Chef Essentials on AWS

Introduction
Your Chef Team

Jody Wolfborn – Solutions Architect
jody@chef.io
@joderita

Jeff Mery – Solutions Architect
jmery@chef.io
Introduce Yourselves

Name
Current job role
Previous job roles/background
Experience with Chef and/or config management
Favorite Text Editor
Expectations

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

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

After completing this course, you should be able to:

- Use Chef Resources to define the state of your system
- Write and use Chef recipes and cookbooks
- Automate testing of cookbooks
- Create Organizations
- Provision and bootstrap nodes
Chef

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

Chef can integrate with cloud-based platforms such as Amazon Elastic Compute Cloud to automatically provision and configure new machines.
Chef is a large set of tools that are able to be used on multiple platforms and in numerous configurations.

Learning Chef is like learning a language. You will reach fluency very fast but it will take practice until you become comfortable.

A great way to learn Chef is to use Chef.
Chef Essentials

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

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

In this course you will use two different architectures:

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

Architecture 1

Virtual Workstation
Preconfigured with Chef tools

Your Laptop

©2015 Chef Software Inc.
Chef Lab System Architecture

Architecture 2

Chef Server

Your Local Workstation

Nodes
Getting a Workstation

Your instructor will assign you a virtual workstation hosted in EC2. The credentials for login are:

Username: Administrator
Password: Cod3Can!
Hands-on Legend

- GE or Group Exercise: All participants and the instructor do this task together with the instructor often leading the way and explaining things as we proceed.

- Lab: You perform this task on your own.
Chef Resources
Objectives

After completing this module, you should be able to:

- Use Chef to manage files on a system
- Use the chef-apply command
- Create a basic Chef recipe file
- Define Chef Resources
Objective:
- Create a file on the system that states 'Hello, world!'
What is a recipe file?

A recipe file is a ruby file that is mostly a collection of resources. It is a fundamental configuration element within Chef.
Resources

A resource is a statement of configuration policy.

It describes the desired state of an element of your infrastructure and the steps needed to bring that item to the desired state.
Objective:
- Change into a directory where we can do work
- Create a recipe file with a resource to generate a file
- Use chef-apply to execute the recipe file

Let's create a recipe file, add a resource, and then apply it to our system.
> cd ~/Desktop
Managing a text file

Let's create a recipe file, add a resource, and then apply it to our system.

Objective:

- Change into a directory where we can do work
- Create a recipe file with a resource to generate a file
- Use chef-apply to execute the recipe file
Open hello.rb in Atom

```ruby
~/Desktop/hello.rb

file 'greeting.txt' do
  content 'Hello, world!'
  action :create
end
```
EXERCISE

Managing a File

Let's create a recipe file, add a resource, and then apply it to our system.

Objective:
- Change into a directory where we can do work
- Create a recipe file with a resource to generate a file
- Use chef-apply to execute the recipe file
> chef-apply --help
> chef-apply hello.rb
Hello, world!
Managing a File

Let's create a recipe file, add a resource, and then apply it to our system.

Objective:

- Change into a directory where we can do work
- Create a recipe file with a resource to generate a file
- Use chef-apply to execute the recipe file
Deleting a File

- Create a recipe file named 'goodbye.rb'
- Add a file resource:

  The file named 'greeting.txt' is deleted.

- Use chef-apply to apply the policy defined in the recipe file
file 'greeting.txt' do
  action :delete
end
> chef-apply goodbye.rb
Q&A

What questions can we answer for you?
Cookbooks
Organizing Recipes for Fun and Profit
Objectives

After completing this module, you should be able to:

- Generate a Chef cookbook
- Write a Chef recipe that builds a web server
- Commit your code to version control
- Spin up an EC2 Linux instance using Test Kitchen
- Destroy and recreate your test instance
Objective:
- Generate a web cookbook
- Create an instance to test the cookbook against
- Write a recipe to install the apache package
- Add content to your homepage
- Update the recipe to start and enable the apache service
Get into your cookbooks directory

> cd ~/chef-repo/cookbooks
Help!

> chef --help

Usage:

chef -h/--help
chef -v/--version
chef command [arguments...] [options...]

Available Commands:
exec Runs the command in context of the embedded ruby
env Prints environment variables used by ChefDK
gem Runs the `gem` command in context of embedded ruby
generate Generate a new app, cookbook, or component
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
...
Even more help...

> chef generate cookbook --help

Usage: chef generate cookbook NAME [options]

- `b, --berks` Generate cookbooks with berkshelf integration
- `-C, --copyright COPYRIGHT` Name of the copyright holder - defaults to 'The Authors'
- `-m, --email EMAIL` Email address of the author - defaults to 'you@example.com'
- `-a, --generator-arg KEY=VALUE` Use to set arbitrary attribute KEY to VALUE in the code_generator
- `-I, --license LICENSE` all_rights, apache2, mit, gplv2, gplv3 - defaults to all_rights

...
Generate a New Cookbook

> chef generate cookbook web

Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[C:/Users/Administrator/chef-repo/cookbooks/web] action create
    - create new directory C:/Users/Administrator/chef-repo/cookbooks/web
  * template[C:/Users/Administrator/chef-repo/cookbooks/web/metadata.rb] action create_if_missing
    - create new file C:/Users/Administrator/chef-repo/cookbooks/web/metadata.rb
    - update content in file C:/Users/Administrator/chef-repo/cookbooks/web/metadata.rb from none to 5367bd
      (diff output suppressed by config)
  * template[C:/Users/Administrator/chef-repo/cookbooks/web/README.md] action create_if_missing
    - create new file C:/Users/Administrator/chef-repo/cookbooks/web/README.md
    - update content in file C:/Users/Administrator/chef-repo/cookbooks/web/README.md from none to 05e1d0
      (diff output suppressed by config)
Objective:

- Generate a web cookbook
- Create an instance to test the cookbook against
- Write the recipe to install the apache package
- Update the recipe to start and enable the apache service
Get into the web cookbook

> cd web
Commit Your Changes

- `git status`
- `git add .`
- `git commit -m "Initial Commit"`

*After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean*
Update your metadata.rb

```ruby
name 'web'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'all_rights'
description 'Installs/Configures web'
long_description 'Installs/Configures web'
version '0.1.0'
```
Basic Git Commands

git status
Edit metadata.rb, save it
git status
git add .
git status
git log
git log --oneline --all --decorate --graph
git commit -m "Customized metadata.rb"
git status
Replace the Existing Kitchen Configuration

> cp ~/chef-repo/.kitchen.linux.yml .kitchen.yml
Edit your `.kitchen.yml` file

`> atom .kitchen.yml`
Put `web::default` in your run_list

```yaml
~/chef-repo/cookbooks/web/.kitchen.yml

...suites:
  - name: default
    run_list:
      - recipe[web::default]

attributes:
```
Commit Your Changes

- git status
- git add .
- git commit -m "Updated run list, added resource tag"

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
Check your instance status

> kitchen list

<table>
<thead>
<tr>
<th>Instance</th>
<th>Driver</th>
<th>Provisioner</th>
<th>Verifier</th>
<th>Transport</th>
<th>Last Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-linux</td>
<td>Ec2</td>
<td>ChefZero</td>
<td>Busser</td>
<td>Ssh</td>
<td>&lt;Not Created&gt;</td>
</tr>
</tbody>
</table>
Create a new test instance

> kitchen create

----> Creating <default-linux>...

If you are not using an account that qualifies under the AWS free-tier, you may be charged to run these suites. The charge should be minimal, but neither Test Kitchen nor its maintainers are responsible for your incurred costs.

Instance <i-5c56adef> requested.

EC2 instance <i-5c56adef> created.

Waited 0/300s for instance <i-5c56adef> to become ready.

Waited 5/300s for instance <i-5c56adef> to become ready.

Waited 10/300s for instance <i-5c56adef> to become ready.

Waited 15/300s for instance <i-5c56adef> to become ready.
Show your instance status

<table>
<thead>
<tr>
<th>Instance</th>
<th>Driver</th>
<th>Provisioner</th>
<th>Verifier</th>
<th>Transport</th>
<th>Last Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>default-linux</td>
<td>Ec2</td>
<td>ChefZero</td>
<td>Busser</td>
<td>Ssh</td>
<td>Created</td>
</tr>
</tbody>
</table>
Objective:

✓ Generate a web cookbook
✓ Create an instance to test the cookbook against
☐ Write the recipe to install the apache package
☐ Update the recipe to start and enable the apache service
Add a Package Resource

```ruby
package 'httpd' do
  action :install
end
```
Run Chef on your instance

> kitchen converge

-----> Converging <default-linux>...
Preparing files for transfer
Preparing dna.json
Resolving cookbook dependencies with Berkshelf 4.0.1...
Removing non-cookbook files before transfer
Preparing validation.pem
Preparing client.rb
...

...
Log onto your instance

> kitchen login

Last login: Fri Dec 25 22:46:30 2015 from ip-10-100-20-236.ec2.internal
[centos@ip-10-100-20-167 ~]$
Test your Changes

```
> rpm -q httpd

httpd-2.4.6-40.el7.centos.x86_64
```
Remove the Package Manually

> rpm -e httpd
Run Chef again to repair the machine...

> kitchen converge

-----> Converging <default-linux>...
Preparing files for transfer
Preparing dna.json
Resolving cookbook dependencies with Berkshelf 4.0.1...
Removing non-cookbook files before transfer
Preparing validation.pem
Preparing client.rb
...


Test your Changes

`> rpm -q httpd`

`httpd-2.4.6-40.el7.centos.x86_64`
Commit Your Changes

- git status
- git add .
- git commit -m "Added a package resource."

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
Add a file resource

```ruby
package 'httpd' do
  action :install
end

file '/var/www/html/index.html' do
  content 'Hello, world!'
end
```
Converge your node again

> kitchen converge
Test your Changes

`> cat /var/www/html/index.html`

Hello world!
Commit Your Changes

- git status
- git add .
- git commit -m "Added a file resource."

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
Add a service resource

```
package 'httpd' do
  action :install
end

file '/var/www/html/index.html' do
  content 'Hello, world!' 
end

service 'httpd' do
  action [ :enable, :start ]
end
```
Converge your node again

> kitchen converge
Test your Changes

Hello world!
Commit Your Changes

- git status
- git add .
- git commit -m "Added a service resource."

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
Objective:
- Generate a web cookbook
- Create an instance to test the cookbook against
- Write the recipe to install the apache package
- Update the recipe to start and enable the apache service
Bonus Exercise – Disaster Recovery

- kitchen destroy
- kitchen converge
Bonus Exercise – Time Travel

- `git log --oneline`
- `git checkout f947881`
- `git checkout master`

You can go back to any previous commit if you need to!
Q&A

What questions can we answer for you?
Testing Cookbooks
Objectives

After completing this module, you should be able to:

- Write a test in ServerSpec
- Verify the test passes with Test Kitchen
Test Kitchen

What does this test when kitchen converges a recipe?

What does it NOT test when kitchen converges a recipe?
Test Kitchen

What is left to validate to ensure that the cookbook successfully applied the policy defined in the recipe?
Objective:
- Write a test to validate a working web server
- Execute the test to see it pass
require 'spec_helper'

describe 'web::default' do
  
  # Serverspec examples can be found at
  # http://serverspec.org/resource_types.html
  it 'does something' do
    skip 'Replace this with meaningful tests'
  end

  describe command('curl localhost') do
    its(:stdout) { should match(/ip-204-54-11-27/) }
  end
end
Objective:

- Write a test to validate a working web server
- Execute the test to see it pass

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

> kitchen verify

```
web::default
    Command "curl localhost"
    stdout
        should match /Hello World!/

    Finished in 0.18118 seconds (files took 0.26053 seconds to load)
    1 example, 0 failures
```
Commit Your Changes

- git status
- git add .
- git commit -m "Wrote test to verify homepage is working"

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?

**Objective:**
- Write a test to validate a working web server
- Execute the test to see it pass
Q&A

What questions can we answer for you?
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
Status Page – Hello World v2.0

We can do better than 'Hello, world!'

Objective:

- Update the default test page to include the host name and IP address of the system
- Converge and verify that the web page displays the host name and IP address
- Add other system data and an image to our homepage
Find our Public IP Address & Hostname

- `hostname`

52.90.82.1
ip-10-100-20-167
Add the data to our recipe

```ruby
package 'httpd' do
  action :install
end

file '/var/www/html/index.html' do
  content '
52.90.82.1
ip-10-100-20-167
'
end

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

> kitchen converge

...,
- update content in file /var/www/html/index.html from 7f83b1 to 6228bf
@@ -1,2 +1,4 @@
-Hello World!
+
+52.90.82.1
Run your Test

> kitchen verify
Commit Your Changes

- git status
- git add .
- git commit -m "Changed index.html into status page"

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
EXERCISE

Status Page – Hello World v2.0

Adding more system data to our status page

Objective:
✓ Update the default test page to include the host name and IP address of the system
✓ Converge and verify that the web page displays the host name and IP address
☐ Add other system data and an image to our homepage
Generate Content for our Homepage

- sh sysinfo.sh

<html><body>
<img src=https://upload.wikimedia.org/wikipedia/commons/thumb/1/1d/AmazonWebservices_Logo.svg/2000px-AmazonWebservices_Logo.svg.png width=200>
<pre>
Hostname:   ip-10-100-20-167
IP:         52.90.82.1
Memory:     1015944 kB
# CPUs:     1
Kernel:     3.10.0-229.14.1.el7.x86_64
OS Version: CentOS Linux release 7.1.1503 (Core)
</pre></body></html>
Copy Pasta, Paisano!

```ruby
file '/var/www/html/index.html' do
  action :create
  content:
  <html><body>
  <img src=https://upload.wikimedia.org/wikipedia/commons/thumb/1/1d/AmazonWebservices_L.png/>
  <pre>
  Hostname:  ip-10-100-20-167
  IP:        52.90.82.1
  Memory:    1015944 kB
  # CPUs:    1
  Kernel:    3.10.0-229.14.1.el7.x86_64
  OS Version: CentOS Linux release 7.1.1503 (Core)
  </pre></body></html>
end
```
Converge Again

> kitchen converge

... update content in file /var/www/html/index.html from 7f83b1 to 6228bf


...
Run your Test

> kitchen verify
Commit Your Changes

- `git status`
- `git add .`
- `git commit -m "Added more data to status page."`

After committing, your git status should be clean:

On branch master

nothing to commit, working directory clean
EXERCISE

Status Page – What's Next?

In the next chapter we'll make it more dynamic...

Objective:

- Update the default test page to include the host name and IP address of the system
- Converge and verify that the web page displays the host name and IP address
- Add other system data and an image to our homepage
Q&A

What questions can we answer for you?
Managing Files with Chef Templates

Learn to separate code from 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
Cleaner Apache Recipe

Adding all this HTML to the index page made it somewhat unwieldy and harder to manage.

Objective:
- Create a template with chef generate
- Define the contents of the ERB template
- Change the file resource to the template resource in the 'web' cookbook
How do we generate a template?

> chef generate template --help
Generate a Template for our Homepage

> chef generate template index.html
Clean up the code

Storing files inside a recipe is not efficient. Let's move all that code into the template.

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
Move the HTML into our new template

```
~/chef-repo/cookbooks/web/templates/default/index.html.erb

<html>
  <body>
    <img src=https://upload.wikimedia.org/wikipedia/commons/thumb/1/1d/AmazonWebservice.png/s_Logo.svg/2000px-AmazonWebservices_Logo.svg.png width=200>
    <pre>
      Hostname: ip-10-100-20-229
      IP: 54.84.76.148
      Memory: 1015944 kB
      # CPUs: 1
      Kernel: 3.10.0-229.14.1.el7.x86_64
      OS Version: CentOS Linux release 7.1.1503 (Core)
    </pre>
  </body>
</html>
```
Update your recipe to look like this:

```ruby
package 'httpd' do
  action :install
end

template '/var/www/html/index.html' do
  action :create
  source 'index.html.erb'
end

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

> kitchen converge

... update content in file /var/www/html/index.html from 7f83b1 to 6228bf
...

> kitchen converge
Run your Test

> kitchen verify
Objective:

- Update the version of the 'web' cookbook's version for this patch
- Commit the changes to the 'web' cookbook to version control
Bump your cookbook version

```ruby
name 'web'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'all_rights'
description 'Installs/Configures web cookbook'
long_description 'Installs/Configures web cookbook'
version '0.2.0'
```
Commit Your Changes

- `git status`
- `git add .`
- `git commit -m "Moved HTML code into template, bumped version."`

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
Q&A

What questions can we answer for you?
Chef Templates and Node Attributes

More refactoring! Let's make this recipe more dynamic.
Objectives

After completing this module, you should be able to:

- Use Ohai to gather data about your instance
- Use ERB tags to display node data in a template
- Create dynamic, automatically updating templates
Objective:
- Use the Ohai tool to examine the state of our machine and its attributes
- Remove hard-coded data from our template, replace with attributes
- Test and verify that our homepage is updated dynamically
Ohai!

Ohai is a tool that already captures all the data that we dug up manually in the previous sections.

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

```json
{
    "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": {}
        }
    }
}
```
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 + 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
The node object is a representation of our system. It stores all the attributes found about the system.

http://docs.chef.io/nodes.html#attributes
The Node

IPADDRESS: 104.236.192.102
The Node

HOSTNAME: banana-stand
The Node

MEMORY: 502272kB
The Node

CPU: 2399.998MHz
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

```erb
<% 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 %>

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

```erb
<% if (50 + 50) == 100 %>  
50 + 50 = <%= 50 + 50 %>  
</% if %>

<% else %>
At some point all of MATH I learned in school changed.
</% else %>

<% end %>

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

The Angry Squid

< % =
Update your template

<pre>
Hostname: <%= node['hostname'] %>
IP: 54.84.76.148
Memory: 1015944 kB
# CPUs: 1
Kernel: 3.10.0-229.14.1.el7.x86_64
OS Version: CentOS Linux release 7.1.1503 (Core)
Zone: us-east-1a
</pre>
Converge the Instance

> kitchen converge

```bash
... update content in file /var/www/html/index.html from 7f83b1 to 6228bf
...```
Update your template

```html
<%=
  Hostname: node['hostname']

  IP: % node['cloud']['public_ipv4']

  Memory: 1015944 kB

  # CPUs: 1

  Kernel: 3.10.0-229.14.1.el7.x86_64

  OS Version: CentOS Linux release 7.1.1503 (Core)

  Zone: us-east-1a

%>
```
Add node attributes for the other items:

```python
node['memory']['total']
node['cpu']['total']
node['kernel']['release']
node['platform']
node['platform_version']
node['ec2']['placement_availability_zone']
```
<html>
  <body>
    <img src=https://upload.wikimedia.org/wikipedia/commons/thumb/1/1d/AmazonWebservices_Logo.svg>
    <pre>
      Hostname: <%= node['hostname'] %>
      IP: <%= node['cloud']['public_ipv4'] %>
      Memory: <%= node['memory']['total'] %>
      # CPUs: <%= node['cpu']['total'] %>
      Kernel: <%= node['kernel']['release'] %>
      OS Version: <%= node['platform'] -%> <%= node['platform_version'] %>
      Zone: <%= node['ec2']['placement_availability_zone'] %>
    </pre>
  </body>
</html>
Converge the Instance

> kitchen converge

...  
- update content in file /var/www/html/index.html from 7f83b1 to 6228bf
  
--- /var/www/html/index.html
  2015-12-25 22:50:09.497834997 +0000

+++ /var/www/html/.index.html
  2015-12-25 23:14:51.505056903 +0000

@@ -1,2 +1,4 @@

-Hello World!

+ 
  
+52.90.82.1
Run your Test

> kitchen verify
Commit Your Changes

- `git status`
- `git add .`
- `git commit -m "Converted template to use dynamic attributes"

After committing, your git status should be clean:

On branch master
nothing to commit, working directory clean
What questions can we answer for you?
User-defined Attributes

What if we need custom attributes to describe our infrastructure?
Objectives

After completing this module, you should be able to:

- Generate a chef default attributes file
- Define custom attributes that you can use in templates
- Build a simple web app with custom attributes
How do we generate an attributes file?

> chef generate attribute --help
Generate a default attributes file

> chef generate attribute default

Compiling Cookbooks...

Recipe: code_generator::attribute
  * directory[C:/Users/Administrator/chef-repo/cookbooks/web/attributes] action create
    - create new directory C:/Users/Administrator/chef-repo/cookbooks/web/attributes
  * template[C:/Users/Administrator/chef-repo/cookbooks/web/attributes/default.rb] action create
    - create new file C:/Users/Administrator/chef-repo/cookbooks/web/attributes/default.rb
    - update content in file C:/Users/Administrator/chef-repo/cookbooks/web/attributes/default.rb from none to e3b0c4

(diff output suppressed by config)
Fetch some code for your webapp

http://www.kongregate.com/games_for_your_site
Insert the code into your template

```html
<embed width="720" height="480"
base="http://external.kongregate-games.com/gamez/0020/3"
src="http://external.kongregate-games.com/gamez/0020/38"
type="application/x-shockwave-flash"></embed><br/>Play
<a href="http://www.kongregate.com/">Kongregate</a>
```
Converge the Instance

> kitchen converge

... update content in file /var/www/html/index.html from 7f83b1 to 6228bf

...
Edit the default.rb attributes file:

```
default['web']['width'] = '1280'
default['web']['height'] = '768'
```
Update your template with the new attributes:

```html
<embed width="<%= node["web"]["width"] -%>" height="<%= node["web"]["height"] -%>"
base="<%= node["web"]["base"] %>
src="<%= node["web"]["src"] %>"
```
Converge the Instance

> kitchen converge

... update content in file /var/www/html/index.html from 7f83b1 to 6228bf
+++ /var/www/html/.index.html20151225-12310-5fw0wy 2015-12-25 23:14:51.505056903 +0000
...

> kitchen converge
Test your app in a browser