Details About the System

Finding and Displaying Information About Our System

Objectives

After completing this module, you should be able to

- Capture details about a system
- Use the node object within a recipe
- Use Ruby's string interpolation
- Update the version of a cookbook

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

GL: Discover the IP Address

```
$ hostname -I
```

```
172.31.8.68 172.17.42.1
```


CHEF

GL: Discover the Host Name

```
$ hostname
```

```
banana-stand
```


CHEF

GL: 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
```


CHEF

GL: 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
```

GL: Adding the CPU

~/cookbooks/workstation/recipes/setup.rb

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


  IPADDRESS: 104.236.192.102
  HOSTNAME : banana-stand
  MEMORY   : 502272 kB
  CPU      : 2399.998 MHz
'
  mode '0644'
  owner 'root'
  group 'root'
end
```

GL: Introducing a Change

By creating a change we have introduced risk.

Lets run our cookbook tests before we apply the updated recipe.

GL: Change into Our Cookbook

```
$ cd ~/cookbooks/workstation
```


CHEF

GL: Run Our Tests

```
$ kitchen test
```

```
-----> Starting Kitchen (v1.4.0)
-----> Setting up <default-ubuntu-1404>...
$$$$$$ Running legacy setup for 'Docker' Driver
-----> Installing Busser (busser)
Fetching: thor-0.19.0.gem (100%)
       Successfully installed thor-0.19.0
Fetching: busser-0.7.1.gem (100%)
       Successfully installed busser-0.7.1
       2 gems installed
-----> Setting up Busser
       Creating BUSSER_ROOT in /tmp/verifier
       Creating busser binstub
       Installing Busser plugins: busser-serverspec
```

GL: Return Home and Apply workstation Cookbook

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

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

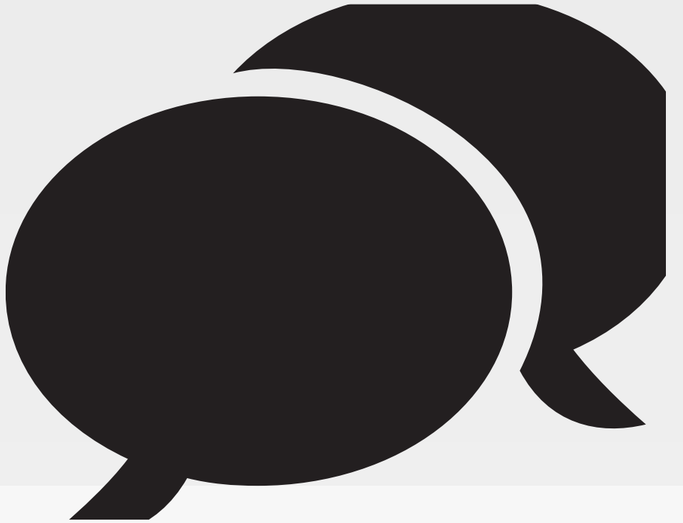
GL: Verify that the /etc/motd Has Been Updated

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

```
Property of ...

  IPADDRESS: 172.31.8.68
  HOSTNAME : ip-172-31-8-68
  MEMORY   : 605048 kB
  CPU      : 1795.672
```

DISCUSSION



Capturing System Data

What are the limitations of the way we captured this data?

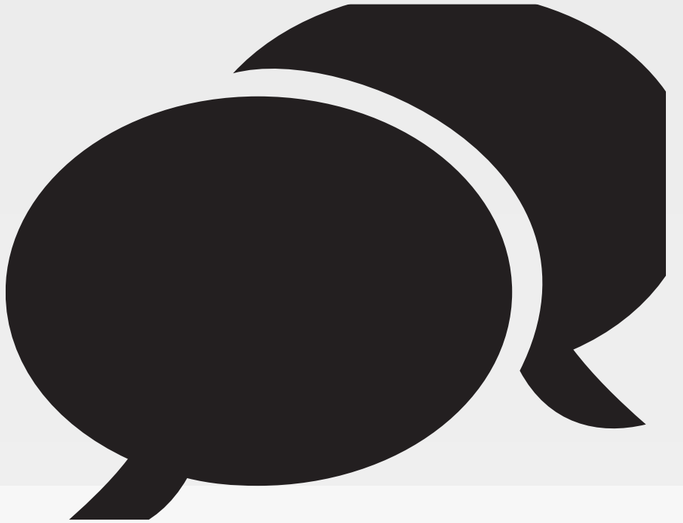
How accurate will our MOTD be when we deploy it on other systems?

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?

Ohai!

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

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

Ohai!

```
$ 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": {
```

All About The System

Ohai queries the operating system with a number of commands, similar to the ones demonstrated.

The data is presented in JSON (JavaScript Object Notation).

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

ohai + chef-client = <3

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

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

The Node Object

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

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

The Node

```
IPADDRESS: 104.236.192.102
```

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

The Node

```
HOSTNAME: banana-stand
```

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

The Node


```
MEMORY: 502272kB
```

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

The Node

```
CPU: 2399.998MHz
```

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

String Interpolation

```
I have 4 apples
```

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

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

String Interpolation

```
I have 4 apples
```

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

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

String Interpolation

```
I have 4 apples
```

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


CHEF

GL: Using the Node's Attributes

~/cookbooks/workstation/recipes/setup.rb

```
# ... PACKAGE RESOURCES ...
file '/etc/motd' do
  content "Property of ...

  IPADDRESS: #{node['ipaddress']}
  HOSTNAME : #{node['hostname']}
  MEMORY   : #{node['memory']['total']}
  CPU      : #{node['cpu']['0']['mhz']}
"
  mode '0644'
  owner 'root'
  group 'root'
end
```

GL: Verify the Changes

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

Lab: Test the Workstation's Default Recipe

```
$ cd cookbooks/workstation
$ kitchen test
```

```
-----> Starting Kitchen (v1.4.0)
-----> Cleaning up any prior instances of <default-centos-67>
-----> Destroying <default-centos-67>...
       Finished destroying <default-centos-67> (0m0.00s).
-----> Testing <default-centos-67>
-----> Creating <default-centos-67>...
       Sending build context to Docker daemon  2.56 kB
       Sending build context to Docker daemon
       Step 0 : FROM centos:centos6
        ---> 72703a0520b7
       Step 1 : RUN yum clean all
```

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...
```

GL: Verify the Changes

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

Changes Mean a New Version

Let's bump the version number and check in the code to source control.

Objective:

- ❑ Update the version of the "workstation" cookbook
- ❑ Commit the changes to the "workstation" cookbook to version control

Cookbook Versions

A cookbook version represents a set of functionality that is different from the cookbook on which it is based.

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

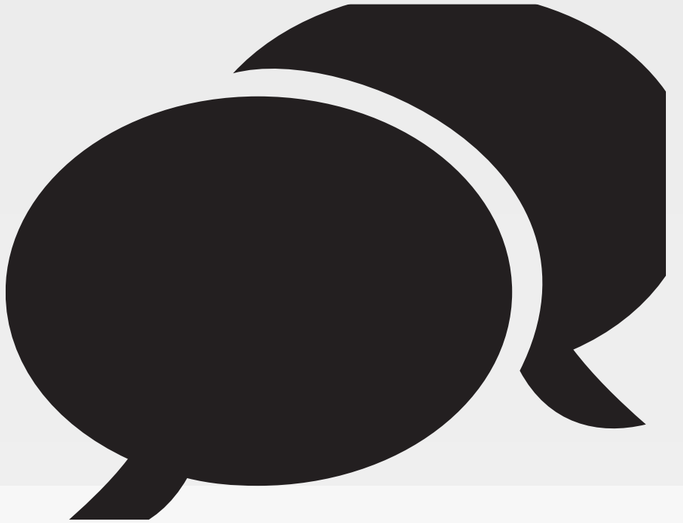
Semantic Versions

Given a version number **MAJOR**.**MINOR**.**PATCH**, increment the:

- **MAJOR** version when you make incompatible API changes
- **MINOR** version when you add functionality in a backwards-compatible manner
- **PATCH** version when you make backwards-compatible bug fixes

http://semver.org

DISCUSSION



Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

GL: Update the Cookbook Version

~/cookbooks/workstation/metadata.rb

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

GL: Commit Your Work

```
$ cd ~/cookbooks/workstation
$ 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

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: Test the Apache Cookbook's Default Recipe

```
$ cd cookbooks/apache
$ kitchen test
```

```
-----> Starting Kitchen (v1.4.0)
-----> Cleaning up any prior instances of <default-centos-67>
-----> Destroying <default-centos-67>...
       Finished destroying <default-centos-67> (0m0.00s).
-----> Testing <default-centos-67>
-----> Creating <default-centos-67>...
       Sending build context to Docker daemon  2.56 kB
       Sending build context to Docker daemon
       Step 0 : FROM centos:centos6
        ---> 72703a0520b7
       Step 1 : RUN yum clean all
```

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'
```

Lab: Commit Your Work

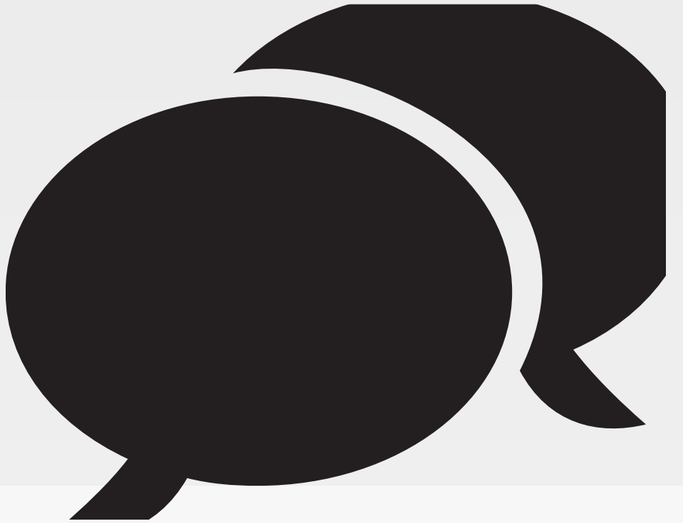
```
$ cd ~/cookbooks/apache
$ git add .
$ git status
$ git commit -m "Release version 0.2.0"
```

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

What is the major difference between a single-quoted string and a double-quoted string?

How are the details about the system available within a recipe?

How does the version number help convey information about the state of the cookbook?

CHEF™