four lines of code. The first line is a beginning tag: `<% if (50 + 50) == 100 %>`. The second line is a text line: `50 + 50 = <%= 50 + 50 %>`. The third line is an ending tag: `<% else %>`. The fourth line is a text line: `At some point all of MATH I learned in school changed.`. The fifth line is another ending tag: `<% end %>`. There are four arrows pointing from the text lines to their corresponding end tags. Two blue arrows point from the text lines to the `<% else %>` tag. Two yellow arrows point from the text lines to the `<% end %>` tag.

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

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

## Text Within an ERB Template

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

A diagram showing the structure of an ERB template. It contains a code block with an if-else statement. Blue arrows point from the opening tags (<% if, <% else, <% end) to their corresponding closing tags (%>). Yellow arrows point from the closing tags back to the opening tags, illustrating the matching process.

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

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

## Text Within an ERB Template

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

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

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

## Text Within an ERB Template

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

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

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


# CONCEPT

## The Angry Squid



©2015 Chef Software Inc.

7-32



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

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



## GE: Move Our Source to the Template

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

```
<h1>Hello, world!</h1>
<h2>ipaddress: #{node['ipaddress']}</h2>
<h2>hostname: #{node['hostname']}</h2>
```

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

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

## GE: Replace String Interpolation with ERB

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

<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: <%= node['ipaddress'] %></h2>
    <h2>hostname: <%= node['hostname'] %></h2>
  </body>
</html>
```

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



## Cleaner Recipes

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

### Objective:

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

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

## GE: Remove the Existing Content Attribute

```
~/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
```

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

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

## GE: Change File Resource to a Template Resource

```
~/cookbooks/apache/recipes/server.rb  
  
template '/var/www/html/index.html' do  
  
end
```

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

## What to Specify as the Source?

```
~/cookbooks/apache/recipes/server.rb  
  
template '/var/www/html/index.html' do  
  source '????????????????????'   
end
```

Lastly we need to specify a source attribute which contains that path to the template we generated. This path is relative starting from within the cookbook's template directory.

## GE: Viewing the Partial Path to the Template



```
$ tree cookbooks/apache/templates/default
```

```
cookbooks/apache/templates/default/
```

```
└─ index.html.erb
```

```
0 directories, 1 file
```

To visualize that with 'tree' we can run it with a path that places us right at the templates directory. So the results will be relative paths from the point specified.

And we see the filepath `index.html.erb`.

## GE: Update the Source Attribute

```
~/cookbooks/apache/recipes/server.rb  
  
template '/var/www/html/index.html' do  
  source 'index.html.erb'  
end
```

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





## Cleaner Recipes

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

### Objective:

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

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



## Lab: Update the Version

- ☐ Use kitchen test on the "apache" cookbook
- ☐ Use chef-client to apply the "apache" cookbook's "default" recipe
- ☐ Update the "apache" cookbook's version for this patch
- ☐ Commit the changes

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

## Lab: Test the Cookbook



```
$ 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
```

Since kitchen is a cookbook testing tool, you need to move into the cookbook's directory.

Then run the 'kitchen test' command, addressing any issues if they show up.

## Lab: Change Directories and Apply the Cookbook



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

[2015-09-16T14:18:05+00:00] WARN: No config file found or specified on command line,
using command line options.
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
[2015-09-16T14:18:09+00:00] WARN: Cloning resource attributes for service[httpd]
from prior resource (CHEF-3694)
[2015-09-16T14:18:09+00:00] WARN: Previous service[httpd]: /root/.chef/local-mode-
cache/cache/cookbooks/apache/recipes/server.rb:8:in `from_file'
[2015-09-16T14:18:09+00:00] WARN: Current service[httpd]: /root/.chef/local-mode-
cache/ ...
```

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

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

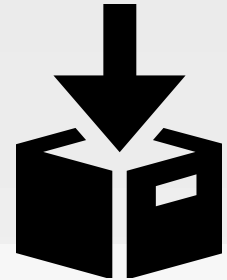
## Lab: Update the Cookbook's Patch Number

```
~/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             '0.2.1'
```

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

## Lab: Commit the Changes



```
$ cd ~/cookbooks/apache  
$ git add .  
$ git status  
$ git commit -m "Changed file resource to template  
resource and defined a template"
```

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



## Lab: Use the Template

For the "workstation" cookbook:

- ☐ Use chef generate to create a template named "motd.erb".
- ☐ Copy the source attribute from the file named '/etc/motd' into the template file "motd.erb"
- ☐ Remove a resource: The file named '/etc/motd'
- ☐ Add a resource: The template named '/etc/motd' is created with the source 'motd.erb'
- ☐ Use kitchen to test it and chef-client to locally apply the default recipe.

It's time to do that again--this time for the workstation cookbook.

Generate a template named 'motd', copy in the source attribute from the file resource, and then update it to use ERB tags.

Then come back to the recipe. Change it to a template resource and then add a source attribute whose value is that partial path to the new template you created.

## Lab: Return Home and Generate the Template



```
$ cd ~  
$ chef generate template cookbooks/workstation motd
```

```
Compiling Cookbooks...  
Recipe: code_generator::cookbook  
  * directory[/home/chef/template] action create  
    - create new directory /home/chef/template  
  * template[/home/chef/template/metadata.rb] action create_if_missing  
    - create new file /home/chef/template/metadata.rb  
    - update content in file /home/chef/template/metadata.rb from none to 000bce  
      (diff output suppressed by config)  
  * template[/home/chef/template/README.md] action create_if_missing  
    - create new file /home/chef/template/README.md
```

Return to the home directory. Run the command to generate the template named 'motd' in the workstation cookbook.



## Lab: Copy the Existing Source into the Template

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

Property of ...

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

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

## Lab: Update the motd.erb to Use ERB

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

Property of ...

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

Replace all the string interpolation with ERB tags.

## Lab: Remove the file Resource

 ~/cookbooks/workstation/recipes/setup.rb

```
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
```

Remove the file resource from the setup recipe.

## Lab: Replace it with the Template Resource

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

```
template '/etc/motd' do
  source 'motd.erb'
  mode '0644'
  owner 'root'
  group 'root'
end
```

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

## Lab: Test the Cookbook



```
$ 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
```

Since kitchen is a cookbook testing tool, you need to move into the cookbook's directory.

Then run the 'kitchen test' command, addressing any issues if they show up.

## Lab: Change Directories and Apply the Cookbook



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

[2015-09-16T14:18:05+00:00] WARN: No config file found or specified on command
line, using command line options.
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
[2015-09-16T14:18:09+00:00] WARN: Cloning resource attributes for
service[httpd] from prior resource (CHEF-3694)
[2015-09-16T14:18:09+00:00] WARN: Previous service[httpd]: /root/.chef/local-
mode-cache/cache/cookbooks/apache/recipes/server.rb:8:in `from_file'
[2015-09-16T14:18:09+00:00] WARN: Current service[httpd]: /root/.chef/local-
mode-cache/...
```

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

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



## Lab: Update the Version

- ☐ Update the "workstation" cookbook's version for this patch
- ☐ Commit the changes to the "workstation" cookbook to version control

With everything working it is time to update the patch version and commit the changes.

## Lab: Update the Cookbook's Patch Number

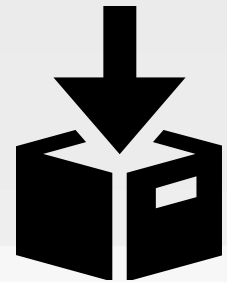
```
~/cookbooks/workstation/metadata.rb
```

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

Update the patch version number for the workstation cookbook.




## Lab: Commit the Changes



```
$ cd ~/cookbooks/workstation  
$ git add .  
$ git status  
$ git commit -m "Changed file resource to template  
resource and defined a template"
```

Add and then commit the changes to the workstation cookbook.

# DISCUSSION




## Discussion

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

What do each of the ERB tags accomplish?

©2015 Chef Software Inc.


7-58



Answer these questions.

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

# DISCUSSION




## Q&A

What questions can we help you answer?

- Resources (file, cookbook\_file, template, and remote\_file)
- Templates
- ERB

---

©2015 Chef Software Inc. 7-59 

What questions can we help you answer?

Generally or specifically about resources, templates, and ERB.



**CHEF**™