

Desired State and Data

Extracting the Content for Clarity

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



Cleaner Recipes

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

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

Webserver Recipe

 ~\cookbooks\webserver\recipes\default.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>
  <h2>MEMORY: #{node['memory']['total']}</h2>
  <h2>CPU: #{node['cpu']['0']['mhz']}</h2>
"
end

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

Double Quotes Close Double Quotes



Double quoted strings are terminated by double quotes.

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

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 shows the HTML tag `<h1 style="color: red;">Hello, World!</h1>` with the opening and closing double quotes highlighted in red. Blue arrows point from the bottom to each red quote, indicating the need to escape them with backslashes in the code representation shown above.

Backslash



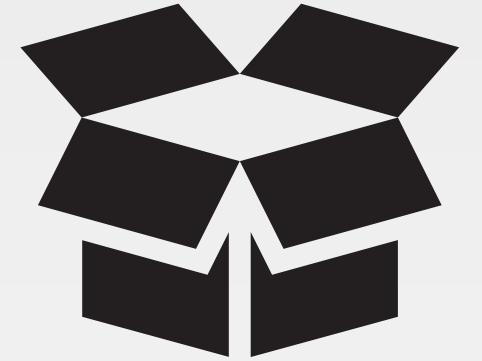
Backslashes are reserved characters. So to use them you need to use a backslash.

"Root Path: \"



CONCEPT

Backslash



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

"Root Path: \\"

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!
```

Copy Paste



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

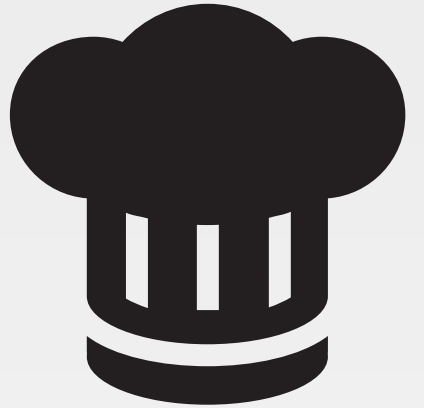
CONCEPT

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.



GL: Cleaner Recipes

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

Objective:

- ❑ Decide which resource will help us address this issue

DOCS



GL: Let's Check the Docs...

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

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

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

DOCS

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

DOCS

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

Demo: cookbook_file's Source Match Up

```
> tree cookbooks\webserver\files\default  
files\default
```

```
└─ index.html
```

```
0 directories, 1 file
```

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

```
  source 'index.html'
```

```
end
```


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

Demo: Template File's Source Matches Up

```
> tree cookbooks\webserver\templates\default
cookbooks\webserver\templates\default
```

```
└─ index.html.erb
```

```
0 directories, 1 file
```

```
template '/var/www/html/index.html' do
```

```
  source 'index.html.erb'
```

```
end
```

DOCS

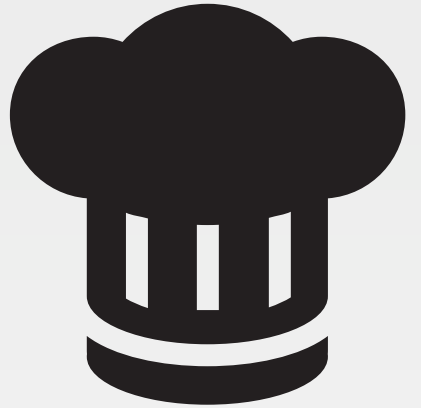
Template



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



GL: Cleaner Webserver 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 'webserver' cookbook

GL: What Can chef generate Do?



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

```
Usage: chef generate GENERATOR [options]
```

```
Available generators:
```

app	Generate an application repo
cookbook	Generate a single cookbook
recipe	Generate a new recipe
attribute	Generate an attributes file
template	Generate a file template
file	Generate a cookbook file
lwrp	Generate a lightweight resource/provider
repo	Generate a Chef policy repository
policyfile	Generate a Policyfile for use with the install/push commands (experimental)

GL: What Can chef generate template Do?



```
> chef generate template --help
```

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

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

GL: Use chef to Generate a Template



```
> cd ~
```

```
> chef generate template cookbooks\webserver index.html
```

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

```
  * directory[cookbooks/webserver/templates/default] action create
```

```
    - create new directory cookbooks/webserver/templates/default
```

```
  * template[cookbooks/webserver/templates/index.html.erb] action create
```

```
    - create new file cookbooks/webserver/templates/index.html.erb
```

```
    - update content in file cookbooks/webserver/templates/index.html.erb from  
none to e3b0c4
```

```
      (diff output suppressed by config)
```

GL: Lets Look at the Template File



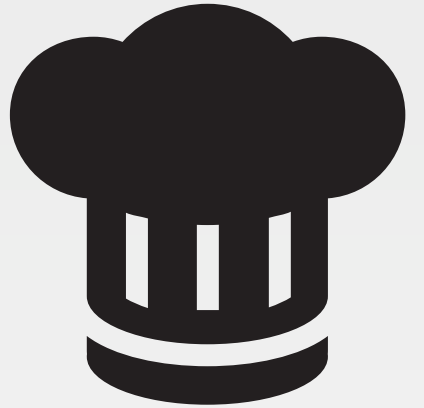
```
> tree /f cookbooks\webserver\templates
```

```
cookbooks/webserver/templates/
```

```
├── default
```

```
└── index.html.erb
```

```
1 directory, 1 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 'webserver' cookbook

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>

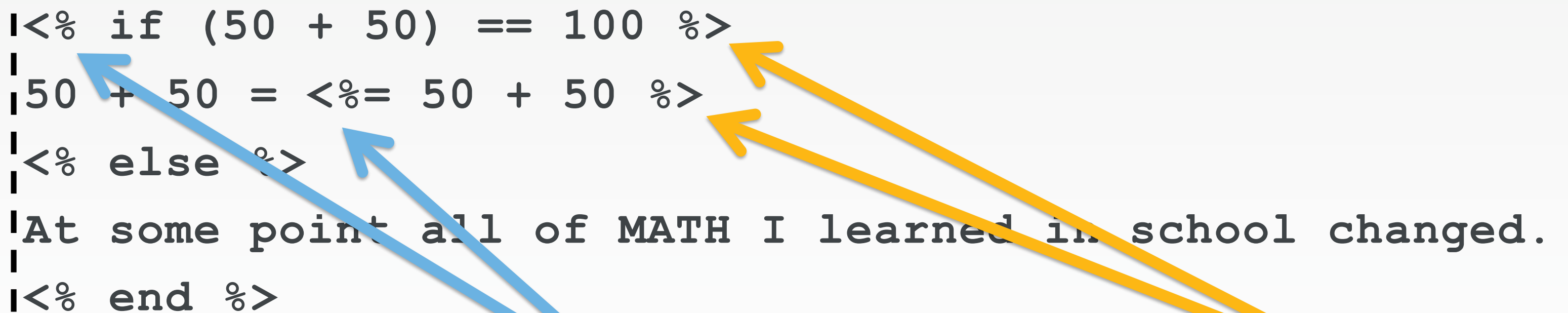
Text Within an ERB Template

```
<% 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.

Text Within an ERB Template

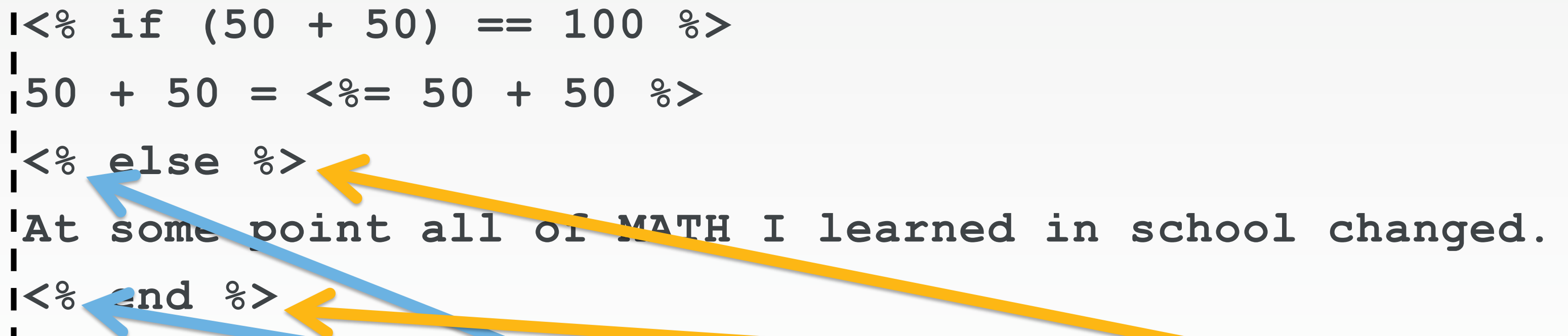
```
<% if (50 + 50) == 100 %>  
50 + 50 = <%= 50 + 50 %>  
<% else %>  
At some point all of MATH I learned in school changed.  
<% end %>
```

A diagram illustrating ERB tag matching. It shows a code snippet within a dashed rectangular border. The code contains an if-else block. Two blue arrows originate from the bottom of the dashed box and point to the beginning tags: one to the opening <% if tag and one to the opening <% else tag. Two yellow arrows originate from the right side of the dashed box and point to the closing tags: one to the closing %> tag of the if block and one to the closing %> tag of the else block.

Each ERB tag has a beginning tag and a matched 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 illustrating ERB tag matching. A dashed rectangular box encloses the first four lines of code. Two yellow arrows originate from the right side of the box: one points to the opening tag '<% if (50 + 50) == 100 %>' and the other points to the closing tag '<% end %>'. Two blue arrows originate from the left side of the box: one points to the opening tag '<% else %>' and the other points to the closing tag '<% end %>'. The text 'At some point all of MATH I learned in school changed.' is the content between the 'else' and 'end' tags.

Each ERB tag has a beginning tag and a matched 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 %>
```

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

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.

CONCEPT

The Angry Squid



<%=

GL: Move Our Source to the Template

 ~\cookbooks\webserver\templates\index.html.erb

```
<h1>Hello, world!</h1>
<h2>IPADDRESS: #{node['ipaddress']}</h2>
<h2>HOSTNAME:  #{node['hostname']}</h2>
<h2>MEMORY:   #{node['memory']['total']}</h2>
<h2>CPU:      #{node['cpu']['0']['mhz']}</h2>
```

GL: Replace String Interpolation with ERB

~\cookbooks\webserver\templates\index.html.erb

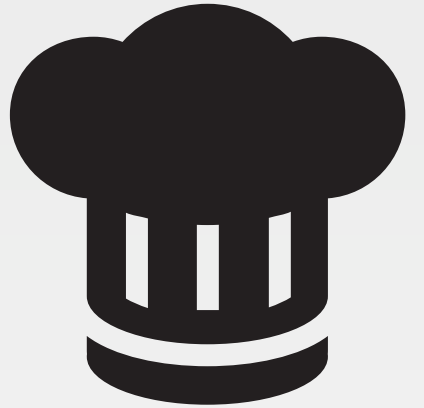
```
<h1>Hello, world!</h1>
```

```
<h2>IPADDRESS: <%= node['ipaddress'] %></h2>
```

```
<h2>HOSTNAME: <%= node['hostname'] %></h2>
```

```
<h2>MEMORY: <%= node['memory']['total'] %></h2>
```

```
<h2>CPU: <%= node['cpu']['0']['mhz'] %></h2>
```



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 'webserver' cookbook

GL: Remove the Existing Content Attribute

 ~\cookbooks\webserver\recipes\default.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>  
|    <h2>MEMORY: #{node['memory']['total']}</h2>  
|    <h2>CPU: #{node['cpu']['0']['mhz']}</h2>  
|  "  
|end  
|  
|service 'httpd' do  
|  action [ :enable, :start ]  
|end
```

GL: Change File Resource to a Template Resource

~\cookbooks\webserver\recipes\default.rb

```
package 'httpd'
```

```
template '/var/www/html/index.html' do
```

```
end
```

```
service 'httpd' do
```

```
  action [ :enable, :start ]
```

```
end
```

What to Specify as the Source?

 ~\cookbooks\webserver\recipes\server.rb

```
package 'httpd'

template '/var/www/html/index.html' do
  source '????????????????'
end

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

GL: Viewing the Partial Path to the Template



```
> tree cookbooks\webserver\templates
```

```
cookbooks/webserver/templates
```

```
├── default
```

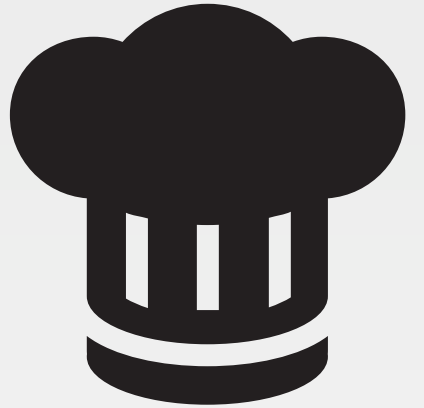
```
└── index.html.erb
```

```
1 directories, 1 file
```

GL: Update the Source Attribute

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

```
package 'httpd'  
  
template '/var/www/html/index.html' do  
  source 'index.html.erb'  
end  
  
service 'httpd' do  
  action [ :enable, :start ]  
end
```

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 'webserver' cookbook



Lab: Update the Version

- ☐ Use kitchen test on the "webserver" cookbook
- ☐ Update the "webserver" cookbook's version for this patch
- ☐ Commit the changes

Lab: Converge and Verify the Test Instance



```
> cd ~\cookbooks\webserver  
> kitchen converge  
> kitchen verify
```

```
...  
-----> Starting Kitchen (v1.13.2)  
-----> Converging <default-centos-6>...  
    Preparing files for transfer  
    Preparing dna.json  
    Resolving cookbook dependencies with Berkshelf 5.1.0...  
    Removing non-cookbook files before transfer  
    Preparing validation.pem  
    Preparing client.rb  
-----> Chef Omnibus installation detected (install only if missing)
```

Lab: Update the Cookbook's Patch Number

```
~\cookbooks\webserver\metadata.rb
```

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

Lab: Commit the Changes

- > `cd ~\cookbooks\webserver`
- > `git add .`
- > `git status`
- > `git commit -m "Update default recipe to use template"`





Lab: Update the Version

- ✓ Use kitchen test on the "webserver" cookbook
- ✓ Update the "webserver" cookbook's version for this patch
- ✓ Commit the changes

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?

DISCUSSION



Q&A

What questions can we help you answer?

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



CHEF™