Installing the ChefDK

Installing the tools on your system

Objective:

- ❑ Install the ChefDK
- ❑ Open a Terminal / Command Prompt
- ❑ Execute a series of commands to ensure everything is installed
- ❑ Install a text editor (optional)

©2016 Chef Software Inc.

CHEF

ChefDK

The ChefDK contains tools like chef, chef-client, and kitchen.

You can find the ChefDK to download at the website downloads.chef.io. In most cases you should just download the latest version.

https://downloads.chef.io/chef-dk/

©2016 Chef Software Inc.

GL: Download the ChefDK

ChefDK is a tool chain built on top of the Ruby programming language.

The ChefDK installer does not install any particular graphical-user-interface—installs CLI instead

https://downloads.chef.io/chef-dk/

GL: Installing ChefDK

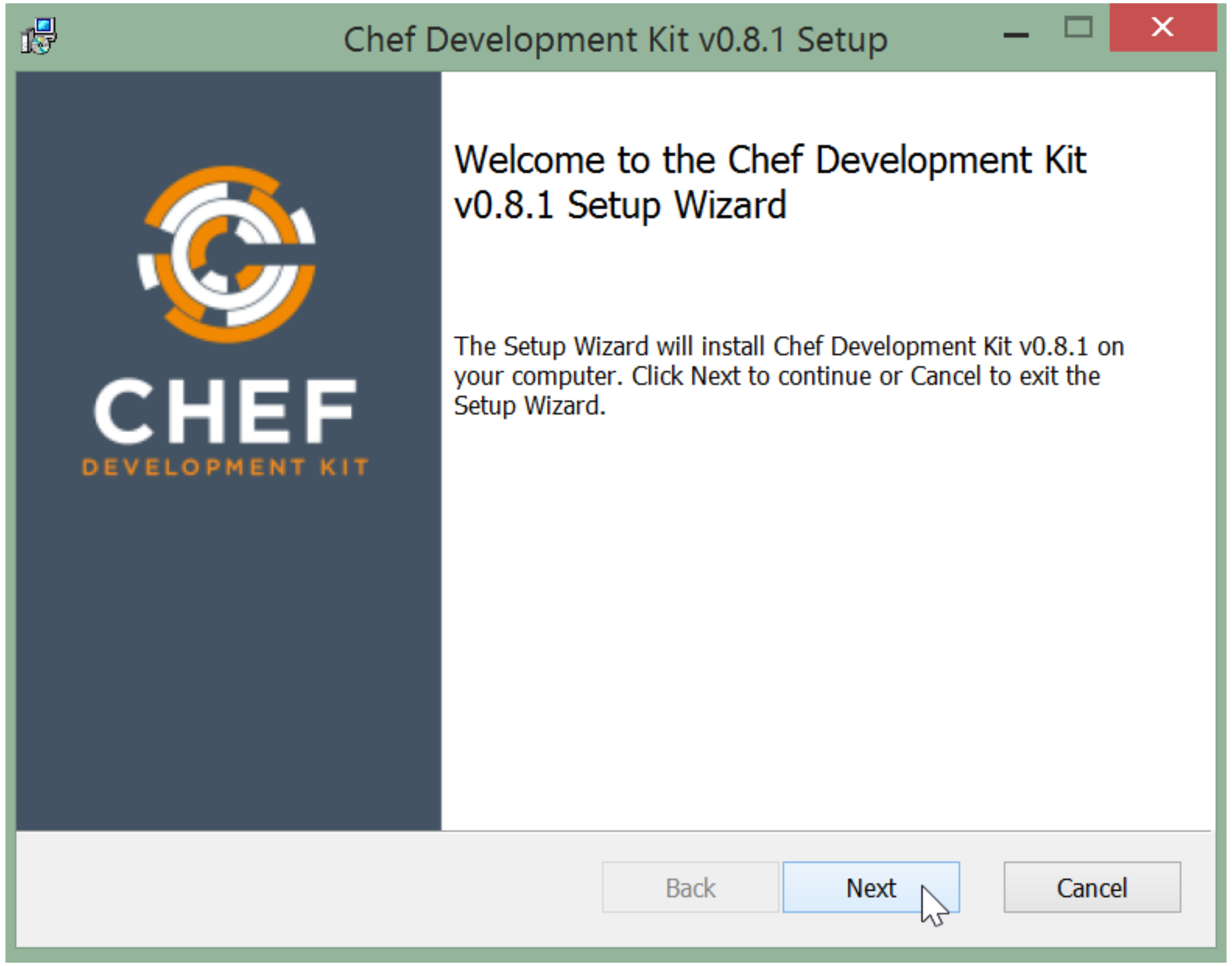
The omnibus installer is used to set up the Chef development kit on a workstation, including the chef-client itself, an embedded version of Ruby, RubyGems, OpenSSL, key-value stores, parsers, libraries, command line utilities, and community tools such as Kitchen, Berkshelf, and ChefSpec.

https://downloads.chef.io/chef-dk/

©2016 Chef Software Inc.

CHEF™

GL: Installing ChefDK



Welcome to the Chef Development Kit v0.8.1 Setup Wizard

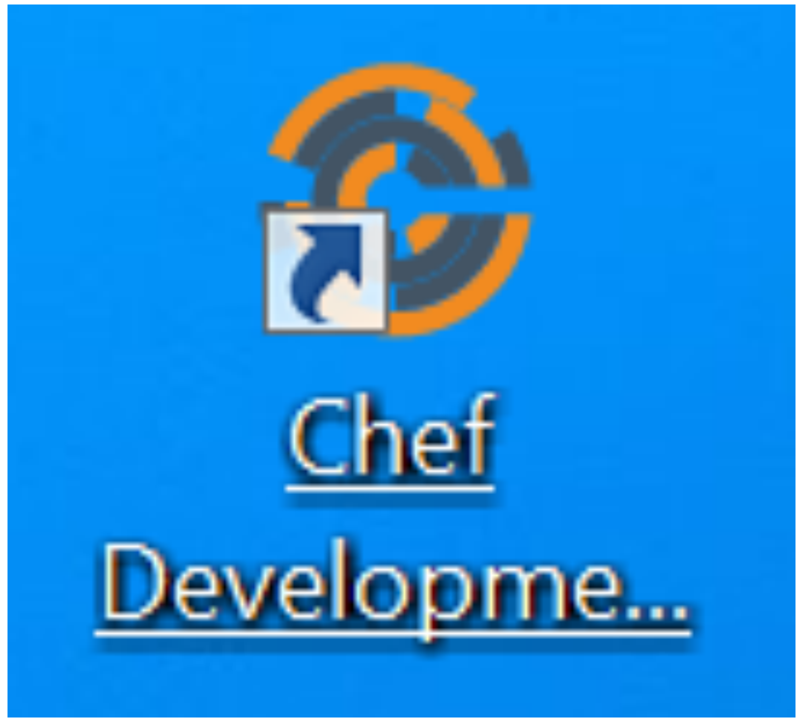
The Setup Wizard will install Chef Development Kit v0.8.1 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

©2016 Chef Software Inc.

GL: ChefDK PowerShell

For Windows users, ChefDK installs a Chef version of PowerShell on your desktop that ensures everything is properly set up for the shell.

You should run all the commands in the rest of the labs from within this ChefDK shell.



Mac users' native terminals will be properly set up by ChefDK.

Lab: Run All These Commands

```
$ chef --version
$ chef-client --version
$ knife --version
$ ohai --version
$ berks --version
$ kitchen --version
$ foodcritic --version
$ cookstyle --version
```

©2016 Chef Software Inc.

CHEF™

CONCEPT

git

Git was initially designed and developed by Linus Torvalds for Linux kernel development in 2005, and has since become the most widely adopted version control system for software development.

http://git-scm.com/downloads

CONCEPT

Text Editors

When working with Chef you spend a large time editing files.

Whatever editor you use should optimize for this workflow.

CONCEPT

ATOM Editor

Atom is modern, approachable, and hackable to the core. We can't wait to see what you build with it.

https://atom.io

CONCEPT

Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs.

https://code.visualstudio.com

©2016 Chef Software Inc.

CHEF™

Configure Your Development Environment:

Vagrant and VirtualBox

Chef Workflow

When building and testing Chef code a normal workflow involves managing servers directly from your workstation. In this class, you'll start by logging into a server directly to get to know the way Chef works.

In the second half of class, we'll manage remote servers (nodes) using a workstation connected to a Chef Server. For now, we'll start by managing a node using Vagrant and VirtualBox.

Objectives:

- Check pre-reqs
- Verify your ssh client
- Install VirtualBox
- Install Vagrant
- Launch Centos VM

Ensure your system meets the minimum requirements

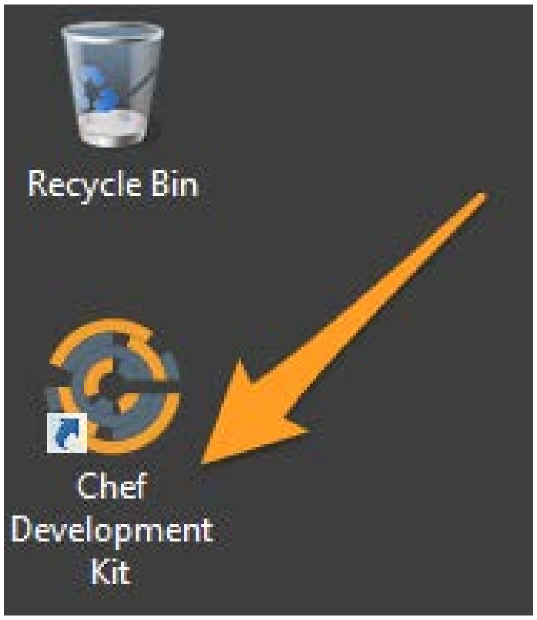
Running virtual machines can be demanding on your system's hardware. Before proceeding with the labs, please ensure that your system:

1. Supports virtualization. This is typically enabled in your BIOS
2. Meets the minimum requirements to run VirtualBox
3. Has at least 5GB of hard-disk space available
4. Has at least 512MB RAM available for each VM you would like to run (this class may have a maximum of 3 VM's running at any time, so please ensure you have at least 1.5GB RAM available)
5. If using VMware Fusion, ensure nested virtualization is enabled
6. If your system doesn't meet the pre-reqs, consider one of the other options for completing the class exercises.

Verify your ssh client

This class uses ssh to connect to our instances. This usually involves an ssh client. If using MacOS or most Linux workstations the terminal should work.

If using Windows or having trouble connecting, the ChefDK includes Git for Windows, an ssh client.



Disclaimers:

Chef integrates easily with VirtualBox and Vagrant, but these projects are not maintained by Chef. If something isn't working, we recommend the following:

- Check the Vagrant Issues or VirtualBox Bugtracker pages
- Refer to documentation:
  - Vagrant
  - VirtualBox

This class is known to work with the versions of the software we suggest installing. If something isn't working, check that you're running with the tested versions of the following:

- ChefDK 0.18.30
- Vagrant 1.8.6
- VirtualBox 5.1.8r111374
- Optional: Git 2.8.2

Class Workflow

For the first half of the class we will log into a Vagrant instance and work with Chef by directly managing the virtual machine

On the virtual CentOS instance we will install the Chef Development Kit (ChefDK) and write code using a command-line text editor, like Vi, Emac or Nano

In the second half of the class we will manage several Vagrant instances remotely using a Chef Server.

For these exercises we will be using your local machine, where the ChefDK will also be installed. You can use any text editor you prefer for these exercises. I'll be using Vim or Sublime Text throughout the video demos.

Install the Chef Development Kit

You can install ChefDK from here
or
On Windows you can run the
installation script->

```
PS > . { iwr -useb https://omnitruck.chef.io/install.ps1 } | iex; install -project chefdk -channel stable -version  1.0.3
```

After installing on Linux and MacOS, check that the tools can be found by running:

```
$ chef --version
```

On Windows, launch the ChefDK to run *chef --version* or update your PATH to include the tools in your Powershell session

Install VirtualBox

- Windows: Consider using the Chocolatey Installer or
- Download VirtualBox directly from Oracle

Verify the installation by running:

```
$ VBoxManage --version
5.1.8r111374
```

Note: On Windows, VirtualBox is installed to C:\Program Files\Oracle\VirtualBox
If the command fails, you may need to update your system PATH:

```
PS> $path = [Environment]::GetEnvironmentVariable("PATH", "Machine")
PS> $vbox_path = "C:\Program Files\Oracle\VirtualBox"
PS> [Environment]::SetEnvironmentVariable("PATH", "$path;$vbox_path", "Machine")
```

Install Vagrant

Windows: Consider using the Chocolatey Installer
Download Vagrant directly from HashiCorp

Verify the installation by running:

```
$ vagrant --version
Vagrant 1.8.6
```

Note: On Windows, Vagrant is installed to
C:\HashiCorp\Vagrant\bin
To add Vagrant to your system PATH, run:

```
PS> $path = [Environment]::GetEnvironmentVariable(“PATH”, “Machine”)
PS> $vagrant_path = “C:\HashiCorp\Vagrant\bin”
PS> [Environment]::SetEnvironmentVariable(“PATH”, “$path;$vagrant_path”, “Machine”)
```

Download a CentOS 7.2 Vagrant Box

```
$ vagrant box add bento/centos-7.2 --provider=virtualbox

==> box: Loading metadata for box ‘bento/centos-7.2’
    box: URL: https://atlas.hashicorp.com/bento/centos-7.2
==> box: Adding box ‘bento/centos-7.2’ (v2.3.0) for provider: virtualbox
    box: Downloading: https://atlas.hashicorp.com/bento/boxes/centos-7.2/versions/2.3.0/providers/virtualbox.box
```

This downloads a VirtualBox-compatible CentOS 7.2 Vagrant box. The bento/centos-7.2 image is retrieved from the HashiCorp Atlas, and refers to a Chef project that provides Vagrant boxes to make testing on common platforms easy.

Note: On Windows, if the command fails with a blank error message you may need to install Microsoft Visual C++ 2010 SP1. See the Vagrant issues here.

Launch a CentOS 7.2 Instance

```
$ vagrant init bento/centos-7.2

A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.

$ vagrant up

Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'bento/centos-7.2'...
==> default: Checking if box 'bento/centos-7.2' is up to date...
==> default: Setting the name of the VM: root_default_1476898305221_53382
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Mounting shared folders...
    default: /vagrant => /root
```

Log Into the CentOS 7.2 Instance

```
$ vagrant ssh
```

And Install the ChefDK

```
[vagrant@localhost ~]$ curl https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk -c stable -v 0.18.30

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 20051  100 20051    0     0  48583      0 --:--:-- --:--:-- --:--:-- 48549
warning: /tmp/install.sh.12800/chefdk-0.18.30-1.el7.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 83ef826a: NOKEYel 7 x86_64
Getting information for chefdk stable 0.18.30 for el...
downloading https://omnitruck.chef.io/stable/chefdk/metadata?v=0.18.30&p=el&pv=7&m=x86_64
  to file /tmp/install.sh.12800/metadata.txt
[...]
Installing chefdk 0.18.30
installing with rpm...
Preparing...                          #################################
Updating / installing...
chefdk-0.18.30-1.el7                  #################################
Thank you for installing Chef Development Kit!
```

Setup Your Text Editor

We'll be writing code in this class to configure remote machines. Install your text editor of choice.

If you're new to command-line text editors, we recommend trying Nano.

Learn Vim

```
[vagrant@localhost ~]$ sudo yum install vim -y
```

Learn Emacs

```
[vagrant@localhost ~]$ sudo yum install emacs -y
```

Learn Nano

```
[vagrant@localhost ~]$ sudo yum install vim -y
```

Manage Your Vagrant Instance

Run these common commands in the same directory as your Vagrantfile:

```
$ vagrant init
```

creates a Vagrantfile, used to specify virtual machine settings

```
$ vagrant up
```

spins up the virtual machine using the Vagrantfile

```
$ vagrant ssh-config
```

list connection details for running instances

```
$ vagrant status
```

lists virtual machines and current status. 'running' means machine is available for ssh

```
$ vagrant suspend
```

save machine state and shut down

```
$ vagrant destroy --force
```

destroy all running virtual machines

Configure Your Development Environment:

Amazon Web Services

Chef Workflow

When building and testing Chef code a normal workflow involves managing servers directly from your workstation. In this class, you'll start by logging into a server directly to get to know the way Chef works.

In the second half of class, we'll manage remote servers (nodes) using a workstation connected to a Chef Server. For now, we'll start by managing a node using AWS.

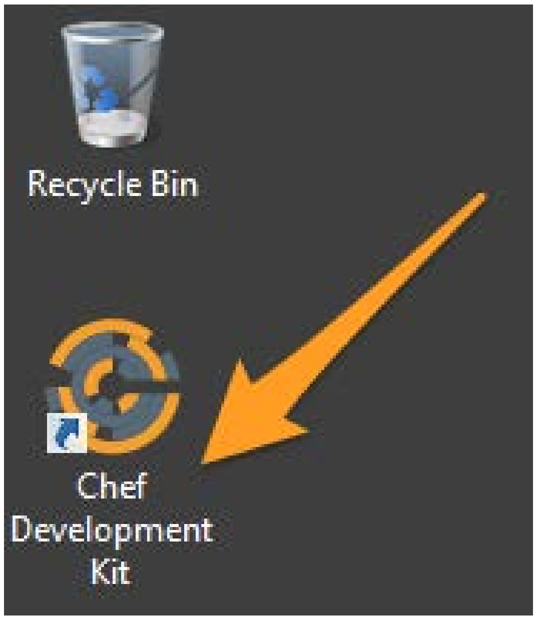
Objectives:

- Check pre-reqs
- Verify your ssh client
- Create AWS Account
- Launch Centos VM

Verify your ssh client

This class uses ssh to connect to our instances. This usually involves an ssh client. If using MacOS or most Linux workstations the terminal should work.

If using Windows or having trouble connecting, the ChefDK includes Git for Windows, an ssh client.



Disclaimers:

The output that you witness may be different depending on what Amazon Machine Images you use. If you have issues with Amazon Web Services, visit the discussion forums.

This class is known to work with the versions of the software we suggest installing. If something isn't working, check that you're running with the tested versions of the following:

- ChefDK 0.18.30
- Optional: Git 2.8.2

Install the Chef Development Kit

You can install ChefDK from here
or
On Windows you can run the installation script->

```
PS > . { iwr -useb https://omnitruck.chef.io/install.ps1 } | iex; install -project chefdk -channel stable -version  1.0.3
```

Create/Access your AWS Account

You will need to log into your AWS account; if you don't have one yet you can take advantage of AWS's one-year free account that gives you 750 hours of the t2.micro tier use each month

If you are unable to use AWS, you may try using one of the other options offered for this class

Launch a CentOS 7 Instance

Spin up a CentOS instance on Amazon EC2
If you haven't used AWS before, this tutorial will show you how to do so

Launch a CentOS 7 instance from either the AWS console or from the AWS Marketplace
Select t2.micro to implement the free usage tier

In this process when you create your security group,
You must open ports 22 (ssh), 80 (http), and 443 (https)

Class Workflow

For the first half of the class we will log into an AWS instance and work with Chef by directly managing the virtual machine

On the virtual CentOS instance we will install the Chef Development Kit (ChefDK) and write code using a command-line text editor, like Vi, Emac or Nano

In the second half of the class we will manage several AWS instances remotely using a Chef Server.

For these exercises we will be using your local machine, where the ChefDK will also be installed. You can use any text editor you prefer for these exercises. I'll be using Vim or Sublime Text throughout the video demos.

Connect to your AWS Instance

You will now need to connect to your CentOS VM instance

Amazon provides documentation of how to connect using a web browser or with ssh

```
$ ssh -i /path/to/private/key centos@52.87.191.240
```

Install the Chef Development Kit

From your ssh connection, run the following command to install the ChefDK:

```
$ curl https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk -c stable -v 0.18.30
```

After installing, check that the tools can be found by running:

```
$ chef --version
```

Setup Your Text Editor

We'll be writing code in this class to configure remote machines. Install your text editor of choice.

If you're new to command-line text editors, we recommend trying Nano.

Learn Vim

```
[vagrant@localhost ~]$ sudo yum install vim -y
```

Learn Emacs

```
[vagrant@localhost ~]$ sudo yum install emacs -y
```

Learn Nano

```
[vagrant@localhost ~]$ sudo yum install vim -y
```

Clean Up

To reduce the costs of running your instances, you will need to stop your VM

At the completion of this class you will need to destroy your instances

DOCS

Resources

A resource is a statement of configuration policy.

It describes the desired state of an element of your infrastructure and the steps needed to bring that item to the desired state.

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

©2016 Chef Software Inc.

Example: Package

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

The package named 'httpd' is installed.

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

Example: Service

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

The service named 'ntp' is enabled (start on reboot) and started.

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

Example: File

```
file '/etc/motd' do
  content 'This computer is the property ...'
end
```

The file name '/etc/motd' is created with content 'This computer is the property ...'

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

Example: File

```
file '/etc/php.ini.default' do
  action :delete
end
```

The file name '/etc/php.ini.default' is deleted.

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

GL: Hello, World?

I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application.

Objective:

- ❑ Create a recipe that writes out a file with the contents "Hello, world!"
- ❑ Apply that recipe to the workstation
- ❑ Verify the contents of the file

GL: Create and Open a Recipe File

```
$ nano hello.rb
```

GL: Create a Recipe File Named hello.rb

```
~/hello.rb
```

```
file '/hello.txt' do
  content 'Hello, world!'
end
```

The file named '/hello.txt' is created with the content 'Hello, world!'

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

GL: Apply the Recipe File

```
$ sudo chef-client --local-mode hello.rb
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-02-19T13:08:13+00:00] WARN: Node ip-172-31-12-176.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/chef/hello.rb
  * file[hello.txt] action create
    - create new file hello.txt
    - update content in file hello.txt from non to 315f5b
    +++ ./.hello.txt20160224-8559-19kqial
      2016-02-24 16:51:04.400844959 +0000
    @@ -1 +1,2 @@
    +Hello, world!
```

GL: What Does hello.txt Say?

```
$ cat /hello.txt
```

```
Hello, world!
```

GL: Hello, World?

I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application.

Objective:

✓ Create a recipe that writes out a file with the contents "Hello, world!"
✓ Apply that recipe to the workstation
✓ Verify the contents of the file

©2016 Chef Software Inc.

CONCEPT

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

©2016 Chef Software Inc.

CONCEPT

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

CONCEPT

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be ACTION'd with PROPERTIES

CHEF

CONCEPT

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

CONCEPT

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

?

The TYPE named NAME should be **ACTION'd** with PROPERTIES

CONCEPT

Resource Definition

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

?

The **TYPE** named **NAME** should be **ACTION'd** with **PROPERTIES**

CHEF

CONCEPT

Resource Definition

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

If no **ACTION** is listed, the default action is applied.

https://docs.chef.io/resource.html#resources-syntax

CHEF

CONCEPT

chef-client

chef-client is an agent that runs locally on every node that is under management by Chef.

When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state.

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

CHEF

CONCEPT

--local-mode (or -z)

chef-client's default mode attempts to contact a Chef Server and ask it for the recipes to run for the given node.

We are overriding that behavior to have it work in a local mode.

GL: Apply the Setup Recipe

```
$ sudo chef-client --local-mode hello.rb
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-02-19T13:08:13+00:00] WARN: Node ip-172-31-12-176.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/chef/moo.rb
  * yum_package[nano] action install
    - install version 3.03-8.e16 of package cowsay
Running handlers:
Running handlers complete
Chef Client finished, 1/1 resources updated in 38 seconds
```

©2015 Chef Software Inc.

CHEF

Discussion

What would happen if the 'hello.txt' file contents were modified?

©2016 Chef Software Inc.

Test and Repair

What would happen if the file permissions (mode), owner, or group changed?

Have we defined a policy for these properties?

©2016 Chef Software Inc.

CHEF™

CONCEPT

Test and Repair

`chef-client` takes action only when it needs to. Think of it as test and repair.

Chef looks at the current state of each resource and takes action only when that resource is out of policy.

Lab: The `file` Resource

- ❑ **Read https://docs.chef.io/resources.html**
- ❑ **Discover the file resource's:**
  - default action.
  - default values for **`mode`**, **`owner`**, and **`group`**.

- ❑ **Update the `file` policy in "hello.rb" to:**

  The file named 'hello.txt' should be created with the content 'Hello, world!', mode '0644', owner is 'root', and group is 'root'.

Lab: The Updated file Resource

~/hello.rb

```
file 'hello.txt' do
  content 'Hello, world!'
  mode '0644'                    +
  owner 'root'
  group 'root'
  action :create
end
```

The default mode is set by the POSIX Access Control Lists.

The default owner is the current user (could change).

The default group is the POSIX group (if available).

The default action is to create (not necessary to define it).

Lab: The `file` Resource

✓ **Read** https://docs.chef.io/resources.html

✓ **Discover the file resource's:**

- default action.
- default values for **`mode`**, **`owner`**, and **`group`**.

✓ **Update the `file` policy in "hello.rb" to:**

The file named 'hello.txt' should be created with the content 'Hello, world!', mode '0644', owner is 'root', and group is 'root'.

©2016 Chef Software Inc.

Lab: Workstation Setup

- ❑ Create a recipe file named "setup.rb" that defines the policy:
  - The packages named 'tree' and ‘ntp’ are installed.
  - The file named '/etc/motd' is created with the content 'This server is the property of <Your Name>', and an owner and group of ‘root’
- ❑ Use chef-client to apply the recipe file named "setup.rb"

©2016 Chef Software Inc.

Lab: Workstation Setup Recipe File

~/setup.rb

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

package 'ntp' do
  action :install
end

file '/etc/motd' do
  content 'Property of ...'
  owner ‘root’
  group ‘root’
end
```

The packages named 'tree' and ‘ntpd’ are installed.

The file named '/etc/motd' is created with the content 'Property of ...' and an owner and group of ‘root’

©2016 Chef Software Inc.

CHEF

GL: Apply the Recipe File

```
$ sudo chef-client --local-mode setup.rb
```

```
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install
    - install version 1.5.3-3.el6 of package tree
  * file[/etc/motd] action create
    - update content in file /etc/motd from e3b0c4 to d100eb
    --- /etc/motd     2010-01-12 13:28:22.000000000 +0000
    +++ /etc/.motd20160224-8754-1xczeyn     2016-02-24 16:57:57.203844958 +0000
    @@ -1 +1,2 @@
    +Property of ...
Running handlers:
Running handlers complete
Chef Client finished, 2/2 resources updated in 17 seconds
```

©2015 Chef Software Inc.

Discussion

What is a resource?

What are some other possible examples of resources?

How did the example resources we wrote describe the desired state of an element of our infrastructure?

What does it mean for a resource to be a statement of configuration policy?

©2016 Chef Software Inc.

Cookbooks

Organizing Recipes

©2016 Chef Software Inc.

Questions You May Have

1. Thinking about the workstation recipe, could we do something like that for a web server?
2. Is there a way to package up recipes you create with a version number (and maybe a README)?
3. I think `chef` is able to generate something called a cookbook. Shouldn't we start thinking about some version control so we don't lose all our hard work?

©2016 Chef Software Inc.

CHEF

Objectives

After completing this module, you should be able to:

- ➢ Modify a recipe
- ➢ Use version control
- ➢ Generate a Chef cookbook
- ➢ Define a Chef recipe that sets up a web server

©2016 Chef Software Inc.

GL: Create a Cookbook

How are we going to manage this file? Does it need a README?

Objective:

- ❑ Use chef to generate a cookbook
- ❑ Move the setup recipe into the new cookbook
- ❑ Add the new cookbook to version control

Cookbooks

A Chef cookbook is the fundamental unit of configuration and policy distribution.

Each cookbook defines a scenario, such as everything needed to install and configure MySQL, and then it contains all of the components that are required to support that scenario.

Read the first three paragraphs here: http://docs.chef.io/cookbooks.html

Cookbooks

- Recipes that specify the resources to use and the order in which they are to be applied
- Attribute values
- File distributions
- Templates
- Extensions to Chef, such as libraries, definitions, and custom resources
- Version Control

©2016 Chef Software Inc.

Cookbooks

- Common Components
  a. README
  b. metadata
  c. recipes
  d. testing directories (spec + test)

GL: Create a Cookbooks Directory

```
$ mkdir cookbooks
```

CONCEPT

What is 'chef'?

An executable program that allows you generate cookbooks and cookbook components.

What can 'chef' do?

```
$ chef --help
```

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

Available Commands:
    exec        Runs the command in context of the embedded ruby
    gem         Runs the `gem` command in context of the embedded ruby
    generate    Generate a new app, cookbook, or component
    shell-init  Initialize your shell to use ChefDK as your primary ruby
    install     Install cookbooks from a Policyfile and generate a locked cookboo...
    update      Updates a Policyfile.lock.json with latest run_list and cookbooks
```

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
```

GL: Let's Create a Cookbook

```
$ chef generate cookbook cookbooks/workstation
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
* directory[/home/chef/workstation] action create
    - create new directory /home/chef/workstation
  * template[/home/chef/workstation/metadata.rb] action create_if_missing
    - create new file /home/chef/workstation/metadata.rb
    - update content in file /home/chef/workstation/metadata.rb from none to bd85d3
    (diff output suppressed by config)
  * template[/home/chef/workstation/README.md] action create_if_missing
    - create new file /home/chef/workstation/README.md
    - update content in file /home/chef/workstation/README.md from none to 44d165
    (diff output suppressed by config)
  * cookbook_file[/home/chef/workstation/chefignore] action create
```

GL: The Cookbook Has a README

```
$ tree cookbooks/workstation
```

```
workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
10 directories, 9 files
```

CONCEPT

README.md

The description of the cookbook's features written in Markdown.

http://daringfireball.net/projects/markdown/syntax

GL: The Cookbook Has Some Metadata

```
$ tree cookbooks/workstation
```

```
workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
10 directories, 9 files
```

©2016 Chef Software Inc.

DOCS

metadata.rb

Every cookbook requires a small amount of metadata. Metadata is stored in a file called metadata.rb that lives at the top of each cookbook's directory.

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

GL: Let's Take a Look at the Metadata

```
$ cat 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          '0.1.0'
```

GL: The Cookbook Has a Folder for Recipes

```
$ tree cookbooks/workstation
```

```
workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
10 directories, 9 files
```

GL: The Cookbook Has a Default Recipe

```
$ cat cookbooks/workstation/recipes/default.rb
```

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

GL: Create a Cookbook

How are we going to manage this file? Does it need a README?

Objective:

- ✓ Use chef to generate a cookbook
- ✓ Move the setup recipe into the new cookbook
- ✓ Add the new cookbook to version control

©2016 Chef Software Inc.

GL: Copy the Recipe into the Cookbook

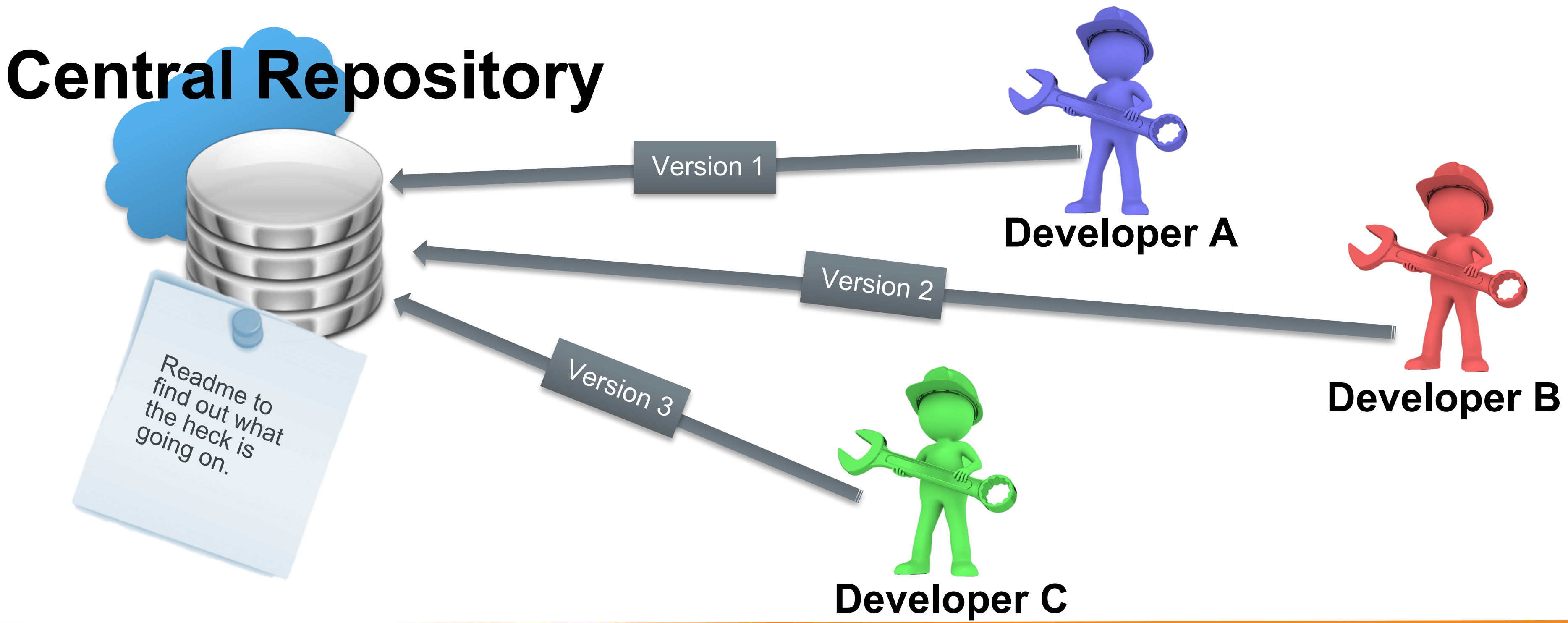
```
$ mv setup.rb cookbooks/workstation/recipes
```

GL: Verify the Cookbook has the Recipe

```
$ tree cookbooks/workstation
```

```
cookbooks/workstation
├── Berksfile
├── chefignore
├── metadata.rb
├── README.md
├── recipes
│   ├── default.rb
│   └── setup.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
```

Collaboration and Version Control



Git Version Control

git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows.

git is commonly used to track changes to your cookbooks.

©2016 Chef Software Inc.

Lab: Install git

- ❑ Add the additional policy to the "setup.rb":

  The package named 'git' is installed.

- ❑ Then apply this recipe with chef-client.

©2016 Chef Software Inc.

Lab: Adding the git Package

~/setup.rb

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

package 'git' do                                        +
  action :install
end

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

Lab: Re-apply the Setup Recipe

```
$ sudo chef-client --local-mode setup.rb
```

```
Converging 4 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * yum_package[git] action install
    - install version 1.7.1-3.el6_4.1 of package git
  * file[/etc/motd] action create (up to date)
```

Lab: Install git

✓ Add the additional policy to the "setup.rb":

The package named 'git' is installed.

✓ Then apply this recipe with chef-client.

Group Exercise: Version Control

This is a probably a good point to capture the initial state of our cookbook.

Objective:

- ✓ Use chef to generate a cookbook
- ✓ Move the setup recipe into the new cookbook
- ✓ Add the new cookbook to version control

©2016 Chef Software Inc.

GL: Move into the Cookbook Directory

```
$ cd cookbooks/workstation
```

GL: Initialize the Directory as a git Repository

```
$ git init
```

```
Reinitialized existing Git repository in /home/chef/cookbooks/workstation/.git/
```

©2016 Chef Software Inc.

GL: Use 'git add' to Stage Files to be Committed

```
$ git add .
```

CONCEPT

Staging Area

The staging area has a file, generally contained in your Git directory, that stores information about what will go into your next commit.

It's sometimes referred to as the "index", but it's also common to refer to it as the staging area.

http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

GL: Use 'git status' to View the Staged Files

```
$ git status
```

```
On branch master


Initial commit


Changes to be committed:
  (use "git rm --cached <file>..." to unstage)


        new file:   .gitignore
        new file:   .kitchen.yml
        new file:   Berksfile
        new file:   README.md
        new file:   chefignore
        new file:   metadata.rb
```

©2016 Chef Software Inc.

CHEF

GL: Use 'git commit' to Save the Staged Changes

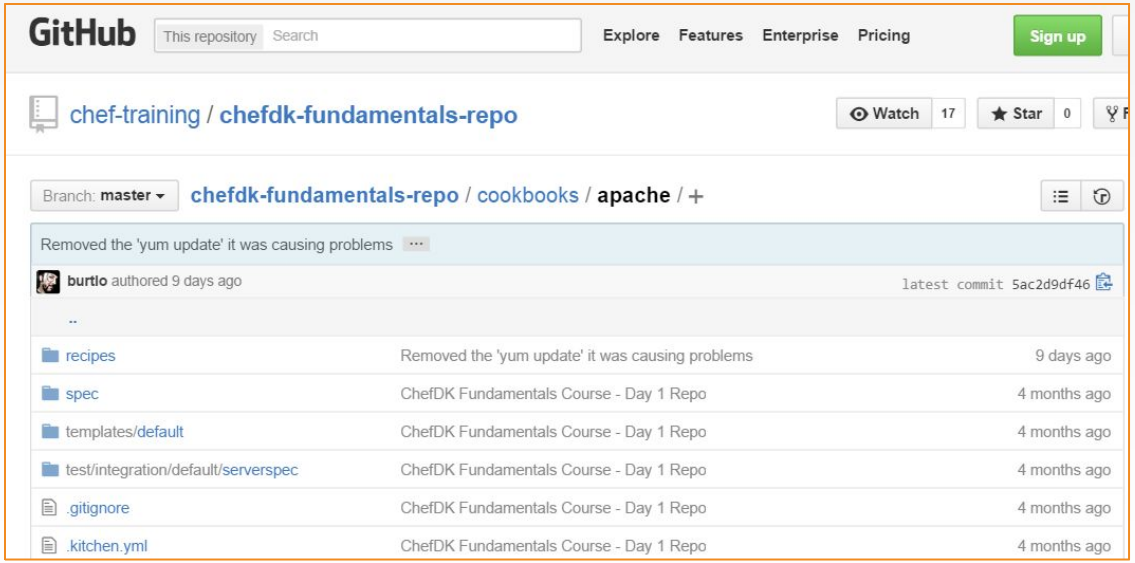
```
$ git commit -m "Initial commit"
```

```
master (root-commit) 9998472] Initial workstation cookbook
 Committer: ChefDK User <chef@ip-172-31-59-191.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author
```

Git Version Control

If you use git versioning you should ultimately push the local git repository to a shared remote git repository.

In this way others could collaborate with you from a centralized location.



chef-training / chefdk-fundamentals-repo

Branch: master

chefdk-fundamentals-repo / cookbooks / apache / +

Removed the 'yum update' it was causing problems ···

burtlo authored 9 days ago latest commit 5ac2d9df46

..

recipes	Removed the 'yum update' it was causing problems	9 days ago
spec	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
templates/default	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
test/integration/default/serverspec	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
.gitignore	ChefDK Fundamentals Course - Day 1 Repo	4 months ago
.kitchen.yml	ChefDK Fundamentals Course - Day 1 Repo	4 months ago

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CONCEPT

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

CONCEPT

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: Setting up a Web Server

- ❑ Use `chef generate` to create a cookbook named "apache".
- ❑ Write and apply a recipe named **`"server.rb"`** with the policy:

  The package named 'httpd' is installed.

  The file named '/var/www/html/index.html' is created with the content '<h1>Hello, world!</h1>'

  The service named 'httpd' is started and enabled.
- ❑ Apply the recipe with chef-client
- ❑ Verify the site is available by running **`curl localhost`**

©2016 Chef Software Inc.

GL: Return to the Home Directory

```
$ cd ~
```

Lab: Create a Cookbook

```
$ chef generate cookbook cookbooks/apache
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[/home/chef/cookbooks/apache] action create
    - create new directory /home/chef/cookbooks/apache
  * template[/home/chef/cookbooks/apache/metadata.rb] action create_if_missing
    - create new file /home/chef/cookbooks/apache/metadata.rb
    - update content in file /home/chef/cookbooks/apache/metadata.rb from none
to 37ed5f
    (diff output suppressed by config)
  * template[/home/chef/cookbooks/apache/README.md] action create_if_missing
    - create new file /home/chef/cookbooks/apache/README.md
    - update content in file /home/chef/cookbooks/apache/README.md from none to
5c3d3a
    (diff output suppressed by config)
```

©2016 Chef Software Inc.

CHEF

Lab: Create a Cookbook

```
$ chef generate recipe cookbooks/apache server
```

```
Compiling Cookbooks...
Recipe: code_generator::recipe
  * directory[cookbooks/apache/spec/unit/recipes] action create (up to date)
  * cookbook_file[cookbooks/apache/spec/spec_helper.rb] action create_if_missing
 (up to date)
  * template[cookbooks/apache/spec/unit/recipes/server_spec.rb] action
create_if_missing
    - create new file cookbooks/apache/spec/unit/recipes/server_spec.rb
    - update content in file cookbooks/apache/spec/unit/recipes/server_spec.rb
from none to a43970
    (diff output suppressed by config)
  * template[cookbooks/apache/recipes/server.rb] action create
    - create new file cookbooks/apache/recipes/server.rb
    - update content in file cookbooks/apache/recipes/server.rb from none to
3d6b92
```

Lab: Create the Server Recipe

```
~/cookbooks/apache/recipes/server.rb
```

```
package 'httpd'

file '/var/www/html/index.html' do
  content '<h1>Hello, world!</h1>'
end

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

Lab: Apply the Server Recipe

```
$ sudo chef-client -z cookbooks/apache/recipes/server.rb
```

```
Converging 3 resources
Recipe: @recipe_files::/home/chef/cookbooks/apache/recipes/server.rb
  * yum_package[httpd] action install
    - install version 2.2.15-47.el6.centos.3 of package httpd
  * file[/var/www/html/index.html] action create
    - create new file /var/www/html/index.html
    - update content in file /var/www/html/index.html from none to 17d291
    --- /var/www/html/index.html   2016-02-24 21:41:45.494844958 +0000
    +++ /var/www/html/.index.html20160224-10036-6y8on7 2016-02-24 21:41:45.493844958
+0000
    @@ -1 +1,2 @@
    +<h1>Hello, world!</h1>
  * service[httpd] action enable
    - enable service service[httpd]
```

Lab: Verify That the Website is Available

```
$ curl localhost
```

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

Lab: Setting up a Web Server

- ✓ Use `chef generate` to create a cookbook named "apache".
- ✓ Write and apply a recipe named **`"server.rb"`** with the policy:

  The package named 'httpd' is installed.

  The file named '/var/www/html/index.html' is created with the content '<h1>Hello, world!</h1>'

  The service named 'httpd' is started and enabled.
- ✓ Apply the recipe with chef-client
- ✓ Verify the site is available by running **`curl localhost`**

©2016 Chef Software Inc.

CHEF

COMMIT

GL: Commit Your Work

```
$ cd cookbooks/apache
$ git init
$ git add .
$ git commit -m "Initial commit"
```

©2016 Chef Software Inc.

CHEF

Discussion

What file would you read first when examining a cookbook?

What other recipes might you include in the apache or workstation cookbook?

Can resources accept multiple actions?

How often would you commit changes with version control?

chef-client

Applying Recipes from Multiple Cookbooks

©2016 Chef Software Inc.

chef-client

```
$ sudo chef-client --local-mode RECIPE_FILE
```

How would we apply both the workstation's setup recipe and apache's server recipe?

Objectives

After completing this module, you should be able to use chef-client to:

- Locally apply multiple cookbooks' recipes with chef-client.
- Include a recipe from within another recipe.

CONCEPT

--local-mode

chef-client's default mode attempts to contact a Chef Server and ask it for the recipes to run for the given node.

We are overriding that behavior to have it work in a local mode.

CONCEPT

--run-list "recipe[COOKBOOK::RECIPE]"

In local mode, we need to provide a list of recipes to apply to the system. This is called a **run list**. A run list is an ordered collection of recipes to execute.

Each recipe in the run list must be addressed with the format **recipe[COOKBOOK::RECIPE]**.

©2016 Chef Software Inc.

Demo: Using 'chef-client' to Locally Apply Recipes

```
$ sudo chef-client --local-mode -r "recipe[workstation::setup]"
```

Applying the following recipes locally:

The 'setup' recipe from the 'workstation' cookbook

Demo: Using 'chef-client' to Locally Apply Recipes

```
$ sudo chef-client --local-mode -r "recipe[apache::server]"
```

Applying the following recipes locally:

The 'server' recipe from the 'apache' cookbook

Demo: Using 'chef-client' to Locally Apply Recipes

```
$ sudo chef-client --local-mode \
  -r "recipe[workstation::setup],recipe[apache::server]"
```

Applying the following recipes locally:

- The 'setup' recipe from the 'workstation' cookbook
- The 'server' recipe from the 'apache' cookbook

©2016 Chef Software Inc.

CHEF™

Applying a Run List

Using a run list will allow us to specify things more 'logically' instead of with paths.

Objective:

- ❑ Individually apply the apache cookbook's server recipe and workstation cookbook's setup recipe
- ❑ Apply both the apache cookbook's server recipe and workstation cookbook's setup recipe

©2016 Chef Software Inc.

GL: Return Home First

```
$ cd ~
```

GL: Apply the Cookbook Recipe Locally

```
$ sudo chef-client --local-mode -r "recipe[apache::server]"
```

```
[2016-09-15T14:54:45+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::server"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 4 resources
Recipe: apache::server
  * yum_package[httpd] action install (up to date)
  * file[/var/www/html/index.html] action create (up to date)
  * service[httpd] action enable (up to date)
```

©2016 Chef Software Inc.

GL: Apply the Cookbook Recipe Locally

```
$ sudo chef-client --local-mode -r "recipe[workstation::setup]"
```

```
[2016-09-15T15:15:26+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: ["workstation::setup"]
Synchronizing Cookbooks:
  - workstation (0.1.0)
Compiling Cookbooks...
Converging 4 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * yum_package[git] action install (up to date)
  * file[/etc/motd] action create (up to date)
```

Applying a Run List

Using a run list will allow us to specify things more 'logically' instead of with paths.

Objective:

- ✓ Individually apply the apache cookbook's server recipe and workstation cookbook's setup recipe
- ✓ Apply both the apache cookbook's server recipe and workstation cookbook's setup recipe

GL: Apply Both Recipes Locally

```
$ sudo chef-client --local-mode \
  -r "recipe[apache::server],recipe[workstation::setup]"
```

```
[2016-09-15T15:17:27+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::server","workstation::setup"]
Synchronizing Cookbooks:
  - apache
  - workstation
Compiling Cookbooks...


Running handlers:
[2016-09-15T15:17:30+00:00] ERROR: Running exception handlers
Running handlers complete
```

©2016 Chef Software Inc.

A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- ❑ Load the workstation cookbook's setup recipe in the default recipe
- ❑ Apply the workstation cookbook's default recipe
- ❑ Commit the changes

©2016 Chef Software Inc.

CHEF

CONCEPT

-r "recipe[COOKBOOK(::default)]"

When you are referencing the default recipe within a cookbook you may optionally specify only the name of the cookbook.

chef-client understands that you mean to apply the default recipe from within that cookbook.

©2016 Chef Software Inc.

DOCS

include_recipe

A recipe can include one (or more) recipes located in cookbooks by using the include_recipe method. When a recipe is included, the resources found in that recipe will be inserted (in the same exact order) at the point where the include_recipe keyword is located.

https://docs.chef.io/recipes.html#include-recipes

CHEF

Demo: Including a Recipe

```
include_recipe 'workstation::setup'
```

Include the 'setup' recipe from the 'workstation' cookbook in this recipe

©2016 Chef Software Inc.

Demo: Including a Recipe

```
include_recipe 'apache::server'
```

Include the 'server' recipe from the 'apache' cookbook in this recipe

©2016 Chef Software Inc.

GL: The Default Recipe Includes the Setup Recipe

~/cookbooks/workstation/recipes/default.rb

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

include_recipe 'workstation::setup'
```

A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

✓ Load the workstation cookbook's setup recipe in the default recipe
✓ Apply the workstation cookbook's default recipe
✓ Commit the changes

©2016 Chef Software Inc.

GL: Apply the Cookbook's Default Recipe

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

```
 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: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
Converging 4 resources

Running handlers:
Running handlers complete
Chef Client finished, 0/4 resources updated in 3.300489827 seconds
```

A Succinct Run List

The cookbook only has one recipe that we care about. Could we set that up as the default?

Objective:

- ✓ Load the workstation cookbook's setup recipe in the default recipe
- ✓ Apply the workstation cookbook's default recipe
- ✓ Commit the changes

©2016 Chef Software Inc.

CHEF™

COMMIT

GL: Commit Your Work

```
$ cd cookbooks/workstation
$ git add .
$ git commit -m "Update default recipe to include setup
recipe"
```

Lab

Lab: Update the apache Cookbook

- ❑ Update the "apache" cookbook's "default" recipe to:

  Include the 'server' recipe from the 'apache' cookbook

- ❑ Run chef-client and locally apply the run_list: "recipe[apache]"

- ❑ Commit the changes with version control

©2016 Chef Software Inc.

CHEF

GL: Return Home

```
$ cd ~
```

Lab: The Default Recipe Includes the Apache Recipe

```
~/cookbooks/apache/recipes/default.rb
```

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

include_recipe 'apache::server'
```

©2016 Chef Software Inc.

CHEF

Lab: Applying the apache Default Recipe

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

```
[2016-09-15T15:23:18+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...
Converging 0 resources

Running handlers:
Running handlers complete
Chef Client finished, 0/0 resources updated in 3.310768509 seconds
```

©2016 Chef Software Inc.

COMMIT

Lab: Commit Your Work

```
$ cd cookbooks/apache
$ git add .
$ git commit -m "Update default recipe to include server
recipe"
```

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?

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

CHEF

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
```

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
```

DISCUSSION

Capturing System Data

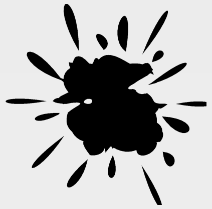
What are the limitations of the way we captured this data?

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

Are these values we would want to capture in our tests?

©2016 Chef Software Inc.

CHEF



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?

©2016 Chef Software Inc.

CHEF

CONCEPT

Ohai!

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

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

CONCEPT

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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": {
```

Running Ohai to Show the IP Address

```
> ohai ipaddress
```

```
[
  "172.31.57.153"
]
```

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"
]
```

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"
]
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

CONCEPT

The Node Object

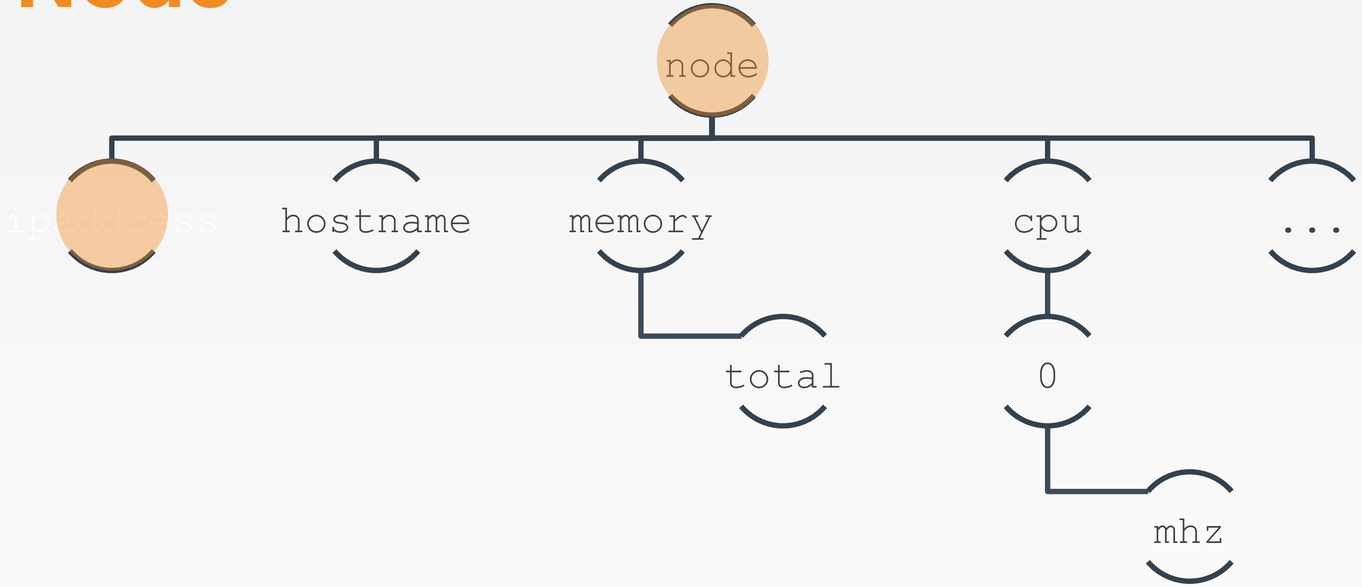
The node object is a representation of our system.
It stores all the attributes found about the system.

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

©2016 Chef Software Inc.

CHEF

The Node

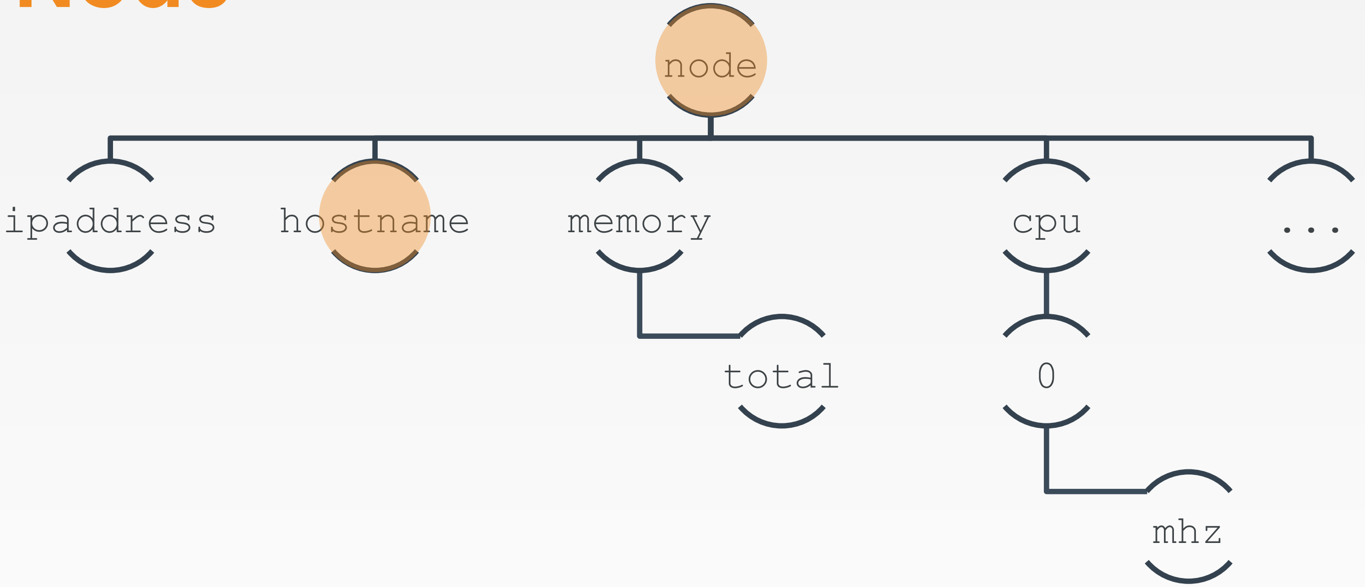


```
IPADDRESS: 172-31-57-153
```

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

©2016 Chef Software Inc.

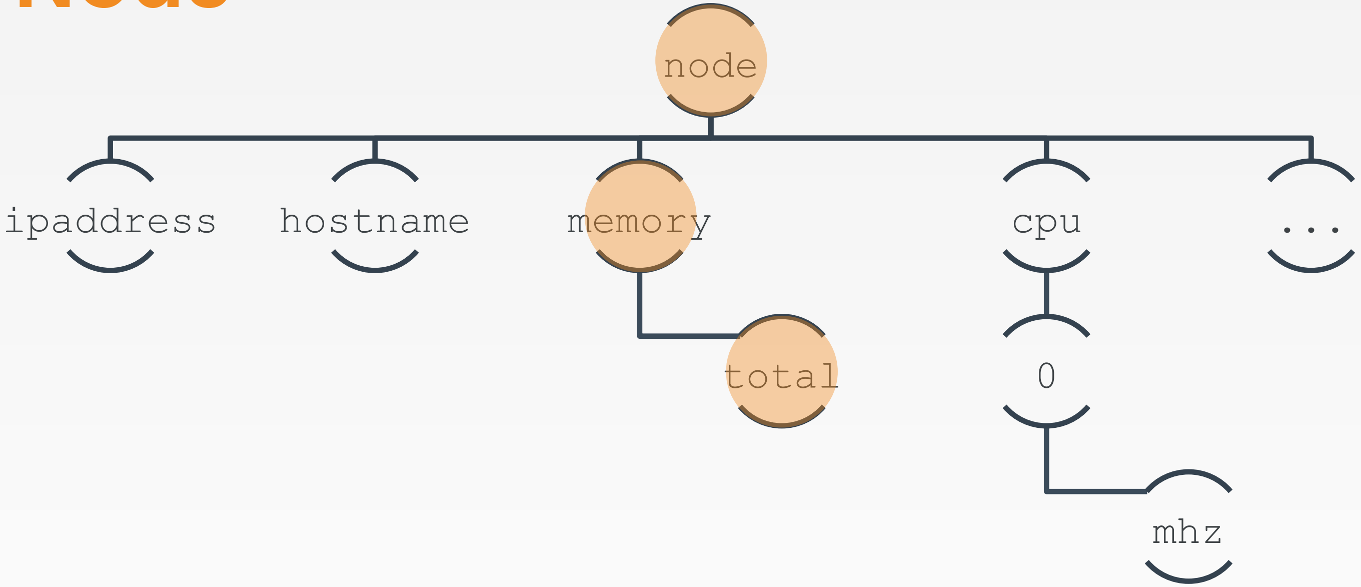
The Node



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

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

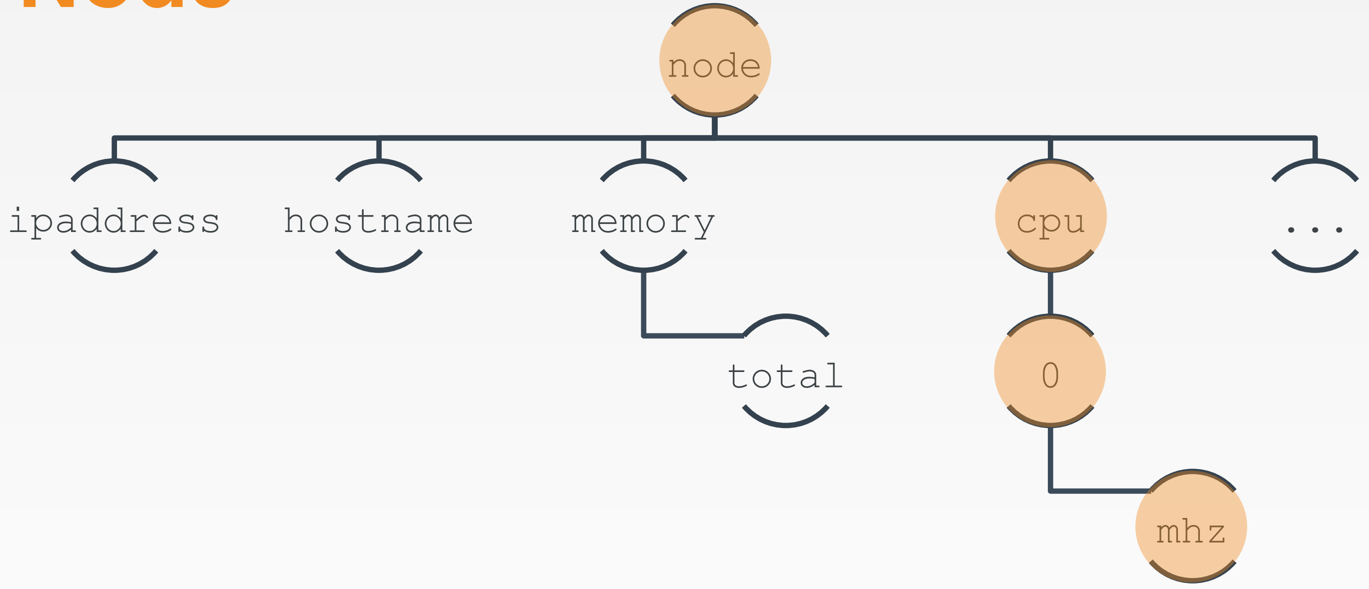
The Node



```
MEMORY: 502272kB
```

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

The Node



```
CPU: 2399.998MHz
```

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

CONCEPT

String Interpolation

```
I have 4 apples
```

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

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

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
```

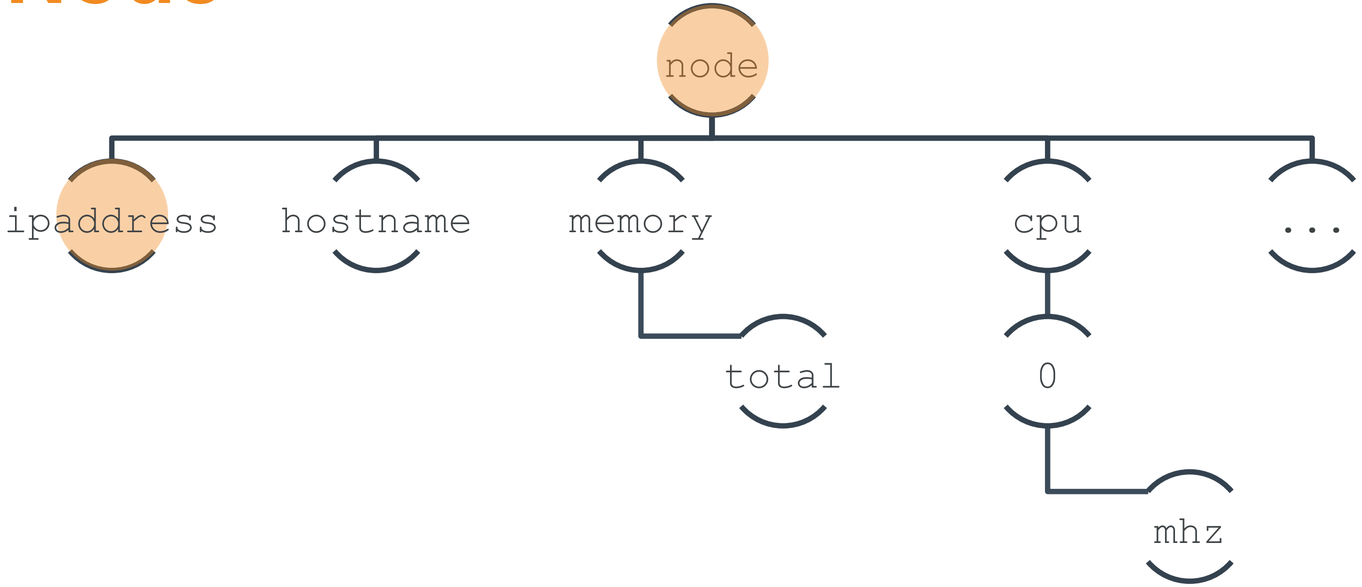
```
apple_count = 4
puts "I have #{apple_count} apples"
```

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

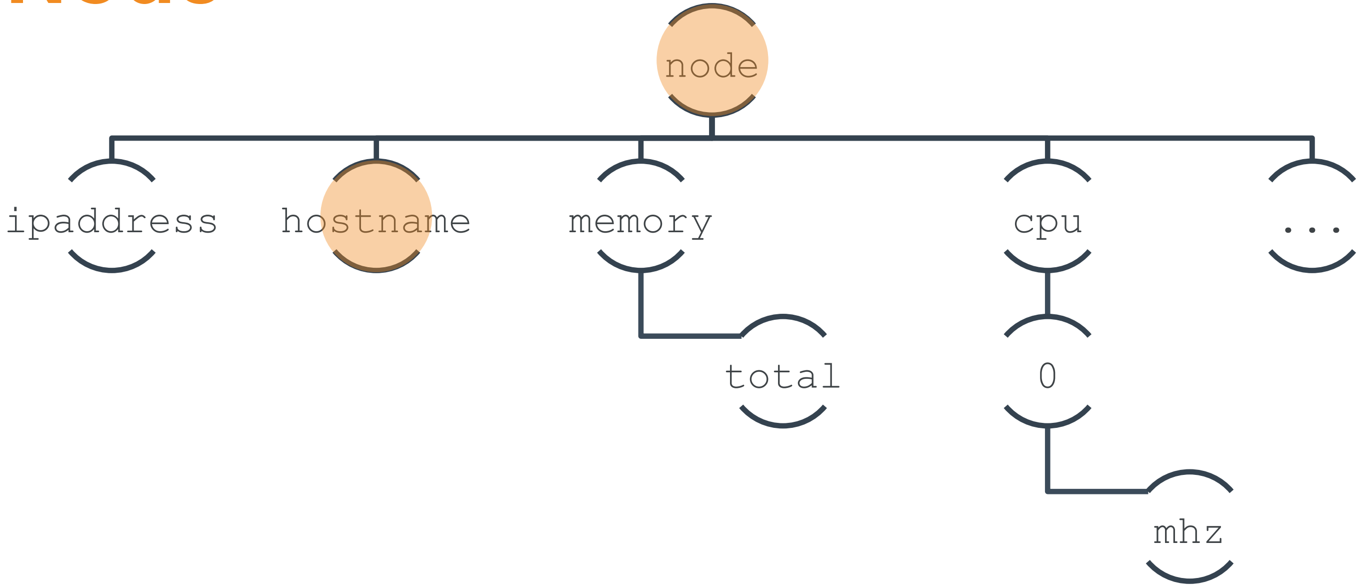


```
IPADDRESS: 104.236.192.102
```

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

©2016 Chef Software Inc.

The Node

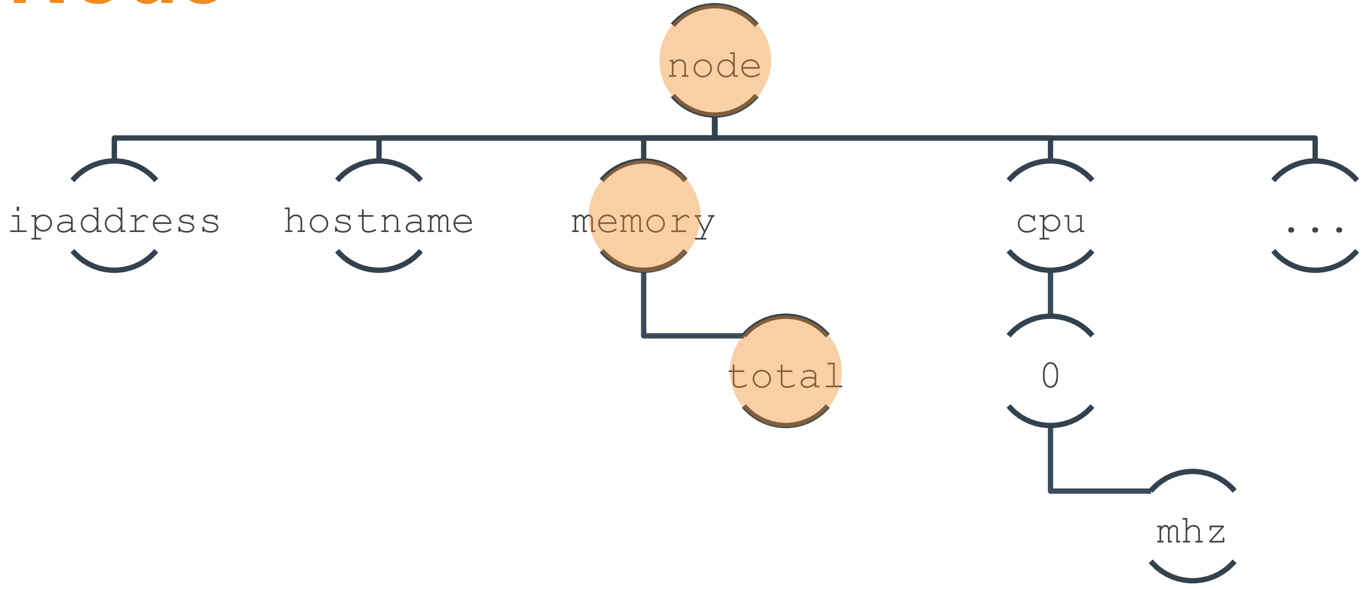


```
HOSTNAME: banana-stand
```

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

©2016 Chef Software Inc.

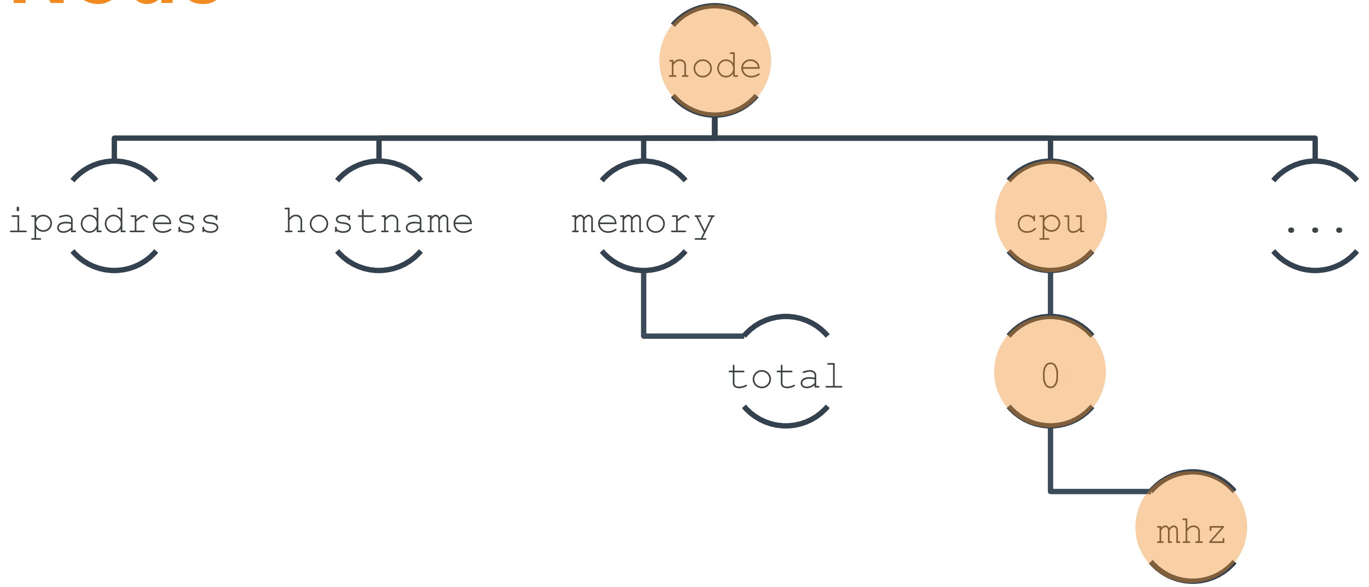
The Node



```
MEMORY: 502272kB
```

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

The Node



```
CPU: 2399.998 MHz
```

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

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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
```

CHEF

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

COMMIT

Lab: Commit Your Work

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

Cleaner Recipes

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

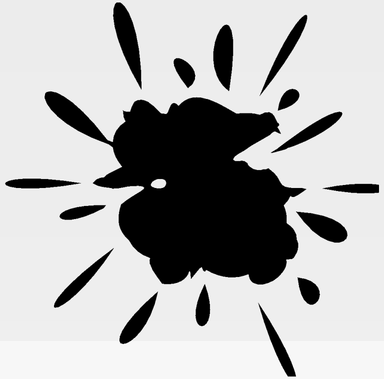
We added this content to the file resource in their respective recipes.

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>"
```

CONCEPT

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: \"
```

CONCEPT

Backslash

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

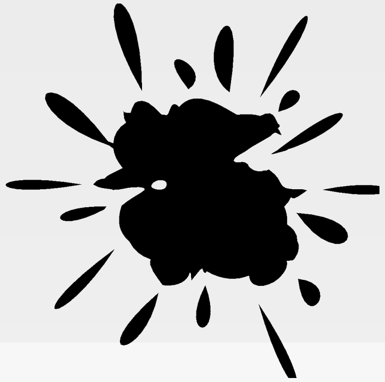
```
"Root Path: \\"
```

Unexpected Formatting

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

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

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

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

Copy Paste



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

©2016 Chef Software Inc.

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.

DOCS

Template

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

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

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF™

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

©2016 Chef Software Inc.

CHEF™

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
```

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)
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

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

<%=

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF™

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
```

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

©2016 Chef Software Inc.

CHEF™

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'
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF™

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?

©2016 Chef Software Inc.