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.

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/

©2016 Chef Software Inc.

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.

GL: Installing ChefDK

©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.

©2016 Chef Software Inc.

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.

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

©2016 Chef Software Inc.

CHEF

CONCEPT

Text Editors

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

Whatever editor you use should optimize for this workflow.

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF

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.

CHEF

Example: Package

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

The package named 'httpd' is installed.

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

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.

GL: Create and Open a Recipe File

```
$ nano hello.rb
```

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF

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!
```

©2015 Chef Software Inc.

CHEF

GL: What Does hello.txt Say?

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

```
Hello, world!
```

©2016 Chef Software Inc.

CHEF

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.

CHEF

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.

CHEF

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.

CHEF

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.

CHEF

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.

CHEF

CONCEPT

Resource Definition

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

?

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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

©2015 Chef Software Inc.

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.

©2015 Chef Software Inc.

CHEF

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.

Discussion

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

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

CHEF

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: 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).

©2016 Chef Software Inc.

CHEF

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.

CHEF

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.

CHEF

Cookbooks

Organizing Recipes

©2016 Chef Software Inc.

CHEF™

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.

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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)

©2016 Chef Software Inc.

GL: Create a Cookbooks Directory

```
$ mkdir cookbooks
```

[redacted]

CHEF

CONCEPT

What is 'chef'?

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

©2016 Chef Software Inc.

CHEF™

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

CONCEPT

README.md

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

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

©2016 Chef Software Inc.

CHEF

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.

CHEF

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

©2016 Chef Software Inc.

CHEF

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'
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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.
```

©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

©2016 Chef Software Inc.

CHEF

GL: Copy the Recipe into the Cookbook

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

©2016 Chef Software Inc.

CHEF

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
```

CHEF

Collaboration and Version Control

©2016 Chef Software Inc.

CHEF

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.

CHEF

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.

CHEF

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
```

©2016 Chef Software Inc.

CHEF

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)
```

©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.

CHEF

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.

CHEF

GL: Move into the Cookbook Directory

```
$ cd cookbooks/workstation
```

©2016 Chef Software Inc.

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 .
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

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.

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
```

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

CHEF

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.

CHEF™

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF

DISCUSSION

Major, Minor, or Patch?

What kind of changes did you make to the cookbook?

©2016 Chef Software Inc.

CHEF

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'
```

©2016 Chef Software Inc.

CHEF

COMMIT

GL: Commit Your Work

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

©2016 Chef Software Inc.

CHEF

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

GL: Return to the Home Directory

```
$ cd ~
```

©2016 Chef Software Inc.

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.

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
```

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

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]
```

©2016 Chef Software Inc.

Lab: Verify That the Website is Available

```
$ curl localhost
```

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

©2016 Chef Software Inc.

CHEF

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?

©2016 Chef Software Inc.

CHEF

chef-client

Applying Recipes from Multiple Cookbooks

©2016 Chef Software Inc.

CHEF™

chef-client

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

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

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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.

CHEF

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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.

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)
```

©2016 Chef Software Inc.

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: 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.

CHEF

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

©2016 Chef Software Inc.

CHEF

Demo: Including a Recipe

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

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

©2016 Chef Software Inc.

CHEF

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'
```

©2016 Chef Software Inc.

CHEF

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

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
```

©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.

COMMIT

GL: Commit Your Work

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

©2016 Chef Software Inc.

CHEF

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"
```

©2016 Chef Software Inc.

CHEF

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.

CHEF

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

©2016 Chef Software Inc.

Some Useful System Data

- IP Address
- hostname
- memory
- CPU - MHz

©2016 Chef Software Inc.

Discover the IP Address

```
$ hostname -I
```

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

©2016 Chef Software Inc.

CHEF

Discover the Host Name

```
$ hostname
```

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

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

CHEF

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)
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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?

©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!

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF

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.

CHEF

CONCEPT

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.

CHEF

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": {
```

©2016 Chef Software Inc.

Running Ohai to Show the IP Address

```
> ohai ipaddress
```

```
[
  "172.31.57.153"
]
```

©2016 Chef Software Inc.

CHEF

Running Ohai to Show the Hostname

```
> ohai hostname
```

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

©2016 Chef Software Inc.

CHEF

Running Ohai to Show the Memory

```
> ohai memory
```

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

©2016 Chef Software Inc.

Running Ohai to Show the Total Memory

```
> ohai memory/total
```

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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",
```

©2016 Chef Software Inc.

CHEF

Running Ohai to Show the First CPU Mhz

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

```
[
  "1795.673"
]
```

©2016 Chef Software Inc.

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

©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.

CHEF

The Node

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

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

©2016 Chef Software Inc.

CHEF

The Node

```
MEMORY: 502272kB
```

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

©2016 Chef Software Inc.

CHEF

The Node

```
CPU: 2399.998MHz
```

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

CONCEPT

String Interpolation

```
I have 4 apples
```

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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']}"
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

CHEF

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.

CHEF™

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...
```

©2016 Chef Software Inc.

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
```

©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 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

...
```

©2016 Chef Software Inc.

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.

CHEF

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.

CHEF

COMMIT

Lab: Commit Your Work

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

©2016 Chef Software Inc.

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 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.

CHEF

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?

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

Double Quotes Close Double Quotes

Double quoted strings are terminated by double quotes.

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

©2016 Chef Software Inc.

CHEF

CONCEPT

Backslash

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

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

©2016 Chef Software Inc.

CHEF

Backslash

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

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

©2016 Chef Software Inc.

CHEF

CONCEPT

Backslash

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

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

©2016 Chef Software Inc.

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!
```

©2016 Chef Software Inc.

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.

CHEF

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.

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

DOCS

Template

To use a template, two things must happen:

1. A template resource must be added to a recipe
2. An Embedded Ruby (ERB) template must be added to a cookbook

https://docs.chef.io/resource_template.html#using-templates

©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.

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)
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

CHEF

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.

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

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

CONCEPT

The Angry Squid

<%=

©2016 Chef Software Inc.

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']}
```

©2016 Chef Software Inc.

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'] %>
```

©2016 Chef Software Inc.

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.

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.

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
```

©2016 Chef Software Inc.

Changing from file to template Resource

~/cookbooks/workstation/recipes/setup.rb

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

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

©2016 Chef Software Inc.

CHEF

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.

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
```

©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.

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.

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

©2016 Chef Software Inc.

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**.

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.

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

Signing Up for a Hosted Chef Account

Steps

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

©2016 Chef Software Inc.

Signing Up for a Hosted Chef Account

Steps

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

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

CHEF

Download Your Cookbooks

Download cookbooks to your local machine

Section: 10

The Chef Server

2 / 7

49. Overview: Interfacing with a Chef Server 2:02

50. Why use a Chef Server? 6:13

About the Chef Server
Installing Your Own Chef Server

51. Exercise: Getting Started with Hosted Chef 7:46

Hosted Chef
About the Management Console
cookbooks.zip

52. Exercise: Uploading Cookbooks 6:41

knife cookbook upload

53. Exercise: Reconfigure Your Vagrant Environment 6:13

Vagrantfile.zip

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CONCEPT

knife

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

©2016 Chef Software Inc.

CHEF

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)
```

©2016 Chef Software Inc.

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)
```

©2016 Chef Software Inc.

GL: knife client list

```
$ knife client list
```

```
ORGNAME-validator
```

©2016 Chef Software Inc.

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)
```

©2016 Chef Software Inc.

GL: knife cookbook list

```
$ knife cookbook list
```

©2016 Chef Software Inc.

CHEF

GL: knife cookbook list

```
$ knife cookbook upload workstation
```

GL: knife cookbook list

```
$ knife cookbook upload apache
```

©2016 Chef Software Inc.

GL: knife cookbook list

```
$ knife cookbook list
```

```
apache       0.2.1
workstation  0.2.1
```

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF™

CONCEPT

The Node

©2016 Chef Software Inc.

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)
```

©2016 Chef Software Inc.

Run 'knife node list'

```
$ knife node list
```

CONCEPT

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

©2016 Chef Software Inc.

CHEF

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.
```

©2016 Chef Software Inc.

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                    ta                    ef
ec2-54-175-46-24.compute-1.am                    ta                    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:
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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...
```

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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]
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

Login to web1

```
$ vagrant ssh web1
```

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

©2016 Chef Software Inc.

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....
```

©2016 Chef Software Inc.

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>
```

©2016 Chef Software Inc.

CHEF

Return to your Workstation

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

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

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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?

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

Capture Node's Public Host Name and IP

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

```
web1:
  ipaddress: 192.168.10.43
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

CHEF

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'
```

©2016 Chef Software Inc.

CHEF

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'
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

Community Cookbooks

Find, Explore and View Chef Cookbooks

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

Load Balancer

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

Receives requests and relays them to other systems.

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

Community Cookbooks

Someone already wrote that cookbook?

Available through the community site called Supermarket/

https://supermarket.chef.io

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

Searching in the Supermarket

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

©2016 Chef Software Inc.

Searching in the Supermarket

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

©2016 Chef Software Inc.

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...

RSS

haproxy (24) Versions 2.0.0 Follow 104

Installs and configures haproxy

Berkshelf/Librarian Policyfile Knife

```
cookbook 'haproxy', '= 2.0.0'
```

README Dependencies Changelog Quality

haproxy Cookbook

build unknown cookbook v2.0.2

sous-chefs
Sous Chefs

DETAILS

View Source

View Issues

UPDATED NOVEMBER 10, 2016
Created on October 25, 2009

PLATFORMS

©2016 Chef Software Inc.

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",
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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/

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

Change to the cookbooks/myhaproxy Directory

```
$ cd cookbooks/myhaproxy
```

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

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)
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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)
  ....
```

©2016 Chef Software Inc.

Upload the Cookbook to the Chef Server

```
$ berks upload
```

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

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

CHEF

Lab: Bootstrap a New Node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N NAME --bootstrap-version=12
```

```
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
```

©2016 Chef Software Inc.

Verify the port and identity file

```
$ 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
```

©2016 Chef Software Inc.

Bootstrap Your Node

```
$ knife bootstrap localhost -p LB_PORT -x vagrant --sudo -i PATH_TO_KEY -N
load-balancer -r "recipe[myhaproxy]" --bootstrap-version=12
```

```
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...
```

©2016 Chef Software Inc.

Run 'knife node list' Again

```
$ knife node list
```

```
web1
load-balancer
```

©2016 Chef Software Inc.

CHEF

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:
```

©2016 Chef Software Inc.

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 ~]$
```

©2016 Chef Software Inc.

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>
```

©2016 Chef Software Inc.

Return to your Workstation

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

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

©2016 Chef Software Inc.

CHEF

Verify the Load Balancer

©2016 Chef Software Inc.

Managing Multiple Nodes

Create another web server and add it as a proxy member

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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...
```

©2016 Chef Software Inc.

Run 'knife node list' Again

```
$ knife node list
```

```
load-balancer
web1
web2
```

©2016 Chef Software Inc.

CHEF

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:
```

©2016 Chef Software Inc.

Login to web2

```
$ vagrant ssh web2
```

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

©2016 Chef Software Inc.

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>
```

©2016 Chef Software Inc.

Return to your Workstation

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

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF™

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.

©2016 Chef Software Inc.

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'
   ...
```

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

CHEF

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
```

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

CHEF

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'
```

©2016 Chef Software Inc.

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 .
```

©2016 Chef Software Inc.

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'
```

©2016 Chef Software Inc.

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 ~]$
```

©2016 Chef Software Inc.

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)
....
```

©2016 Chef Software Inc.

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>
```

©2016 Chef Software Inc.

CHEF

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>
```

©2016 Chef Software Inc.

Return to your Workstation

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

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

©2016 Chef Software Inc.

CHEF

Lab: Test the Load Balancer

localhost:9000

Hello, world!

ipaddress: 192.168.10.43

hostname: web1

localhost:9000

Hello, world!

ipaddress: 192.168.10.43

hostname: web1

localhost:9000

Hello, world!

ipaddress: 192.168.10.44

hostname: web2

©2016 Chef Software Inc.

CHEF

Lab: Test the Load Balancer

localhost:9000

Hello, world!

ipaddress: 192.168.10.44

hostname: web2

localhost:9000

Hello, world!

ipaddress: 192.168.10.43

hostname: web1

localhost:9000

Hello, world!

ipaddress: 192.168.10.44

hostname: web2

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

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?

©2016 Chef Software Inc.

CHEF

Roles

Giving Your Nodes a Role

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

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.

©2016 Chef Software Inc.

CHEF

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.

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

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)
```

©2016 Chef Software Inc.

Run 'knife role list'

```
$ knife role list
```

Create a Roles Directory

```
$ mkdir roles
```

©2016 Chef Software Inc.

Create the load_balancer.rb

~/chef-repo/roles/web.rb

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

©2016 Chef Software Inc.

CHEF

Upload it to the Chef Server

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

```
Updated Role web!
```

©2016 Chef Software Inc.

CHEF

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]
```

©2016 Chef Software Inc.

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)
```

©2016 Chef Software Inc.

Set the Web Role to web1

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

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

©2016 Chef Software Inc.

CHEF

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:
```

©2016 Chef Software Inc.

CHEF

Login to web1

```
$ vagrant ssh web1
```

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

©2016 Chef Software Inc.

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....
```

©2016 Chef Software Inc.

Return to your Workstation

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

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

©2016 Chef Software Inc.

CHEF

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:
```

©2016 Chef Software Inc.

CHEF

Set the Web Role to web2

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

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

©2016 Chef Software Inc.

CHEF

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:
```

©2016 Chef Software Inc.

CHEF

Login to web2

```
$ vagrant ssh web2
```

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

©2016 Chef Software Inc.

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....
```

©2016 Chef Software Inc.

Return to your Workstation

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

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

©2016 Chef Software Inc.

CHEF

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:
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

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

©2016 Chef Software Inc.

Lab: Create the load-balaner.rb File

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

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

©2016 Chef Software Inc.

CHEF

Lab: Upload the web.rb File

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

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

©2016 Chef Software Inc.

Lab: Verify the Role on the Chef Server

```
$ knife role list
```

```
load-balancer
web
```

©2016 Chef Software Inc.

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]
```

©2016 Chef Software Inc.

Lab: Set node1's Run List

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

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

©2016 Chef Software Inc.

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:
```

©2016 Chef Software Inc.

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 ~]$
```

©2016 Chef Software Inc.

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)
....
```

©2016 Chef Software Inc.

Return to your Workstation

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

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

©2016 Chef Software Inc.

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:       load-balancer
Recipes:     myhaproxy, myhaproxy::default, haproxy::default, haproxy::install_package
Platform:    centos 7.2
Tags:
```

©2016 Chef Software Inc.

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

©2016 Chef Software Inc.

CHEF

DISCUSSION

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?

©2016 Chef Software Inc.

CHEF

SSH Woes

Logging into multiple systems is a pain. We can use another knife tool to allow us to send commands to sets of our nodes.

©2016 Chef Software Inc.

CHEF

Using knife ssh

```
$ knife ssh --help
```

```
knife ssh QUERY COMMAND (options)
    -a, --attribute ATTR             The attribute to use for opening the connection -
default depends on the context
    -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 like 1000,1010 or 8889-9999 will try all given ports until one works.
    -k, --key KEY                    API Client Key
        --[no-]color                 Use colored output, defaults to false on Windows,
true otherwise
    -C, --concurrency NUM            The number of concurrent connections
    -c, --config CONFIG              The configuration file to use
        --defaults                   Accept default values for all questions
```

©2016 Chef Software Inc.

Run chef-client on all nodes

```
$ knife ssh "*:*" -x USERNAME -P PASSWORD "sudo chef-client"
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```

©2016 Chef Software Inc.

Verify the port and identity file

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

```
Host load-balancer
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/USER/chef-repo/.vagrant/machines/load-balancer/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

©2016 Chef Software Inc.

Run chef-client on a Vagrant instance

```
$ knife ssh localhost "sudo chef-client" --manual-list --ssh-port PORT
--ssh-user vagrant --identity-file /PATH/TO/KEY
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```

©2016 Chef Software Inc.

Run chef-client on Vagrant - short options

```
$ knife ssh localhost "sudo chef-client" -m -p PORT -x vagrant -i /PATH/TO/KEY
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```

©2016 Chef Software Inc.

Example - Run chef-client on Vagrant

```
$ knife ssh localhost "sudo chef-client" -m -p 2222 -x vagrant \
-i /Users/technotrainer/chef-repo/.vagrant/machines/load-balancer/virtualbox/private_key
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```

©2016 Chef Software Inc.

SSH Woes

When using a Cloud Provider like Amazon EC2, knife ssh is a great way to issue a remote command to a set of nodes, not just a single node.

©2016 Chef Software Inc.

CHEF

Example - Converge All Nodes

```
$ knife ssh "*:*" -x USER -P PWD "sudo chef-client"
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-166.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:
["workstation","apache"]
ec2-54-210-86-164.compute-1.amazonaws.com  resolving cookbooks for run list:
["workstation","apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["myhaproxy"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - workstation
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
....
```

©2016 Chef Software Inc.

Example - Converge All Webserver Nodes

```
$ knife ssh "role:web" -x USER -P PWD "sudo chef-client"
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
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: ["apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
ec2-54-175-46-24.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com    - apache
```

©2016 Chef Software Inc.

Example - Converge All Proxy Nodes

```
$ knife ssh "role:load-balancer" -x USER -P PWD "sudo chef-client"
```

```
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:
["myhaproxy"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list:
["myhaproxy"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
  - myhaproxy (0.2.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - ohai (4.2.3)
  - compat_resource (12.16.3)
  - mingw (1.2.4)
....
```

©2016 Chef Software Inc.

CHEF

SSH Woes

When using a Cloud Provider like Amazon EC2, knife ssh allows us to specify what node attribute to use as the “ipaddress”

This allows us to ssh in using the node[‘cloud’] attribute, which contains the public ipaddress.

©2016 Chef Software Inc.

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
```

©2016 Chef Software Inc.

Example - Converge All Nodes

```
$ knife ssh "*:*" -x USER -P PWD "sudo chef-client" -a cloud.public_ipv4
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-166.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:
["workstation","apache"]
ec2-54-210-86-164.compute-1.amazonaws.com  resolving cookbooks for run list:
["workstation","apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["myhaproxy"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - workstation
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
....
```

©2016 Chef Software Inc.

Example - Converge All Webserver Nodes

```
$ knife ssh "role:web" -x USER -P PWD "sudo chef-client" -a cloud.public_ipv4
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
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: ["apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
ec2-54-175-46-24.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com    - apache
```

©2016 Chef Software Inc.

Search

Update a Cookbook to Dynamically Use Nodes with the Web Role

©2016 Chef Software Inc.

CHEF™

Objectives

After completing this module, you should be able to

- Describe the query syntax used in search
- Build a search into your recipe code
- Create a Ruby Array and Ruby Hash
- Update the myhaproxy wrapper cookbook (for the load balancer) to dynamically use nodes with the web role

©2016 Chef Software Inc.

How Do We See Detail Across All Nodes?

`knife node show` only works with individual nodes so how do I see this info for multiple nodes?

©2016 Chef Software Inc.

CHEF

How Do We See Detail Across All Nodes?

`knife node show` only works with individual nodes so how do I see this info for multiple nodes?

Search allows you to retrieve information across multiple nodes.

©2016 Chef Software Inc.

CHEF

The Chef Server and Search

Chef Server maintains a searchable index of all nodes within our infrastructure.

Chef Server

Web Nodes	Proxy Nodes
Node 1	Node 2
Node 3	Node 5
Node 8	Node 6

©2016 Chef Software Inc.

CHEF

What is search?

Use search to query data indexed on the chef server

```
$ knife search INDEX SEARCH_QUERY
```

The search runs on the server and is invoked from within a recipe or using knife

INDEX can be 'client', 'environment', 'node', 'role', (or the name of a data bag)

SEARCH_QUERY is of the format "attribute:value"

Querying ***:*** returns everything

©2016 Chef Software Inc.

CHEF

GL: View Information for All Nodes

```
$ knife search node "*:*"
```

```
3 items found

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:

Node Name:   load-balancer
Environment: _default
FQDN:        load-balancer
IP:          10.0.2.15
Run List:    role[load-balancer]
….
```

©2016 Chef Software Inc.

CHEF

GL: Narrow the Search

```
$ knife search node "*:*" -a ipaddress
```

```
2 items found

web1:
 ipaddress : 192.168.10.43

load-balancer:
 ipaddress : 10.0.2.15

web2:
 ipaddress : 192.168.10.44
```

©2016 Chef Software Inc.

CHEF

CONCEPT

Search

So far we have seen how Chef is able to manage the policy of the nodes.

We have two web servers and one load balancer.

©2016 Chef Software Inc.

CHEF

CONCEPT

Search

To add new servers as load balancer members, we would need to bootstrap a new web server and then update our load balancer's myhaproxy cookbook recipe.

That seems inefficient to have to update a cookbook recipe.

©2016 Chef Software Inc.

CHEF

The Chef Server and Search

Chef Server maintains a representation of all the nodes within our infrastructure that can be searched on.

Search is a service discovery tool that allows us to query the Chef Server.

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

https://docs.chef.io/chef_search.html#search-indexes

©2016 Chef Software Inc.

CHEF

The Chef Server and Search

We can ask the Chef Server to return all the nodes or a subset of nodes based on the query syntax that we provide it through `knife search` or within our recipes through `search`.

©2016 Chef Software Inc.