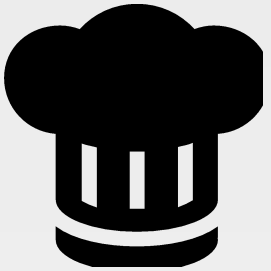
Managing a Large Number of Servers

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

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



Details About the Node

Displaying system details in the MOTD definitely sounds useful.

Objective:

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

Some Useful System Data

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

Discover the IP Address

```
$ hostname -I
```

```
172-31-57-153
```

Discover the Host Name

```
$ hostname
```

```
ip-172-31-57-153
```

Discovering the Memory

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

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

Discover the CPU - MHz

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


CHEF

Adding the CPU

~/cookbooks/workstation/recipes/setup.rb

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

  IPADDRESS: 172-31-57-153
  HOSTNAME : ip-172-31-57-153
  MEMORY   : 502272 kB
  CPU      : 2399.998 MHz
'
  mode '0644'
  owner 'root'
  group 'root'
end
```

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)
```

Verify that the /etc/motd Has Been Updated

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

```
Property of ...

  IPADDRESS: 172-31-57-153
  HOSTNAME : ip-172-31-8-68
  MEMORY   : 605048 kB
  CPU      : 1795.672
```


CHEF

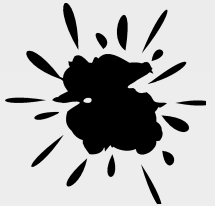
DISCUSSION

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?



Hard Coded Values

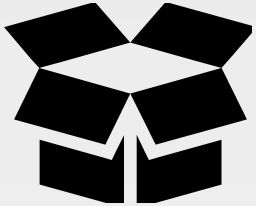
The values that we have derived at this moment may not be the correct values when we deploy this recipe again even on the same system!

DISCUSSION

Data In Real Time

How could we capture this data in real-time?

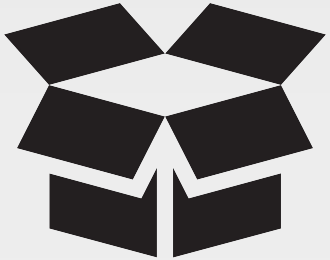
CONCEPT



Ohai!

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

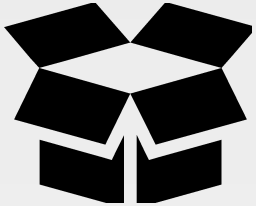
http://docs.chef.io/ohai.html



Ohai!

Ohai is a tool that is used to detect attributes on a node, and then provide these attributes to the chef-client at the start of every chef-client run. Ohai is required by the chef-client and must be present on a node. (Ohai is installed on a node as part of the chef-client install process.)

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



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

Running Ohai to Show All Attributes

```
> ohai
```


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


CHEF

Running Ohai to Show the IP Address

```
> ohai ipaddress
```

```
[
  "172.31.57.153"
]
```


CHEF

Running Ohai to Show the Hostname

```
> ohai hostname
```

```
[
  "ip-172-31-57-153"
]
```

Running Ohai to Show the Memory

```
> ohai memory
```

```
{
  "swap": {
    "cached": "0kB",
    "total": "0kB",
    "free": "0kB"
  },
  "total": "604308kB",
  "free": "297940kB",
  "buffers": "24824kB",
  "cached": "198264kB",
```

Running Ohai to Show the Total Memory

```
> ohai memory/total
```

```
[
  "604308kB"
]
```


CHEF

Running Ohai to Show the CPU

```
> ohai cpu
```

```
{
  "0": {
    "vendor_id": "GenuineIntel",
    "family": "6",
    "model": "45",
    "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",
    "stepping": "7",
    "mhz": "1795.673",
    "cache_size": "20480 KB",
    "physical_id": "34
```

Running Ohai to Show the First CPU

```
> ohai cpu/0
```

```
{
  "vendor_id": "GenuineIntel",
  "family": "6",
  "model": "45",
  "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",
  "stepping": "7",
  "mhz": "1795.673",
  "cache_size": "20480 KB",
  "physical_id": "34",
  "core_id": "0",
```

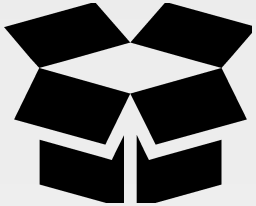
Running Ohai to Show the First CPU Mhz

```
> ohai cpu/0/mhz
```

```
[
  "1795.673"
]
```


CHEF

CONCEPT

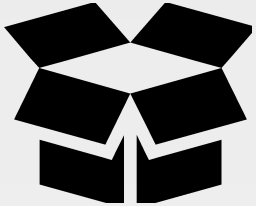


ohai + chef-client = <3

chef-client and chef-apply automatically executes ohai and stores the data about the node in an object we can use within the recipes named node.

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

CONCEPT

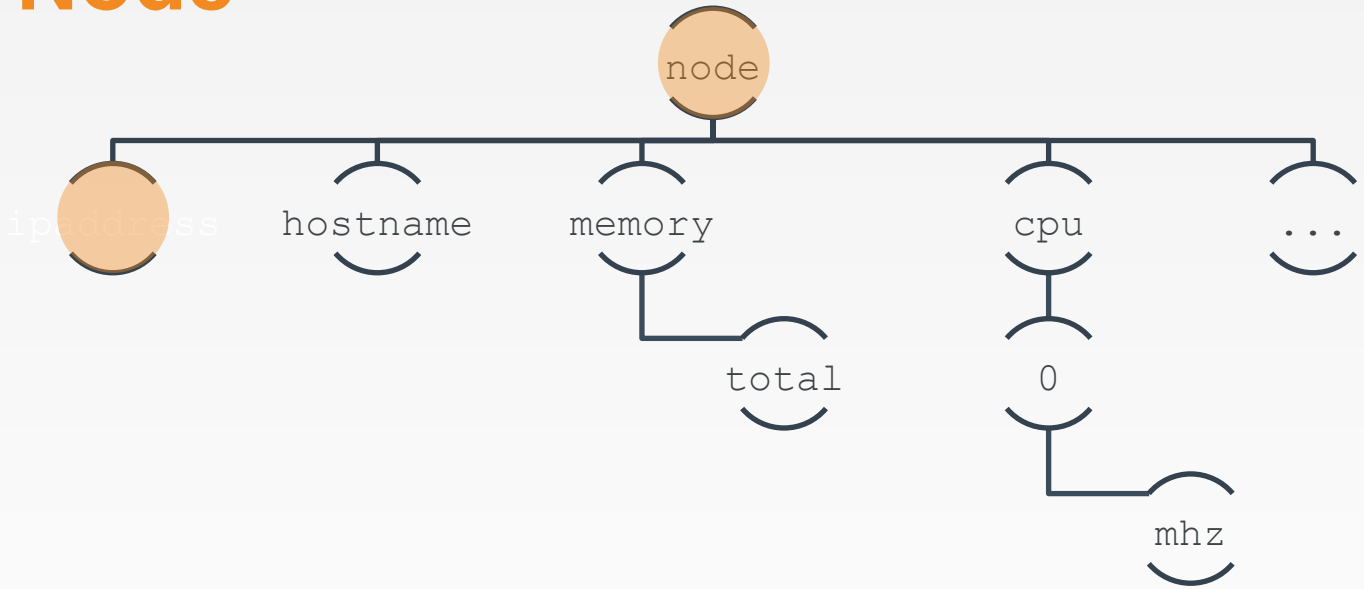


The Node Object

The node object is a representation of our system.
It stores all the attributes found about the system.

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

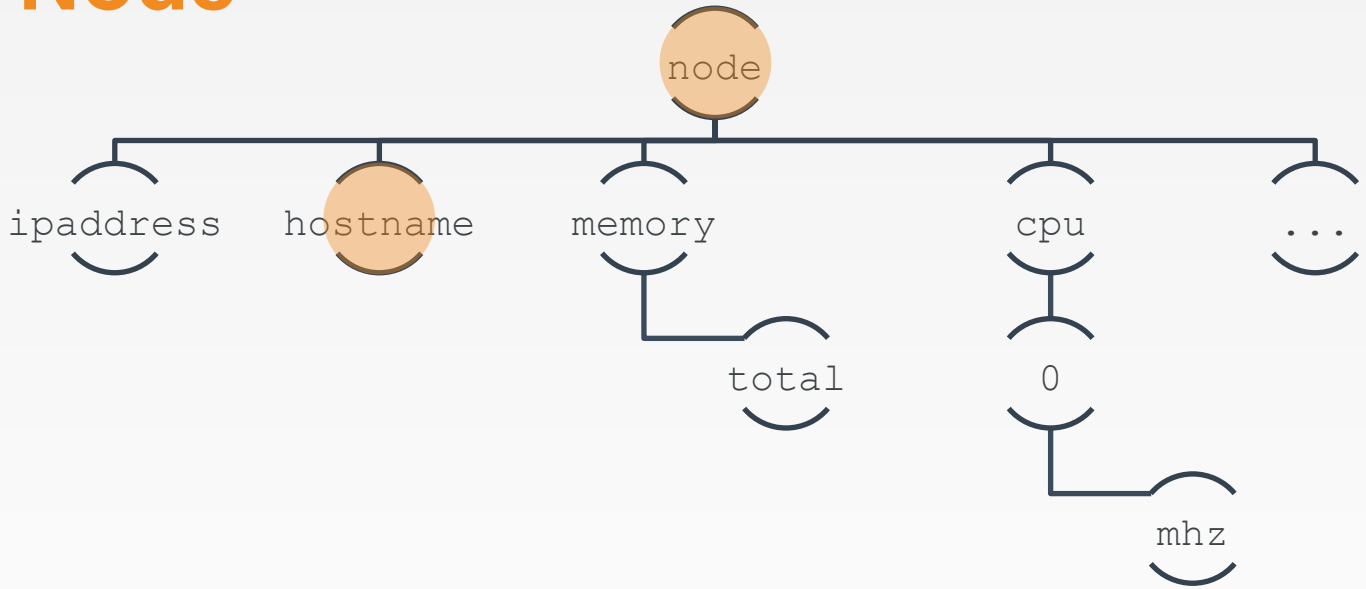
The Node




```
IPADDRESS: 172-31-57-153
```

```
node['ipaddress']
```

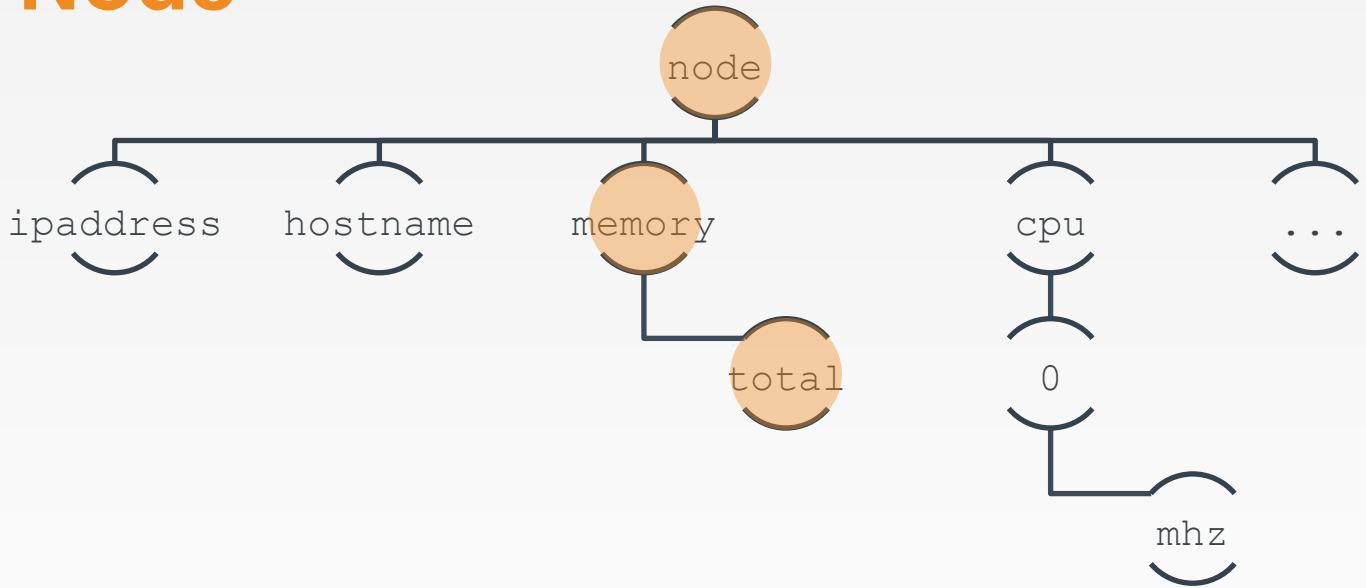
The Node




```
HOSTNAME: ip-172-31-57-153
```

```
node['hostname']
```

The Node

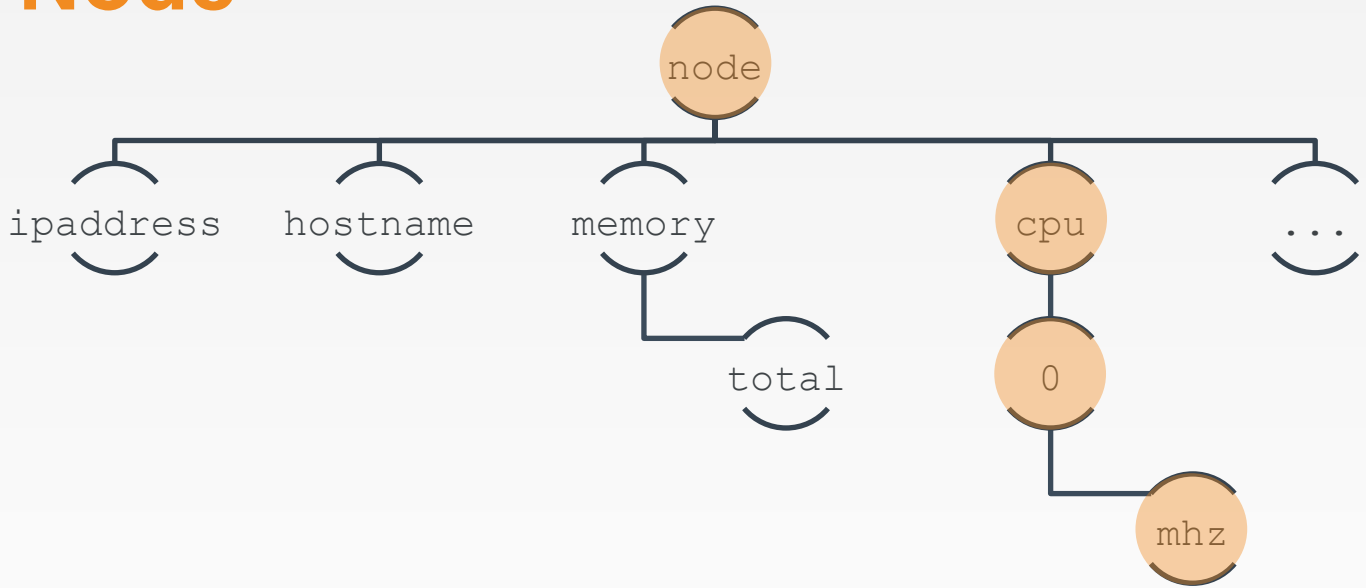



```
MEMORY: 502272kB
```

```
node['memory']['total']
```


CHEF

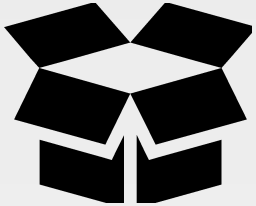
The Node




```
CPU: 2399.998MHz
```

```
node['cpu']['0']['mhz']
```

CONCEPT



String Interpolation

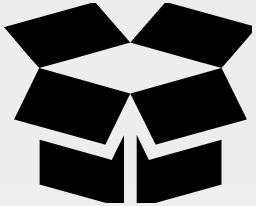
```
I have 4 apples
```

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

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

CHEF

CONCEPT



String Interpolation

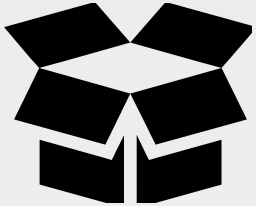
```
I have 4 apples
```

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

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

CHEF

CONCEPT



String Interpolation

```
I have 4 apples
```

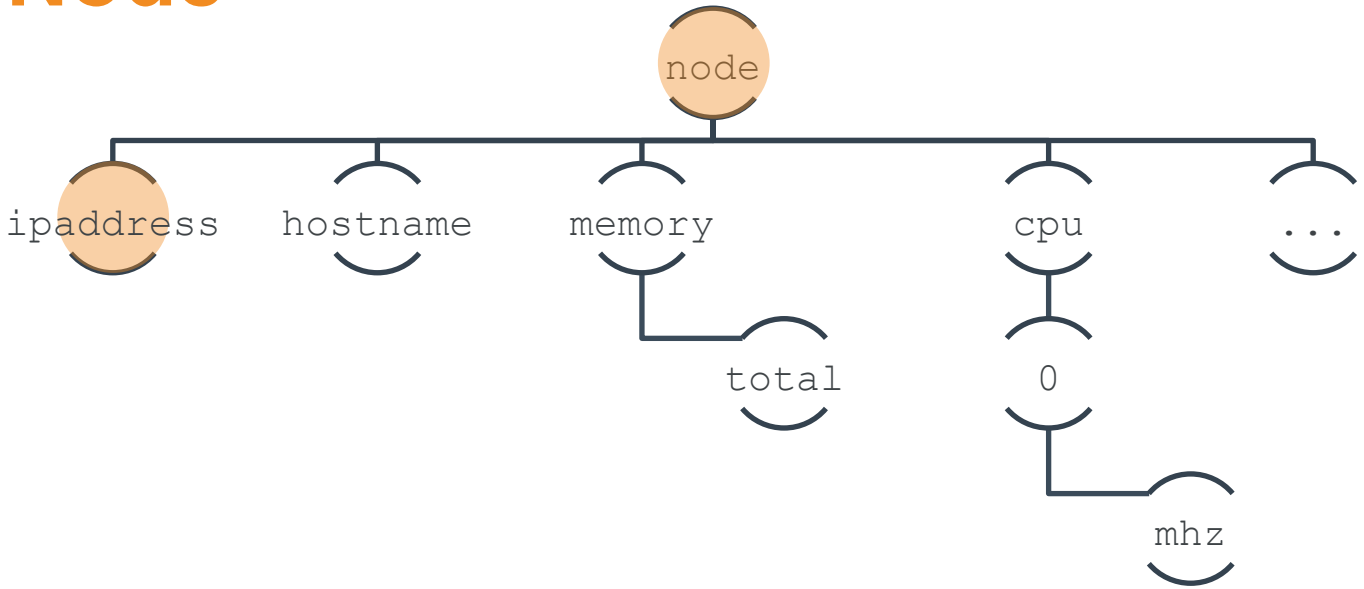
```
apple_count = 4
puts "I have #{apple_count} apples"
```


CHEF

How do I access attributes inside of a recipe?

- node[‘attribute’] resolves to the value of the attribute
- Use string interpolation to access node values inside of a string, like the ‘content’ property in the files resource

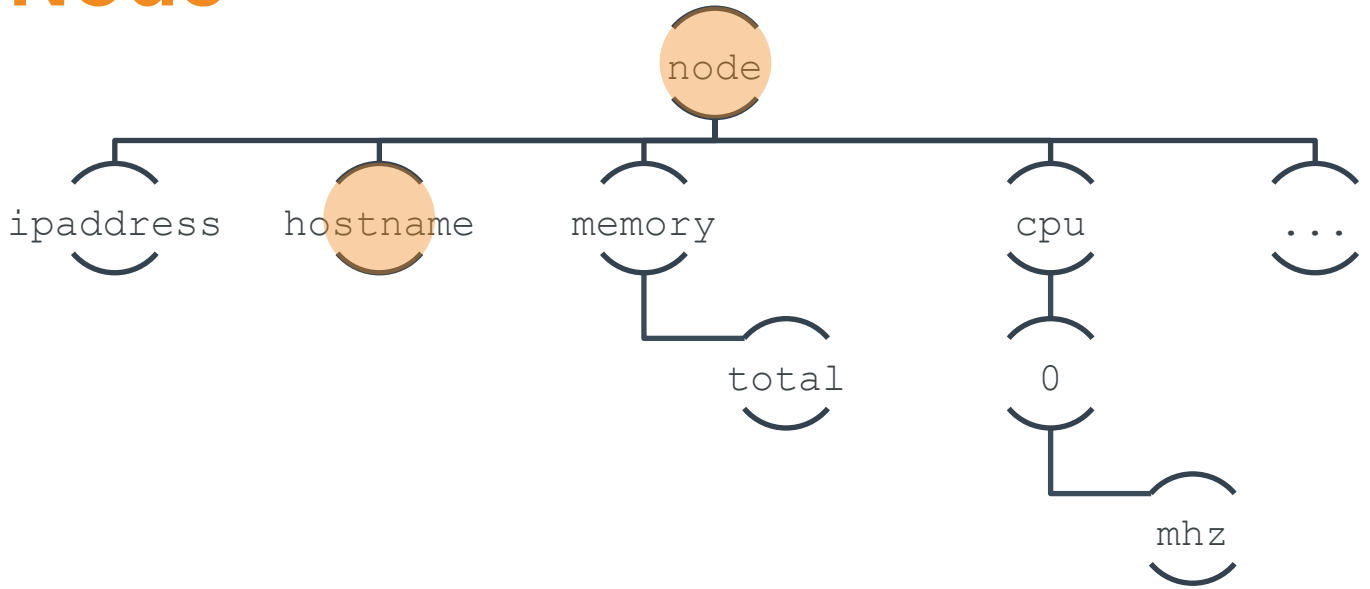
The Node




```
IPADDRESS: 104.236.192.102
```

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

The Node

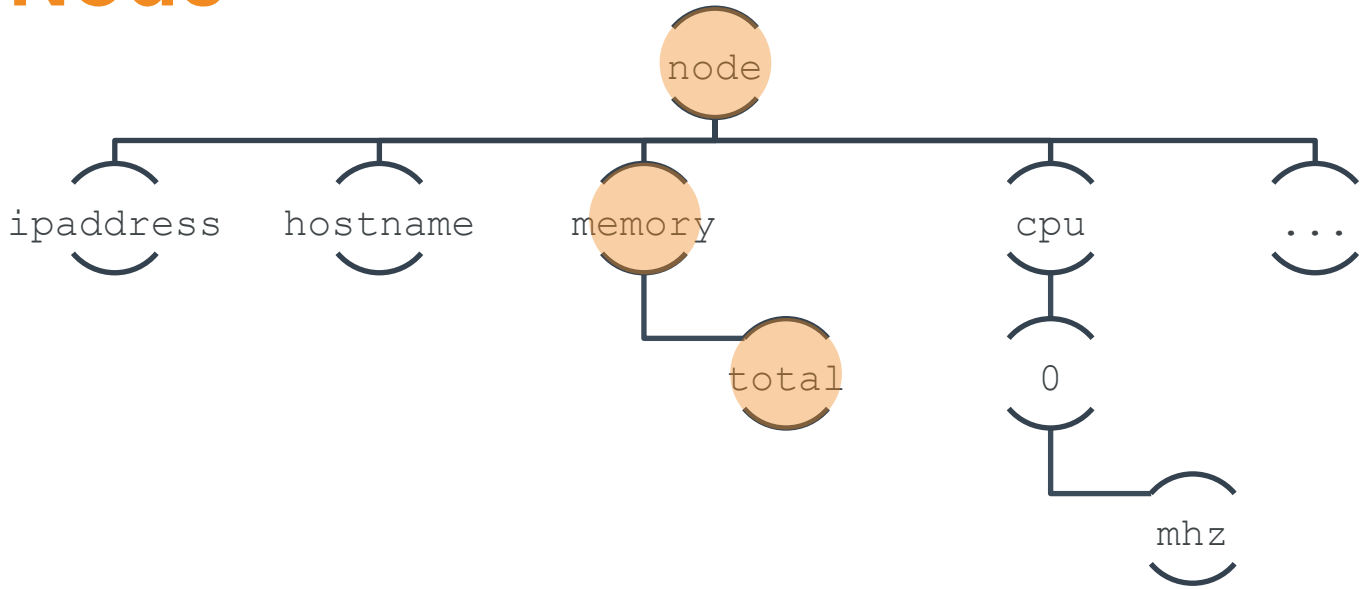



```
HOSTNAME: banana-stand
```

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


CHEF

The Node

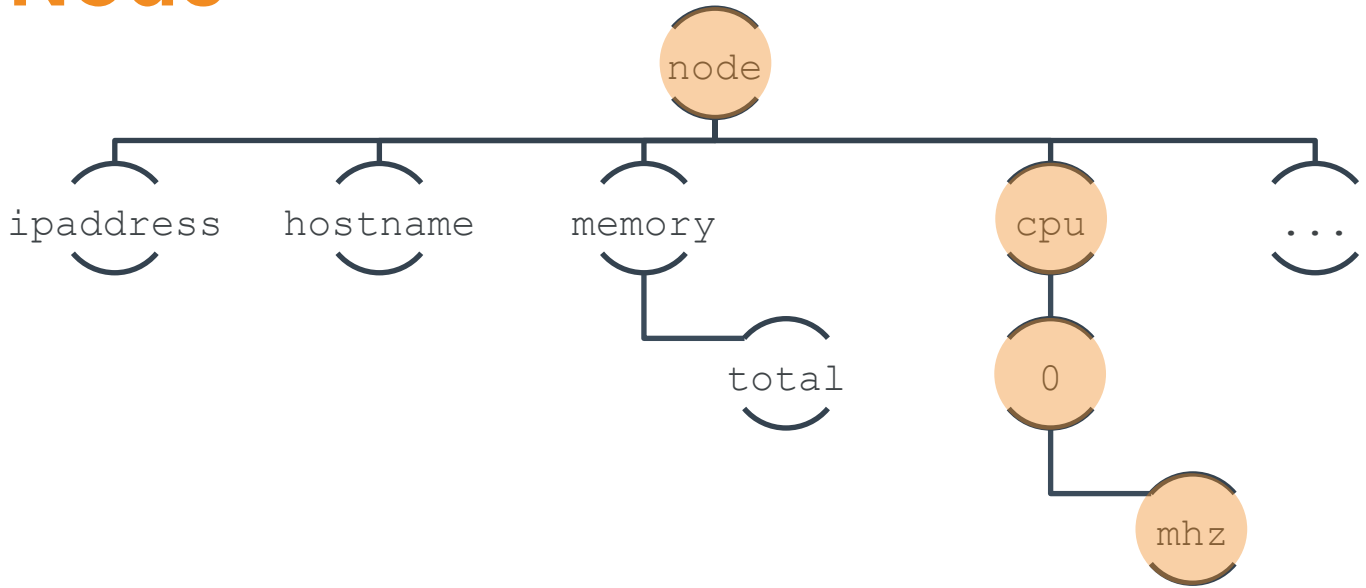



```
MEMORY: 502272kB
```

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


CHEF

The Node




```
CPU: 2399.998 MHz
```

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

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
```

Verify the Changes

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

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...
```

Verifying that the MOTD has been Updated

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

```
Property of ...

  IPADDRESS: 172.31.57.153
  HOSTNAME : ip-172-31-57-153
  MEMORY   : 604308kB
  CPU      : 1795.673
```

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

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

...
```

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
```

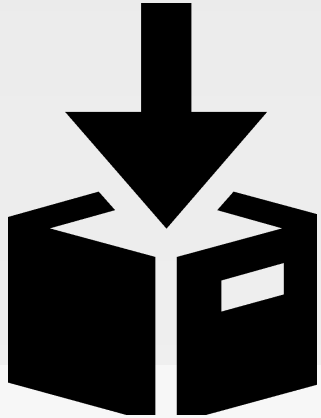
Lab: Update the Cookbook Version

~/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'
```

COMMIT

Lab: Commit Your Work



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

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

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?

Using Templates

Extracting the Content for Clarity

CHEF

Cleaner Recipes

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

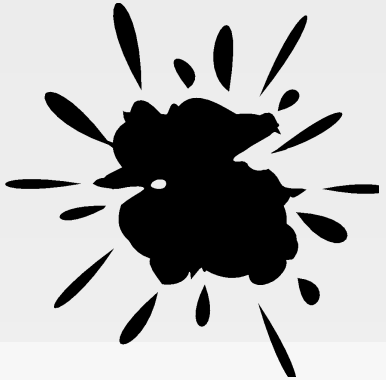
We added this content to the file resource in their respective recipes.

Viewing the workstation's setup recipe

~/cookbooks/workstation/recipes/setup.rb

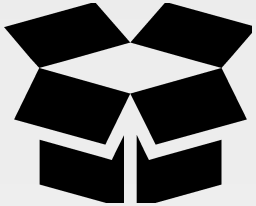
```
package 'tree'

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

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

Double Quotes Close Double Quotes



Double quoted strings are terminated by double quotes.

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

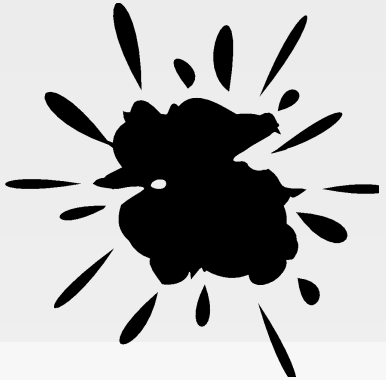


Backslash

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

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


CHEF



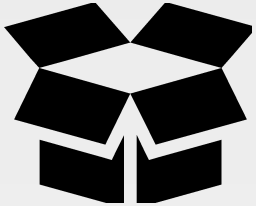
Backslash

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

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


CHEF

CONCEPT



Backslash

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

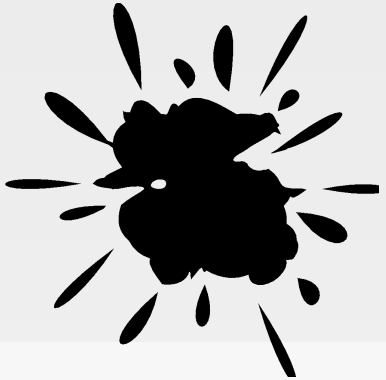
```
"Root Path: \\"
```


CHEF

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
```

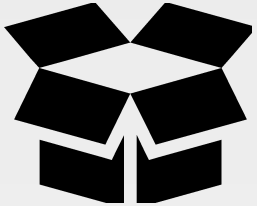
```
This is the first line of the file.
           This is the second line. If I try and line
it up...


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.

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/apache/templates/default
templates/default
└── index.html.erb

0 directories, 1 file
```

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

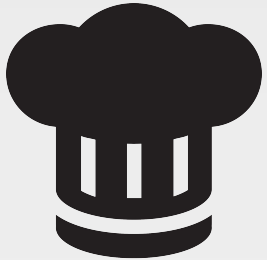
Template

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

EXERCISE



Let's Add a Template!

Adding all the information into the recipe did make it hard to read.

Objective:

- ❑ Create a template with chef generate
- ❑ Define the contents of the ERB template
- ❑ Change the file resource to the template resource
- ❑ Update the cookbook's version number
- ❑ Apply the updated recipe and verify the results

GL: What Can `chef generate` Do?

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

```
UsaGL: 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)
```


CHEF

GL: What Can `chef generate template` Do?

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

```
UsaGL: 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
```


CHEF

Generating a motd Template

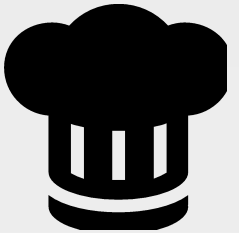
```
> chef generate template cookbooks/workstation motd
```

```
Recipe: code_generator::template
  * directory[cookbooks/workstation/templates/default] action
create
    - create new directory
cookbooks/workstation/templates/default
  * template[cookbooks/workstation/templates/motd.erb] action
create
    - create new file cookbooks/workstation/templates/motd.erb
    - update content in file
cookbooks/workstation/templates/motd.erb from none to e3b0c4
    (diff output suppressed by config)
```

Examining the templates Directory

```
> tree cookbooks/workstation/templates
```

```
cookbooks/workstation/templates/
└── default
└── motd.erb

1 directory, 1 file
```



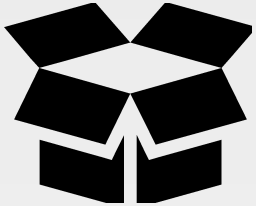
Cleaner Recipes

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 'apache' 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

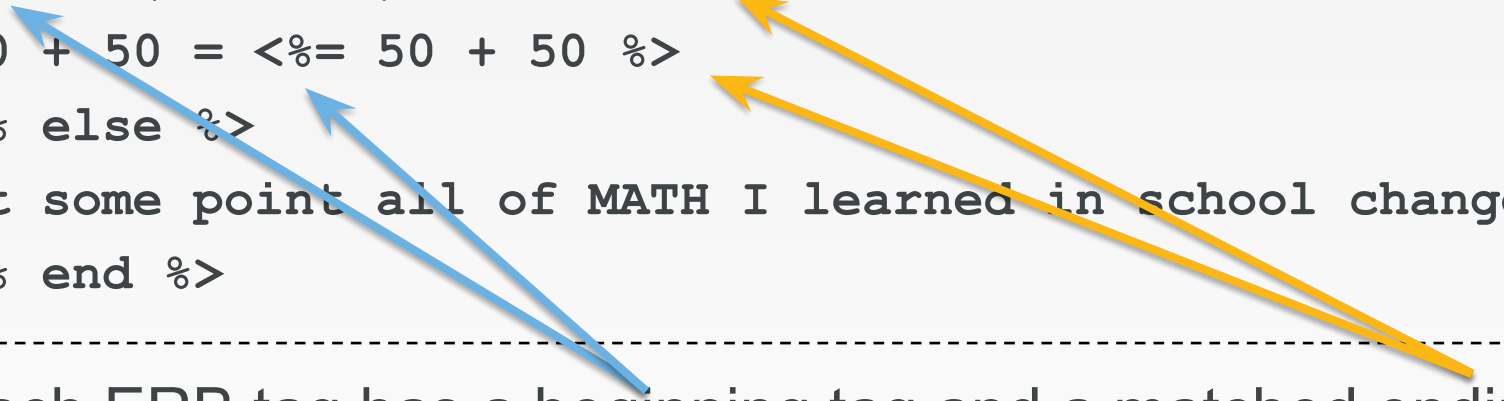
Text Within an ERB Template

```
<% 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

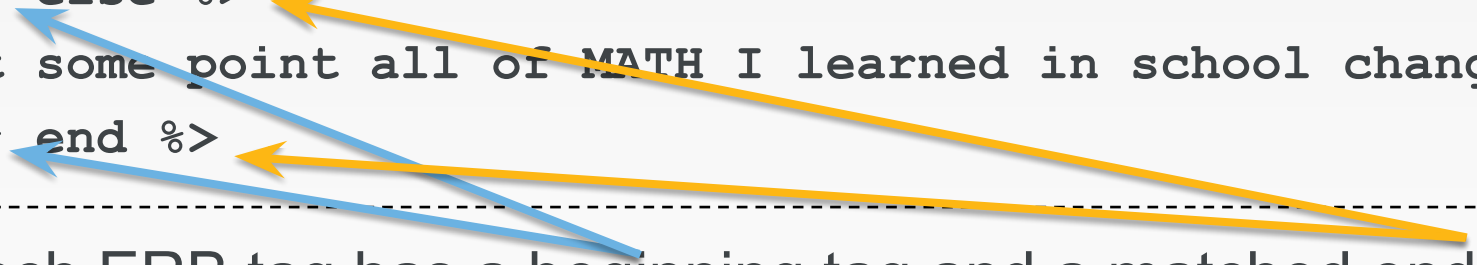
```
<% if (50 + 50) == 100 %>
50 + 50 = <%= 50 + 50 %>
<% else %>
At some point all of MATH I learned in school changed.
<% end %>
```



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 %>
```

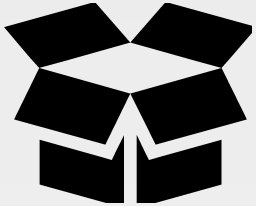
Executes the ruby code within the brackets and do not display the result.

Text Within an ERB Template

```
<% 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

<%=

CHEF

Copying the Existing Content into the Template

~/cookbooks/workstation/templates/motd.erb

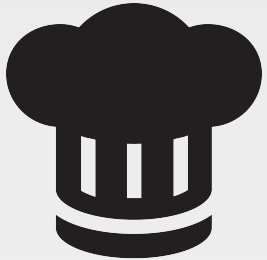
```
Property of ...

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

Changing String Interpolation to ERB Tags

~/cookbooks/workstation/templates/motd.erb

```
Property of ...

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

EXERCISE



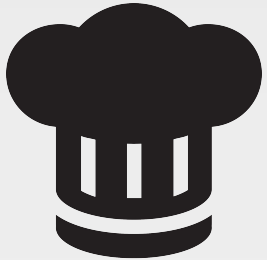
Cleaner Recipes

The template is created and defined. It now needs to be used within the recipe.

Objective:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results

EXERCISE



Using the Template Resource

Adding all the information into the recipe did make it hard to read.

Objective:

- ❑ Create a template with chef generate
- ❑ Define the contents of the ERB template
- ❑ Change the file resource to the template resource
- ❑ Update the cookbook's version number
- ❑ Apply the updated recipe and verify the results

Removing the file Resource

~/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']}
"
end
```

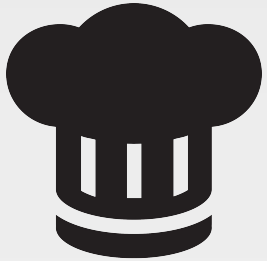

CHEF

Changing from file to template Resource

~/cookbooks/workstation/recipes/setup.rb

```
# ... PACKAGE RESOURCES ...

template '/etc/motd' do
  source 'motd.erb'
end
```

EXERCISE



Cleaner Recipes

This is a change to the cookbook so it is time to update the version again.

Objective:

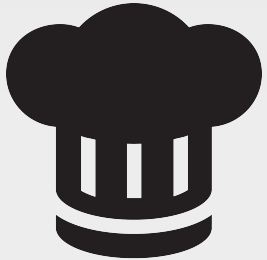
- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results

Updating the Cookbook's Version 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'
```

EXERCISE



Cleaner Recipes

This is a change to the cookbook so it is time to update the version again.

Objective:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results

Applying the Updated Cookbook

```
> sudo chef-client --local-mode --runlist "recipe[workstation]"
```

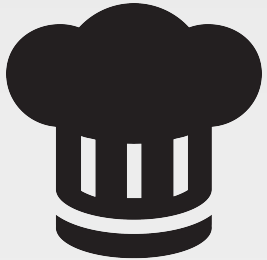
```
- workstation (0.2.1)
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * template[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/2 resources updated in 12 seconds
```

Verifying the Contents of the MOTD File

```
> cat /etc/motd
```

```
Property of ...

  IPADDRESS: 172.31.57.153
  HOSTNAME : ip-172-31-57-153
  MEMORY   : 604308kB
  CPU      : 1795.673
```

EXERCISE



Cleaner Recipes

This is a change to the cookbook so it is time to update the version again.

Objective:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ✓ Change the file resource to the template resource
- ✓ Update the cookbook's version number
- ✓ Apply the updated recipe and verify the results

Discussion

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

What are the two types of ERB tags we talked about?

What do each of the ERB tags accomplish?

Lab: Use the Template

For the "apache" cookbook:

- ❑ Use chef generate to create a template named “index.html.erb".
- ❑ Copy the content property from the file named '/var/www/html/index.html' into the template file "index.html.erb"
- ❑ Remove a resource: The file resource named '/var/www/html/index.html'
- ❑ Add a resource: The template named '/var/www/html/index.html' is created with the source ‘index.html.erb'
- ❑ Run chef-client to locally apply the default recipe from the apache cookbook

GL: What Can `chef generate template` Do?

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

```
UsaGL: 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
```


CHEF

GL: 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)
```


CHEF

GL: Lets Look at the Template File

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

1 directory, 1 file
```


CHEF

GL: Move Our Source to the Template

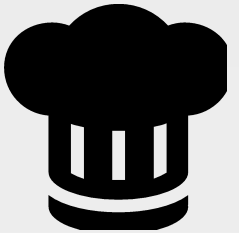
~/cookbooks/apache/templates/default/index.html.erb

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

GL: Move Our Source to the Template

~/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>
```



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

GL: Remove the Existing Content Attribute

~/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
```

GL: Change File Resource to a Template Resource

~/cookbooks/apache/recipes/server.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/apache/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/apache/templates/default
cookbooks/apache/templates/default/
└── index.html.erb

0 directories, 1 file
```


CHEF

GL: Update the Source Attribute

~/cookbooks/apache/recipes/server.rb

```
package 'httpd'

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

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

Lab: Change Directories and Apply the Cookbook

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

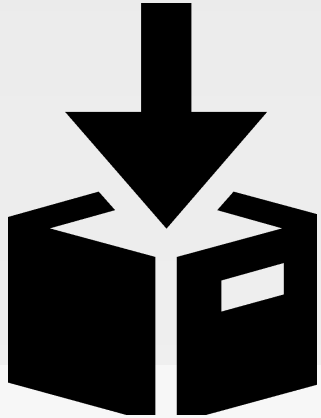
```
[2016-09-16T14:18:05+00:00] WARN: No config file found or specified on command line, using command line options.
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
[2016-09-16T14:18:09+00:00] WARN: Cloning resource attributes for service[httpd] from prior resource (CHEF-3694)
[2016-09-16T14:18:09+00:00] WARN: Previous service[httpd]: /root/.chef/local-mode-cache/cache/cookbooks/apache/recipes/server.rb:8:in `from_file'
[2016-09-16T14:18:09+00:00] WARN: Current  service[httpd]: /root/.chef/local-mode-cache/ ...
```

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'
```

Lab: Commit the Changes



```
$ cd ~/cookbooks/apache
$ git add .
$ git status
$ git commit -m "Update default recipe to use template"
```

The Chef Server

A Hub for Configuration Data

CHEF

Managing an Additional System

To manage another system, you would need to:

1. Provision a new node within your company or appropriate cloud provider with the appropriate access to login to administrate the system.
2. Install the Chef tools.
3. Transfer the apache cookbook.
4. Run chef-client on the new node to apply the apache cookbook's default recipe.

Managing Additional Systems

Installing the Chef tools, transferring the apache cookbook, and applying the run list is not terribly expensive.

- Chef provides a one-line curl install.
- You could use **git** to clone the repository from a common **git** repository.
- Applying the run list.

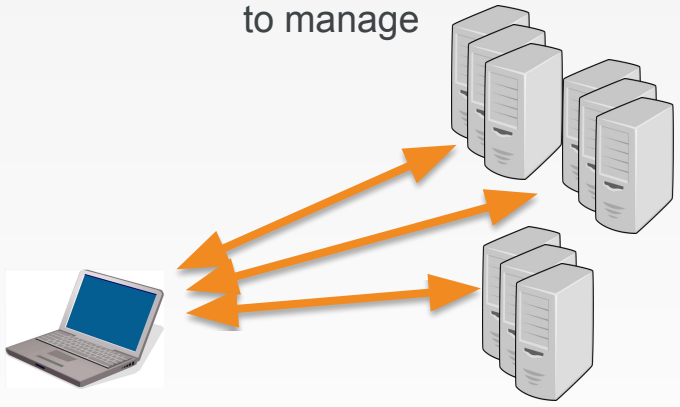
Managing Additional Systems

Now

Future

More complex

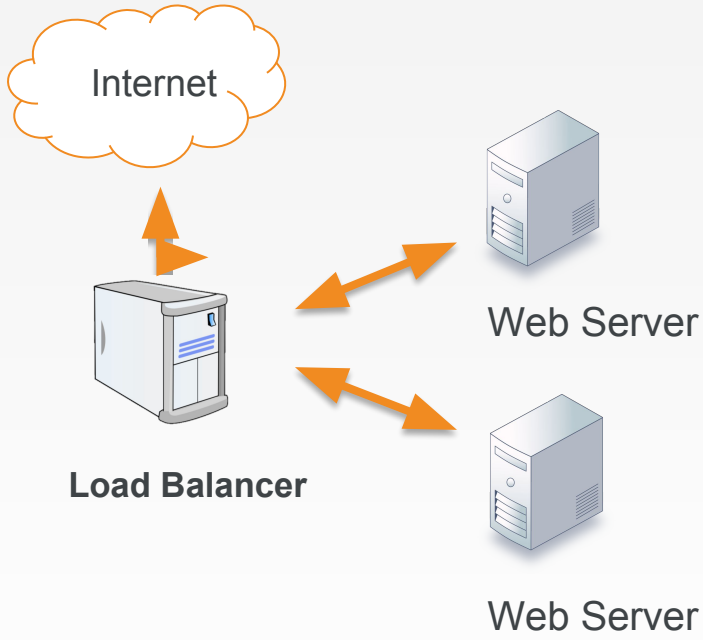
to manage



CHEF

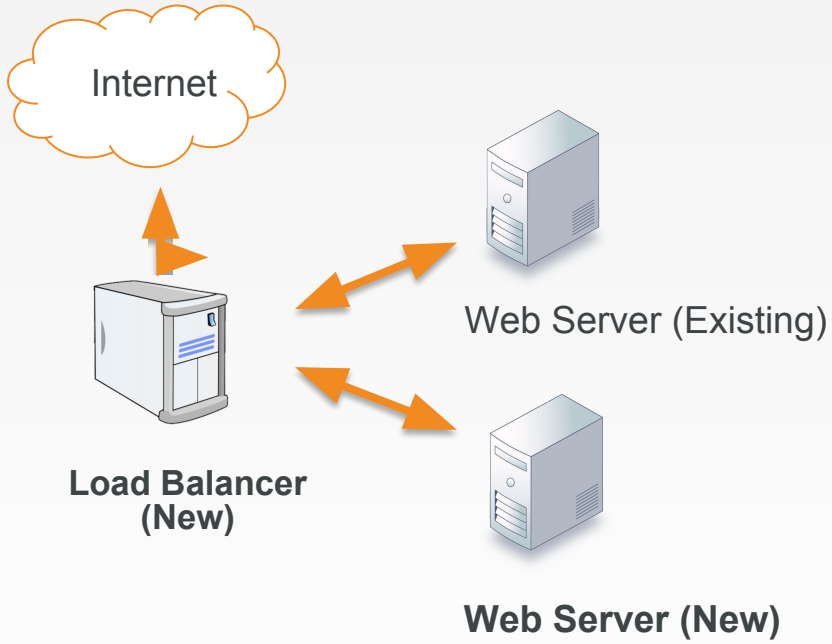
Managing User Traffic

A load balancer can forward incoming user web requests to other nodes.



Managing User Traffic

Today you will set up a new load balancer that will direct web requests to similarly-configured nodes.



CHEF

Steps to Set up Load Balancer and Web Servers

Web Server

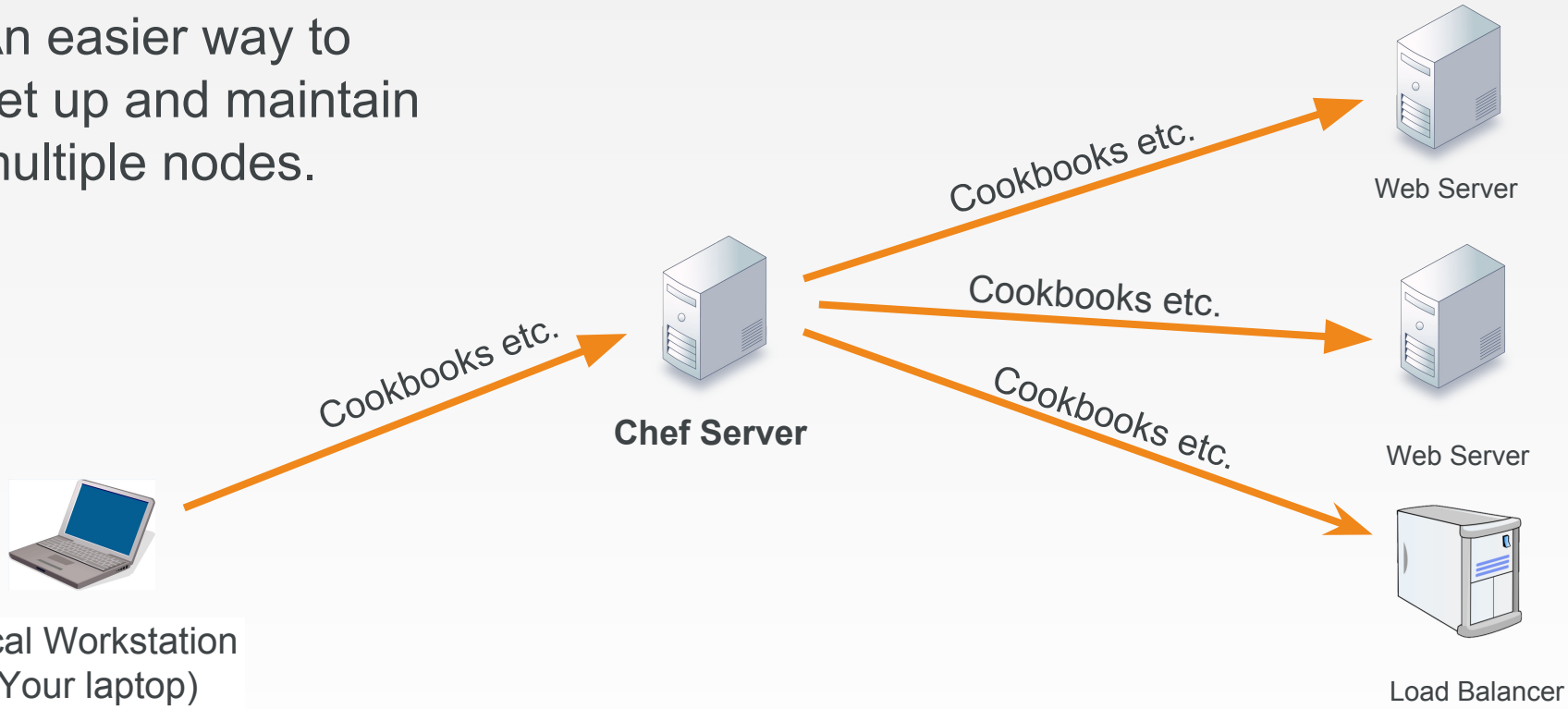
1. Provision the instance
2. Install Chef
3. Copy the Web Server cookbook
4. Apply the cookbook

Load Balancer

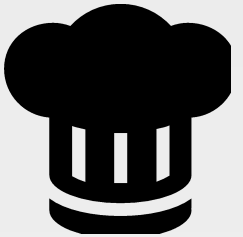
1. Create the haproxy (load balancer) cookbook
2. Provision the instance
3. Install Chef
4. Copy the haproxy cookbook
5. Apply the cookbook

The Chef Server

An easier way to set up and maintain multiple nodes.



CHEF



Hosted Chef

More easily manage multiple nodes

Objective:

- ❑ Create a Hosted Chef Account
- ❑ Upload your cookbooks to the Hosted Chef Server
- ❑ Add your old workstation as a managed node

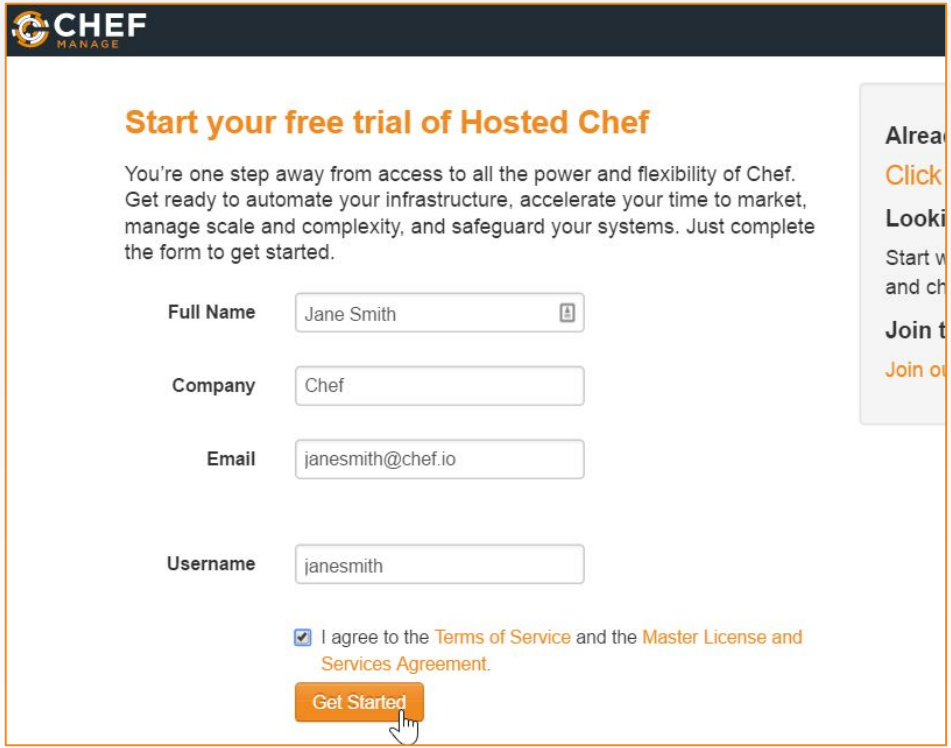
CHEF

Signing Up for a Hosted Chef Account

Steps

1. Navigate to https://manage.chef.io/signup

2. Fill out the form as indicated in this image using your name and a valid email address and then click **Get Started**.

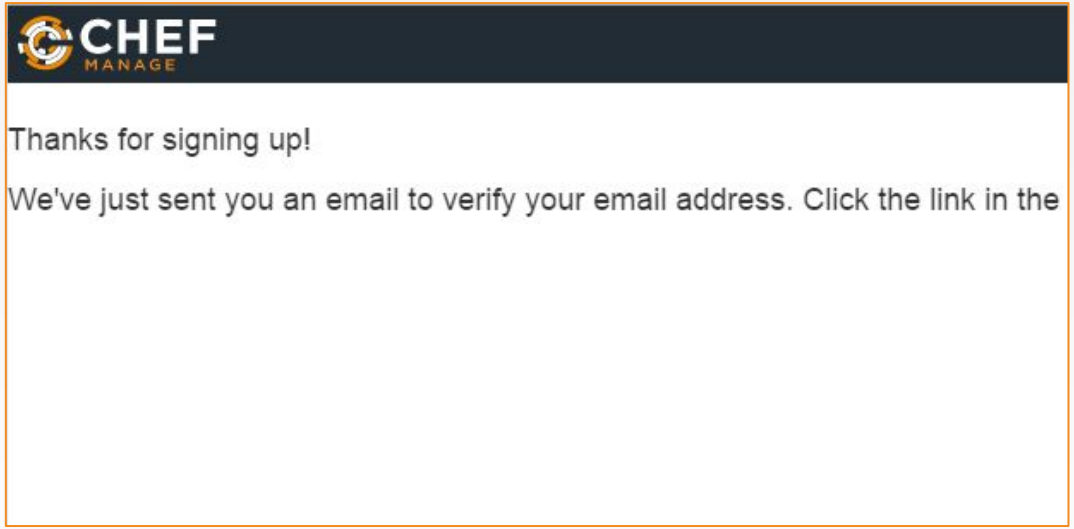


CHEF

Signing Up for a Hosted Chef Account

Steps

3. When prompted, open the email just sent to you and click the link in the email to finish the creation of your account.

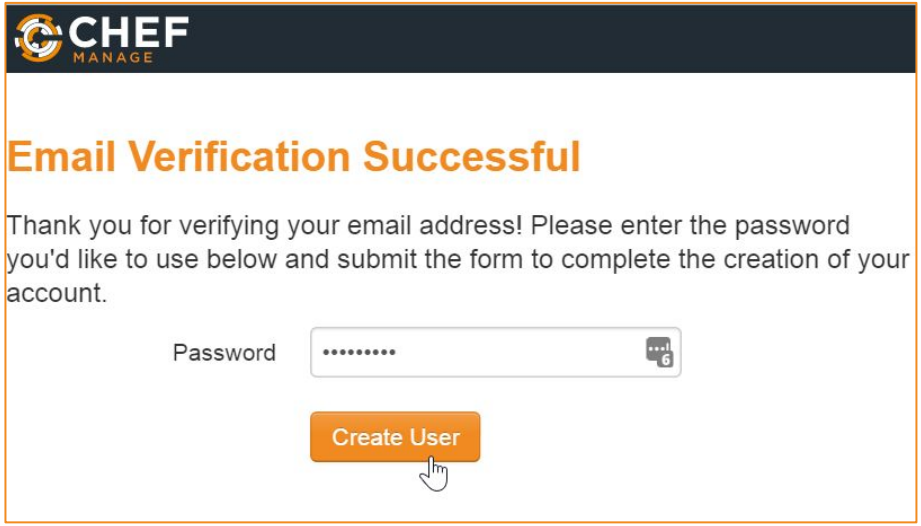


Signing Up for a Hosted Chef Account

Steps

4. Enter a password when prompted and then click **Create User**.

   You should write down your password in case you forget it.



Signing Up for a Hosted Chef Account

Steps

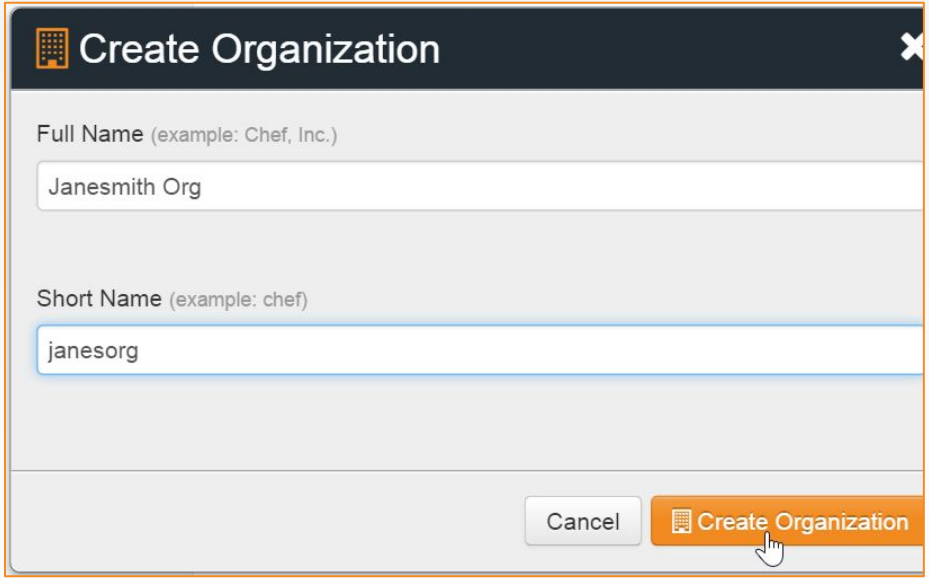
5. From the resulting page, click the **Create New Organization** button.

CHEF

Signing Up for a Hosted Chef Account

Steps

6. Fill out the resulting Create Organization form and then click **Create Organization**.



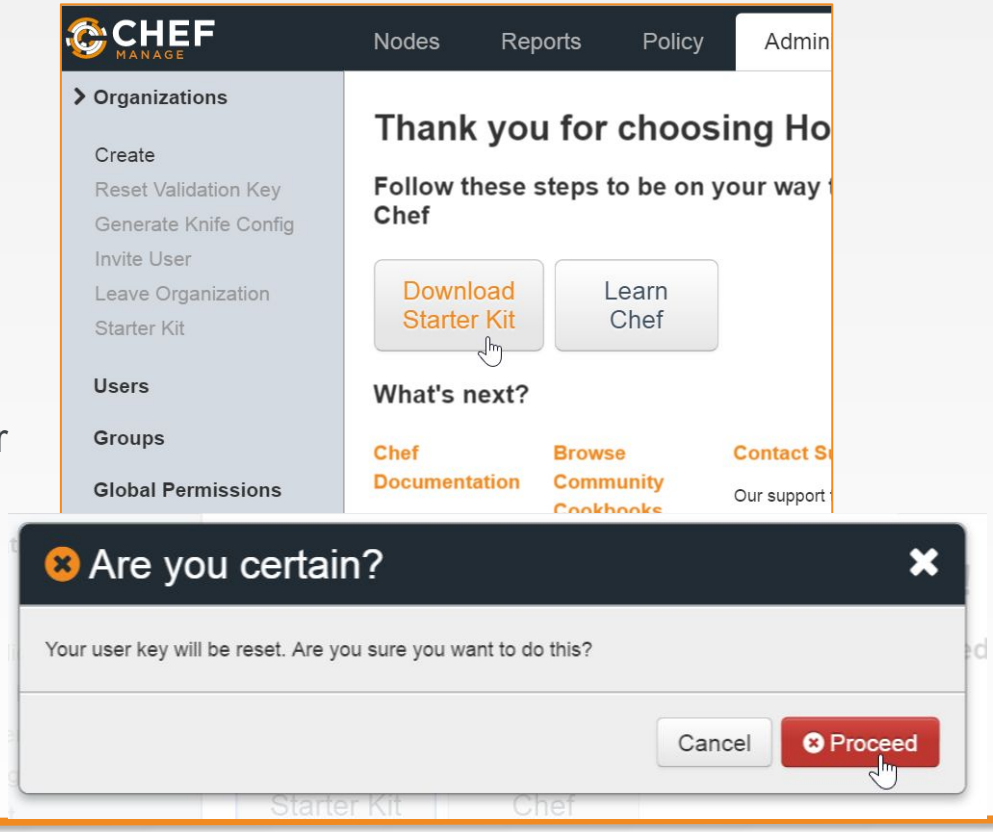
CHEF

Signing Up for a Hosted Chef Account

Steps

7. From the resulting page, click **Download Starter Kit** and then click **Proceed** when prompted.

A chef-starter zip file should download to your laptop.



CHEF

Signing Up for a Hosted Chef Account

Steps

8. Open the downloaded zip file and copy chef-repo folder that's contained in the zip file.
9. Paste the chef-repo folder to a location on your laptop, such as your home directory.

Note: Ensure that the path to the chef-repo does not have a space in it. Examples:

Mac: /home/*username*/chef-repo

Windows: C:\Users*username*\chef-repo

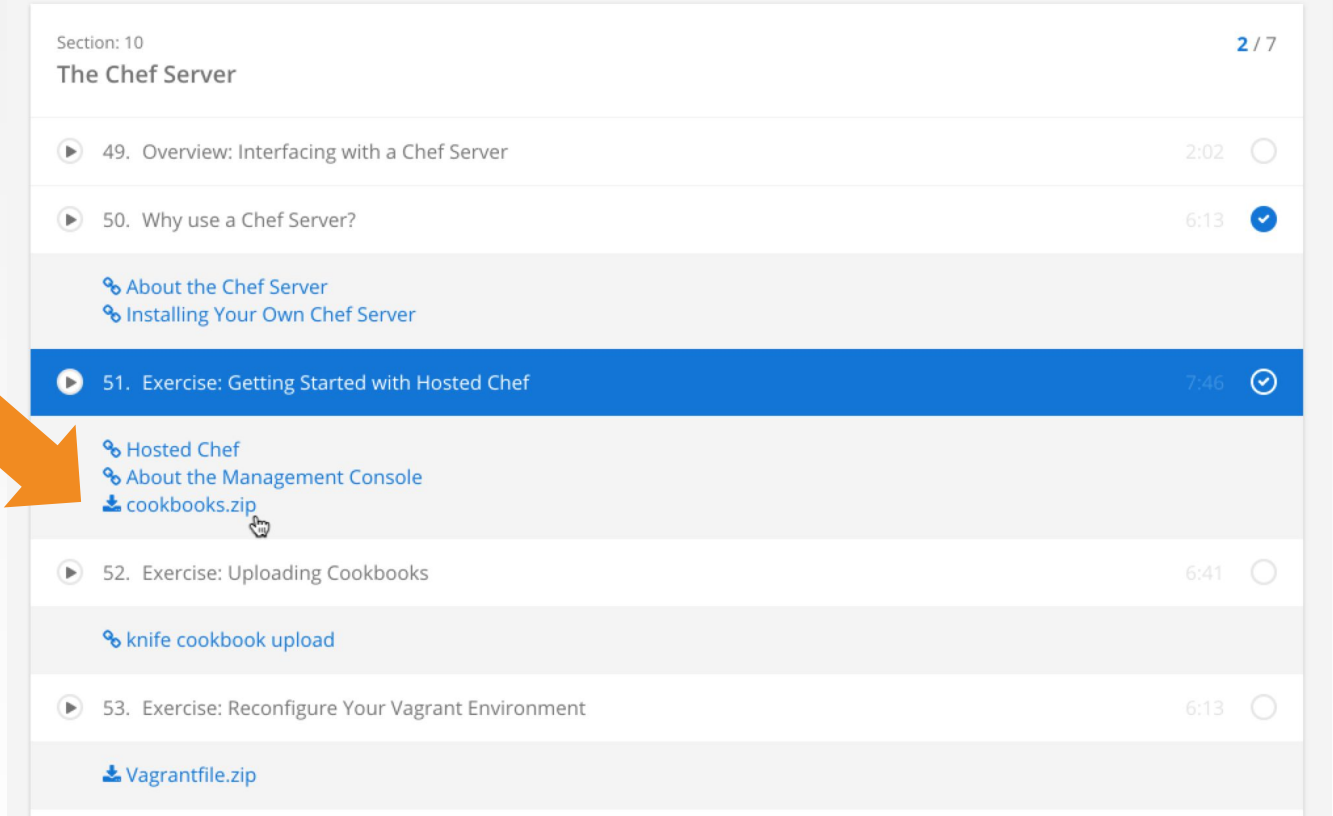
Download Your Cookbooks

Attached to this video you will find a zipped copy of the cookbooks we will use going forward.

Please download these cookbooks and place them in your chef-repo directory.

Download Your Cookbooks

Download cookbooks to your local machine

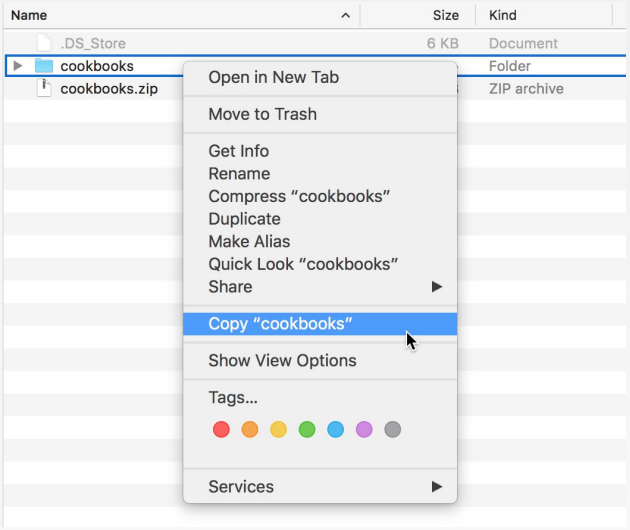


CHEF

Paste the cookbooks Folder

Steps

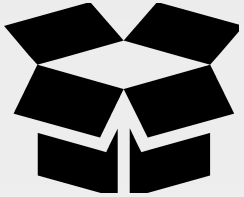
- copy the **cookbooks** folder that's contained in the zip file.
- Replace the **cookbooks** folder that's in your chef-repo folder with the copied cookbooks folder.



chef-repo

CHEF

CONCEPT



knife

knife is a command-line tool that provides an interface between a local chef-repo and the Chef Server.

GL: Navigate to the chef-repo

```
$ cd ~/chef-repo
```

GL: knife --help

```
$ knife --help
```

```
Available subcommands: (for details, knife SUB-COMMAND --help)

** BOOTSTRAP COMMANDS **
knife bootstrap FQDN (options)
knife bootstrap windows ssh FQDN (options)
knife bootstrap windows winrm FQDN (options)

** CLIENT COMMANDS **
knife client bulk delete REGEX (options)
knife client create CLIENT (options)
knife client delete CLIENT (options)
knife client edit CLIENT (options)
```

GL: knife client --help

```
$ knife client --help
```

```
Available client subcommands: (for details, knife SUB-COMMAND --help)

** CLIENT COMMANDS **
knife client bulk delete REGEX (options)
knife client create CLIENT (options)
knife client delete CLIENT (options)
knife client edit CLIENT (options)
knife client list (options)
knife client reregister CLIENT (options)
knife client show CLIENT (options)
```

GL: knife client list

```
$ knife client list
```

```
ORGNAME-validator
```

GL: knife cookbook --help

```
$ knife cookbook --help
```

```
** COOKBOOK COMMANDS **
knife cookbook bulk delete REGEX (options)
knife cookbook create COOKBOOK (options)
knife cookbook delete COOKBOOK VERSION (options)
knife cookbook download COOKBOOK [VERSION] (options)
knife cookbook list (options)
knife cookbook metadata COOKBOOK (options)
knife cookbook metadata from FILE (options)
knife cookbook show COOKBOOK [VERSION] [PART] [FILENAME] (options)
knife cookbook test [COOKBOOKS...] (options)
knife cookbook upload [COOKBOOKS...] (options)
```

GL: knife cookbook list

```
$ knife cookbook list
```

GL: knife cookbook list

```
$ knife cookbook upload workstation
```

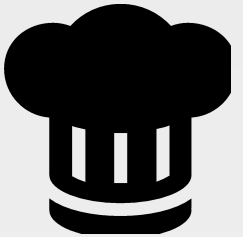
GL: knife cookbook list

```
$ knife cookbook upload apache
```

GL: knife cookbook list

```
$ knife cookbook list
```

```
apache        0.2.1
workstation   0.2.1
```



Hosted Chef

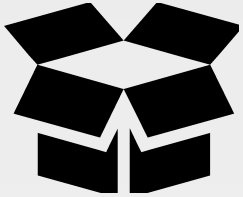
Adding nodes to your Chef Server

Objective:

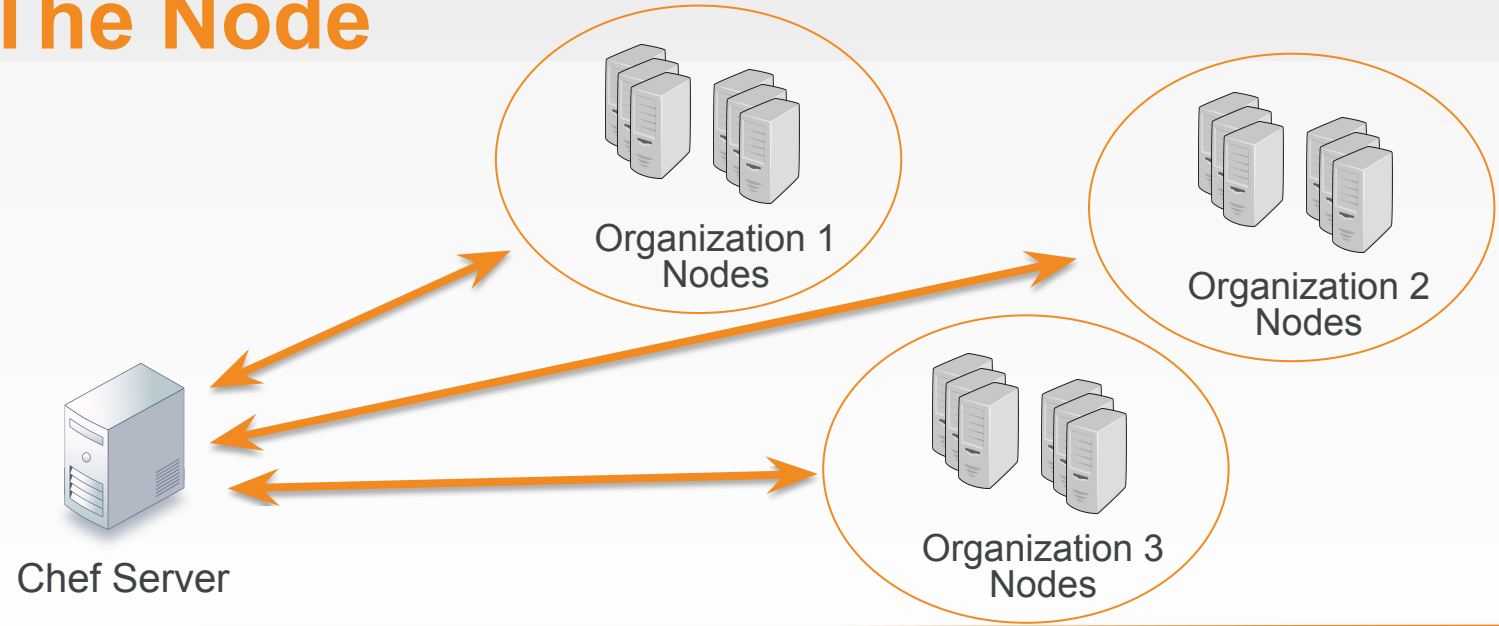
- ✓ Create a Hosted Chef Account
- ✓ Upload your cookbooks to the Hosted Chef Server
- ✓ Bootstrap a node and update its runlist

CHEF

CONCEPT



The Node



CHEF

Change to the chef-repo

```
$ cd ~/chef-repo
```

Run 'knife node –help'

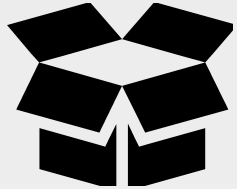
```
$ knife node --help
```

```
** NODE COMMANDS **
knife node bulk delete REGEX (options)
knife node create NODE (options)
knife node delete NODE (options)
knife node edit NODE (options)
knife node environment set NODE ENVIRONMENT
knife node from file FILE (options)
knife node list (options)
knife node run_list add [NODE] [ENTRY[,ENTRY]] (options)
knife node run_list remove [NODE] [ENTRY[,ENTRY]] (options)
knife node run_list set NODE ENTRIES (options)
knife node show NODE (options)
```


CHEF

Run 'knife node list'

```
$ knife node list
```



Bootstrapping a Node

The node may not have Chef installed. It may also not have details of where the Chef Server is located or the credentials to securely talk to that Server. To add those credentials we can **bootstrap** that node to install all those components.

https://learn.chef.io/skills/beyond-essentials-1

Run 'knife bootstrap –help'

```
$ knife bootstrap --help
```

```
knife bootstrap FQDN (options)
        --bootstrap-curl-options OPTIONS
                                     Add options to curl when install chef-client
        --bootstrap-install-command COMMANDS
                                     Custom command to install chef-client
        --bootstrap-no-proxy [NO_PROXY_URL|NO_PROXY_IP]
                                     Do not proxy locations for the node being
bootstrapped; this option is used interna
lly by Opscode
        --bootstrap-proxy PROXY_URL  The proxy server for the node being bootstrapped
    -t TEMPLATE,                     Bootstrap Chef using a built-in or custom
template. Set to the full path of an erb
template or use one of the built-in templates.
```


CHEF

Bootstrap Your Node - options

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node_name
```

Fully Qualified Domain Name

user name

password

sudo flag

node name

```
Creating new client for node1
Creating new node for node1
C                                  -24.compute-1.amazonaws.com
e                                  am          tar          ef
ec2-54-175-46-24.compute-1.am      tar          ent          3.0
ec2-54-175-46-24.compute-1.amazonaws.com resolving cookbooks for run list: []
ec2-54-175-46-24.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-175-46-24.compute-1.amazonaws.com [2016-09-16T16:51:21+00:00] WARN: Node node1 has an empty run list.
ec2-54-175-46-24.compute-1.amazonaws.com Converging 0 resources
ec2-54-175-46-24.compute-1.amazonaws.com
ec2-54-175-46-24.compute-1.amazonaws.com Running handlers:
```


CHEF

Verify the port and identity file for web1

```
$ vagrant ssh-config web1
```

```
Host web1
  HostName 127.0.0.1
  User vagrant
  Port 2200
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/USER/chef-repo/.vagrant/machines/web1/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```


CHEF

Bootstrap Your Node

```
$ knife bootstrap localhost --ssh-port WEB1_PORT --ssh-user vagrant --sudo
--identity-file PATH_TO_KEY -N web1
```

```
Creating new client for web1
Creating new node for web1
Connecting to localhost
localhost -----> Installing Chef Omnibus (-v 12)
localhost downloading https://omnitruck-direct.chef.io/chef/install.sh
localhost   to file /tmp/install.sh.12058/install.sh
localhost trying wget...
localhost el 7 x86_64
localhost Getting information for chef stable 12 for el...
localhost downloading
https://omnitruck-direct.chef.io/stable/chef/metadata?v=12&p=el&pv=7&m=x86_64
localhost   to file /tmp/install.sh.12063/metadata.txt
localhost trying wget...
```


CHEF

Run 'knife node list' Again

```
$ knife node list
```

```
web1
```

View More Information About Your Node

```
$ knife node show web1
```

```
Node Name:   web1
Environment: _default
FQDN:        web1
IP:          10.0.2.15
Run List:
Roles:
Recipes:
Platform:    centos 7.2.1511
Tags:
```

Notice that the IPAddress is not what we defined in the Vagrantfile. It's the internal IP instead.

CHEF

Add a Recipe to web1's Run List

```
$ knife node run_list add web1 "recipe[workstation],recipe[apache]"
```

```
Web1:
  run_list: recipe[workstation]
  run_list: recipe[apache]
```

View More Information About Your Node

```
$ knife node show web1
```

```
Node Name:   web1
Environment: _default
FQDN:        web1
IP:          10.0.2.15
Run List:    recipe[workstation], recipe[apache]
Roles:
Recipes:
Platform:    centos 7.2.1511
Tags:
```

Your Run List for web1 should contain the workstation and apache cookbooks

CHEF

Login to web1

```
$ vagrant ssh web1
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web1 ~]$
```

Run chef-client to converge web1

```
[vagrant@web1 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["workstation", "apache"]
Synchronizing Cookbooks:
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

Verify the state of your web application

```
[vagrant@web1 ~]$ curl localhost
```

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: 192.168.10.43</h2>
    <h2>hostname: web1</h2>
  </body>
</html>
```

Return to your Workstation

```
[vagrant@web1 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

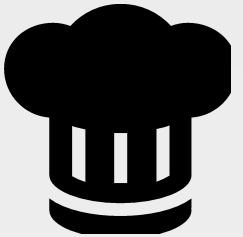
View More Information About Your Node

```
$ knife node show web1
```

```
Node Name:   web1
Environment: _default
FQDN:        web1
IP:          192.168.10.43
Run List:    recipe[workstation], recipe[apache]
Roles:
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::vagrant, workstation::setup, apache::server
Platform:    centos 7.2.1511
Tags:
```

The IPAddress should now match what we defined in the Vagrantfile.

CHEF



Hosted Chef

More easily manage multiple nodes

Objective:

- ✓ Create a Hosted Chef Account
- ✓ Upload your cookbooks to the Hosted Chef Server
- ✓ Add web1 as a managed node

DISCUSSION

Discussion

What is the benefit of storing cookbooks in a central repository?

What is the primary tool for communicating with the Chef Server?

How did you add a node to your organization?

Community Cookbooks

Find, Explore and View Chef Cookbooks

CHEF

Objectives

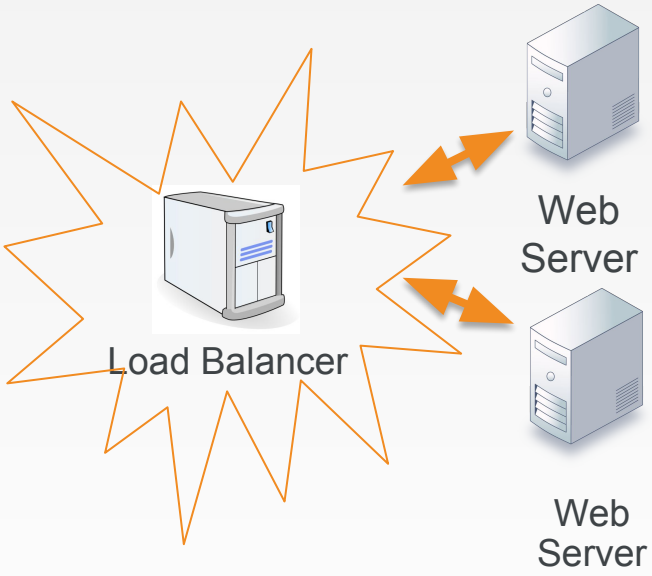
After completing this module, you should be able to

- Find cookbooks on the Chef Super Market
- Create a wrapper cookbook
- Replace the existing default values
- Upload a cookbook to Chef Server
- Bootstrap a new node that runs the cookbook

Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Receives requests and relays them to other systems.

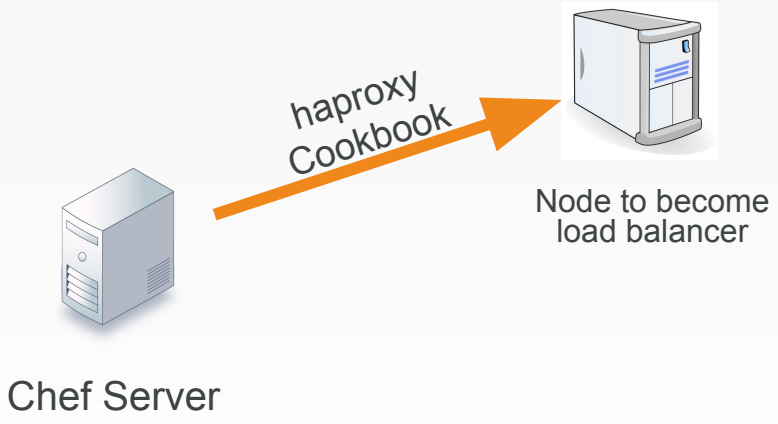


Load Balancer

Work that needs to be accomplished to setup a load balancer within our infrastructure:

Write a haproxy (load balancer) cookbook.

We will need to establish a new node within our organization to which we apply that cookbook.



Community Cookbooks

Someone already wrote that cookbook?

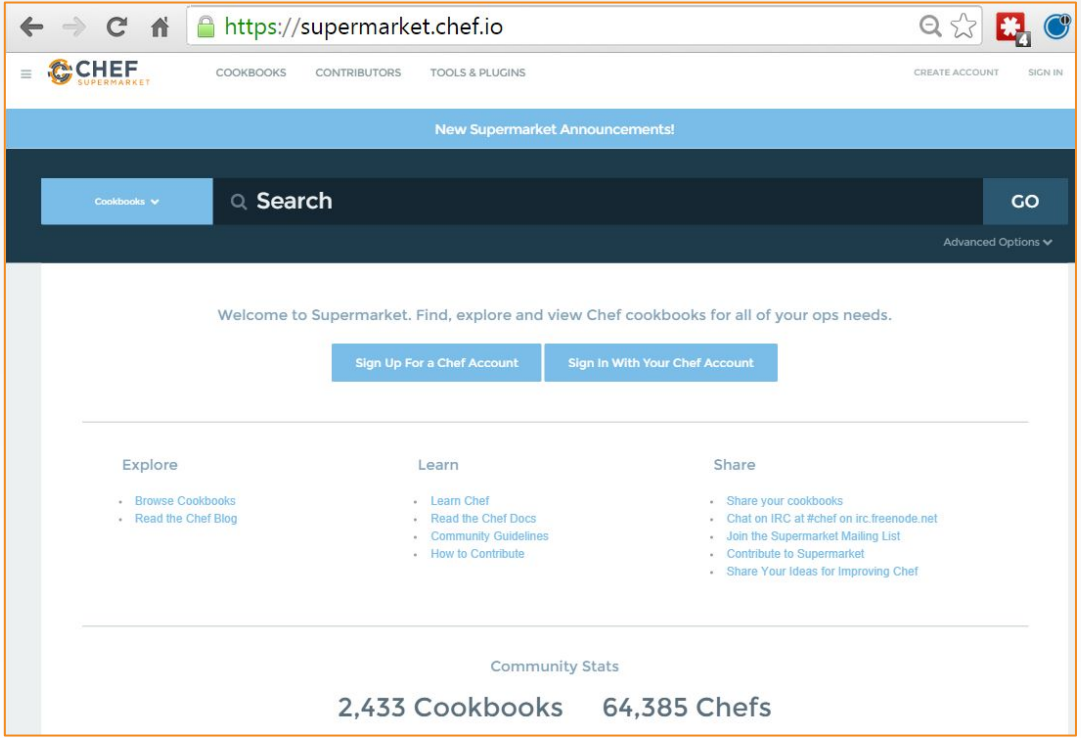
Available through the community site called Supermarket/

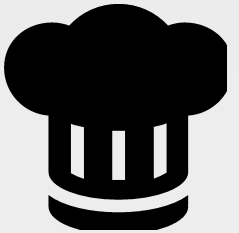
https://supermarket.chef.io

CHEF

Community Cookbooks

- Community cookbooks are managed by individuals.
- Chef does not verify or approve cookbooks in the Supermarket.
- Cookbooks may not work for various reasons.
- Still, there are real benefits to community cookbooks.





Load Balancer

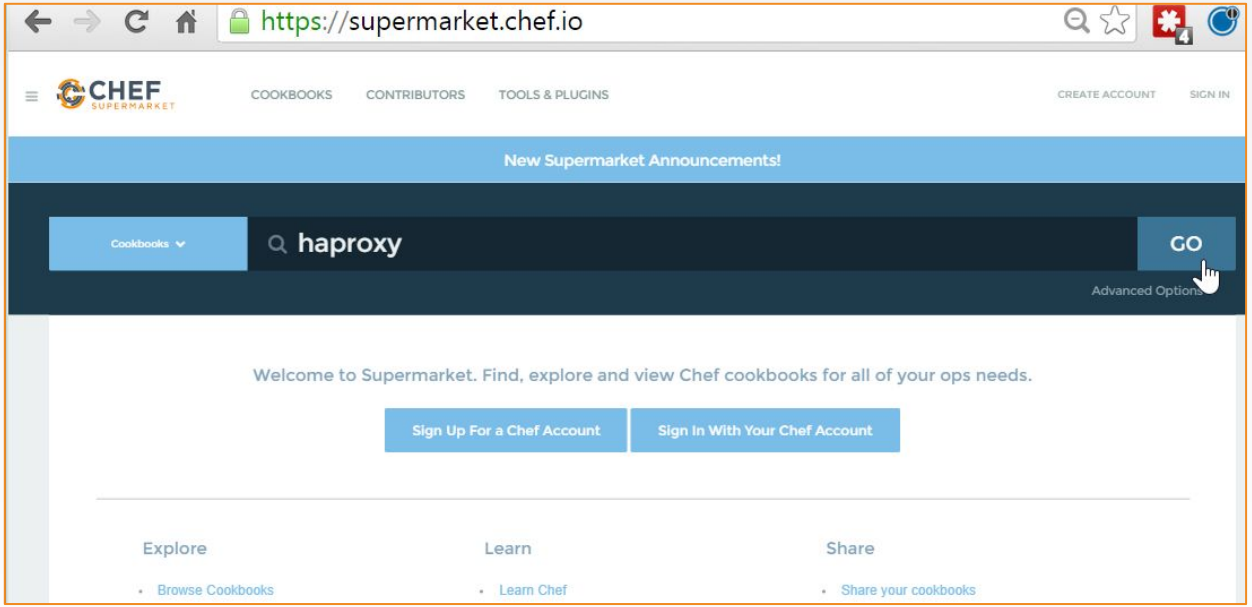
Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ❑ Find or Create a Cookbook to Manage a load balancer
- ❑ Configure the load balancer to send traffic to the new node
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the haproxy (load balancer) cookbook

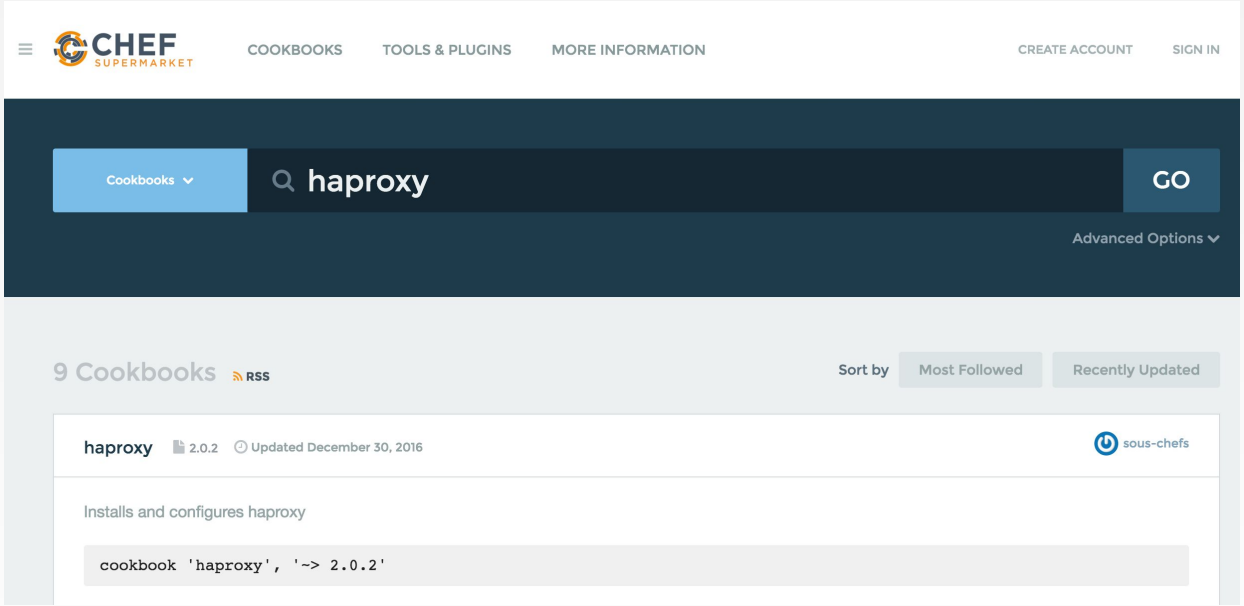
Searching in the Supermarket

1. From the https://supermarket.chef.io page, type **haproxy** in the search field and then click the **GO** button.



Searching in the Supermarket

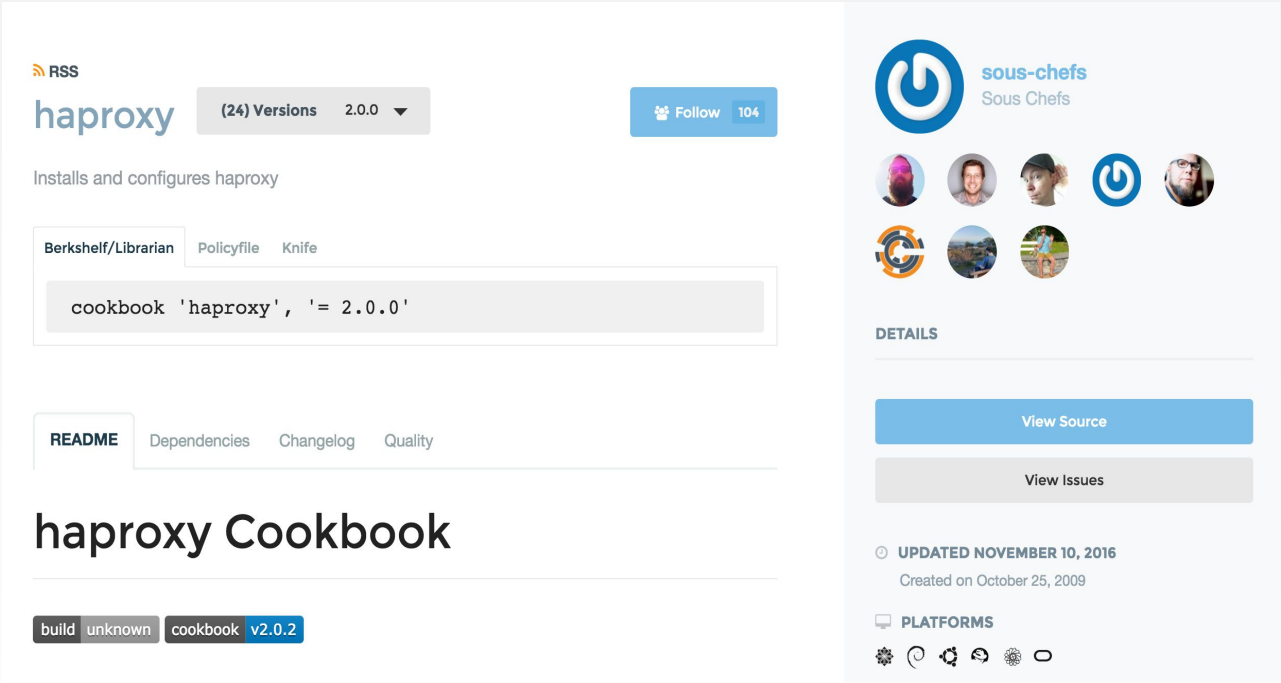
2. Click the resulting **haproxy** link.



Supermarket Cookbooks

On the right-hand side we can see the individuals that maintain the cookbook...

On the left, we are presented with the various ways we can install the cookbook...

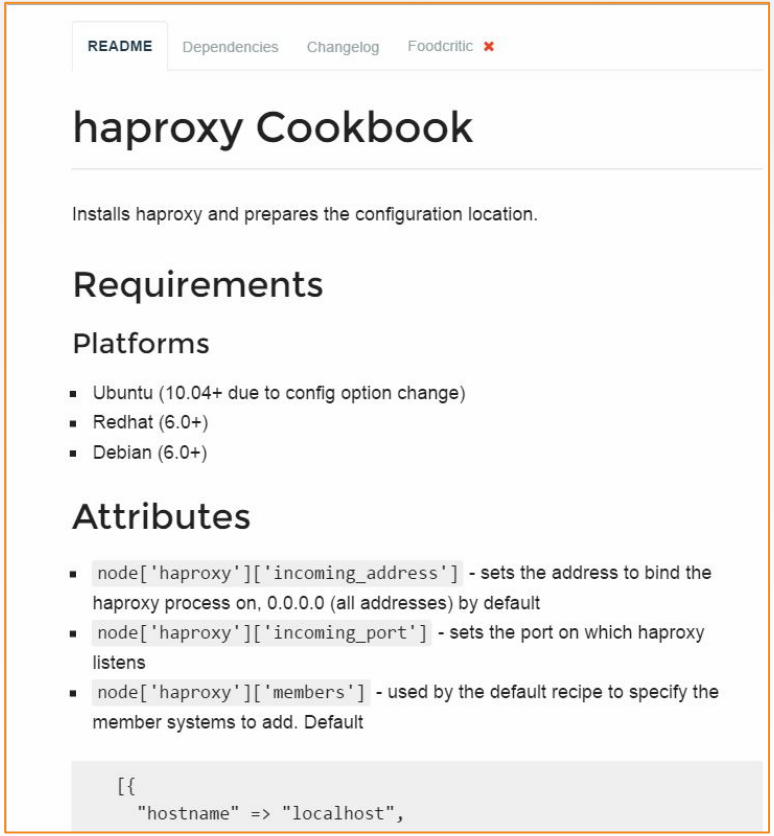


CHEF

Supermarket Cookbooks

The area to focus most of your attention from the beginning is the README.

Reading and understanding the README at a glance is difficult. It is a skill that comes with time.


README Dependencies Changelog Foodcritic ✖

haproxy Cookbook

Installs haproxy and prepares the configuration location.

Requirements

Platforms

- Ubuntu (10.04+ due to config option change)
- Redhat (6.0+)
- Debian (6.0+)

Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

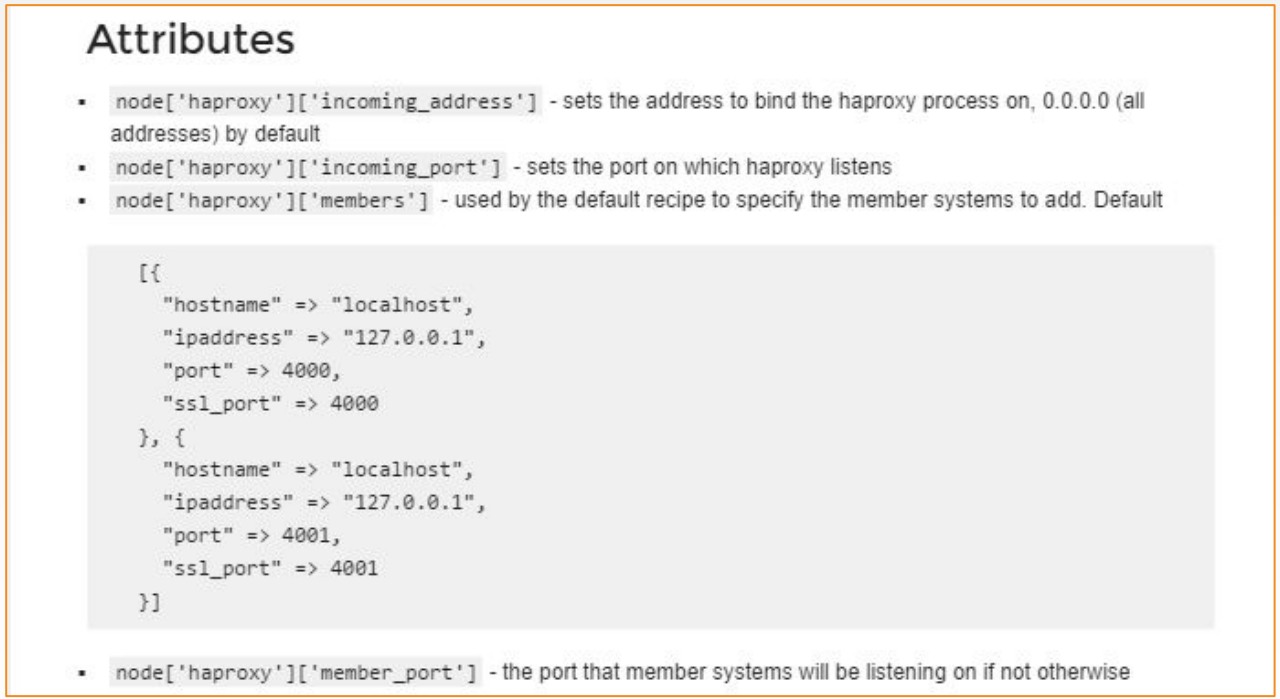
```
[{
  "hostname" => "localhost",
```


CHEF

Supermarket Cookbooks

These node attributes are different than the automatic ones defined by Ohai.

Attributes defined in a cookbook are not considered automatic.



Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4000,
  "ssl_port" => 4000
}, {
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4001,
  "ssl_port" => 4001
}]
```

- `node['haproxy']['member_port']` - the port that member systems will be listening on if not otherwise

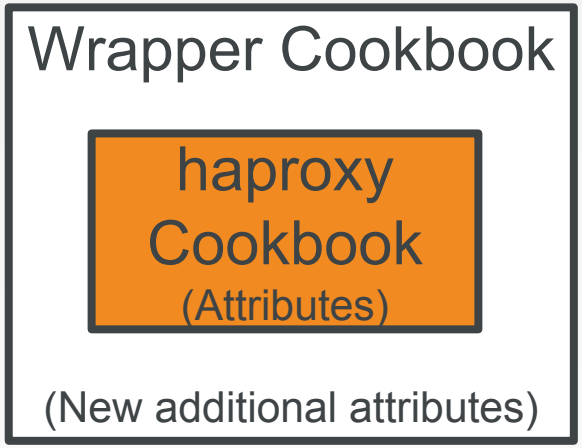
https://docs.chef.io/attributes.html

CHEF

Supermarket Cookbooks

A wrapper cookbook is a new cookbook that encapsulates the functionality of the original cookbook.

It defines new default values for the recipes.



https://docs.chef.io/supermarket.html#wrapper-cookbooks

https://www.chef.io/blog/2013/12/03/doing-wrapper-cookbooks-right/

cd and Generate the Cookbook

```
$ cd ~/chef-repo
$ chef generate cookbook cookbooks/myhaproxy
```

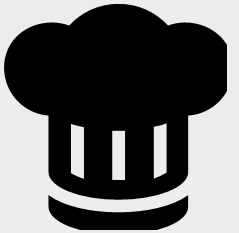
```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy] action create
    - create new directory C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy
  * template[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb] action
create_if_missing
    - create new file C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb
    - update content in file
C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb from none to 899276
    (diff output suppressed by config)
  * template[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/README.md] action
create_if_missing
```


CHEF

Create a Dependency in the Cookbook

~/chef-repo/cookbooks/myhaproxy/metadata.rb

```
name              'myhaproxy'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures myhaproxy'
long_description  'Installs/Configures myhaproxy'
version           '0.1.0'

depends 'haproxy', '= 2.0.0'
```



Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

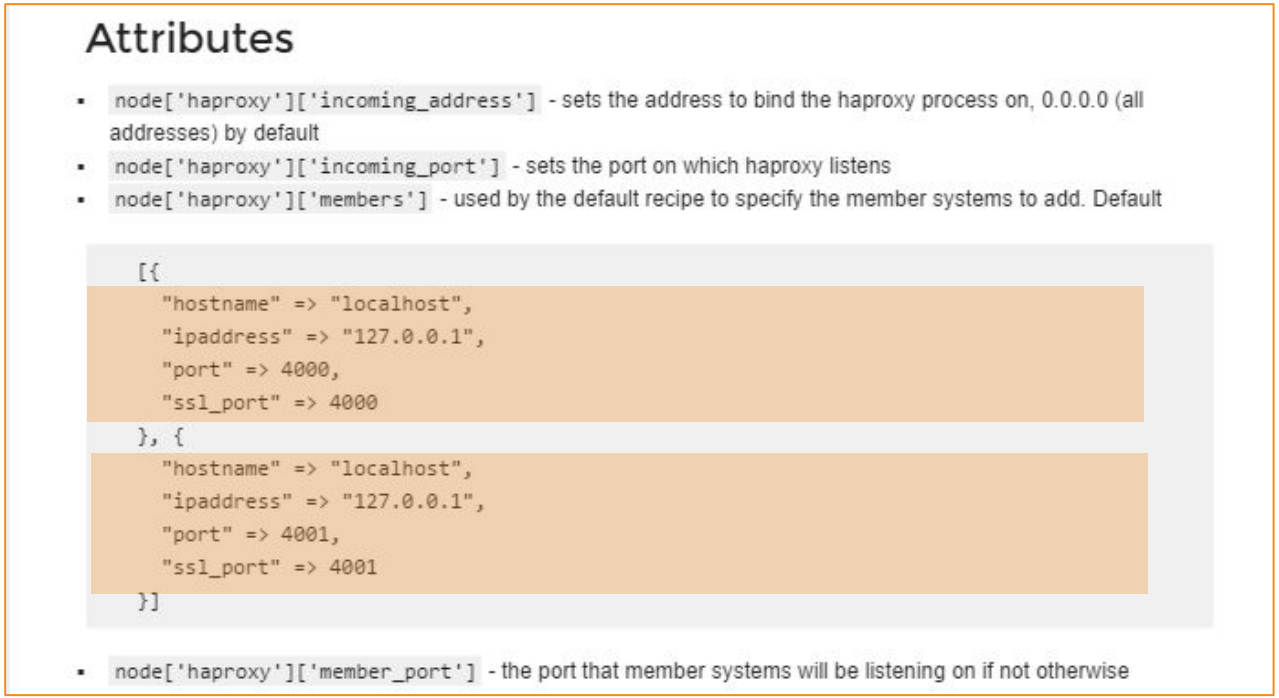
Objective:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- ✓ Upload cookbook to Chef Server
- ✓ Bootstrap a new node that runs the haproxy cookbook

Supermarket Cookbooks

Currently, the haproxy cookbook assumes that there are two different services running on the localhost at port 4000 and port 4001.

In a moment, you'll need to change that.



Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4000,
  "ssl_port" => 4000
}, {
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4001,
  "ssl_port" => 4001
}]
```

- `node['haproxy']['member_port']` - the port that member systems will be listening on if not otherwise

https://docs.chef.io/supermarket.html#wrapper-cookbooks

CHEF

Capture Node's Public Host Name and IP

```
$ knife node show --help
```

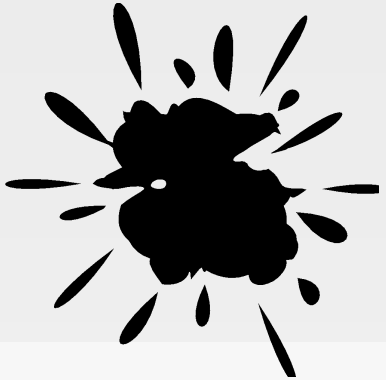
```
knife node show NODE (options)
    -a ATTR1 [--attribute ATTR2] ,   Show one or more attributes
        --attribute
    -s, --server-url URL             Chef Server URL
        --chef-zero-host HOST        Host to start chef-zero on
        --chef-zero-port PORT        Port (or port range) to start chef-zero on.  Port
ranges
    -k, --key KEY                    API Client Key
        --[no-]color                 Use colored output, defaults to false on Windows,
true
    -c, --config CONFIG              The configuration file to use
        --defaults                   Accept default values for all questions
    -d, --disable-editing            Do not open EDITOR, just accept the data as is
    -e, --editor EDITOR              Set the editor to use for interactive commands
```

Capture Node's Public Host Name and IP

```
$ knife node show web1 -a ipaddress
```

```
web1:
  ipaddress: 192.168.10.43
```

NOTE:
Cloud Instances



If using a cloud provider, such as EC2, Azure or Google Compute, you'll need to discover the public ipaddress for your instance. You usually can't simple ask for the node['ipaddress']

Instead, use the cloud.public_ipv4 and cloud.public_hostname attributes

NOTE: cloud web1's Public Host Name and IP

```
$ knife node show web1 -a cloud
```

```
node1:
  cloud:
    local_hostname:  ip-172-31-8-68.ec2.internal
    local_ipv4:      172.31.8.68
    private_ips:     172.31.8.68
    provider:        ec2
    public_hostname: ec2-54-175-46-24.compute-1.amazonaws.com
    public_ips:      54.175.46.24
    public_ipv4:     54.175.46.24
```


CHEF

Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2016 The Authors, All Rights
Reserved.



include_recipe 'haproxy::manual'
```

Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
#

node['haproxy']['members'] = [
{
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4000,
    'ssl_port' => 4000
  }, {
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4001,
    'ssl_port' => 4001
}]

include_recipe 'haproxy::manual'
```

Erase 1 of the members

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node['haproxy']['members'] = [
  {
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4000,
    'ssl_port' => 4000
  },
  {
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',
    'ipaddress' => '52.8.71.11',
    'port' => 80,
    'ssl_port' => 80
  }]

include_recipe 'haproxy::manual'
```

Create a node attribute node['haproxy']['members']

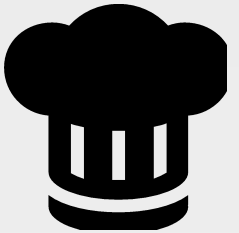
~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4000,
    'ssl_port' => 4000
  }]

include_recipe 'haproxy::manual'
```

GL: Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{
    'hostname' => 'WEB1_PUBLIC_HOSTNAME',
    'ipaddress' => 'WEB1_PUBLIC_IPADDRESS',
    'port' => 80,
    'ssl_port' => 80
  }]

include_recipe 'haproxy::manual'
```



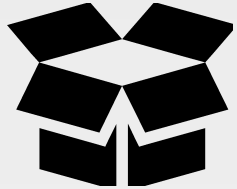
Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
  - Upload cookbook to Chef Server
  - Bootstrap a new node that runs the haproxy cookbook

CONCEPT



Berkshelf

Berkshelf is a cookbook management tool that allows us to upload your cookbooks and all of its dependencies to the Chef Server.

http://berkshelf.com

CHEF

Change to the cookbooks/myhaproxy Directory

```
$ cd cookbooks/myhaproxy
```

Run berks --help

```
$ berks --help
```

```
Commands:
  berks apply ENVIRONMENT     # Apply version locks from Berksfile.lock to a Chef
environment
  berks contingent COOKBOOK   # List all cookbooks that depend on the given cookbook in
your
  berks cookbook NAME [PATH]  # Create a skeleton for a new cookbook
  berks help [COMMAND]        # Describe available commands or one specific command
  berks info [COOKBOOK]       # Display name, author, copyright, and dependency
information
  berks init [PATH]           # Initialize Berkshelf in the given directory
  berks install               # Install the cookbooks specified in the Berksfile
  berks list                  # List cookbooks and their dependencies specified by your
  berks outdated [COOKBOOKS]  # List dependencies that have new versions available that
  berks package [PATH]        # Vendor and archive the dependencies of a Berksfile
```

Run berks install

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myhaproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using build-essential (7.0.2)
Using cpu (1.0.0)
Using compat_resource (12.16.2)
Installing haproxy (2.0.0)
Using myhaproxy (1.0.1) from source at .
Using ohai (4.2.3)
Using seven_zip (2.0.2)
Using mingw (1.2.4)
Using windows (2.1.1)
```

See the Berksfile.lock

```
$ ls -al (or ls -Force if using Powershell)
```

```
drwxr-xr-x 7 chef chef 4096 Aug 27 18:44 .
drwxr-xr-x 4 chef chef 4096 Aug 27 16:17 ..
drwxr-xr-x 8 chef chef 4096 Aug 27 16:07 .git
-rw-r--r-- 1 chef chef  126 Aug 27 15:46 .gitignore
drwxr-xr-x 3 chef chef 4096 Aug 27 18:45 .kitchen
-rw-r--r-- 1 chef chef  183 Aug 27 18:44 .kitchen.yml
-rw-r--r-- 1 chef chef   47 Aug 27 15:46 Berksfile
-rw------- 1 chef chef   77 Aug 27 18:45 Berksfile.lock
-rw-r--r-- 1 chef chef   54 Aug 27 15:46 README.md
-rw-r--r-- 1 chef chef  974 Aug 27 15:46 chefignore
-rw-r--r-- 1 chef chef  198 Aug 27 15:46 metadata.rb
drwxr-xr-x 2 chef chef 4096 Aug 27 16:34 recipes
```


CHEF

See the Contents of the Berksfile.lock

```
$ cat Berksfile.lock
```

```
DEPENDENCIES
  myhaproxy
    path: .
    metadata: true

GRAPH
  build-essential (7.0.2)
    compat_resource (>= 12.14)
    mingw (>= 1.1)
    seven_zip (>= 0.0.0)
  ....
```

Upload the Cookbook to the Chef Server

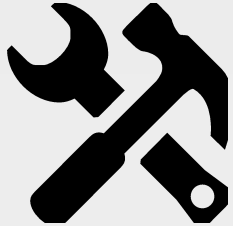
```
$ berks upload
```

```
Uploaded myhaproxy (0.1.0) to:
'https://api.opscode.com:443/organizations/ORG_NAME'
```

Display Cookbooks within Your Org

```
$ knife cookbook list
```

```
apache                  0.2.1
build-essential         7.0.2
compat_resource         12.16.2
cpu                     1.0.0
haproxy                 2.0.0
mingw                   1.2.4
myhaproxy               0.1.0
ohai                    4.2.3
seven_zip               2.0.2
windows                 2.1.1
workstation             0.2.1
```



Lab: Bootstrap a Load Balancer

- ❑ Bootstrap a new node
- ❑ Update the run list of the new node to include the wrapper proxy server cookbook
- ❑ SSH to that system and run chef-client
- ❑ Verify that traffic to the load balancer is relayed to the web server.

Lab: Bootstrap a New Node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N NODE_NAME
```

```
Creating new client for node2
Creating new node for node2
Connecting to ec2-54-210-192-12.compute-1.amazonaws.com
ec2-54-210-192-12.compute-1.amazonaws.com Starting first Chef Client run...
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version
12.3.0
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
[]
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-192-12.compute-1.amazonaws.com [2016-09-16T17:13:10+00:00] WARN:
Node node2 has an empty run list.
ec2-54-210-192-12.compute-1.amazonaws.com Converging 0 resources
```


CHEF

Verify the port and identity file for web1

```
$ vagrant ssh-config load-balancer
```

```
Host load-balancer
  HostName 127.0.0.1
  User vagrant
  Port 2202
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/USER/chef-repo/.vagrant/machines/load-balancer/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```


CHEF

Bootstrap Your Node

```
$ knife bootstrap localhost --ssh-port LOAD_BAL_PORT --ssh-user vagrant
--sudo --identity-file PATH_TO_KEY -N load-balancer -r "recipe[myhaproxy]"
```

```
Creating new client for load-balancer
Creating new node for load-balancer
Connecting to localhost
localhost -----> Installing Chef Omnibus (-v 12)
localhost downloading https://omnitruck-direct.chef.io/chef/install.sh
localhost   to file /tmp/install.sh.12058/install.sh
localhost trying wget...
localhost el 7 x86_64
localhost Getting information for chef stable 12 for el...
localhost downloading
https://omnitruck-direct.chef.io/stable/chef/metadata?v=12&p=el&pv=7&m=x86_64
localhost   to file /tmp/install.sh.12063/metadata.txt
localhost trying wget...
```


CHEF

Run 'knife node list' Again

```
$ knife node list
```

```
web1
load-balancer
```

View More Information About Your Node

```
$ knife node show load-balancer
```

```
Node Name:   load-balancer
Environment: _default
FQDN:        load-balancer
IP:          10.0.2.16
Run List:    recipe[myhaproxy]
Roles:
Recipes:     myhaproxy, myhaproxy::default, haproxy::manual,
haproxy::install_package
Platform:    centos 7.2.1511
Tags:
```


CHEF

Login to Load Balancer

```
$ vagrant ssh load-balancer
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@load-balancer ~]$
```

Verify the Load Balancer

```
[vagrant@load-balancer ~]$ curl localhost
```

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: 192.168.10.43</h2>
    <h2>hostname: web1</h2>
  </body>
</html>
```

Return to your Workstation

```
[vagrant@load-balancer ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Verify the Load Balancer

Managing Multiple Nodes

Create another web server and add it as a proxy member

CHEF

Objectives

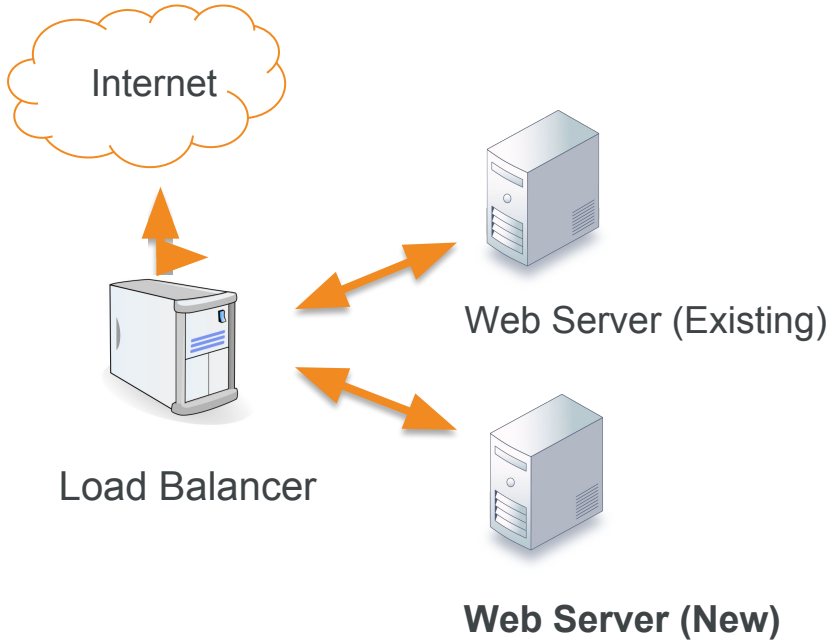
After completing this module, you should be able to

- Bootstrap, update the run_list, and run chef-client on a node
- Append values to an attribute within a recipe
- Version a cookbook and upload it to the Chef Server

Managing User Traffic

You already configured the load balancer and one web server node.

In this module you'll add another node to the load balancer's list of web server's it is serving.



Lab: Another Web Node

- ❑ Bootstrap a new webserver node
- ❑ Update the run list of the new node to include the web server cookbook
- ❑ Run chef-client on that system
- ❑ Verify that the node's web server is functional

Lab: Bootstrap the New Node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N web2 -r "recipe[workstation],recipe[apache]"
```

```
Connecting to ec2-54-210-86-164.compute-1.amazonaws.com
ec2-54-210-86-164.compute-1.amazonaws.com Starting first Chef Client run...
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list: []
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com [2016-09-16T17:36:14+00:00] WARN: Node node3
has an empty run list.
ec2-54-210-86-164.compute-1.amazonaws.com Converging 0 resources
ec2-54-210-86-164.compute-1.amazonaws.com
ec2-54-210-86-164.compute-1.amazonaws.com Running handlers:
ec2-54-210-86-164.compute-1.amazonaws.com Running handlers complete
ec2-54-210-86-164.compute-1.amazonaws.com Chef Client finished, 0/0 resources updated
in
```


CHEF

Verify the port and identity file for web1

```
$ vagrant ssh-config web2
```

```
Host web2
  HostName 127.0.0.1
  User vagrant
  Port 2201
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/USER/chef-repo/.vagrant/machines/web2/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

Bootstrap Your Node

```
$ knife bootstrap localhost --ssh-port WEB2_PORT --ssh-user vagrant --sudo
--identity-file PATH_TO_KEY -N web2 --run-list "recipe[workstation],recipe[apache]"
```

```
Creating new client for web2
Creating new node for web2
Connecting to localhost
localhost -----> Installing Chef Omnibus (-v 12)
localhost downloading https://omnitruck-direct.chef.io/chef/install.sh
localhost   to file /tmp/install.sh.12058/install.sh
localhost trying wget...
localhost el 7 x86_64
localhost Getting information for chef stable 12 for el...
localhost downloading
https://omnitruck-direct.chef.io/stable/chef/metadata?v=12&p=el&pv=7&m=x86_64
localhost   to file /tmp/install.sh.12063/metadata.txt
localhost trying wget...
```


CHEF

Run 'knife node list' Again

```
$ knife node list
```

```
load-balancer
web1
web2
```

Lab: Verify the New Node

```
$ knife node show web2
```

```
Node Name:   web2
Environment: _default
FQDN:        web2
IP:          192.168.10.44
Run List:    recipe[workstation], recipe[apache]
Roles:
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Login to web2

```
$ vagrant ssh web2
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web2 ~]$
```

Verify the state of your web application

```
[vagrant@web2 ~]$ curl localhost
```

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: 192.168.10.44</h2>
    <h2>hostname: web1</h2>
  </body>
</html>
```

Return to your Workstation

```
[vagrant@web1 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Lab: Another Web Node

- ✓ Bootstrap a new node
- ✓ Update the run list of the new node to include the web server cookbook
- ✓ Run chef-client on that system
- ✓ Verify that the node's web server is functional

Lab: Update the Load Balancer

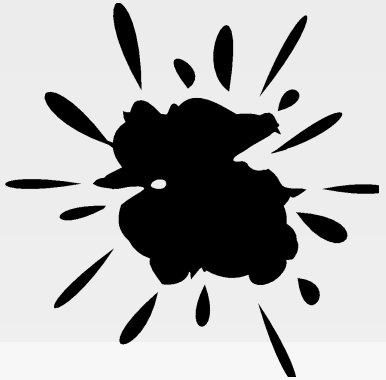
- ❑ Update the wrapped proxy server cookbook to include the new web node as a member.
- ❑ Upload that cookbook to the Chef Server
- ❑ Run chef-client on that system
- ❑ Verify that the load balancer delivers traffic to both web server nodes.

Verify web1's hostname and IP

~/chef-repo/Vagrantfile

```
...
Vagrant.configure(2) do |config|

 config.vm.define 'web1' do |n|
    n.vm.box = 'bento/centos-7.2'
    n.vm.box_version = '2.2.9'
    n.vm.hostname = 'web1'
    n.vm.network :private_network, ip: '192.168.10.43'
    n.vm.provision :shell, inline: NODE_SCRIPT.dup
   set_hostname(n)
  end

 config.vm.define 'web2' do |n|
    n.vm.box = 'bento/centos-7.2'
    n.vm.box_version = '2.2.9'
    n.vm.hostname = 'web2'
    n.vm.network :private_network, ip: '192.168.10.44'
   ...
```

NOTE:
Cloud Instances



If using a cloud provider, such as EC2, Azure or Google Compute, you'll need to discover the public ipaddress for your instance. You usually can't simply ask for the node['ipaddress']

Instead, use the cloud.public_ipv4 and cloud.public_hostname attributes

NOTE: cloud web1's Public Host Name and IP

```
$ knife node show web1 -a cloud
```

```
node1:
  cloud:
    local_hostname:  ip-172-31-8-68.ec2.internal
    local_ipv4:      172.31.8.68
    private_ips:     172.31.8.68
    provider:        ec2
    public_hostname: ec2-54-175-46-24.compute-1.amazonaws.com
    public_ips:      54.175.46.24
    public_ipv4:     54.175.46.24
```


CHEF

NOTE: cloud web2's Public Host Name and IP

```
$ knife node show web2 -a cloud
```

```
node1:
  cloud:
    local_hostname:  ip-172-31-8-69.ec2.internal
    local_ipv4:      172.31.8.69
    private_ips:     172.31.8.69
    provider:        ec2
    public_hostname: ec2-54-175-46-25.compute-1.amazonaws.com
    public_ips:      54.175.46.25
    public_ipv4:     54.175.46.25
```


CHEF

Lab: Add the Other Web Server to LB

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{
    'hostname' => 'web1',
    'ipaddress' => '192.168.10.43',
    'port' => 80,
    'ssl_port' => 80
  },{
    'hostname' => 'web2',
    'ipaddress' => '192.168.10.44',
    'port' => 80,
    'ssl_port' => 80
  }
]

include_recipe 'haproxy::default'
```

Lab: Update the Version

~/chef-repo/cookbooks/myhaproxy/metadata.rb

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

depends 'haproxy', '= 2.0.0'
```

Lab: CD and Then Run berks install

```
$ cd ~/chef-repo/cookbooks/myhaproxy
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myhaproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using build-essential (2.2.3)
Using cpu (0.2.0)
Using haproxy (2.0.0)
Using myhaproxy (0.2.0) from source at .
```

Lab: Upload the Cookbook to Chef Server

```
$ berks upload
```

```
Skipping build-essential (2.2.3) (frozen)
Skipping cpu (0.2.0) (frozen)
Skipping haproxy (2.0.0) (frozen)
Uploaded myhaproxy (0.2.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
```


CHEF

Login to Load Balancer

```
$ vagrant ssh load-balancer
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@load-balancer ~]$
```

Converge the Load Balancer

```
[vagrant@load-balancer ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["myhaproxy"]
Synchronizing Cookbooks:
  - myhaproxy (0.1.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - windows (2.1.1)
  - ohai (4.2.3)
....
```

Verify the Load Balancing - curl 1

```
[vagrant@load-balancer ~]$ curl localhost
```

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: 192.168.10.43</h2>
    <h2>hostname: web1</h2>
  </body>
</html>
```

Verify the Load Balancing - curl 2

```
[vagrant@load-balancer ~]$ curl localhost
```

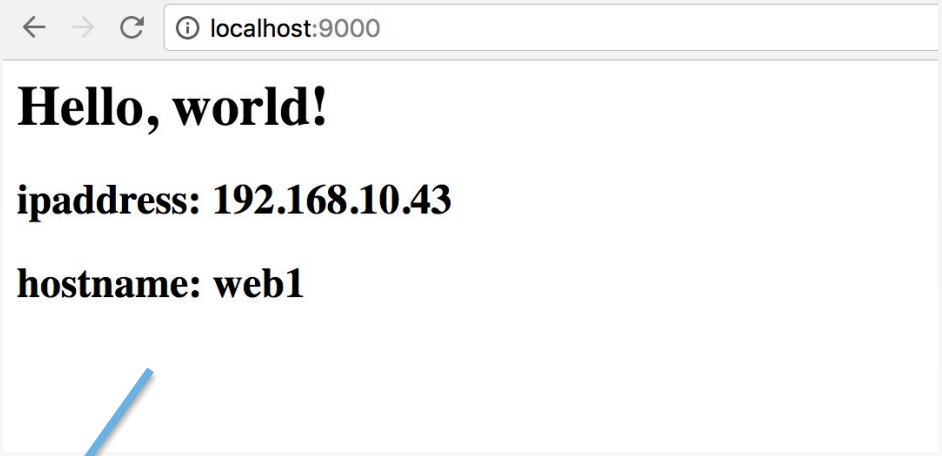
```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: 192.168.10.44</h2>
    <h2>hostname: web2</h2>
  </body>
</html>
```

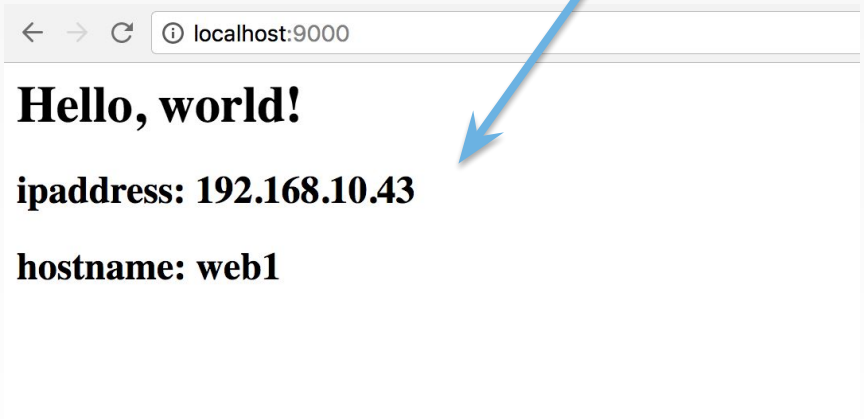
Return to your Workstation

```
[vagrant@load-balancer ~]$ exit
```

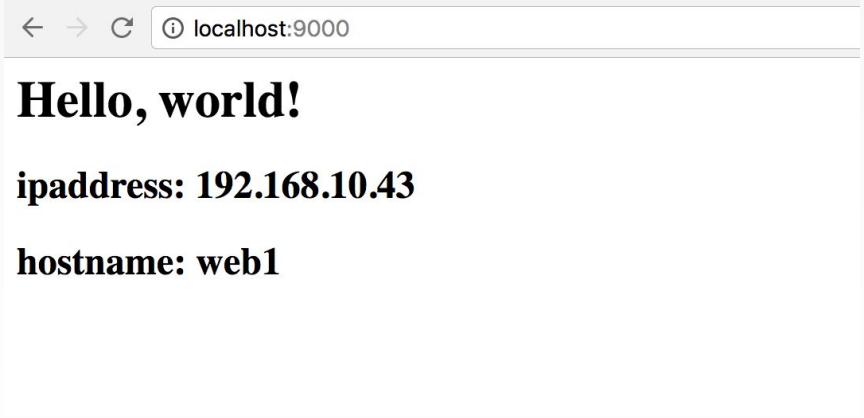
```
logout
Connection to 127.0.0.1 closed.
```

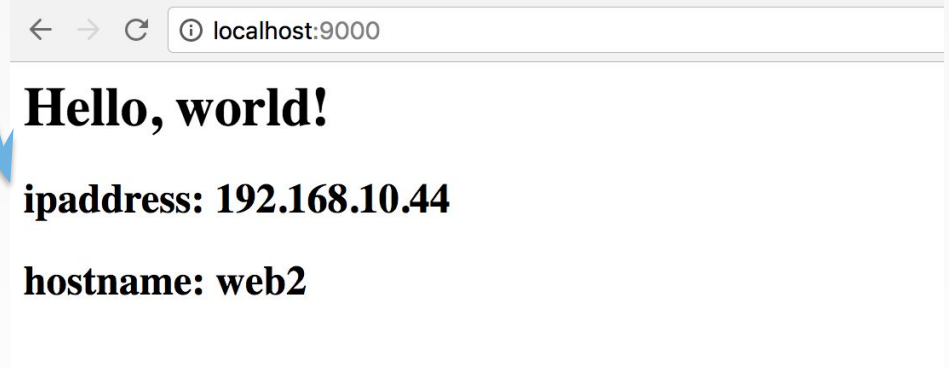
Lab: Test the Load Balancer





Lab: Test the Load Balancer





Lab: Update the Load Balancer

- ✓ Update the wrapped proxy server cookbook to include the new web node as a member.
- ✓ Upload that cookbook to the Chef Server
- ✓ Run chef-client on that system
- ✓ Verify that the load balancer delivers traffic to both web server nodes.

DISCUSSION

Discussion

What is the process to setup a third web node?

What is the process for removing a web node?

What is the most manual part of the process?

Roles

Giving Your Nodes a Role

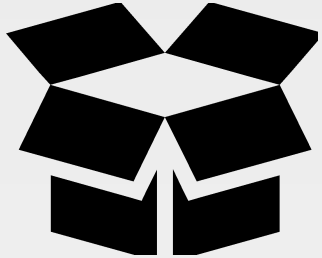
CHEF

Objectives

After completing this module, you should be able to

- Assign roles to nodes so you can better describe them and configure them in a similar manner.

CONCEPT

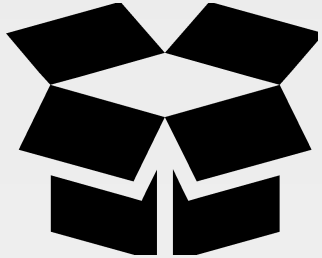


Roles

A role describes a run list of recipes that are executed on the node.

A role may also define new defaults or overrides for existing cookbook attribute values.

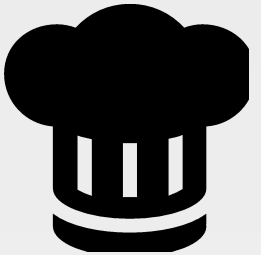
CONCEPT



Roles

When you assign a role to a node you do so in its run list.

This allows you to configure many nodes in a similar fashion.



Roles for Everyone

We will give our nodes a role to better describe them and so we can configure them in a similar manner.

Objective:

- ❑ Give our load balancer node a "load_balancer" Role
- ❑ Give our web nodes a "web" Role

What Can 'knife role' Do?

```
$ cd ~/chef-repo
$ knife role --help
```

```
** ROLE COMMANDS **
knife role bulk delete REGEX (options)
knife role create ROLE (options)
knife role delete ROLE (options)
knife role edit ROLE (options)
knife role env_run_list add [ROLE] [ENVIRONMENT] [ENTRY[,ENTRY]] (options)
knife role env_run_list clear [ROLE] [ENVIRONMENT]
knife role env_run_list remove [ROLE] [ENVIRONMENT] [ENTRIES]
knife role env_run_list replace [ROLE] [ENVIRONMENT] [OLD_ENTRY]
[NEW_ENTRY]
knife role env_run_list set [ROLE] [ENVIRONMENT] [ENTRIES]
knife role from file FILE [FILE...] (options)
```

Run 'knife role list'

```
$ knife role list
```

Create a Roles Directory

```
$ mkdir roles
```

Create the load_balancer.rb

~/chef-repo/roles/web.rb

```
name 'web'
description 'Web Server Role'
run_list 'recipe[workstation]','recipe[apache]'
```

Upload it to the Chef Server

```
$ knife role from file roles/web.rb
```

```
Updated Role web!
```

Validate Chef Server Received It

```
$ knife role list
```

```
web
```

View Details of the Role

```
$ knife role show web
```

```
chef_type:            role
default_attributes:
description:          Web Server Role
env_run_lists:
json_class:           Chef::Role
name:                 web
override_attributes:
run_list:
  recipe[workstation]
  recipe[apache]
```


CHEF

Run 'knife node --help'

```
$ knife node --help
```

```
** NODE COMMANDS **
knife node bulk delete REGEX (options)
knife node create NODE (options)
knife node delete NODE (options)
knife node edit NODE (options)
knife node environment set NODE ENVIRONMENT
knife node from file FILE (options)
knife node list (options)
knife node run_list add [NODE] [ENTRY[,ENTRY]] (options)
knife node run_list remove [NODE] [ENTRY[,ENTRY]] (options)
knife node run_list set NODE ENTRIES (options)
knife node show NODE (options)
```

Set the Web Role to web1

```
$ knife node run_list set web1 "role[web]"
```

```
web1:
  run_list: role[web]
```

Lab: Verify the New Node

```
$ knife node show web1
```

```
Node Name:   web1
Environment: _default
FQDN:        web1
IP:          192.168.10.43
Run List:    role[web]
Roles:
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Login to web1

```
$ vagrant ssh web1
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web1 ~]$
```

Run chef-client to converge web1

```
[vagrant@web1 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["workstation", "apache"]
Synchronizing Cookbooks:
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

Return to your Workstation

```
[vagrant@web1 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Lab: Verify the New Node

```
$ knife node show web1
```

```
Node Name:   web1
Environment: _default
FQDN:        web1
IP:          192.168.10.43
Run List:    role[web]
Roles:       web
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Set the Web Role to web2

```
$ knife node run_list set web2 "role[web]"
```

```
web2:
  run_list: role[web]
```

Lab: Verify the New Node

```
$ knife node show web2
```

```
Node Name:   web2
Environment: _default
FQDN:        web2
IP:          192.168.10.44
Run List:    role[web]
Roles:
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Login to web2

```
$ vagrant ssh web2
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web2 ~]$
```

Run chef-client to converge web2

```
[vagrant@web2 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["workstation", "apache"]
Synchronizing Cookbooks:
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

Return to your Workstation

```
[vagrant@web2 ~]$ exit
```

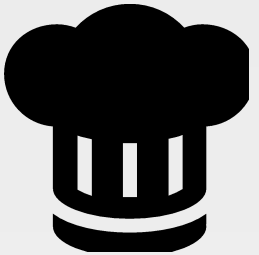
```
logout
Connection to 127.0.0.1 closed.
```

Lab: Verify the New Node

```
$ knife node show web2
```

```
Node Name:   web2
Environment: _default
FQDN:        web2
IP:          192.168.10.44
Run List:    role[web]
Roles:       web
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF



Roles for Everyone

We will give our nodes a role to better describe them and so we can configure them in a similar manner.

Objective:

- ✓ Give our web nodes a "web" Role
  Give our load balancer node a "load_balancer" Role

Lab: Define a Load Balancer Role

- ❑ Create a role named 'load-balancer' that has the run list 'recipe[myhaproxy]'
- ❑ Upload the role with 'knife role from file'
- ❑ Set the load-balancer node's run list to be "role[load-balancer]"
- ❑ Converge your load balancer node

Lab: Create the load-balaner.rb File

~/chef-repo/roles/load-balancer.rb

```
name 'load-balancer'
description 'Load Balancer Role'
run_list 'recipe[myhaproxy]'
```

Lab: Upload the web.rb File

```
$ knife role from file roles/load-balancer.rb
```

```
Updated Role load-balancer!
```

Lab: Verify the Role on the Chef Server

```
$ knife role list
```

```
load-balancer
web
```


CHEF

Lab: Verify Specific Information About the Role

```
$ knife role show load-balancer
```

```
chef_type:            role
default_attributes:
description:          Load Balancer Role
env_run_lists:
json_class:           Chef::Role
name:                 load-balancer
override_attributes:
run_list:             recipe[myhaproxy]
```

Lab: Set node1's Run List

```
$ knife node run_list set load-balancer "role[load-balancer]"
```

```
load-balancer:
  run_list: role[load-balancer]
```


CHEF

Verify the Run List

```
$ knife node show load-balancer
```

```
Node Name:   load-balancer
Environment: _default
FQDN:        load-balancer
IP:          10.0.2.15
Run List:    role[load_balancer]
Roles:
Recipes:     myhaproxy, myhaproxy::default, haproxy::default, haproxy::install_package
Platform:    centos 7.2
Tags:
```


CHEF

Login to Load Balancer

```
$ vagrant ssh load-balancer
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@load-balancer ~]$
```

Converge the Load Balancer

```
[vagrant@load-balancer ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["myhaproxy"]
Synchronizing Cookbooks:
  - myhaproxy (0.1.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - windows (2.1.1)
  - ohai (4.2.3)
....
```

Return to your Workstation

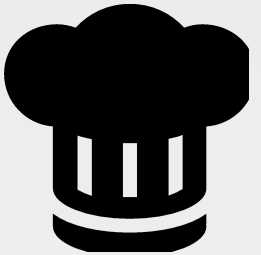
```
[vagrant@load-balancer ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Verify the Run List

```
$ knife node show load-balancer
```

```
Node Name:   load-balancer
Environment: _default
FQDN:        load-balancer
IP:          10.0.2.15
Run List:    role[load-balancer]
Roles:       load-balancer
Recipes:     myhaproxy, myhaproxy::default, haproxy::default, haproxy::install_package
Platform:    centos 7.2
Tags:
```



Roles for Everyone

We will give our nodes a role to better describe them and so we can configure them in a similar manner.

Objective:

- ✓ Give our load balancer node a "load_balancer" Role
- ✓ Give our web nodes a "web" Role

CHEF

Discussion

What are the benefits of using roles? What are the drawbacks?

Roles can contain roles. How many of these nested roles would make sense?