Chef Essentials

Introduction

CHEF

Introduce Yourselves

Name

Current job role

Previous job roles/background

Experience with Chef and/or config management

Favorite Text Editor

CHEF

Expectations

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

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

Course Objectives

After completing this course, you should be able to:

- Use Chef Resources to define the state of your system
- Write and use Chef recipes and cookbooks
- Automate testing of cookbooks
- Create Organizations
- Manage multiple nodes with Chef Server - Bootstrapping nodes
- Assign Roles to nodes
- Deploy nodes to environments

Chef

Chef is a large set of tools that are able to be used on multiple platforms and in numerous configurations.

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

A great way to learn Chef is to use Chef

Chef Fundamentals

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

Break It: If everything works the first time go back and make some changes. Break it!

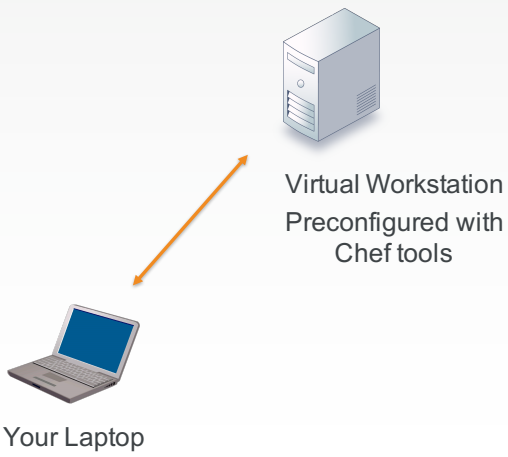
Chef Lab System Architecture

In this course you will use two different architectures:

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

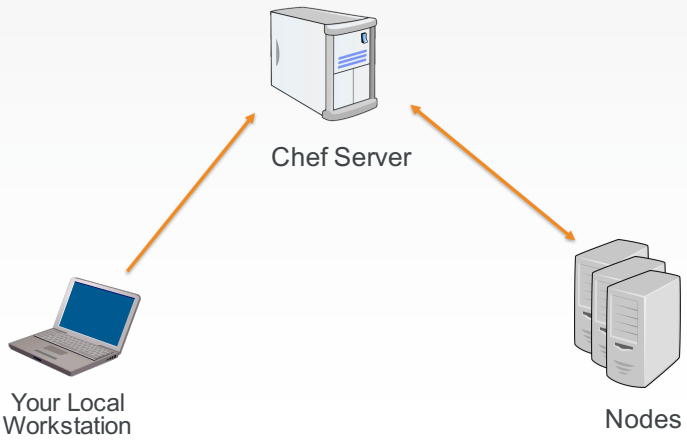
Chef Lab System Architecture

Architecture 1



Chef Lab System Architecture

Architecture 2



Lab: SSH Into the Remote Workstation

```
$ ssh ADDRESS -l chef
```

workstation

Getting a Workstation

The chef user has been granted password-less sudoers access

The following software is installed on the remote workstation:

- Chef DK
- Docker
- kitchen-docker gem

Overview of Chef

CHEF

Lesson Objectives

After completing the lesson, you will be able to

- Describe how Chef thinks about Infrastructure Automation

CHEF

Typical Sys Admin Journey

From a vanilla image…

```
ssh into machine
$ yum install httpd
$ yum install wget
$ yum install unzip
$ wget https://somewhere/master.zip
$ unzip master.zip
$ cd myapp
$ sudo mv html /var/www/
$ sudo su root
$ python myappinstall.py
$ apachectl graceful
```

Log into machine

CHEF

Typical Sys Admin Journey

From a vanilla image…

```
ssh into machine
$ yum install httpd
$ yum install wget
$ yum install unzip
$ wget https://somewhere/master.zip
$ unzip master.zip
$ cd myapp
$ sudo mv html /var/www/
$ sudo su root
$ python myappinstall.py
$ apachectl graceful
```

Install a few packages

CHEF

Typical Sys Admin Journey

From a vanilla image…

```
ssh into machine
$ yum install httpd
$ yum install wget
$ yum install unzip
$ wget https://somewhere/master.zip
$ unzip master.zip
$ cd myapp
$ sudo mv html /var/www/
$ sudo su root
$ python myappinstall.py
$ apachectl graceful
```

Pull in some content

CHEF

Typical Sys Admin Journey

From a vanilla image…

```
ssh into machine
$ yum install httpd
$ yum install wget
$ yum install unzip
$ wget https://somewhere/master.zip
$ unzip master.zip
$ cd myapp
$ sudo mv html /var/www/
$ sudo su root
$ python myappinstall.py
$ apachectl graceful
```

Manipulate directories & content

CHEF

Typical Sys Admin Journey

From a vanilla image…

```
ssh into machine
$ yum install httpd
$ yum install wget
$ yum install unzip
$ wget https://somewhere/master.zip
$ unzip master.zip
$ cd myapp
$ sudo mv html /var/www/
$ sudo su root
$ python myappinstall.py
$ apachectl graceful
```

Re/start services

CHEF

Typical Sys Admin Journey

From a vanilla image…

```
ssh into machine
$ yum install httpd
$ yum install wget
$ yum install unzip
$ wget https://somewhere/master.zip
$ unzip master.zip
$ cd myapp
$ sudo mv html /var/www/
$ sudo su root
$ python myappinstall.py
$ apachectl graceful
```

All commands are manual

All have different syntaxes

They're platform specific (RHEL, Debian, Windows, …)

CHEF

Typical Sys Admin Journey

Write some scripts (setup.sh, fixit.sh, etc.)

Typical Sys Admin Journey

Write some scripts (setup.sh, fixit.sh, etc.)

Store notes in ~/server.txt

Typical Sys Admin Journey

Write some scripts (setup.sh, fixit.sh, etc.)

Store notes in ~/server.txt

Move notes to the wiki

Typical Sys Admin Journey

Write some scripts (setup.sh, fixit.sh, etc.)

Store notes in ~/server.txt

Move notes to the wiki

Version control

Typical Sys Admin Journey

Write some scripts (setup.sh, fixit.sh, etc.)

Store notes in ~/server.txt

Move notes to the wiki

Version control

- setup.sh.BAK
- fixit.sh.OLD

CHEF

Typical Sys Admin Journey

Write some scripts (setup.sh, fixit.sh, etc.)

Store notes in ~/server.txt

Move notes to the wiki

Version control

- setup.sh.BAK
- fixit.sh.OLD

Golden images and snapshots

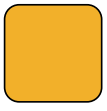
CHEF

Systems grow organically

Application

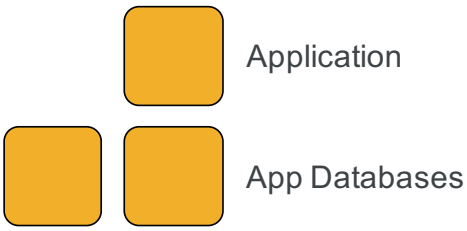
Systems grow organically

Application

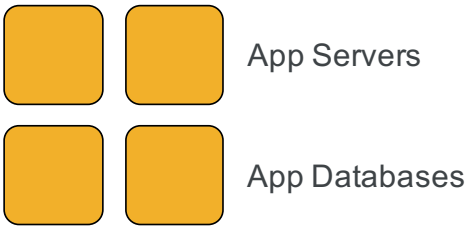


Application Database

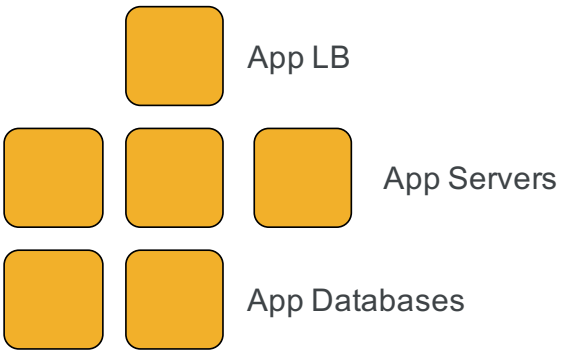
Systems grow organically



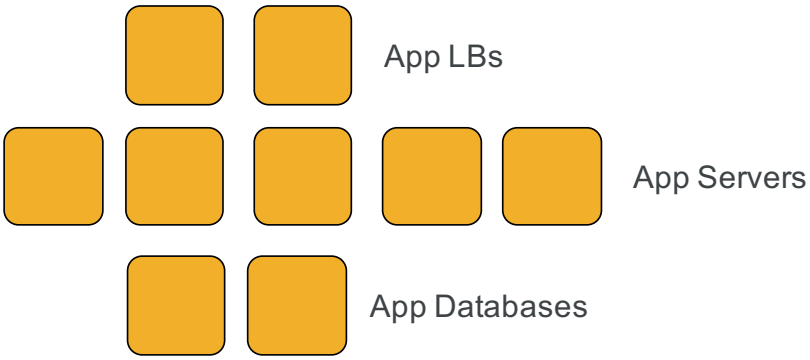
Systems grow organically



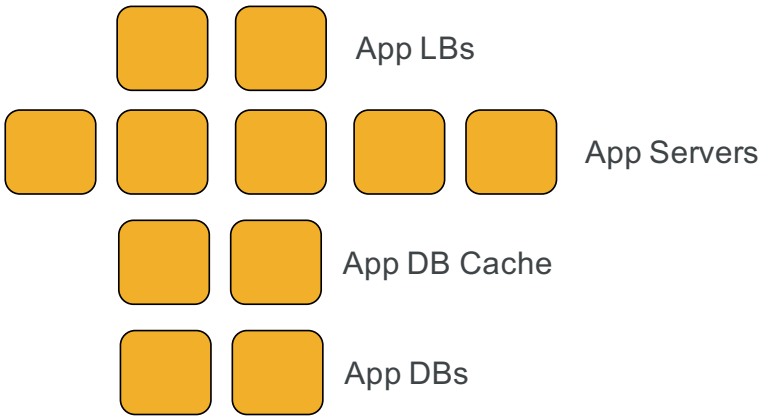
Systems grow organically



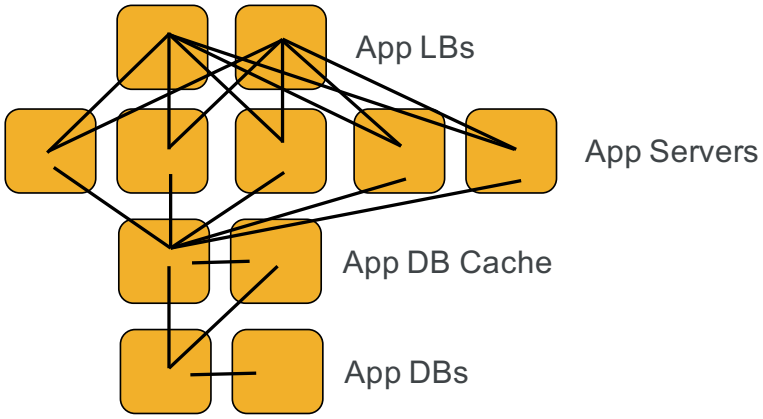
Systems grow organically



Systems grow organically



Systems grow organically



Chef Solves This Problem

The Infrastructure Code

Chef provides consistent DSL that is platform agnostic to manage any configuration component

- packages
- files
- users
- …

Complex implementation code abstracted out

Treat Infrastructure like any code base

Infrastructure configuration files are stored in version control, e.g. GitHub

Infrastructure becomes as testable & repeatable as the application code you're delivering

Items of Manipulation (Resources)

- Networking
- Files
- Directories
- Symlinks
- Mounts
- Registry Keys
- Powershell Scripts
- Users
- Groups
- Packages
- Services
- Filesystems

Items of Manipulation (Resources)

- Networking
- Files
- Directories
- Symlinks
- Mounts
- Registry Keys
- Powershell Scripts
- Users
- Groups
- Packages
- Services
- Filesystems

We'll start by looking at what a resource is...

Chef Resources

Chef's Fundamental Building Blocks

CHEF

Objectives

After completing this module, you should be able to:

- Use Chef to install packages on your virtual workstation
- Use the chef-client command
- Create a basic Chef recipe file
- Define Chef Resources

CHEF

CONCEPT

Resources

A resource is a statement of configuration policy.

Describes the desired state of an element of your infrastructure

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

CONCEPT

Resource Definition

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

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

CONCEPT

Resource Definition

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

The TYPE named NAME should be ACTION'd with ATTRIBUTES

CONCEPT

Resource Definition

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

The TYPE named NAME should be ACTION'd with ATTRIBUTES

CONCEPT

Resource Definition

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

The TYPE named NAME should be ACTION'd with ATTRIBUTES

Example: 'package' resource

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

The package named 'httpd' should be installed.

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

Example: ‘service’ resource

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

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

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

Example: ‘file’ resource

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

The file name '/etc/motd' should exist with content 'This company is the property ...'

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

Example: ‘file’ resource

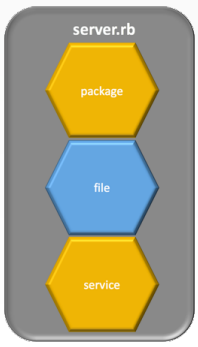
```
file '/etc/php.ini.default' do
  action :delete
end
```

The file name '/etc/php.ini.default' should be deleted if it exists

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

CONCEPT

Recipes



Resources are included in recipes.

Recipes are Ruby files (.rb extension) in which you include the code that defines your infrastructure.

Lab: Workstation Setup

Objective:

- Create a recipe file named "setup.rb" that defines the policy:
  - The package named 'tree' is installed.
  - The file named '/etc/motd' is created with the content 'Property of ...'.
- Use chef-client (local-mode) to apply the recipe file named "setup.rb"

CHEF

Lab: Is the node already in the desired state?

```
$ which tree
```

```
/usr/bin/which: no tree in
(/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/chef/bin)
```

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

```
```


CHEF

Lab: Use Your Editor to Open the Recipe

```
$ [EDITOR NAME] setup.rb
```

Use the editor of your choice to create a file called setup.rb – nano, vi, vim, emacs

Lab: Final 'setup.rb' Recipe

~/setup.rb

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

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  action :create
end
```

SAVE FILE

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 execute the resources in the recipe to bring the node into the expected state.

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

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. More on this later in the course.

We'll override that behavior to have it work in a local mode – i.e. feed it the recipes to run from local machine

'local mode' is often referred to as 'chef zero'

Lab: 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
```


Lab: Is the node in the desired state?

```
$ which tree
```

```
/usr/bin/tree
```

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

```
Property of [INSERT DESIRED TEXT]
```

Lab: Setting Resource Attributes

We'll use the docs to discover the file resource's default values for:

- `mode`,
- `owner`
- `group`

Update the `file` resource in "setup.rb" to do the following:

The file named '/etc/motd' should be created with the content '**Property of [INSERT DESIRED TEXT]**', mode '**0644**', owner is '**chef**', and group is '**chef**'.

More information at https://docs.chef.io/resources.html

Lab: Final 'setup.rb' Recipe

~/setup.rb

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

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  action :create
  mode '0644'
  owner 'chef'
  group 'chef'
end
```

SAVE FILE

Lab: Rerun chef-client

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

```
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create
    - change owner from 'root' to 'chef'
    - change group from 'root' to 'chef'

Running handlers:
Running handlers complete
Chef Client finished, 1/2 resources updated in 08 seconds
```


Test and Repair

1. What would happen if you ran the chef-client command again?
2. What would happen if the package were to become uninstalled?
3. What would happen if the /etc/motd file was manually edited or owner & group changed?

Test and Repair

1. What would happen if you ran the chef-client command again?
2. What would happen if the package were to become uninstalled?
3. What would happen if the /etc/motd file was manually edited or owner & group changed?
4. Try it – set owner & group to ‘root’

Lab: Final ‘setup.rb’ Recipe

~/setup.rb

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

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  action :create
  mode '0644'
  owner 'root'
  group 'root'
end
```

SAVE FILE

Lab: Rerun chef-client

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

```
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create
    - change owner from 'root' to 'root'
    - change group from 'root' to 'root'

Running handlers:
Running handlers complete
Chef Client finished, 1/2 resources updated in 08 seconds
```


CHEF

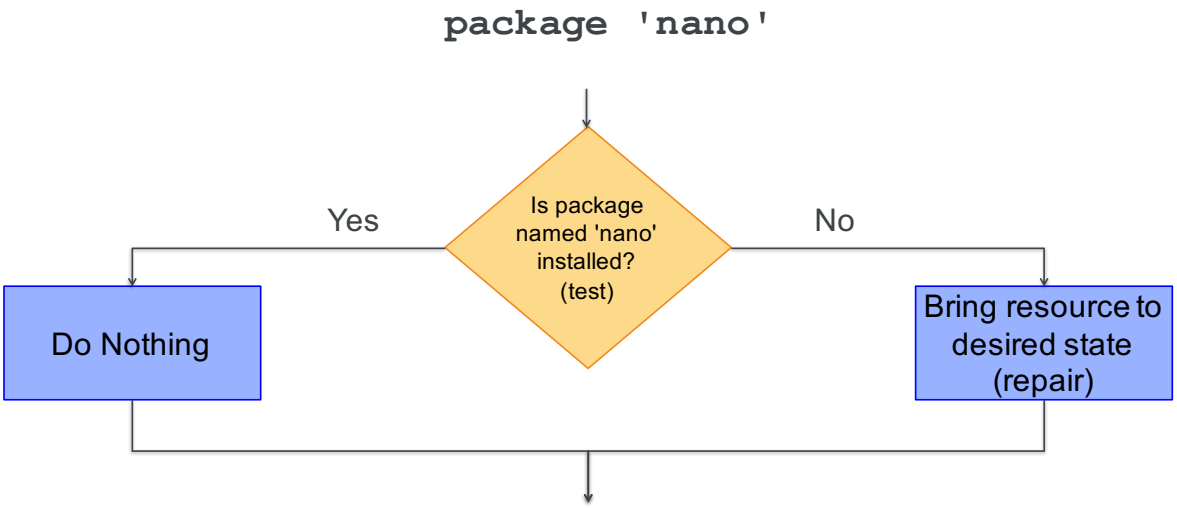
Test and Repair

`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.

CHEF

Test and Repair



Remove default actions

~/setup.rb

```
package 'tree'

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  mode '0644'
  owner 'root'
  group 'root'
end
```

Default actions can be omitted

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?

Q&A

What questions can we answer for you?

- chef-client
- Resources
- Resource - default actions and default attributes
- Test and Repair

Cookbooks

Organizing Recipes

CHEF

Objectives

After completing this module, you should be able to:

- Set up a repo as a working folder
- Modify a recipe
- Discuss version control
- Generate a Chef cookbook
- Define a Chef recipe that sets up a web server

CHEF

Repository

A repository is the working folder on your Workstation where you keep all your Chef files.

CHEF

Lab: Create a Repository

Lets create our workspace

Objective:

- Use chef to generate a repo that will act as our workspace on the workstation

CHEF

CONCEPT

What is 'chef' command?

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

CHEF

What can 'chef' do?

```
$ chef --help
```

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

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


What Can 'chef generate' Do?

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

```
Usage: chef generate GENERATOR [options]

Available generators:
  app          Generate an application repo
  cookbook     Generate a single cookbook
  recipe       Generate a new recipe
  attribute    Generate an attributes file
  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
```

What Can 'chef generate repo' Do?

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

```
Usage: chef generate repo NAME [options]
    -C, --copyright COPYRIGHT        Name of the copyright holder - defaults to 'The Authors'
    -m, --email EMAIL                Email address of the author - defaults to
'you@example.com'
    -a, --generator-arg KEY=VALUE    Use to set arbitrary attribute KEY to VALUE in the
code_generator cookbook
    -I, --license LICENSE            all_rights, apache2, mit, gplv2, gplv3 - defaults to
all_rights
    -P, --policy                     Use policyfiles instead of Berkshelf
    -p, --policy-only                Create a repository for policy only, not cookbooks
    -r, --roles                      Create roles and environments directories instead of
using policyfiles
    -g GENERATOR_COOKBOOK_PATH,      Use GENERATOR_COOKBOOK_PATH for the code_generator
cookbook
        --generator-cookbook
```


Lab: Create your repo

```
$ chef generate repo chef-repo
```

```
...
    - create new file /home/chef/chef-repo/chef-repo/cookbooks/README.md
    - update content in file /home/chef/chef-repo/chef-
repo/cookbooks/README.md from none to 86e9ef
    (diff output suppressed by config)
  * execute[initialize-git] action run
    - execute git init .
  * template[/home/chef/chef-repo/chef-repo/.gitignore] action
create_if_missing
    - create new file /home/chef/chef-repo/chef-repo/.gitignore
    - update content in file /home/chef/chef-repo/chef-repo/.gitignore from
none to 3523c4
    (diff output suppressed by config)
```

Lab: Navigate to the chef-repo directory

```
$ cd chef-repo
```

Navigate to the chef-repo directory. You will perform most the tasks in this class from this directory

CHEF

Lab: View repo structure

```
$ tree
├── chefignore
├── cookbooks
│   ├── example
│   │   ├── attributes
│   │   │   └── default.rb
│   │   ├── metadata.rb
│   │   ├── README.md
│   │   └── recipes
│   │       └── default.rb
│   └── README.md
├── data_bags
│   ├── example
...
```


CHEF

Lab: View repo structure

```
$ tree
├── chefignore
├── cookbooks
│   ├── example
│   │   ├── attributes
│   │   │   └── default.rb
│   │   ├── metadata.rb
│   │   ├── README.md
│   │   └── recipes
│   │       └── default.rb
│   └── README.md
├── data_bags
│   ├── example
...
```

Cookbooks reside in the 'cookbooks' directory

CHEF

Cookbooks

A cookbook is a container for recipes, and supporting files

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.

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

CHEF

Lab: Create a Cookbook

How are we going to manage this 'setup.rb' file? Does it need a README?

Objective:

- Use chef to generate a cookbook to store our setup recipe.

CHEF

What Can 'chef generate' Do?

```
$ chef generate --help
UsaLab: 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
```


CHEF

Lab: Let's Create a Cookbook

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

```
...
- create new file /home/chef/chef-
repo/cookbooks/workstation/recipes/default.rb
    - update content in file /home/chef/chef-
repo/cookbooks/workstation/recipes/default.rb from none to 0d310d
    (diff output suppressed by config)
  * cookbook_file[/home/chef/chef-repo/cookbooks/workstation/.gitignore]
action create
    - create new file /home/chef/chef-repo/cookbooks/workstation/.gitignore
    - update content in file /home/chef/chef-
repo/cookbooks/workstation/.gitignore from none to dd37b2
    (diff output suppressed by config)
```


The Cookbook Has a README

```
$ tree cookbooks/workstation
```

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

CONCEPT

README.md

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

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

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
```

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

CHEF

Lab: 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'
```


CHEF

Lab: 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
```


Lab: The Cookbook Has a Default Recipe

```
$ cat cookbooks/workstation/recipes/default.rb
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.
```

Lab: Copy the Recipe into the Cookbook

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


CONCEPT

chef-client

Up until now we have ran chef-client on individual recipe .rb files at the command line.

But what if we need to run multiple recipes – we cannot add multiple paths/to/.rb/files

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

CONCEPT

-r "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]**

To use this format the cookbooks must be under a directory called 'cookbooks'

Using 'chef-client' to Apply Recipes

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

Use chef-client with '-r' flag to specify the recipe run as 'cookbook::recipe' instead of path/to/recipe.rb

Lab: 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

Using 'chef-client' to Locally Apply Recipes

```
$ sudo chef-client --local-mode \
  -r "recipe[cookbook1::default],recipe[cookbook2::default]"
```

Applying multiple recipes:

- Run the 'default' recipe from the cookbook 'cookbook1'
- Then if it completes successfully
- Run the 'default' recipe from the cookbook 'cookbook2'

Lab: Return Home First

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


CHEF

Lab: Apply the 'workstation::setup' Recipe

```
$ sudo chef-client -z -r "recipe[workstation::setup]"
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-04-07T14:34:55+00:00] WARN: Node ip-172-31-11-163.ec2.internal has an
empty run list.
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
```


CHEF

Lab: Apply the 'workstation::setup' Recipe

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

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-04-07T14:34:55+00:00] WARN: Node ip-172-31-11-163.ec2.internal has an
empty run list.
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
```

Note the flag '-z' can be used instead of '--local-mode'

CHEF

Lab: Apply two recipes

```
$ sudo chef-client -z -r
"recipe[workstation::default],recipe[workstation::setup]"
```

```
Starting Chef Client, version 12.7.2
resolving cookbooks for run list: ["workstation::default", "workstation::setup"]
Synchronizing Cookbooks:
  - workstation (0.0.0)
Compiling Cookbooks...
Converging 2 resources
Recipe: workstation::setup
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/2 resources updated in 08 seconds
```


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.

Using 'chef-client' to apply default recipe

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

Is equivalent to

```
$ sudo chef-client --local-mode -r "recipe[cookbook::default]"
```

When the recipe is specified as "recipe[cookbook]", chef-client assumes the default recipe, default.rb

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

Demo: Including a Recipe

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

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

Lab: The Default Recipe Includes the Setup Recipe

cookbooks/workstation/recipes/default.rb

```
#
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

include_recipe 'workstation::setup'
```


Lab: Apply the Cookbook's Default Recipe

```
$ sudo chef-client -zr "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 0 resources

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

Note the flags '-z –r' can be combined into '-zr'!

Discussion

Why would you want to apply more than one recipe at a time?

What file would you read first when examining a cookbook?

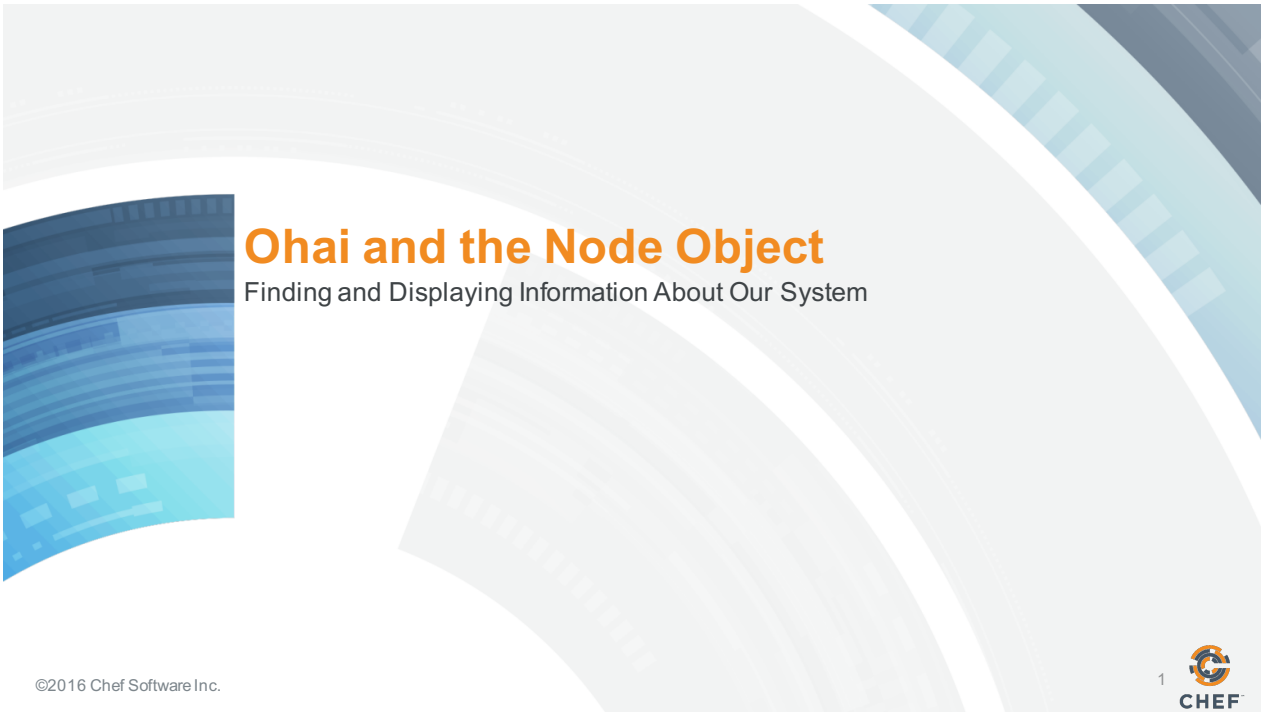
What are the benefits and drawbacks of using "include_recipe" within a recipe?

Do default values make it easier or harder to learn?

Q&A

What questions can we help you answer?

- chef-client
- local mode
- cookbooks
- run list
- include_recipe



Ohai and the Node Object

Finding and Displaying Information About Our System

Objectives

After completing this module, you should be able to

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

CHEF

System Data in MOTD file

- Lets say you needed to update the MOTD file contents, in the "workstation" cookbook, to include node details
  - ❑ IP Address
  - ❑ hostname
  - ❑ memory
  - ❑ CPU - MHz

CHEF

The Manual Way - Discover the IP Address

```
$ ifconfig
```

```
docker0   Link encap:Ethernet  HWaddr 56:84:7A:FE:97:99
          inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::5484:7aff:fefe:9799/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:25870 errors:0 dropped:0 overruns:0 frame:0
          TX packets:128971 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1459392 (1.3 MiB)  TX bytes:190819384 (181.9 MiB)

eth0      Link encap:Ethernet  HWaddr 0A:4D:03:F7:91:D7
          inet addr:172.31.8.68  Bcast:172.31.15.255  Mask:255.255.240.0
          inet6 addr: fe80::84d:3ff:fef7:91d7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```


The Manual Way - Discover the Host Name

```
$ hostname
```

```
banana-stand
```

The Manual Way - Discover the Memory

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


CHEF

The Manual Way - Discover the CPU MHz

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


CHEF

The Manual Way - Add info to the recipe

cookbooks/workstation/recipes/setup.rb

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

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


The Manual Way - Apply workstation Cookbook

```
$ sudo chef-client -zr "recipe[workstation]"
```

```
resolving cookbooks for run list: ["workstation"]
Synchronizing Cookbooks:
  - workstation
Compiling Cookbooks...
...
```

The Manual Way - Verify that the /etc/motd Has Been Updated

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

```
Property of ...

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


DISCUSSION

Capturing System Data

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

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

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

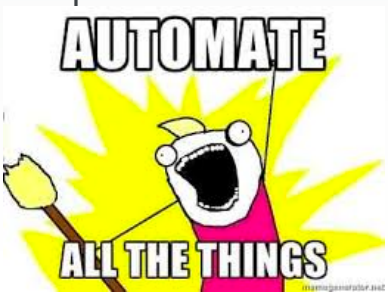
Hard Coded Values

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

DISCUSSION

Data In Real Time

We need to capture this data in real-time!



CONCEPT

Ohai!

Ohai is a system profiling tool that captures all this data (and much more!)

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

Ohai!

```
$ ohai
```

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

CONCEPT

ohai + chef-client

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

The data is presented in JSON (JavaScript Object Notation)

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

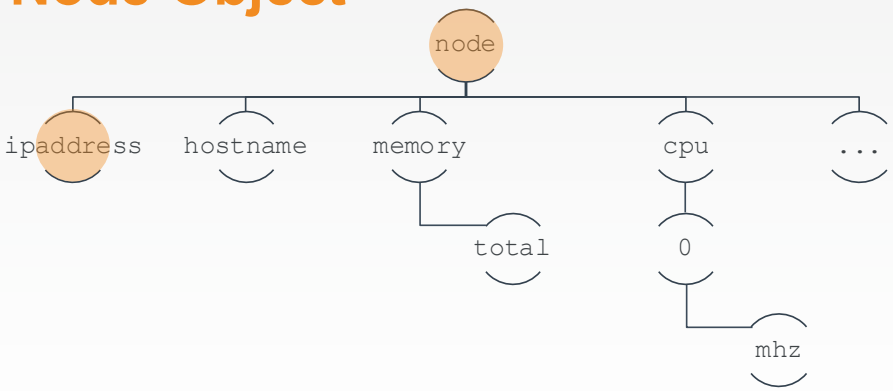
CONCEPT

The Node Object

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

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

The Node Object

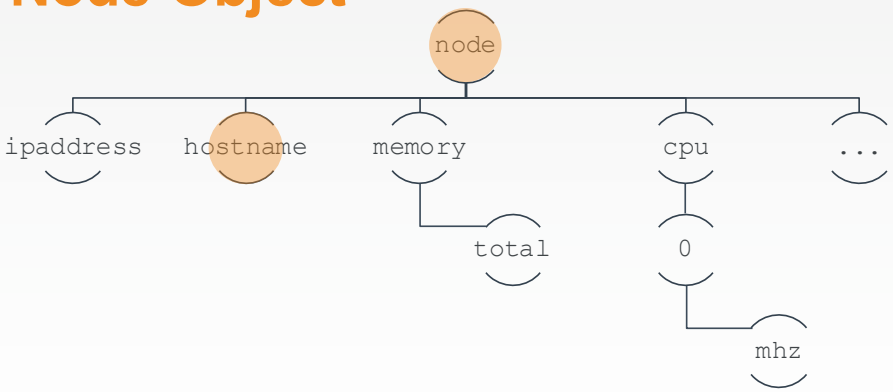



```
IPADDRESS: 104.236.192.102
```

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


CHEF

The Node Object

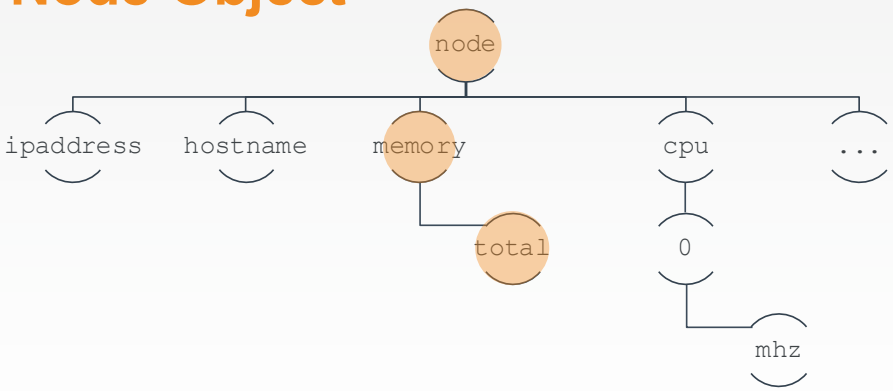



```
HOSTNAME: banana-stand
```

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


CHEF

The Node Object

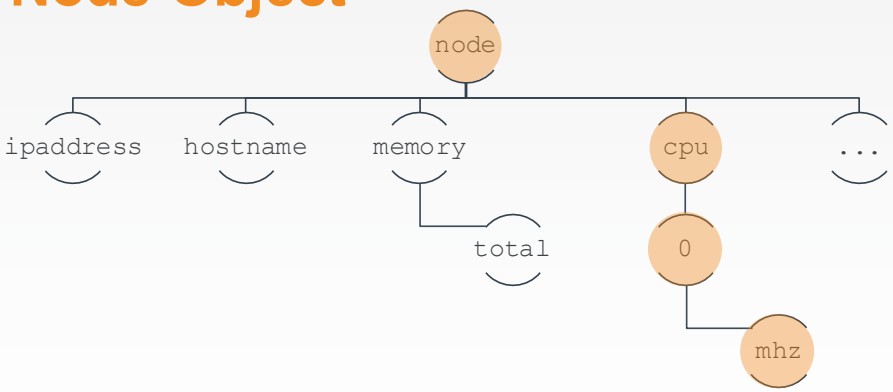



```
MEMORY: 502272kB
```

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


CHEF

The Node Object




```
CPU: 2399.998MHz
```

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


CHEF

CONCEPT

String Interpolation

```
I have 4 apples
```

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

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

CHEF

CONCEPT

String Interpolation

```
I have 4 apples
```

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


CHEF

Lab: 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
```


Lab: Return Home and Apply workstation Cookbook

```
$ cd chef-repo
$ sudo chef-client -zr "recipe[workstation]"
```

```
...
    +
    +  IPADDRESS: 172.31.8.68
    +  HOSTNAME : ip-172-31-8-68
    +  MEMORY   : 604308kB
    +  CPU      : 1795.672

Running handlers:
Running handlers complete
Chef Client finished, 1/2 resources updated in 08 seconds
```

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

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

```
Property of ...

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


Changes Mean a New Version

Let's bump the version number

Objective:

- ❑ Update the version of the "workstation" cookbook

CONCEPT

Semantic Versions

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

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

http://semver.org

Lab: 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'
```

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?

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

DISCUSSION

Q&A

What questions can we help you answer?

- Ohai
- Node Object
- Node Attributes
- String Interpolation
- Semantic Versions

Lab

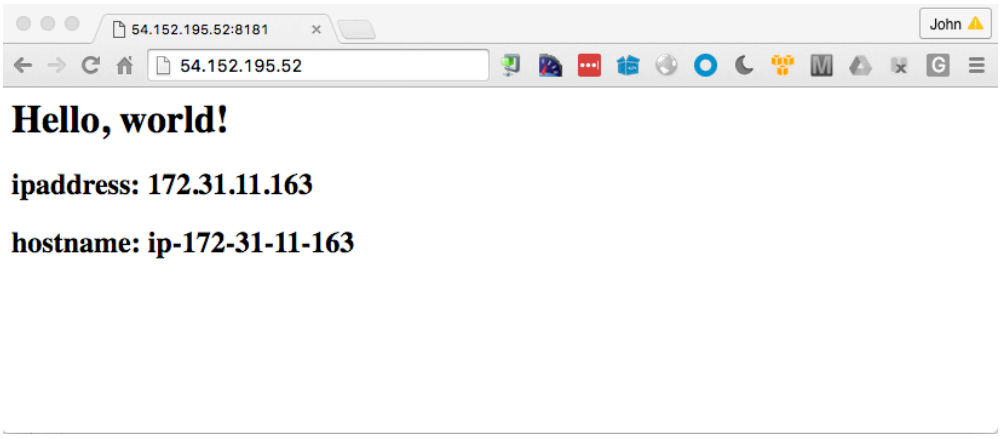
Create a Web Server

CHEF

Lab: Setting up a Web Server

- Use **chef generate** to create a cookbook named "apache"
- Write 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
- Include the following node details in index.html:
  - Node's ipaddress
  - Node's hostname
- Include **server.rb** in the **apache default.rb**
- Update the version of the **apache cookbook** to 0.2.0
- Run **chef-client** to locally apply the apache cookbook's default recipe. Be sure to include the default.rb in the run-list
- Verify the site is available by running `curl localhost` and browsing to your Workstation IP

So this is what we want to see



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
    -
```


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>
<h2>ipaddress: #{node['ipaddress']}</h2>
<h2>hostname: #{node['hostname']}</h2>
"
end

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


CHEF

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'
```


Lab: Update the Cookbook Version

cookbooks/apache/metadata.rb

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

Lab: Run chef-client to Apply the Apache Cookbook

```
$ sudo chef-client -zr "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
```


CHEF

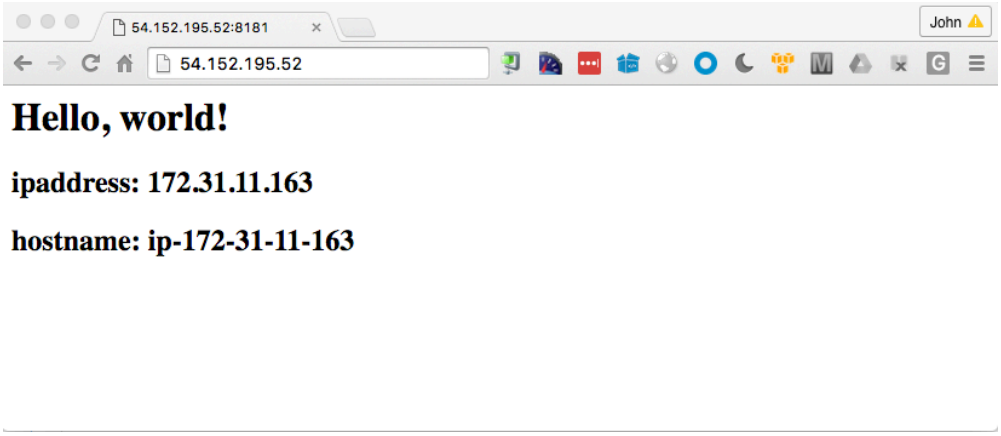
Lab: Verify That the Website is Available

```
$ curl localhost
```

```
<h1>Hello, world!</h1>
<h2>ipaddress: 172.31.11.163</h2>
<h2>hostname: ip-176-31-11-163</h2>
```


CHEF

Lab: Verify That the Website is Available



CHEF™

Templates - Desired State vs Data

Extracting the Content for Clarity

CHEF

Objectives

After completing this module, you should be able to

- Explain when to use a template resource
- Create a template file
- Use ERB tags to display node data in a template
- Define a template resource

Cleaner Recipes

In the last section we installed apache and configured its homepage to display information about our node.

We added this content directly to the file resource

Apache Recipe

cookbooks/apache/recipes/server.rb

```
package 'httpd'

file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>
<h2>ipaddress: #{node['ipaddress']}</h2>
<h2>hostname: #{node['hostname']}</h2>
"
end

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

Double Quotes Close Double Quotes

Double quoted strings are terminated by double quotes.

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


CONCEPT

Backslash

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

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

CONCEPT

Backslash

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

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

CONCEPT

Backslash

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

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

Unexpected Formatting

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

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

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

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


Copy Paste

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

CONCEPT

Separate Logic from Data

We need the ability to store the data in another file, which is in the native format of the file we are writing out but that still allows us to insert ruby code...

...specifically, the node attributes we have defined.

DOCS

'Template' resource

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

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

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

Demo: Template File's Source Matches Up

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

0 directories, 1 file
```

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


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

Lab: Clean workstation::setup recipe

Remove data from the recipe.

Objective:

- ❑ Use chef generate to create a template named "motd.erb".
- ❑ Copy the source attribute from the file named '/etc/motd' into the template file "motd.erb"
- ❑ Remove a resource: The file named '/etc/motd'
- ❑ Add a resource: The template named '/etc/motd' is created with the source 'motd.erb'
- ❑ Update the "workstation" cookbook's version for this patch

CHEF

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
```


CHEF

What Can chef generate template Do?

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

```
Usage: 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
```


Use chef to generate a template file

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

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

Lets Look at the Template File

```
$ tree cookbooks/workstation/templates
```

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

1 directory, 1 file
```


CHEF

https://docs.chef.io/templates.html#variables

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.

CHEF

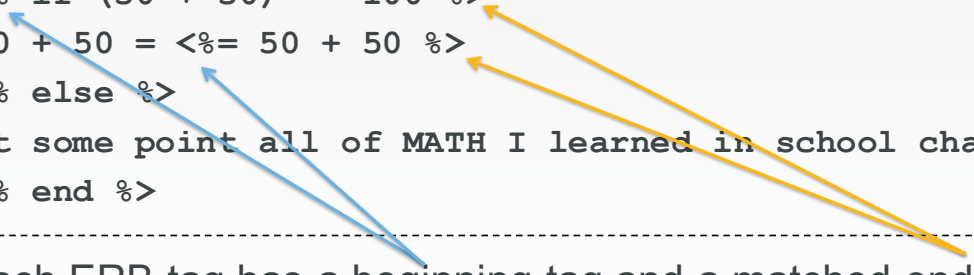
Text Within an ERB Template

```
<% 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.

Text Within an ERB Template

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



Each ERB tag has a beginning tag and a matched ending tag.

Text Within an ERB Template

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

Each ERB tag has a beginning tag and a matched ending tag.

Text Within an ERB Template

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

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

Text Within an ERB Template

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

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

CONCEPT

The Angry Squid

<%=

Lab: Copy Contents from motd to the .erb

cookbooks/workstation/templates/default/motd.erb

```
Property of ...

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


CHEF

Remove 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']}
"
  mode '0644'
  owner 'root'
  group 'root'
end
```


CHEF

Replace it with the Template Resource

cookbooks/workstation/recipes/setup.rb

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

template '/etc/motd' do
  source 'motd.erb'
  mode '0644'
  owner 'root'
  group 'root'
end
```


Change Directories and Apply the Cookbook

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

```
...
  * template[/etc/motd] action create
    - update content in file /etc/motd from f55d46 to 7fec26
    --- /etc/motd       2016-04-08 12:49:50.234093936 +0000
    +++ /etc/.motd20160408-9400-1f8z98b 2016-04-08 13:04:22.726093933 +0000
    @@ -4,4 +4,5 @@
       HOSTNAME : ip-172-31-0-106
       MEMORY   : 604308kB
       CPU      : 1799.999
    +
Running handlers:
Running handlers complete
Chef Client finished, 1/2 resources updated in 08 seconds
```

Update the Cookbook's Patch 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'
```


Lab: Cleaner Apache Recipe

Adding the node attributes to the index page made it hard 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

Lab: Use `chef` to Generate a Template

```
$ cd chef-repo
$ chef generate template cookbooks/apache index.html
```

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


Lab: Lets Look at the Template File

```
$ tree cookbooks/apache/templates
```

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

1 directory, 1 file
```

Lab: Cleaner Apache Recipe

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

Lab: Update the template file

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

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: <%= node['ipaddress'] %></h2>
    <h2>hostname: <%= node['hostname'] %></h2>
</body>
</html>
```

Cleaner Recipes

Adding the node attributes to the default page did make it harder to read the recipe.

Objective:

- ✓ Create a template with chef generate
- ✓ Define the contents of the ERB template
- ❑ Change the file resource to the template resource in the 'apache' cookbook

Lab: Remove the Existing Content Attribute

cookbooks/apache/recipes/server.rb

```
...
file '/var/www/html/index.html' do
  content "<h1>Hello, world!</h1>
<h2>IPADDRESS: #{node['ipaddress']}</h2>
<h2>HOSTNAME : #{node['hostname']}</h2>
"
end
...
```

Lab: Update the Source Attribute

cookbooks/apache/recipes/server.rb

```
...

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

...
```


CHEF

Lab: Change Directories and Apply the Cookbook

```
$ cd chef-repo
$ sudo chef-client -zr "recipe[apache]"
```

```
[2015-09-16T14:18:05+00:00] WARN: No config file found or specified on command line, using command line options.
Starting Chef Client, version 12.3.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
[2015-09-16T14:18:09+00:00] WARN: Cloning resource attributes for service[httpd] from prior resource (CHEF-3694)
[2015-09-16T14:18:09+00:00] WARN: Previous service[httpd]: /root/.chef/local-mode-cache/cache/cookbooks/apache/recipes/server.rb:8:in `from_file'
[2015-09-16T14:18:09+00:00] WARN: Current  service[httpd]: /root/.chef/local-mode-cache/ ...
```


CHEF

Lab: Update the Cookbook's Patch Number

cookbooks/apache/metadata.rb

```
name              'apache'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures apache'
long_description  'Installs/Configures apache'
version           '0.2.1'
```


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

DISCUSSION

Discussion

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

What do each of the ERB tags accomplish?

DISCUSSION

Q&A

What questions can we help you answer?

- Resources
- Templates
- ERB

Testing Cookbooks

Validating Our Recipes in Virtual Environments

CHEF

Objectives

After completing this module, you should be able to

- Use Test Kitchen to verify your recipes converge on a virtual instance
- Read the ServerSpec documentation
- Write and execute tests

Can We Test Cookbooks?

As we start to define our infrastructure as code we also need to start thinking about testing it.

Steps to Verify Cookbooks

What steps would it take to test one of the cookbooks that we have created?

- Create Virtual Machine
- Install Chef Tools
- Copy Cookbooks
- Run/Apply Cookbooks
- Verify Assumptions
- Destroy Virtual Machine

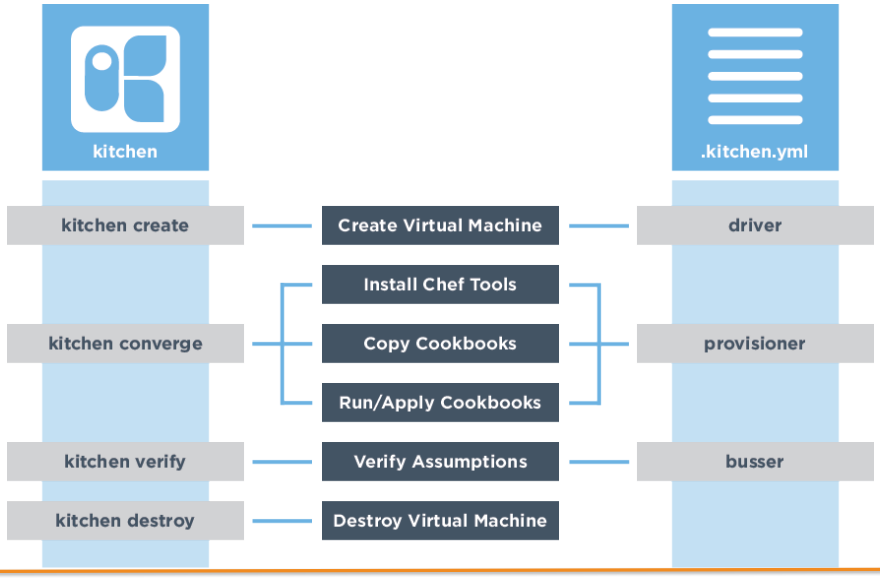
Test Configuration

What are we running in production? Maybe I could test the cookbook against a virtual machine.

Objective:

- ❑ Configure the "workstation" cookbook to test against the centos-6.7 platform
- ❑ Test the "workstation" cookbook on a virtual machine

Test Kitchen Commands and Configuration



Do We Have a .kitchen.yml?

```
$ ls -a cookbooks/workstation/
.       .gitignore    Berksfile     chefignore    recipes       test
..      .kitchen.yml  README.md     metadata.rb   spec
```

What is Inside .kitchen.yml?

```
$ cat cookbooks/workstation/.kitchen.yml
```

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.4

suites:
  - name: default
```


.kitchen.yml

When chef generates a cookbook, a default .kitchen.yml is created. It contains kitchen configuration for the driver, provisioner, platform, and suites.

http://kitchen.ci/docs/getting-started/creating-cookbook

Demo: The kitchen Driver

cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.5

...
```

The driver is responsible for creating a machine that we'll use to test our cookbook.

Example Drivers:

- docker
- vagrant
- ec2
- Azure
- …

What drivers does kitchen support?

```
$ kitchen driver discover
```

```
Gem Name                                Latest
Stable Release

jackal-kitchen-slack                    0.1.2
kitchen-all                             0.2.0
kitchen-ansible                         0.40.1
kitchen-ansiblepush                     0.3.10
kitchen-appbundle-updater               0.1.2
kitchen-azure                           0.1.0
kitchen-azurerm                         0.3.5
kitchen-binding                         0.2.2
kitchen-bluebox                         0.6.2
kitchen-cabinet                         3.0.0
kitchen-centurylink                     0.1.6
kitchen-cfengine                        0.0.5
kitchen-chef-extended-attributes        0.2.0
kitchen-chef_zero_berks_env             1.1.1
kitchen-cloudformation                  1.0.3
kitchen-cloudstack                      0.21.0
kitchen-config                          0.1.1
kitchen-digital_ocean                   0.4.0
kitchen-digitalocean                    0.9.5
kitchen-docker                          2.3.0
kitchen-docker-api                      0.4.0
kitchen-docker_cli                      0.15.0
kitchen-docker_ssh                      0.1.3
kitchen-dokken                          0.0.29
kitchen-driver-sakuracloud              0.1.3
kitchen-driver-vagrant_provision        1.0.0
kitchen-dsc                             0.8.2
kitchen-ec2                             1.0.0
kitchen-environment                     0.1.5
kitchen-fifo                            0.1.0
kitchen-fog                             0.7.3
kitchen-gce                             0.2.0
kitchen-goiardi                         0.1.1
kitchen-google                          1.1.0
kitchen-hyperv                          0.1.10
kitchen-inspec                          0.12.5
kitchen-inspector                       1.3.0
kitchen-itamae                          0.2.4
kitchen-joyent                          0.2.2
kitchen-libvirtlxc                      0.4.0
kitchen-linode                          0.10.0
kitchen-local                           0.0.1
kitchen-localhost                       0.3.0
kitchen-lxc                             0.1.4
kitchen-lxd_api                         0.0.2
kitchen-lxd_cli                         0.1.6
kitchen-machine                         0.1.0
kitchen-nodes                           0.7.0
kitchen-nodes-fqdn                      0.4.1

...
```

Demo: The kitchen Provisioner

cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.5

...
```

This tells Test Kitchen how to run Chef, to apply the code in our cookbook to the machine under test.

The default and simplest approach is to use chef_zero.

Demo: The kitchen Platforms

cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-12.04
  - name: centos-6.5

...
```

This is a list of operation systems on which we want to run our code.

Demo: The kitchen Suites

cookbooks/workstation/.kitchen.yml

```
...

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

This section defines what we want to test. It includes the Chef run-list of recipes that we want to test.

We define a single suite named "default".

Demo: The kitchen Suites

cookbooks/workstation/.kitchen.yml

```
...

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

The suite named "default" defines a run_list.

Run the "workstation" cookbook's "default" recipe file.

Kitchen Test Matrix

Kitchen defines a list of instances, or test matrix, based on the platforms multiplied by the suites.

PLATFORMS x SUITES

Running kitchen list will show that matrix.

What Can 'kitchen' Do?


```
$ kitchen --help
```

```
Commands:
  kitchen console                          # Kitchen Console!
  kitchen converge [INSTANCE|REGEXP|all]   # Converge one or more instances
  kitchen create [INSTANCE|REGEXP|all]     # Create one or more instances
  kitchen destroy [INSTANCE|REGEXP|all]    # Destroy one or more instances
  ...
  kitchen help [COMMAND]                   # Describe available commands or one specif...
  kitchen init                             # Adds some configuration to your cookbook...
  kitchen list [INSTANCE|REGEXP|all]       # Lists one or more instances
  kitchen setup [INSTANCE|REGEXP|all]      # Setup one or more instances
  kitchen test [INSTANCE|REGEXP|all]       # Test one or more instances
  kitchen verify [INSTANCE|REGEXP|all]     # Verify one or more instances
  kitchen version                          # Print Kitchen's version information
```

Example: Kitchen Test Matrix

```
$ kitchen list

Instance              Driver   Provisioner  Verifier  Transport Last Action
default-ubuntu-1204   Vagrant  ChefZero     Busser    Ssh       <Not Created>
default-centos-65     Vagrant  ChefZero     Busser    Ssh       <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```


Example: Kitchen Test Matrix

```
$ kitchen list

Instance              Driver   Provisioner  Verifier  Transport Last Action
default-ubuntu-1204   Vagrant  ChefZero     Busser    Ssh       <Not Created>
default-centos-65     Vagrant  ChefZero     Busser    Ssh       <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```

Lab: Test Configuration

What are we running in production? Maybe I could test the cookbook against a virtual machine.

Objective:

- Configure the "workstation" cookbook's .kitchen.yml to use the Docker driver and centos 6.7 platform
- Use kitchen converge to apply the recipe on a virtual machine

Lab: Move into the Cookbook's Directory

```
$ cd cookbooks/workstation
```

Lab: Edit the Kitchen Configuration File

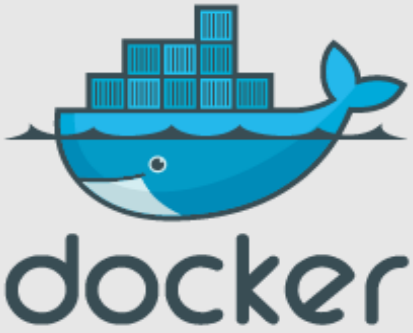
cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: docker

provisioner:
  name: chef_zero

platforms:
  - name: centos-6.7

suites:
# ... REMAINDER OF FILE ...
```



https://github.com/portertech/kitchen-docker

CHEF

Lab: Edit the Kitchen Configuration File

cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: docker

provisioner:
  name: chef_zero

platforms:
  - name: centos-6.7

suites:
# ... REMAINDER OF FILE ...
```

https://www.centos.org

CHEF

Lab: Look at the Test Matrix

```
$ kitchen list
```

Instance	Driver	Provisioner	Verifier	Transport	Last Action
default-centos-67	Docker	ChefZero	Busser	Ssh	<Not Created>

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

```
platforms:
- name: centos-6.7
```


Converging a Cookbook

Before I add features it really would be nice to test these cookbooks against the environments that resemble production.

Objective:

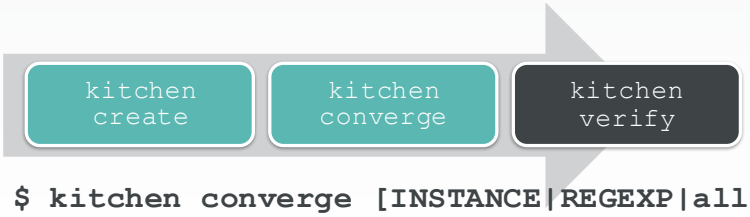
- ✓ Configure the "workstation" cookbook's .kitchen.yml to use the Docker driver and centos-6.7 platform
- ❑ Use kitchen converge to apply the recipe on a virtual machine

Kitchen Create


```
$ kitchen create [INSTANCE|REGEXP|all]

Create one or more instances.
```


Group Exercise: Kitchen Converge




```
$ kitchen converge [INSTANCE|REGEXP|all]

Create the instance (if necessary) and then apply
the run list to one or more instances.
```

Lab: Converge the Cookbook

```
$ kitchen converge
```

```
-----> Starting Kitchen (v1.4.0)
-----> Creating <default-centos-67>...
       Sending build context to Docker daemon  2.56 kB
(skipping)
----->  Finished creating <default-centos-67> (1m18.32s).
-----> Converging <default-centos-67>...
$$$$$$ Running legacy converge for 'Docker' Driver
(skipping)
Synchronizing Cookbooks:
         - workstation
       Compiling Cookbooks...
       Converging 0 resources
       Running handlers:
```


Lab: Converge the Recipe for Apache

- We want to validate that our run-list installs correctly.
- Within the "apache" cookbook use kitchen converge for the default suite on the centos 6.7 platform.

Lab: Configuring Test Kitchen for Apache

cookbooks/apache/.kitchen.yml

```
---
driver:
  name: docker

provisioner:
  name: chef_zero

platforms:
  - name: centos-6.7

suites:
  - name: default
    run_list:
```


Lab: Move into cookbook directory converge the cookbook

```
$ cd cookbooks/apache
$ kitchen converge
```

```
-----> Starting Kitchen (v1.4.0)
-----> Creating <default-centos-67>...
       Sending build context to Docker daemon  2.56 kB
       Sending build context to Docker daemon
(skipping)
 Installing Chef
       installing with rpm...
       warning: /tmp/install.sh.23/chef-12.4.1-1.el6.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID
83ef826a: NOKEY
(skipping)
 Synchronizing Cookbooks:
         - apache
       Compiling Cookbooks...
```

DISCUSSION

Test Kitchen

What is being tested when kitchen converges a recipe without error?

What is NOT being tested when kitchen converges the recipe without error?

DISCUSSION

Test Kitchen

What is left to validate to ensure that the cookbook successfully applied the policy defined in the recipe?

The First Test

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

Objective:

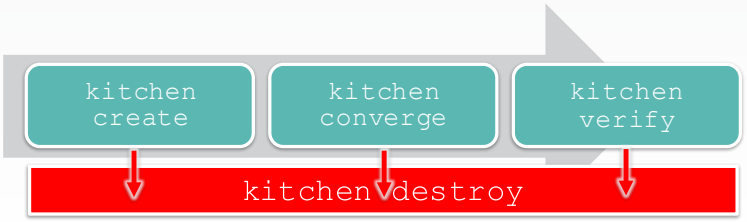
- In a few minutes we'll write and execute a test that asserts that the tree package is installed when the "workstation" cookbook's default recipe is applied.

Kitchen Verify


```
$ kitchen verify [INSTANCE|REGEXP|all]

Create, converge, and verify one or more
instances.
```

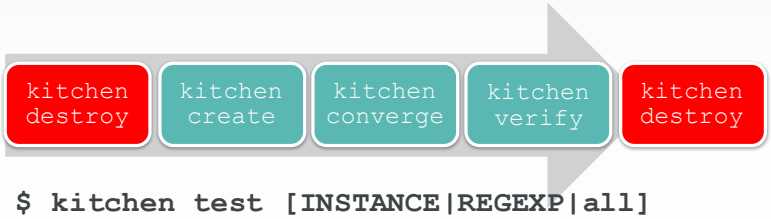
Kitchen Destroy




```
$ kitchen destroy [INSTANCE|REGEXP|all]
```

Destroys one or more instances.

Kitchen Test




```
$ kitchen test [INSTANCE|REGEXP|all]
```

Destroys (for clean-up), creates, converges, verifies and then destroys one or more instances.

ServerSpec

Serverspec tests your servers' actual state by executing command locally, via SSH, via WinRM, via Docker API and so on.

So you don't need to install any agent software on your servers and can use any configuration management tools, Puppet, Chef, CFEngine, Itamae and so on.

http://serverspec.org

Where do Tests Live?

```
workstation/test/integration/default/serverspec/default_spec.rb
```

Test Kitchen will look for tests to run under this directory. It allows you to put unit or other tests in test/unit, spec, acceptance, or wherever without mixing them up. This is configurable, if desired.

http://kitchen.ci/docs/getting-started/writing-test

Where do Tests Live?

`workstation/test/integration/default/serverspec/default_spec.rb`

This corresponds exactly to the Suite name we set up in the .kitchen.yml file. If we had a suite called "server-only", then you would put tests for the server only suite under

http://kitchen.ci/docs/getting-started/writing-test

Where do Tests Live?

`workstation/test/integration/default/serverspec/default_spec.rb`

This tells Test Kitchen (and Busser) which Busser runner plugin needs to be installed on the remote instance.

http://kitchen.ci/docs/getting-started/writing-test

Where do Tests Live?

`workstation/test/integration/default/serverspec/default_spec.rb`

All test files (or specs) are named after the recipe they test and end with the suffix "_spec.rb". A spec missing that will not be found when executing `kitchen verify`.

http://kitchen.ci/docs/getting-started/writing-test

Example: Is the 'tree' Package Installed?

```
describe package('tree') do
  it { should be_installed }
end
```

I expect the package tree should be installed.

http://serverspec.org/resource_types.html#package

Lab: Requiring a Test Helper

cookbooks/workstation/test/integration/default/serverspec/default_spec.rb

```
require 'spec_helper'

describe 'workstation::default' do

  describe package('tree') do
    it { should be_installed }
  end

end
```

Loads a helper file with that name in the same directory.

http://kitchen.ci/docs/getting-started/writing-test

Lab: Describing the Test Context

cookbooks/workstation/test/integration/default/serverspec/default_spec.rb

```
require 'spec_helper'

describe 'workstation::default' do

  describe package('tree') do
    it { should be_installed }
  end

end
```

Describing a body of tests for the 'workstation' cookbook's default recipe.

https://relishapp.com/rspec/rspec-core/v/3-3/docs

Lab: Our Assertion in a spec File

cookbooks/workstation/test/integration/default/serverspec/default_spec.rb

```
require 'spec_helper'

describe 'workstation::default' do

  describe package('tree') do
    it { should be_installed }
  end

end
```

When we converge the workstation cookbook's default recipe we expect the package named tree to be installed.

http://serverspec.org/resource_types.html#package

Lab: Return Home and Move into the Cookbook

```
$ cd cookbooks/workstation
```

Lab: Running the Specification

```
$ kitchen verify
```

```
-----> Starting Kitchen (v1.4.0)
-----> Converging <default-centos-67>...
$$$$$$ Running legacy converge for 'Docker' Driver
(skipping)
-----> Chef Omnibus installation detected (install only if missing)
       Transferring files to <default-centos-67>
       Starting Chef Client, version 12.4.1
(skipping)
       Running handlers:
       Running handlers complete
       Chef Client finished, 6/6 resources updated in 64.426896317 seconds
       Finished converging <default-centos-67> (1m9.02s).
-----> Kitchen is finished. (1m9.69s)
```


DISCUSSION

More Tests

What are other resources within the recipe that we could test?

CONCEPT

Testing a File

ServerSpec can help us assert different characteristics about files on the file system. Like if it is a file, directory, socket or symlink.

The file's mode owner or group. If the file is readable, writeable, or executable. It is even able to verify the data contained within the file.

http://serverspec.org/resource_types.html#file

CHEF

Example: The File Contains Data

```
describe file('/etc/passwd') do
  it { should be_file }
end
```

I expect the file named '/etc/passwd' to be a file (as opposed to a directory, socket or symlink).

http://serverspec.org/resource_types.html#file

Example: The File Contains Specific Content

```
describe file('/etc/httpd/conf/httpd.conf') do
  its(:content) { should match /ServerName www.example.jp/ }
end
```

I expect the file named '/etc/httpd/conf/httpd.conf' to have content that matches 'ServerName www.example.jp'

http://serverspec.org/resource_types.html#file

Example: The File is Owned by a Particular User

```
describe file('/etc/sudoers') do
  it { should be_owned_by 'root' }
end
```

I expect the file named '/etc/sudoers' to be owned by the 'root' user.

DISCUSSION

Testing or webserver

What are some things we could test to validate our web server has deployed correctly?

What manual tests do we use now to validate a working web server?

Lab: Testing Apache

- ❑ Create a test file for the "apache" cookbook's default recipe
- ❑ Add tests that validate a working web server
  http://serverspec.org/resource_types.html#port
  http://serverspec.org/resource_types.html#command
- ❑ Run kitchen verify
- ❑ Commit your changes

Lab: Return Home and 'cd cookbooks/apache'

```
$ cd cookbooks/apache
```


Lab: What Does the Webserver Say?

test/integration/default/serverspec/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  describe port(80) do
    it { should be_listening }
  end

  describe command('curl http://localhost') do
    its(:stdout) { should match /Hello, world!/ }
  end
end
```

Port 80 should be listening.

The standard out from the command 'curl http://localhost' should match 'Hello, world!'

Lab: Run ‘kitchen test’

```
$ kitchen test
```

```
...
apache::default
        Port "80"
          should be listening
        Command "curl http://localhost"
          stdout
            should match /Hello, world!/

      Finished in 0.22502 seconds (files took 0.56581 seconds to load)
      2 examples, 0 failures
...
```


Lab: Chef exit code

```
$ echo $?
```

```
0
```

‘kitchen test’ returns standard return codes, allowing the command to be used in CD pipelines

DISCUSSION

Discussion

Why do you have to run kitchen within the directory of the cookbook?

Where would you define additional platforms?

Why would you define a new test suite?

What are the limitations of using Test Kitchen to validate recipes?

DISCUSSION

Q&A

What questions can we help you answer?

- Test Kitchen
- kitchen commands
- kitchen configuration
- ServerSpec

Workstation Installation

Configuring Your Laptop as a Workstation

CHEF

Objectives

After completing this module, you should be able to

- Ensure that ChefDK is installed on your laptop
- Execute a series of commands to ensure everything is installed
- Install a local editor like Atom

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
- ❑ Download a repository of cookbooks
- ❑ Install git (optional)
- ❑ Install a text editor (optional)

ChefDK

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

You can find the ChefDK to download at the website downloads.chef.io.

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

GE: 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/

GE: 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/

GE: Installing ChefDK

Lab: Run All These Commands

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


CONCEPT

Text Editors

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

Whatever editor you use should optimize for this workflow.

CONCEPT

ATOM Editor

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

https://atom.io

CHEF™

©2015 Chef Software Inc.

The Chef Server

A Hub for Configuration Data

CHEF

Objectives

After completing this module, you should be able to

- Connect your local workstation (laptop) to a Chef Server
- Upload cookbooks to a Chef Server
- Bootstrap a node
- Manage a node via a Chef Server

CHEF

More Web Servers?

More easily manage multiple nodes

Objective:

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

CHEF

Managing an Additional System

To manage another system, you would need to:

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

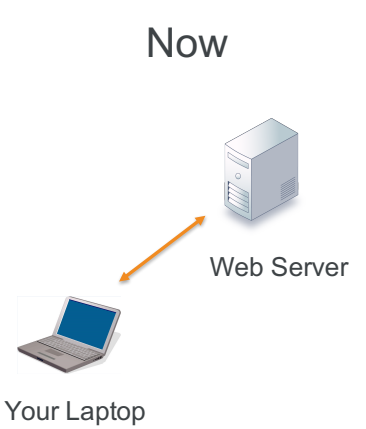
CHEF

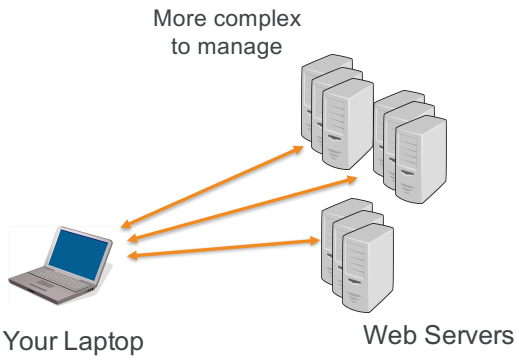
Managing Additional Systems

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

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

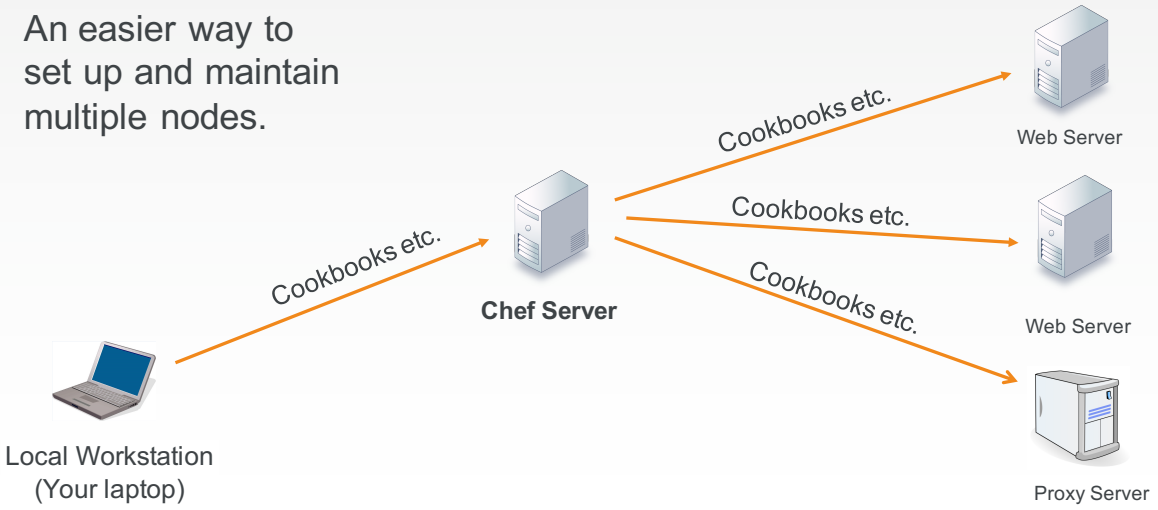
Managing Additional Systems





The Chef Server

An easier way to set up and maintain multiple nodes.



CHEF

Flavors of Chef Server

Open Source	**Chef Server**	Multi-tenant
Chef Server	(Support + Premium Features)	**Hosted Chef Server**

CHEF

Lab: Hosted Chef

More easily manage multiple nodes

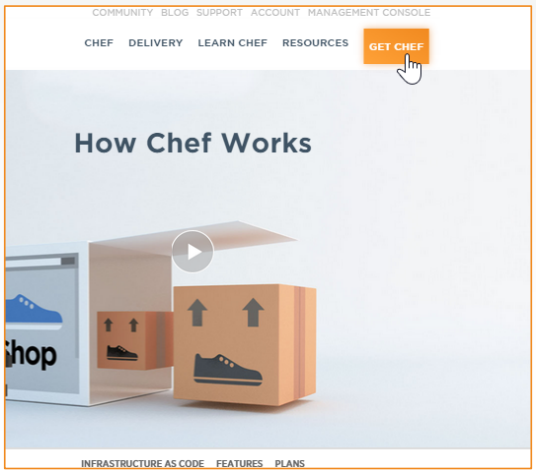
Objective:

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

Lab: Signing Up for a Hosted Chef Account

Steps

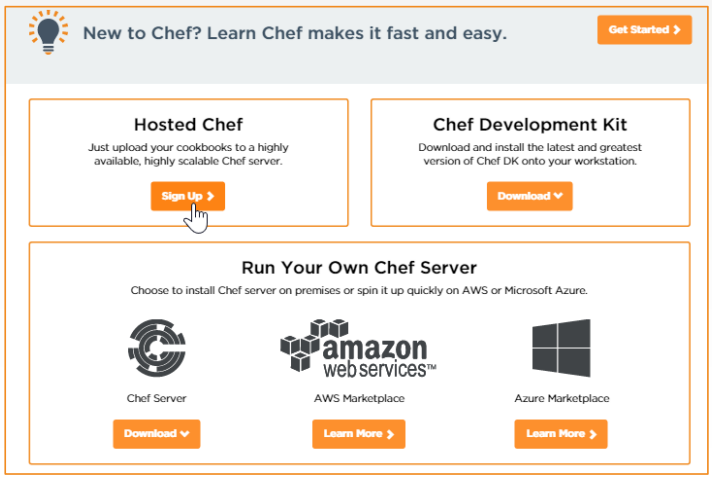
1. Navigate to https://www.chef.io
2. From the resulting window, click **Get Chef**.



Lab: Signing Up for a Hosted Chef Account

Steps

3. From the resulting window, click the Hosted Chef **Sign Up** button.

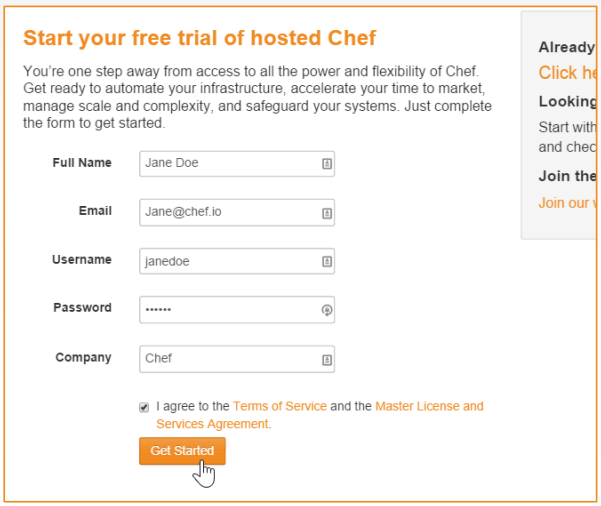


Lab: Signing Up for a Hosted Chef Account

Steps

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

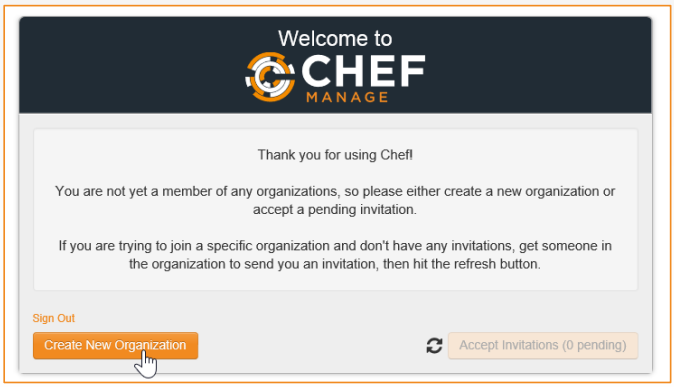
Note: You should write down your new user name and remember your password.



Lab: Signing Up for a Hosted Chef Account

Steps

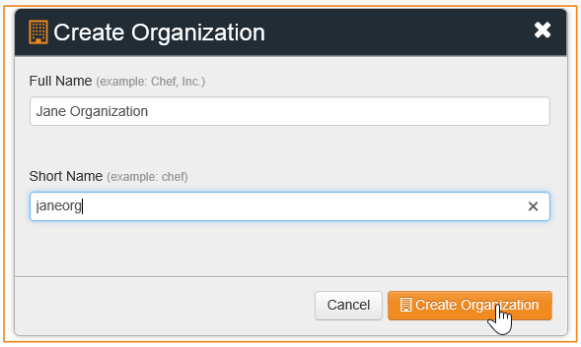
5. From the resulting page, click the **Create New Organization** button.



Lab: Signing Up for a Hosted Chef Account

Steps

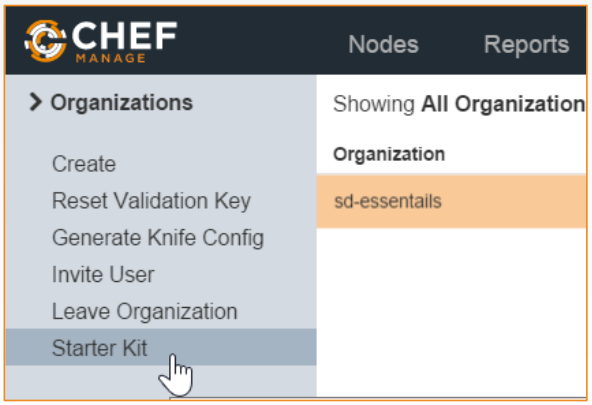
6. Fill out the resulting Create Organization form and then click **Create Organization**.



Lab: Signing Up for a Hosted Chef Account

Steps

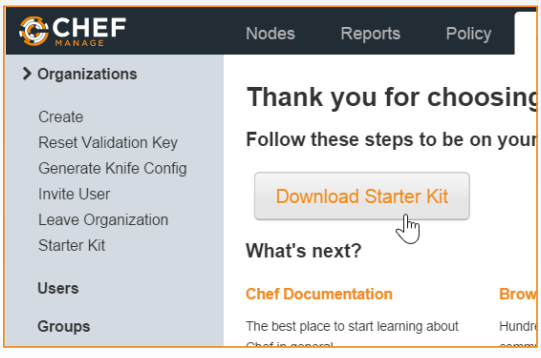
7. From the resulting page, click your new organization to highlight it and then click **Starter Kit**.

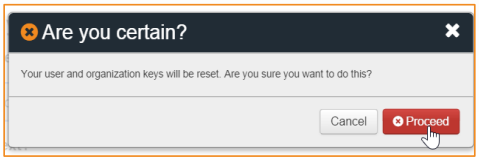


Lab: Signing Up for a Hosted Chef Account

Steps

8. From the resulting window, click the **Download Starter Kit** button.
9. Click the **Proceed** button when prompted.

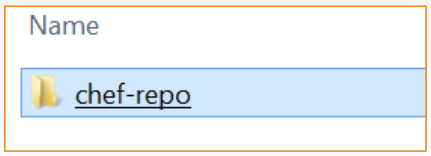




Lab: Signing Up for a Hosted Chef Account

Steps

10. Open the downloaded zip file and copy chef-repo folder that's contained in the zip file.
11. 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

Lab: Download a Repository

A repository containing a similar copy of the work you did previously in this course can be downloaded from here:

https://github.com/chef-training/chefdk-fundamentals-repo

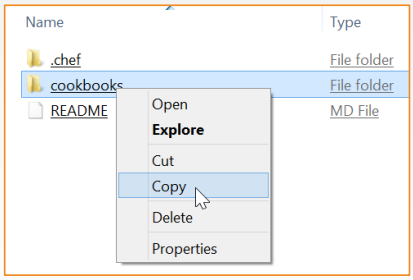
Lab: Download the Repository

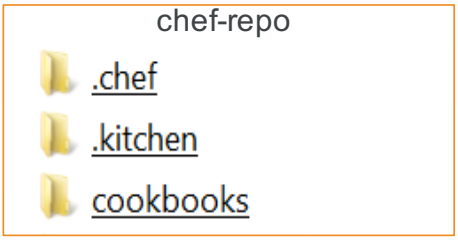
CHEF

Lab: Paste the cookbooks Folder

Steps

- Open the downloaded chefdk-fundamentals-repo-master zip file and then copy **only** 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

Lab: Navigate to the chef-repo

```
$ cd chef-repo
```


CONCEPT

knife

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

Lab: 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)
```


CHEF

Lab: 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)
```


CHEF

Lab: knife client list

```
$ knife client list
```

```
ORGNAME-validator
```


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

Lab: 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)
```


CHEF

Lab: knife cookbook list

```
$ knife cookbook list
```


CHEF

Lab: Change to the cookbooks/apache Directory

```
$ cd cookbooks/apache
```


CONCEPT

Berkshelf

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

berkshelf.com

Lab: 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
  berks search NAME           # Search the remote source for cookbooks matching the partial
```


Lab: Run berks install

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'apache' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using apache (0.2.1) from source at .
```

Lab: 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
```


Lab: See the Contents of the Berksfile.lock

```
$ cat Berksfile.lock
```

```
DEPENDENCIES
  apache
    path: .
    metadata: true

GRAPH
  apache (0.2.1)
```

Lab: Upload the Cookbook to the Chef Server

```
$ berks upload
```

```
Uploaded apache (0.2.1) to:
'https://api.opscode.com:443/organizations/steveessentials2'
```


Lab: Display Cookbooks within Your Org

```
$ knife cookbook list
```

```
apache          0.2.1
```

Lab

Lab: Upload Cookbooks

- ❑ Upload your remaining cookbooks
- ❑ Verify that all cookbooks are uploaded

CHEF

Lab: cd and Run knife cookbook list

```
$ cd chef-repo/cookbooks/workstation
$ knife cookbook list
```

```
apache       0.2.1
```


CHEF

Lab: Install the Cookbook Dependencies

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'workstation' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using workstation (0.2.1) from source at .
```


Lab: Upload the Cookbook to the Chef Server

```
$ berks upload
```

```
Uploaded workstation (0.2.1) to:
'https://api.opscode.com:443/organizations/steveessentials2'
```

Lab: Is the workstation Cookbook Uploaded?

```
$ knife cookbook list
```

```
apache          0.2.1
workstation     0.2.1
```


Hosted Chef

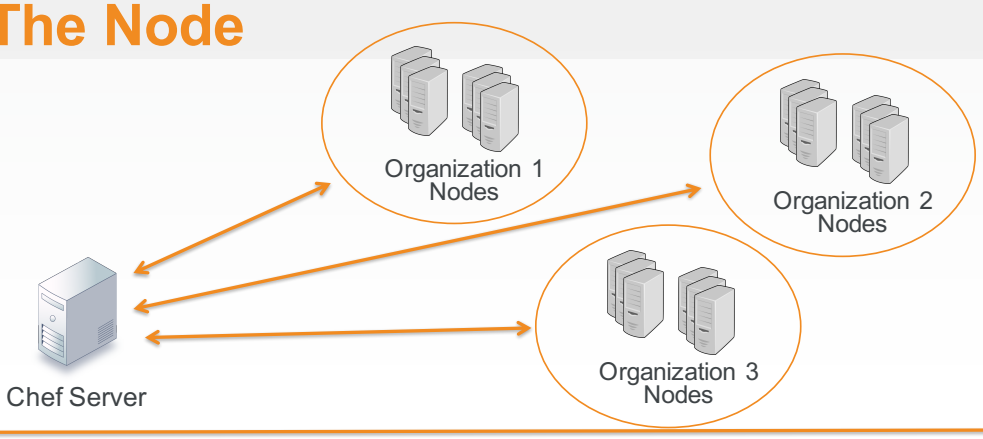
More easily manage multiple nodes

Objective:

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

CONCEPT

The Node



CHEF

Lab: Change to the chef-repo

```
$ cd chef-repo
```


CHEF

Lab: Run 'knife node –help'

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

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


CHEF

Lab: Run 'knife node list'

```
$ knife node list
```


CHEF

Lab: 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 internally 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.
```


Lab: Bootstrap Your Node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node1
```

```
Creating new client for node1
Creating new node for node1
Connecting to ec2-54-175-46-24.compute-1.amazonaws.com
ec2-54-175-46-24.compute-1.amazonaws.com Starting first Chef Client run...
ec2-54-175-46-24.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
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 [2015-010-16T16:51:21+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:
```

Lab: Run 'knife node list' Again

```
$ knife node list
```

```
node1
```


Lab: View More Information About Your Node

```
$ knife node show node1
```

```
Node Name:   node1
Environment: _default
FQDN:        ip-172-31-8-68.ec2.internal
IP:          54.175.46.24
Run List:
Roles:
Recipes:
Platform:    centos 6.7
Tags:
```

DISCUSSION

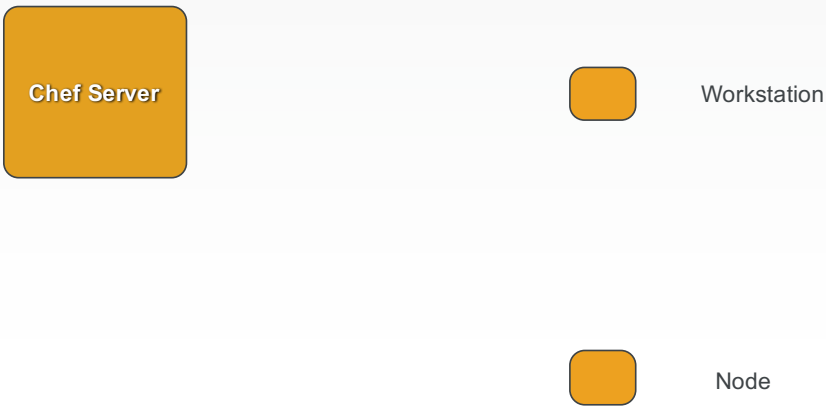
What happened during bootstrap?

Chef and all of its dependencies installed
Installation includes

- The Ruby language - used by Chef
- chef-client - Client application
- ohai - System profiler
- ...and more

CHEF

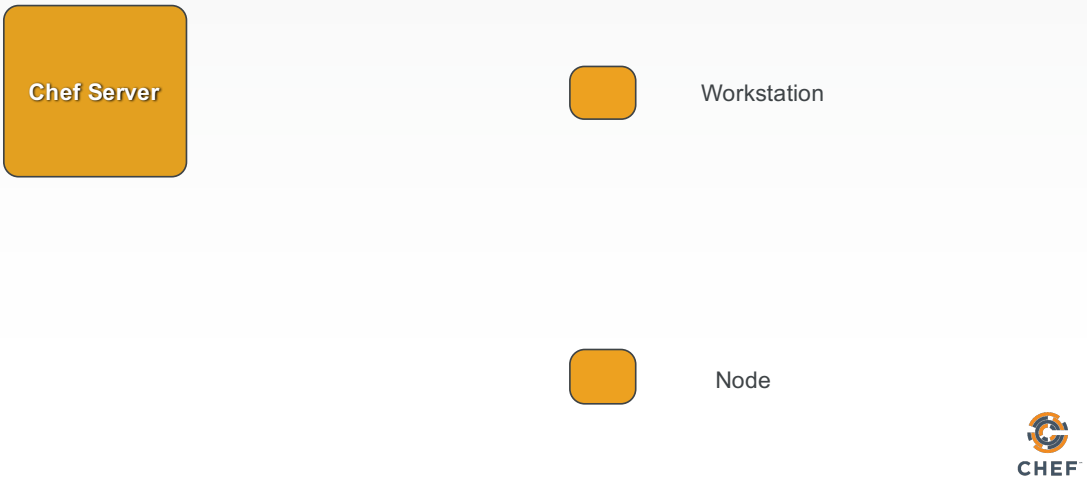
knife bootstrap sequence



CHEF

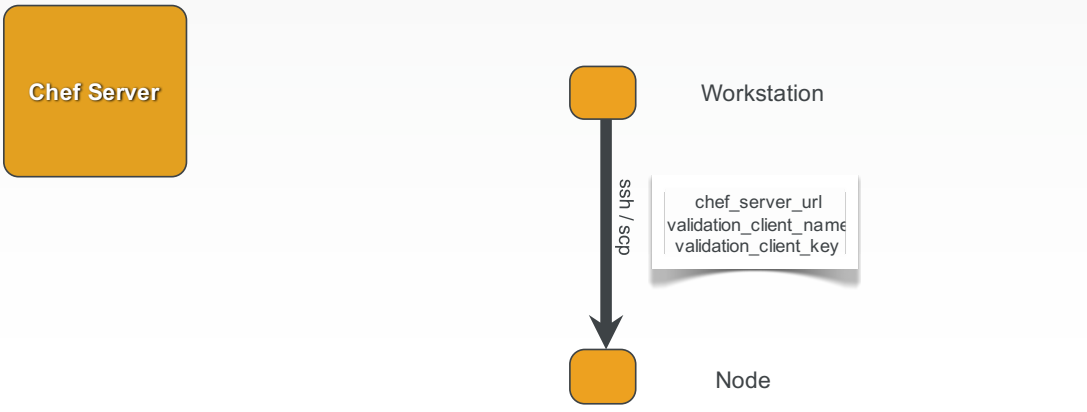
knife bootstrap sequence

```
knife bootstrap <IP> --sudo -x chef -P chef -N
"mynode"
```



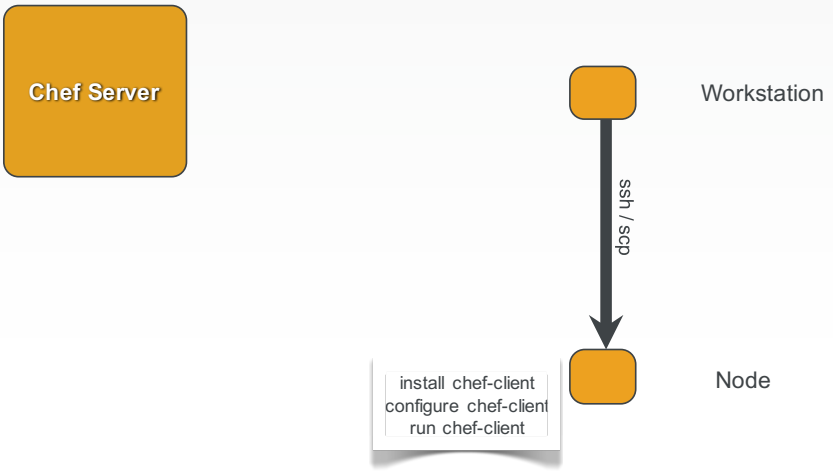
knife bootstrap sequence

```
knife bootstrap <IP> --sudo -x chef -P chef -N
"mynode"
```



knife bootstrap sequence

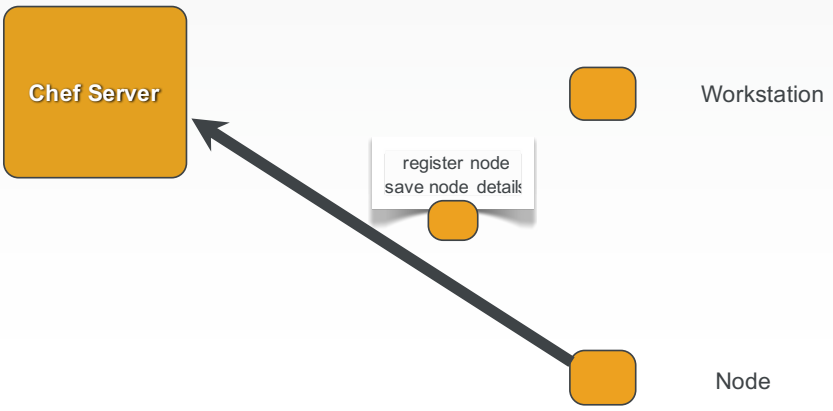
```
knife bootstrap <IP> --sudo -x chef -P chef -N
"mynode"
```



CHEF

knife bootstrap sequence

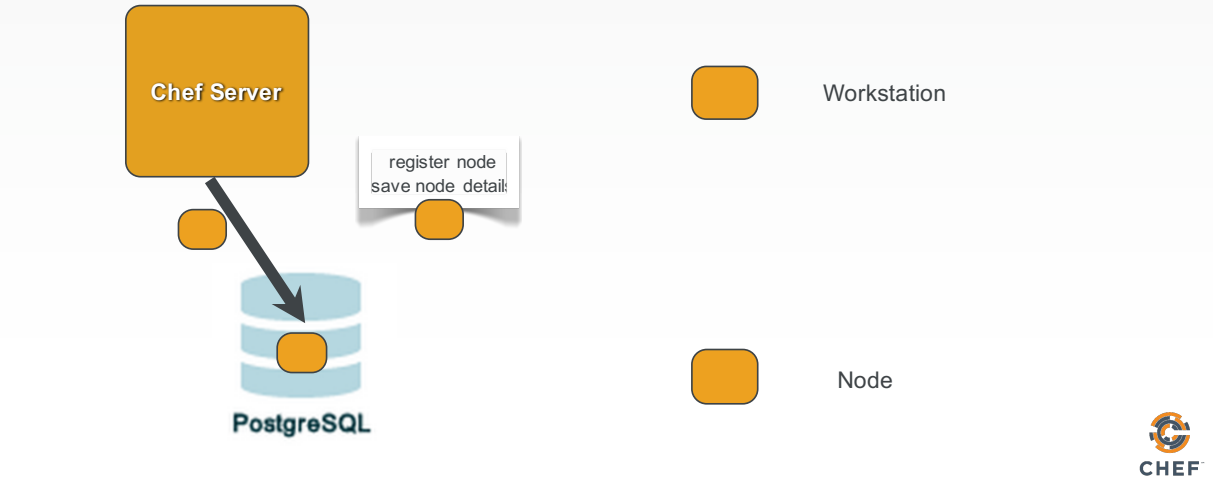
```
knife bootstrap <IP> --sudo -x chef -P chef -N
"mynode"
```



CHEF

knife bootstrap sequence

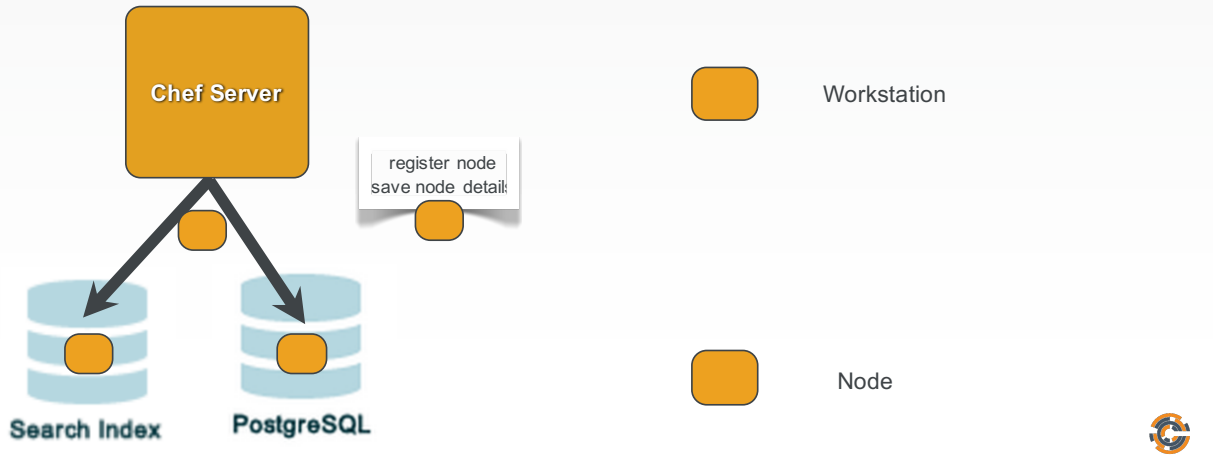
```
knife bootstrap <IP> --sudo -x chef -P chef -N
"mynode"
```



CHEF

knife bootstrap sequence

```
knife bootstrap <IP> --sudo -x chef -P chef -N
"mynode"
```



CHEF

Lab: Add a Recipe to a Run List

```
$ knife node run_list add node1 "recipe[apache]"
```

```
node1:
  run_list: recipe[apache]
```

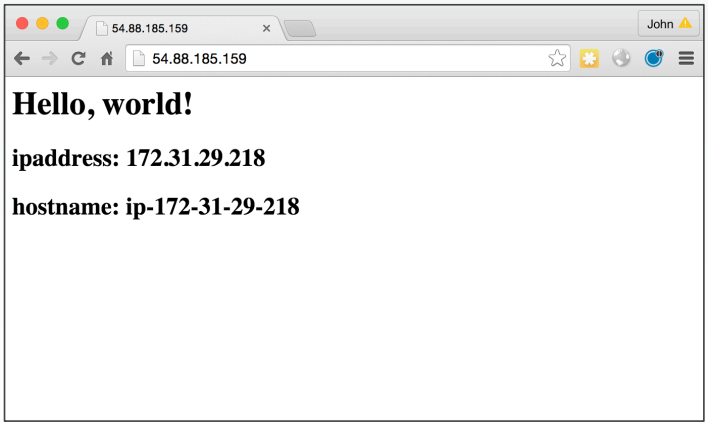

Lab: Rerun chef-client on the node

```
node1$ sudo chef-client
```

```
...
@@ -1 +1,8 @@
    +<html>
    +  <body>
    +    <h1>Hello, world!</h1>
    +    <h2>ipaddress: 172.31.29.218</h2>
    +    <h2>hostname: ip-172-31-210-218</h2>
    +</body>
    +</html>
  * service[httpd] action enable
    - enable service service[httpd]
  * service[httpd] action start
    - start service service[httpd]

Running handlers:
Running handlers complete
Chef Client finished, 4/4 resources updated in 25.003999599 seconds
```

Lab: Testing our webserver



CHEF

Hosted Chef

More easily manage multiple nodes

Objective:

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

CHEF

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?

DISCUSSION

Q&A

What questions can you help you answer?

- Chef Server
- Managed Chef
- Berkshelf
- Bootstrapping Nodes

Templates, Variables & Search

Creating Dynamic Content

CHEF

Objectives

After completing this module, you should be able to

- Set the runlist while bootstrapping a node
- Identify EC2 specific node attributes
- Execute a search using knife and within a recipe

Manage multiple nodes

Our site has just got super busy so we now need to manage multiple nodes

Lab: Lets bootstrap a new node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node2 -r 'recipe[apache]'
```

```
...
54.84.233.7     +    <h2>ipaddress: 172.31.29.219</h2>
54.84.233.7     +    <h2>hostname: ip-172-31-29-219</h2>
54.84.233.7     +</body>
54.84.233.7     +</html>
54.84.233.7   * service[httpd] action enable
54.84.233.7     - enable service service[httpd]
54.84.233.7   * service[httpd] action start
54.84.233.7     - start service service[httpd]
54.84.233.7
54.84.233.7 Running handlers:
54.84.233.7 Running handlers complete
54.84.233.7 Chef Client finished, 4/4 resources updated in 24.447046971 seconds
```


Lab: Lets bootstrap a new node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node2 -r 'recipe[apache]'
```

```
...
54.84.233.7     +    <h2>ipaddress: 172.31.29.219</h2>
54.84.233.7     +    <h2>hostname: ip-172-31-29-219</h2>
54.84.233.7     +</body>
54.84.233.7     +</html>
54.84.233.7   * service[httpd] action enable
54.84.233.7     - enable service service[httpd]
54.84.233.7   * service[httpd] action start
54.84.233.7     - start service service[httpd]
54.84.233.7
54.84.233.7 Running handlers:
54.84.233.7 Running handlers complete
54.84.233.7 Chef Client finished, 4/4 resources updated in 24.447046971 seconds
```

DISCUSSION

Test the Webservers

To test these web servers we need to find their FQDN or public IP Address

CHEF

Lab: List all our nodes

```
$ knife node list
```

```
node1
node2
```


CHEF

Lab: View More Information About Nodes

```
$ knife node show node1
```

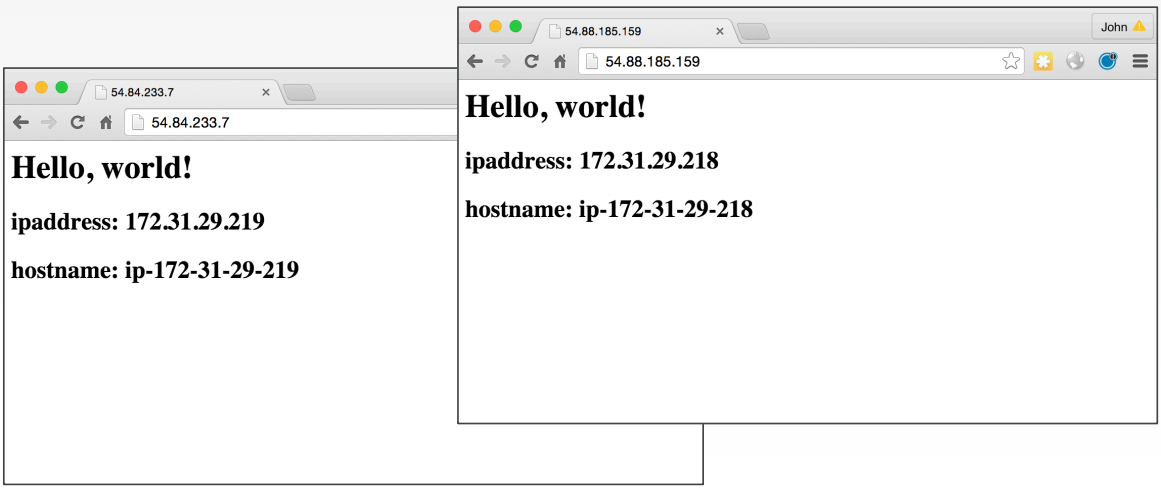
```
Node Name:   node2
Environment: _default
FQDN:        ip-172-31-29-218.ec2.internal
IP:          54.88.185.159
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:
```

```
$ knife node show node2
```

```
Node Name:   node2
Environment: _default
FQDN:        ip-172-31-29-219.ec2.internal
IP:          54.84.233.7
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:
```


CHEF

Lab: Testing our websites



CHEF

How do we see detail across all nodes?

Ugh. `knife node show` only works with individual nodes – how do I see this info for multiple nodes

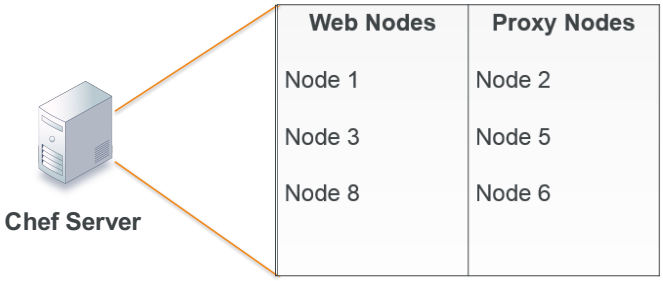
How do we see detail across all nodes?

Ugh. `knife node show` only works with individual nodes – how do I see this info for multiple nodes

Search allows you to retrieve information across multiple nodes

The Chef Server and Search

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



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

CHEF

Lab: View information for all nodes

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

```
2 items found

Node Name:   node1
Environment: _default
FQDN:        ip-172-31-29-218.ec2.internal
IP:          54.88.185.159
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:

Node Name:   node2
Environment: _default
FQDN:        ip-172-31-29-219.ec2.internal
IP:          54.84.233.7
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:
```


Lab: Narrow the search

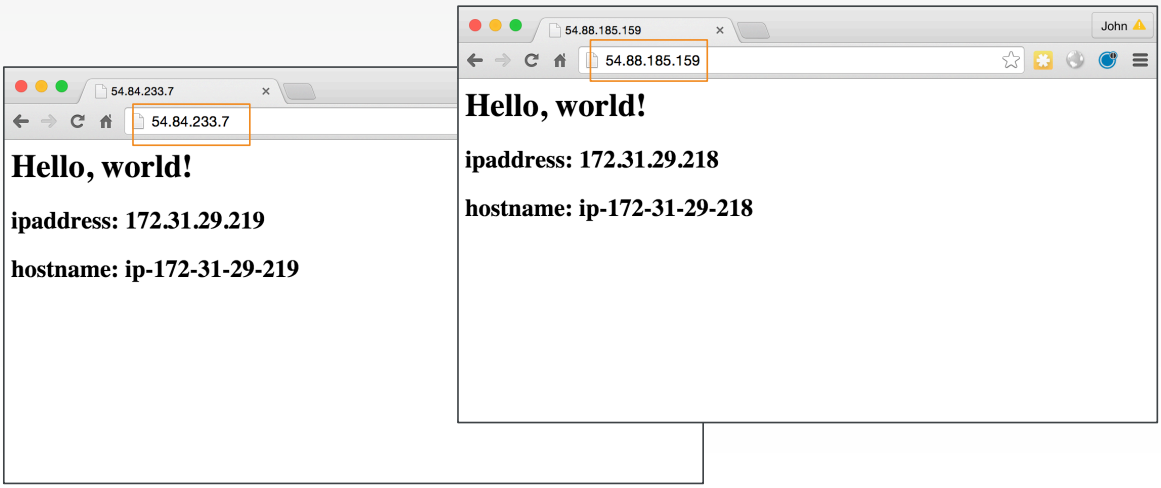
```
$ knife search node "*:*" -a ipaddress
```

```
2 items found

node1:
  name: 172.31.13.2

node2:
  name: 172.31.6.173
```

Back to our websites…

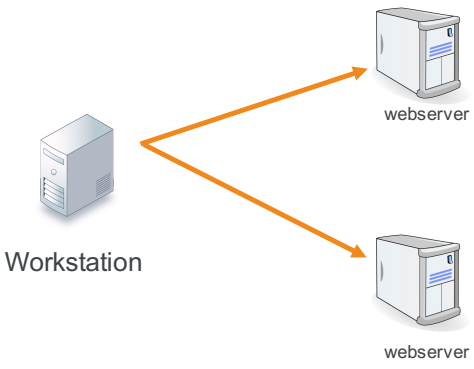


We have to browse to each site individually, ugh

CHEF

Two webservers, one browser

So we have scaled out our webservers, but we still need to hit each webserver individually

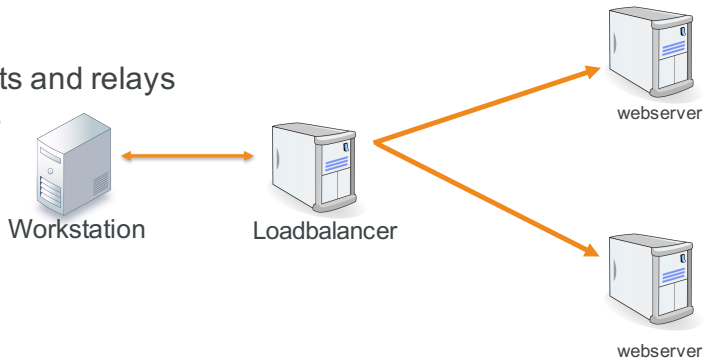


CHEF

Load balancer

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

The LB receives requests and relays them to the web servers



CHEF

Lab: Scaling up

Our site has just got super busy with multiple web servers – so we now need a load balancer.

Objective:

- ❑ Create a load balancer cookbook
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the load balancer cookbook

CHEF

Lab: Generate HAProxy Cookbook

```
$ cd chef-repo
$ chef generate cookbook cookbooks/haproxy
```

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


CHEF

Lab: Edit haproxy cookbook's default recipe

chef-repo/cookbooks/haproxy/recipes/default.rb

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

package 'haproxy'

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
end

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


CHEF

Lab: Edit haproxy cookbook's default recipe

chef-repo/cookbooks/haproxy/recipes/default.rb

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

package 'haproxy'

template '/etc/haproxy/haproxy.cfg' do
 source 'haproxy.cfg.erb'
end

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


DISCUSSION

Configure HAProxy

We need to configure HAProxy to route traffic to our web server.

Have a look at how HAProxy is configured

http://bit.ly/1Xoai9R

Lab: Generate the Template

```
$ cd chef-repo
$ chef generate template cookbooks/haproxy haproxy.cfg
```

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


Lab: Configuring haproxy.cfg.erb

cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...
frontend main *:5000
    acl url_static       path_beg       -i /static /images /javascript /stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    use_backend static          if url_static
    default_backend             app

backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

backend app
    balance     roundrobin
    server app <<IP ADDRESS>>:80 weight 1 maxconn 100 check
```

If you are feeling hardcore, type it
http://bit.ly/1Xoai9R

HAProxy Configuration should look like this

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

We need to add the webserver's IP Addresses to haproxy.cfg

HAProxy Configuration should look like this

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

We need to add the webserver's IP Addresses to haproxy.cfg

Doing it manually seems wrong

HAProxy Configuration should look like this

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

We need to add the webserver's IP Addresses to haproxy.cfg

Doing it manually seems wrong

Search!

HAProxy Configuration should look like this

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

Values from

```
knife search node "recipes:apache\:\:default" -a ipaddress
```

Heuston, we have a problem!

```
$ knife search node "recipes:apache\:\:default" -a ipaddress
```

```
2 items found

node1:
  ipaddress:  172.31.29.218

node2:
  ipaddress:  172.31.29.219
```

- These IP Addresses are not accessible from the outside network

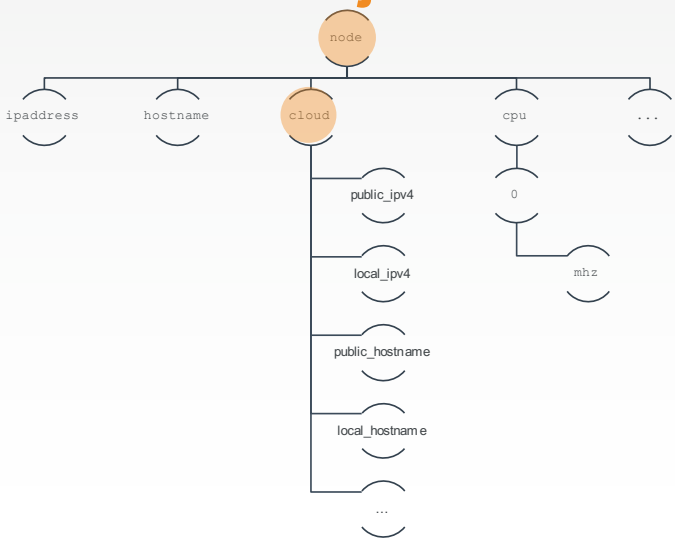
Amazon EC2 Instances

We can't use the `ipaddress` attribute within our recipes – they're on the private (internal) network & we need external access

In a previous section we looked at the node object

Nodes in EC2 have some specific networking attributes in their node object – some private & some public

EC2 and the Node Object



CHEF

Individual Node's EC2 information

```
$ knife node show node1 -a cloud
```

```
node1:
  cloud:
    local_hostname:  ip-172-31-29-218.ec2.internal
    local_ipv4:      172.31.29.218
    private_ips:     172.31.29.218
    provider:        ec2
    public_hostname: ec2-54-88-185-159.compute-1.amazonaws.com
    public_ips:      54.88.185.159
    public_ipv4:     54.88.185.159
```


CHEF

Individual Node's Public IP Address

```
$ knife node show node1 -a cloud.public_ipv4
```

```
node1:
  cloud.public_ipv4: 54.88.185.159
```


View EC2 information for all nodes

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

```
2 items found

node1:
  cloud:
    local_hostname:  ip-172-31-29-218.ec2.internal
    local_ipv4:      172.31.29.218
    private_ips:     172.31.29.218
    provider:        ec2
    public_hostname: ec2-54-88-185-159.compute-1.amazonaws.com
    public_ips:      54.88.185.159
    public_ipv4:     54.88.185.159

node2:
  cloud:
    local_hostname:  ip-172-31-29-219.ec2.internal
    local_ipv4:      172.31.29.219
...
```


View Public IP for all nodes

```
$ knife search node "*:*" -a cloud.public_ipv4
```

```
2 items found

node1:
  cloud.public_ipv4: 54.88.185.159

node2:
  cloud.public_ipv4: 54.84.233.7
```


CHEF

HAProxy Configuration should look like this

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

Values from
`knife search node "recipes:apache\:\:default" -a` **`cloud.public_ipv4`**

CHEF

Lab: Edit haproxy cookbook's default recipe

chef-repo/cookbooks/haproxy/recipes/default.rb

```
...
package 'haproxy'

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  variables(
    :webservers => webservers
  )
 notifies :restart, 'service[haproxy]'
end

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


Lab: Edit haproxy cookbook's default recipe

chef-repo/cookbooks/haproxy/recipes/default.rb

```
...
package 'haproxy'

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  variables(
    :webservers => webservers
  )
 notifies :restart, 'service[haproxy]'
end

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

Invoke a search in the recipe and save the results in the variable 'webservers'

Lab: Edit haproxy cookbook's default recipe

chef-repo/cookbooks/haproxy/recipes/default.rb

```
...
package 'haproxy'

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  variables(
    :webservers => webservers
  )
  notifies :restart, 'service[haproxy]'
end

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

Pass the variable 'webservers' into the template

Lab: Configuring haproxy.cfg.erb

cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...
    use_backend static          if url_static
    default_backend             app

backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

backend app
    balance     roundrobin
    ~~server app <<IP_ADDRESS>>:80 weight 1 maxconn 100 check~~
    <% @webservers.each_with_index do |web, n| -%>
       server <%= "app#{n}" %> <%= web['cloud']['public_ipv4'] %>:80 weight 1 maxconn 100 check
    <% end -%>
```

Remove line in template with hardcoded IP placeholder

Lab: Configuring haproxy.cfg.erb

cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...
    use_backend static          if url_static
    default_backend             app

backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

backend app
    balance     roundrobin
    <% @webservers.each_with_index do |web, n| -%>
       server <%= "app#{n}" %> <%= web['cloud']['public_ipv4'] %>:80 weight 1 maxconn 100 check
    <% end -%>
```

Iterate over the 'webservers' and add a line for each

Lab: Scaling up

Our site has just got super busy with multiple web servers – so we now need a load balancer.

Objective:

- ✓ Create a load balancer cookbook
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the load balancer cookbook

Lab: Upload the Cookbook

- Upload the haproxy cookbook to the Chef Server

Lab: Upload the Cookbook

```
$ cd chef-repo/cookbooks/haproxy
```

Lab: Upload the Cookbook

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'haproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using haproxy (0.1.0) from source at .
```


Lab: Upload the Cookbook

```
$ berks upload
```

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

Lab: Verify the Cookbook Upload

```
$ knife cookbook list
```

```
apache              0.2.1
haproxy             0.1.0
workstation         0.2.1
```


CHEF

Lab: Scaling up

Our site has just got super busy with multiple web servers – so we now need a load balancer.

Objective:

- ✓ Create a load balancer cookbook
- ✓ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the load balancer cookbook

CHEF

Lab: Bootstrap a Load Balancer Node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node3 -r 'recipe[haproxy]'
```

```
...
54.88.169.195     -    server  app3 127.0.0.1:5003 check
54.88.169.195     -    server  app4 127.0.0.1:5004 check
54.88.169.195     -
54.88.169.195     +    server app0 54.88.185.159:80 weight 1 maxconn 100 check
54.88.169.195     +    server app1 54.84.233.7:80 weight 1 maxconn 100 check
54.88.169.195   * service[haproxy] action start
54.88.169.195     - start service service[haproxy]
54.88.169.195   * service[haproxy] action enable
54.88.169.195     - enable service service[haproxy]
54.88.169.195
54.88.169.195 Running handlers:
54.88.169.195 Running handlers complete
54.88.169.195 Chef Client finished, 4/4 resources updated in 11.370356638 seconds
```

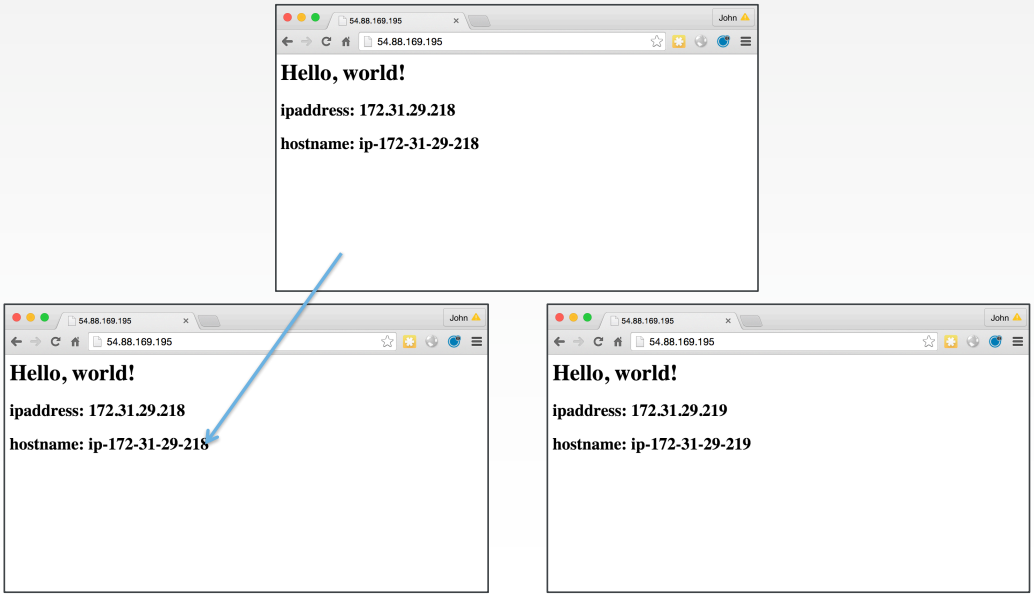

CHEF

Lab: Validate the New Node

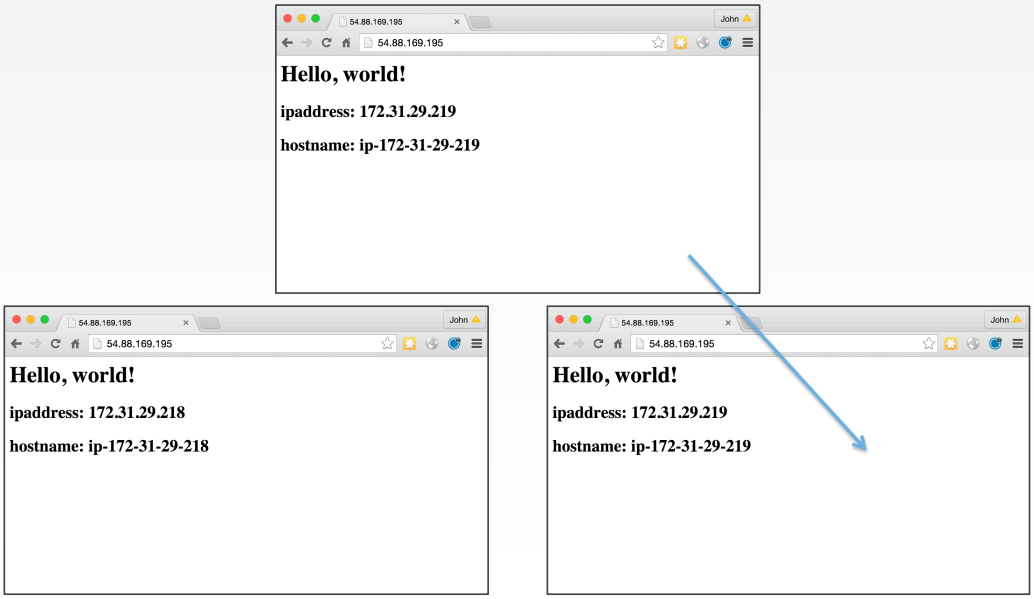
```
$ knife node show node3
```

```
Node Name:   node3
Environment: _default
FQDN:        ip-172-31-29-217.ec2.internal
IP:          54.88.169.195
Run List:    recipe[haproxy]
Roles:
Recipes:     haproxy::default
Platform:    centos 6.7
Tags:
```


CHEF


54.88.169.195
John
Hello, world!
ipaddress: 172.31.29.218
hostname: ip-172-31-29-218
Hello, world!
ipaddress: 172.31.29.218
hostname: ip-172-31-29-218
Hello, world!
ipaddress: 172.31.29.219
hostname: ip-172-31-29-219

CHEF


54.88.169.195
John
Hello, world!
ipaddress: 172.31.29.219
hostname: ip-172-31-29-219
Hello, world!
ipaddress: 172.31.29.218
hostname: ip-172-31-29-218
Hello, world!
ipaddress: 172.31.29.219
hostname: ip-172-31-29-219

CHEF

DISCUSSION

Discussion

How might Search work in the context of a webserver & database?

Where else might you use dynamic content in files?

DISCUSSION

Q&A

What questions can we help you answer?

- Search
- Passing variables into templates

Cookbook Attributes, Attribute Files and Dependencies

Setting attributes within a cookbook

CHEF

Objectives

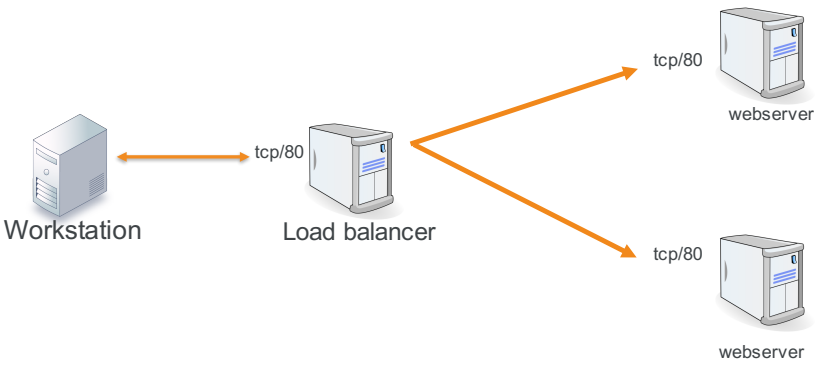
After completing this module, you should be able to

- Explain where cookbook attributes reside
- Configure dependencies between cookbooks
- Use `knife ssh` command

CHEF

Our topology

We now have a load balancer & two web servers, all listening on port 80

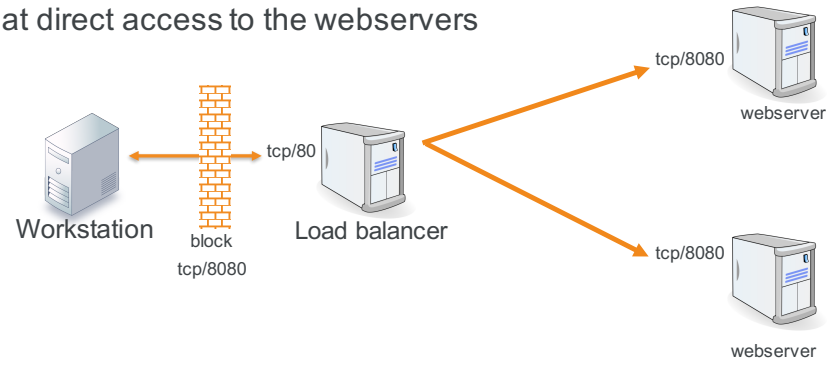


CHEF

Our topology

We now have a load balancer & two web servers, all listening on port 80

IT have mandated that direct access to the webservers should be blocked



CHEF

LAB

Lab: Reconfigure Apache

So we want HAProxy to serve content of port tcp/80, and Apache to serve of tcp/8080

Objective:

- ❑ Reconfigure Apache to serve content of port tcp/8080
- ❑ Reconfigure HAProxy to contact Apache on port tcp/8080

CHEF

Reconfigure Apache

Apache is configured to listen on port 80 in the file `/etc/httpd/conf/httpd.conf`

We will create a template for this file in the apache cookbook and change the value to 8080 via an attribute

CHEF

CONCEPT

Attribute Files

The Node Object contains many automatic attributes generate by OHAI

You can also maintain attributes within a cookbook

These are like variables or parameters for your cookbook and allow recipes to be data driven

CHEF

DISCUSSION

Best Practices

- Well-written cookbooks change behavior based on attributes
- Ideally, you don't have to modify the contents of a cookbook to use it for your specific use case
- Look at the attributes directory for things you can override through roles to affect behavior of the cookbook
- Of course, well written cookbooks have sane defaults, and a README to describe all this.

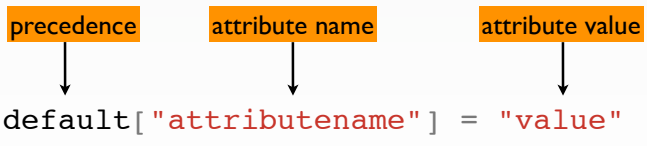
CHEF

Setting attributes in attribute files

Cookbook attributes are set in the attributes file

`./cookbooks/<cookbook>/attributes/default.rb`

Format is



We'll look at precedence later....

CHEF

Lab: Bump the cookbook version number

cookbooks/apache/metadata.rb

```
name              'apache'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures apache'
long_description  'Installs/Configures apache'
version           '0.3.0'
```


CHEF

Lab: Generate the attributes file

```
$ cd chef-repo
$ chef generate attribute cookbooks/apache default
```

```
Compiling Cookbooks...
Recipe: code_generator::attribute
  * directory[cookbooks/apache/attributes] action create
    - create new directory cookbooks/apache/attributes
  * template[cookbooks/apache/attributes/default.rb] action create
    - create new file cookbooks/apache/attributes/default.rb
    - update content in file cookbooks/apache/attributes/default.rb from none to e3b0c4
    (diff output suppressed by config)
```


CHEF

Lab: Set the port value as an attribute

cookbooks/apache/attributes/default.rb

```
default['apache']['port'] = 8080
```

Its good practice to include the name of the cookbook in the attribute name – helps trace where the value is set, although this is not enforced

CHEF

Lab: Update the apache::server recipe

cookbooks/apache/recipes/server.rb

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

template '/etc/httpd/conf/httpd.conf' do
  action :create
  source 'httpd.conf.erb'
  notifies :restart, 'service[httpd]'
end

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


CHEF

Lab: Generate the Template file

```
$ cd chef-repo
$ chef generate template cookbooks/apache httpd.conf
```

```
output suppressed by config)
MacBook-Pro-3:chef-repo johnfitzpatrick$ chef generate template cookbooks/apache
httpd.conf.erb
Compiling Cookbooks...
Recipe: code_generator::template
  * directory[cookbooks/apache/templates/default] action create (up to date)
  * template[cookbooks/apache/templates/default/httpd.conf.erb] action create
    - create new file cookbooks/apache/templates/default/httpd.conf.erb
    - update content in file cookbooks/apache/templates/default/httpd.conf.erb
from none to e3b0c4
    (diff output suppressed by config)
```


Lab: Add the 'port' variable to the Template

cookbooks/apache/templates/default/httpd.conf.erb

```
...
MaxSpareThreads     75
ThreadsPerChild     25
MaxRequestsPerChild  0
</IfModule>

Listen 80
Listen <%= node['apache']['port'] %>

LoadModule auth_basic_module modules/mod_auth_basic.so
LoadModule auth_digest_module modules/mod_auth_digest.so
...
```

If you are feeling hardcore, type it
http://bit.ly/1Hdx7E1

Copy entire contents of file, paste into the template, and edit line 27

Lab: Upload the Cookbook

- Upload the apache cookbook to the Chef Server

Lab: Upload the Cookbook

```
$ cd chef-repo/cookbooks/apache
```

Lab: Upload the Cookbook

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'apache' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using apache (0.3.0) from source at .
```

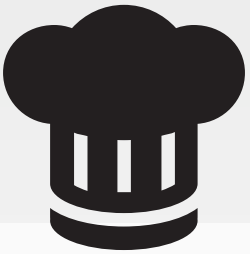

Lab: Upload the Cookbook

```
$ berks upload
```

```
Uploaded apache (0.3.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
```

LAB

Lab: Reconfigure Apache



So we want HAProxy to serve content of port tcp/80, and Apache to serve of tcp/8080

Objective:

- ✓ Reconfigure Apache to serve content of port tcp/8080
- ❑ Reconfigure HAProxy to contact Apache on port tcp/8080

CHEF

Lab: Bump the haproxy cookbook version

cookbooks/haproxy/metadata.rb

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


CHEF

Lab: Configuring haproxy.cfg.erb

cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...

backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

backend app
    balance     roundrobin
   <% @webservers.each_with_index do |web, n| -%>
      server <%= "app#{n}" %> <%= web['cloud']['public_ipv4'] %>:80 weight 1 maxconn 100 check
      server <%= "app#{n}" %> <%= web['cloud']['public_ipv4'] %>:<%= node['apache']['port'] %>
  weight 1 maxconn 100 check
   <% end -%>
```

Add the port attribute as configured in the apache cookbook

DISCUSSION

We've introduced a dependency

We now have the situation where HAProxy cookbook relies on an attribute that is configured in version 0.3.0 of the Apache cookbook

We need to manage this dependency – we do this in the haproxy cookbook `metadata.rb` file

Lab: Bump the cookbook version number

cookbooks/haproxy/metadata.rb

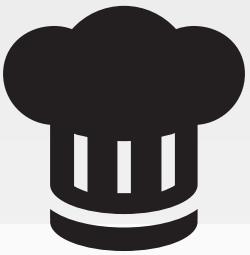
```
name             'haproxy'
maintainer       'The Authors'
maintainer_email 'you@example.com'
license          'all_rights'
description      'Installs/Configures haproxy'
long_description 'Installs/Configures haproxy'
version          '0.2.0'

depends 'apache', '>= 0.3.0'
```


CHEF

LAB

Lab: Reconfigure Apache



So we want HAProxy to serve content of port tcp/80, and Apache to serve of tcp/8080

Objective:

- ✓ Reconfigure Apache to serve content of port tcp/8080
- ✓ Reconfigure HAProxy to contact Apache on port tcp/8080

CHEF

Lab: Upload the Cookbook

- Upload the haproxy cookbook to the Chef Server

Lab: Upload the Cookbook

```
$ cd chef-repo/cookbooks/haproxy
```

DISCUSSION

Configuring Berks

`berks install` retrieves dependencies from the Chef Supermarket by default

We want to override this to use the local apache cookbook instead of the community one

Lab: Edit Berksfile

./Berksfile

```
source "https://supermarket.chef.io"
source 'http://localhost:55555'

metadata

cookbook 'apache', path: '../apache'
```

Lab: Upload the Cookbook

```
$ berks install
```

```
Fetching 'apache' from source at ../apache
Fetching 'haproxy' from source at .
Using haproxy (0.2.0) from source at .
Using apache (0.3.0) from source at ../apache
```


Lab: Upload the Cookbook

```
$ berks upload
```

```
Uploaded apache (0.3.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
Uploaded haproxy (0.2.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
```

SSH Woes

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

Lab: Using knife ssh

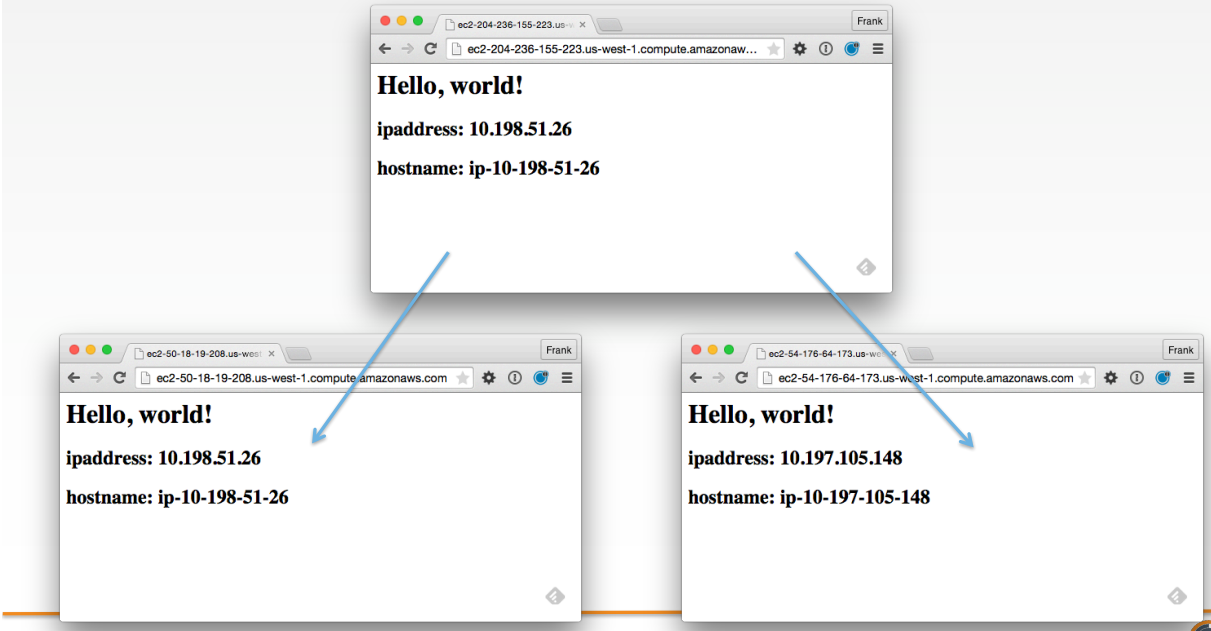
```
$ knife ssh --help
```

```
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
```

Lab: Run chef-client on all nodes

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

```
ec2-54-88-169-195.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-54-84-233-7.compute-1.amazonaws.com   Starting Chef Client, version 12.4.4
ec2-54-88-185-159.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-54-88-169-195.compute-1.amazonaws.com resolving cookbooks for run list: ["haproxy"]
ec2-54-84-233-7.compute-1.amazonaws.com   resolving cookbooks for run list: ["apache"]
ec2-54-88-169-195.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-84-233-7.compute-1.amazonaws.com   Synchronizing Cookbooks:
ec2-54-88-169-195.compute-1.amazonaws.com   - haproxy
ec2-54-84-233-7.compute-1.amazonaws.com     - apache
...
```

DISCUSSION

Discussion

Attributes are like parameters to your cookbook – no hard-coded values in recipes or templates

Can you imagine in complex topologies, where you could have multiple levels of dependencies between cookbooks – berkself handles all of that out the box

CHEF

DISCUSSION

Q&A

What questions can we help you answer?

CHEF

Roles

Giving your Nodes a Role

CHEF

Objectives

After completing this module, you should be able to

- Assign roles to nodes so you can better describe them and configure them in a similar manner
- Set attribute values within roles

CONCEPT

Roles

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

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

CONCEPT

Roles

You assign a role to a node in its run list.

This allows you to configure many similar nodes

Lab: 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 loadbalancer node a "loadbalancer" Role
- ❑ Give our web nodes a "web" Role

Lab: Create the loadbalancer.rb

chef-repo/roles/loadbalancer.rb

```
name 'loadbalancer'
description 'load balancer'
run_list 'recipe[haproxy]'
```


CHEF

Lab: 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)
```


CHEF

Lab: Upload it to the Chef Server & verify

```
$ knife role from file loadbalancer.rb
```

```
Updated Role loadbalancer!
```

```
$ knife role list
```

```
loadbalancer
```


Lab: View Details of the Role

```
$ knife role show loadbalancer
```

```
chef_type:            role
default_attributes:
description:          load balancer
env_run_lists:
json_class:           Chef::Role
name:                 loadbalancer
override_attributes:
run_list:             recipe[haproxy]
```

Lab: Set the loadbalancer role to node3

```
$ knife node run_list set node3 "role[loadbalancer]"
```

```
node3:
  run_list: role[loadbalancer]
```


Lab: Verify the Run List

```
$ knife node show node3
```

```
Node Name:   node3
Environment: _default
FQDN:        ip-172-31-29-217.ec2.internal
IP:          54.88.169.195
Run List:    role[loadbalancer]
Roles:
Recipes:     haproxy::default
Platform:    centos 6.7
Tags:
```

Lab: Converge All the load balancer Nodes

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

```
ec2-54-88-169-195.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-54-88-169-195.compute-1.amazonaws.com resolving cookbooks for run list:
["haproxy"]
ec2-54-88-169-195.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-88-169-195.compute-1.amazonaws.com   - haproxy
ec2-54-88-169-195.compute-1.amazonaws.com   - apache
ec2-54-88-169-195.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-88-169-195.compute-1.amazonaws.com Converging 3 resources
ec2-54-88-169-195.compute-1.amazonaws.com Recipe: haproxy::default
ec2-54-88-169-195.compute-1.amazonaws.com   * yum_package[haproxy] action install
(up to date)
...
```


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 loadbalancer node a "loadbalancer" Role
- ❑ Give our web nodes a "web" Role

Lab: Define a Web Role

- Create a role named 'web' that has the run list 'recipe[apache]'
- Set node1's run list to be "role[web]"
- Set node2's run list to be "role[web]"

Lab: Create the web.rb File

chef-repo/roles/web.rb

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

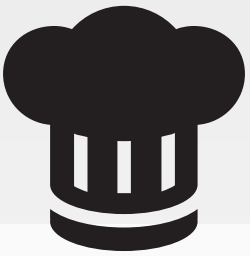
DISCUSSION

Role Attributes

If an attribute is set in a cookbook, and is also set in a role, then the role value wins!

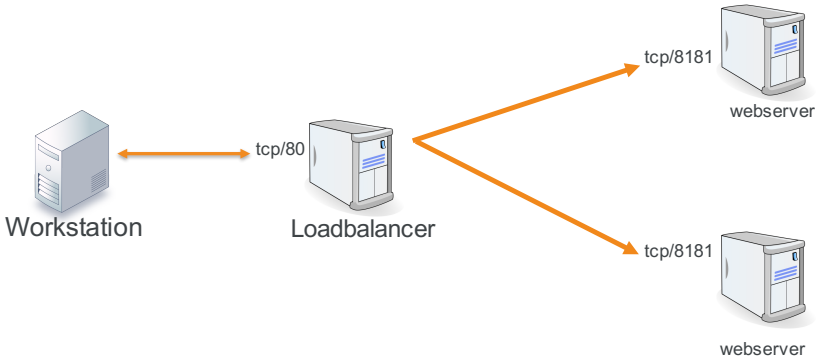
LAB

Lab: Role Attributes



Lets set apache to listen on tcp/8181 instead of tcp/8080 via the role

Our new topology



Lab: Create the web.rb File

chef-repo/roles/web.rb

```
name 'web'
description 'Web Server'
run_list 'recipe[apache]'
default_attributes({
  "apache" => {
        "port" => 8181
      }
})
```

Lab: Upload it to the Chef Server & verify

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

```
Updated Role web!
```

```
$ knife role list
```

```
loadbalancer
web
```


Lab: Verify Specific Information About the Role

```
$ knife role show web
```

```
chef_type:            role
default_attributes:
  apache:
    port: 8181
description:          Web Server
env_run_lists:
json_class:           Chef::Role
name:                 web
override_attributes:
run_list:             recipe[apache]
```

Lab: Set Run List on node1 and node2

```
$ knife node run_list set node1 "role[web]"
```

```
node1:
  run_list: role[web]
```

```
$ knife node run_list set node2 "role[web]"
```

```
node2:
  run_list: role[web]
```

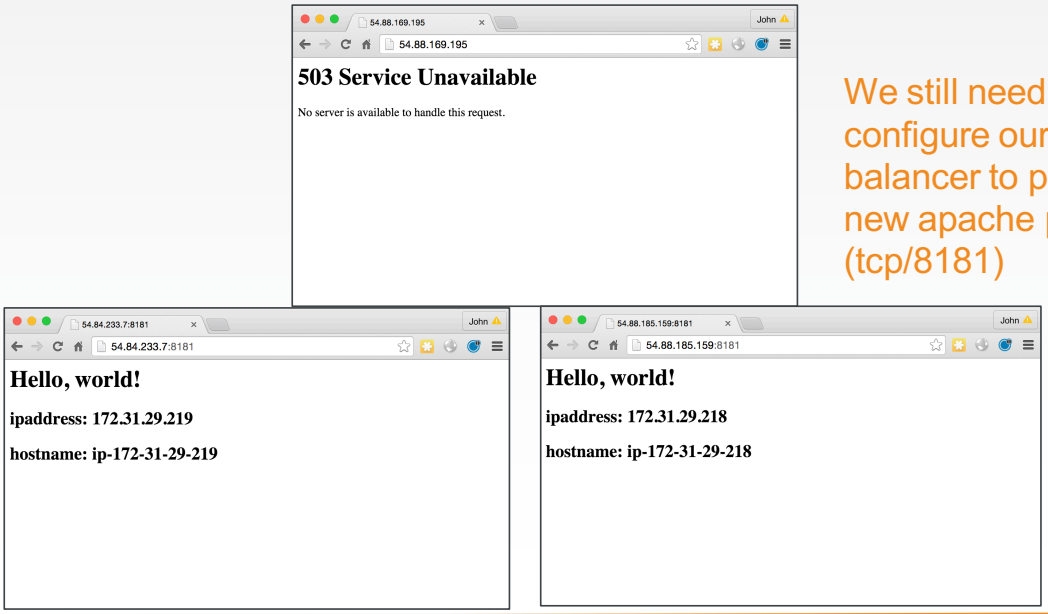

CHEF

Lab: Converge All Web Nodes

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

```
ec2-54-84-233-7.compute-1.amazonaws.com       - restart service service[httpd]
ec2-54-84-233-7.compute-1.amazonaws.com
ec2-54-84-233-7.compute-1.amazonaws.com   Running handlers:
ec2-54-84-233-7.compute-1.amazonaws.com   Running handlers complete
ec2-54-84-233-7.compute-1.amazonaws.com   Chef Client finished, 2/6 resources
updated in 9.758669459 seconds
ec2-54-88-185-159.compute-1.amazonaws.com   * service[httpd] action restart
ec2-54-88-185-159.compute-1.amazonaws.com     - restart service service[httpd]
ec2-54-88-185-159.compute-1.amazonaws.com
ec2-54-88-185-159.compute-1.amazonaws.com Running handlers:
ec2-54-88-185-159.compute-1.amazonaws.com Running handlers complete
ec2-54-88-185-159.compute-1.amazonaws.com Chef Client finished, 2/6 resources
updated in 10.349332394 seconds
```


CHEF



We still need to configure our load balancer to pick up new apache port (tcp/8181)

CHEF

Lab: Create the proxy.rb

chef-repo/roles/loadbalancer.rb

```
name 'loadbalancer'
description 'load balancer'
run_list 'recipe[haproxy]'
default_attributes({
  "apache" => {
        "port" => 8181
      }
})
```


CHEF

Lab: Upload it to the Chef Server

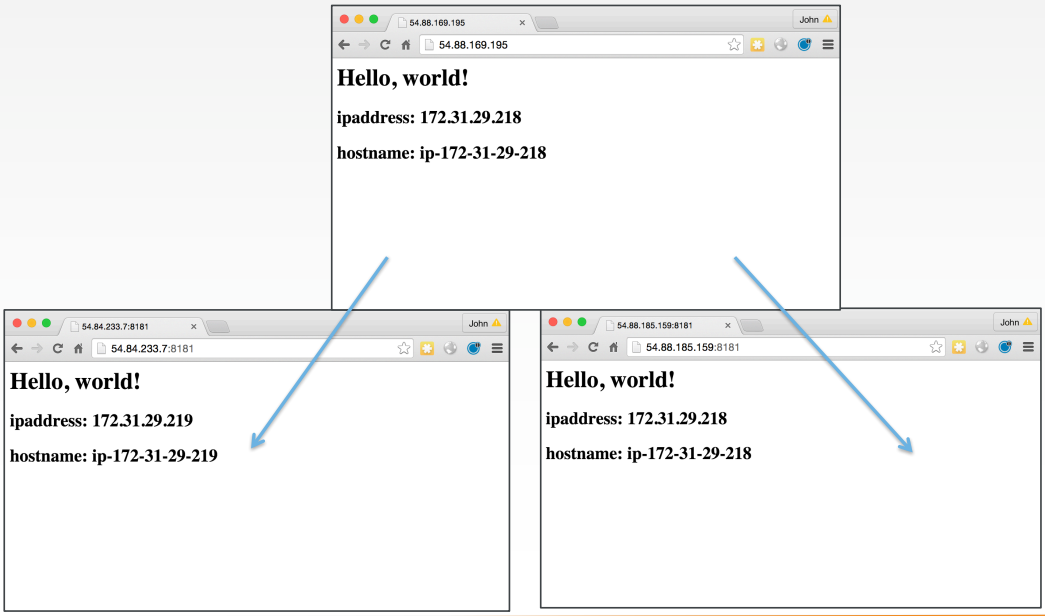
```
$ knife role from file loadbalancer.rb
```

```
Updated Role loadbalancer!
```


Lab: Converge Load Balancer node

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

```
ec2-54-88-169-195.compute-1.amazonaws.com     -    server app1 54.84.233.7:8080 weight 1
maxconn 100 check
ec2-54-88-169-195.compute-1.amazonaws.com     +    server app0 54.88.185.159:8181 weight 1
maxconn 100 check
ec2-54-88-169-195.compute-1.amazonaws.com     +    server app1 54.84.233.7:8181 weight 1
maxconn 100 check
ec2-54-88-169-195.compute-1.amazonaws.com   * service[haproxy] action start (up to date)
ec2-54-88-169-195.compute-1.amazonaws.com   * service[haproxy] action enable (up to date)
ec2-54-88-169-195.compute-1.amazonaws.com   * service[haproxy] action restart
ec2-54-88-169-195.compute-1.amazonaws.com     - restart service service[haproxy]
ec2-54-88-169-195.compute-1.amazonaws.com
ec2-54-88-169-195.compute-1.amazonaws.com Running handlers:
ec2-54-88-169-195.compute-1.amazonaws.com Running handlers complete
ec2-54-88-169-195.compute-1.amazonaws.com Chef Client finished, 2/5 resources updated in
9.831407575 seconds
```



CHEF

Roles for Everyone

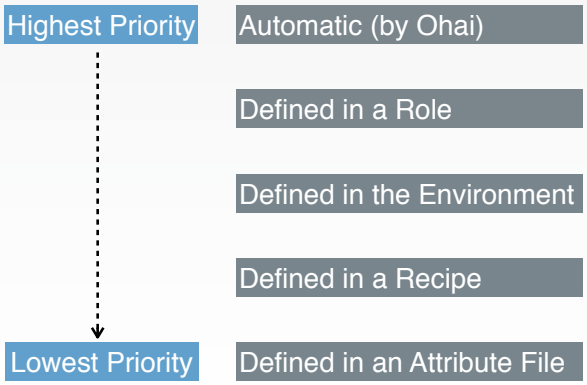
We will give our nodes a role to better describe them and so we can configure them in a similar manner.

Objective:

- ✓ Give our loadbalancer node a "loadbalancer" Role
- ✓ Give our web nodes a "web" Role

CHEF

Default Attribute Precedence



Please note this is a simplified diagram, and the precedence shown can be overridden

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?

Roles are not version controlled – can you think of another way around this?

CHEF

DISCUSSION

Q&A

What questions can we help you answer?

CHEF

©2015 Chef Software Inc.

Community Cookbooks

Introducing the chef-client cookbook & wrapper cookbooks

CHEF

Lesson Objectives

After completing the module, you will be able to

- Find, preview and download cookbooks from the Chef Supermarket
- Use knife to work with the Chef Supermarket site API
- Override community cookbook defaults using wrapper cookbooks
- Upload community cookbooks to Chef Server
- Run chef-client as a service/task

CHEF

Community Cookbooks

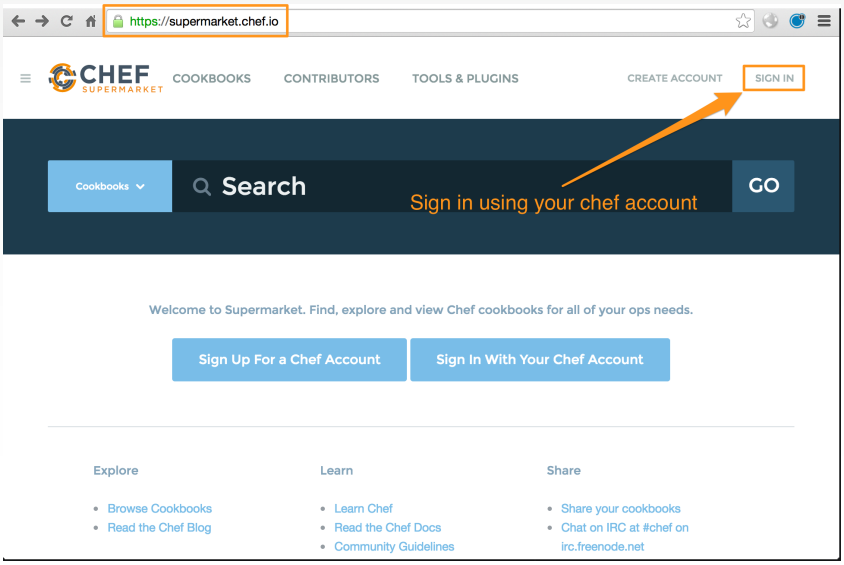
Someone already wrote that cookbook?

Available through the community site called Supermarket/

https://supermarket.chef.io

CHEF

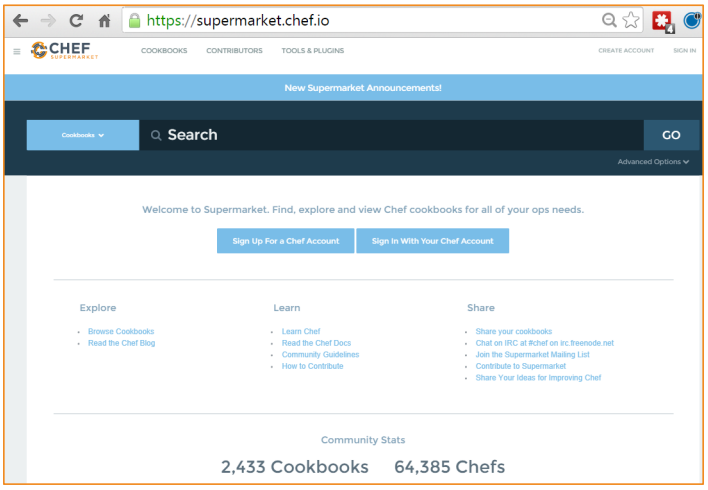
Log into Supermarket



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.



DISCUSSION

Step back: How how is chef-client configured

- How can I run chef-client as a service or Windows task?
- Where can I configure logging?
- How does chef-client know what Chef Server to connect to?
- How does chef-client authenticate with the Chef Server?
- How do I configure where chef-client caches?

Lab: View chef-client config directory

```
$ knife ssh "*:*" -x USER -P PWD "ls -F /etc/chef"
```

```
ec2... client.pem  client.rb  first-boot.json ohai/  validation.pem

ec2... client.pem  client.rb  first-boot.json ohai/  validation.pem

ec2... client.pem  client.rb  first-boot.json ohai/  validation.pem
```

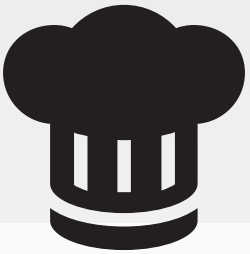

Lab: View chef-client configuration

```
$ knife ssh "*:*" -x USER -P PWD "cat /etc/chef/client.rb"
```

```
ec2...  log_location     STDOUT
ec2...  chef_server_url  "https://api.chef.io/organizations/jaf-291015"
ec2...  validation_client_name "ORG-validator"
ec2...  node_name "node3"
ec2...
ec2...  log_location     STDOUT
ec2...  chef_server_url  "https://api.chef.io/organizations/jaf-291015"
ec2...  validation_client_name "ORG-validator"
ec2...  node_name "node1"
ec2...
ec2...  log_location     STDOUT
ec2...  chef_server_url  "https://api.chef.io/organizations/jaf-291015"
ec2...  validation_client_name "ORG-validator"
ec2...  node_name "node2"
```

LAB

Introducing chef-client cookbook



The Chef-client cookbook allows you to manage and configure chef-client as a service on Linux based nodes, or as a Task on Windows nodes, configure logging, caching, etc

Bootstrapping installs chef-client executable
The chef-client cookbook is used to configure chef-client

CHEF

Introducing chef-client cookbook

We will use the chef-client community cookbook to configured chef-client on each of our nodes to run periodically

Objective:

- ❑ Download chef-client cookbook & upload to Chef Server
- ❑ Configure each of our nodes to run chef-client as a service

Lab: Download the chef-client cookbook

```
$ knife cookbook site download chef-client
```

```
Downloading chef-client from the cookbooks site at version 4.3.1 to /Users/.../chef-repo/chef-client-4.3.1.tar.gz
Cookbook saved: /Users/.../chef-repo/chef-client-4.3.1.tar.gz
```


CHEF

Lab: Extract chef-client cookbook tarball

```
$ tar -zxvf chef-client*.tar.gz -C cookbooks/
```

```
x chef-client/
x chef-client/attributes/
x chef-client/CHANGELOG.md
x chef-client/CONTRIBUTING
x chef-client/LICENSE
x chef-client/metadata.json
x chef-client/metadata.rb
x chef-client/README.md
x chef-client/recipes/
x chef-client/templates/
x chef-client/templates/arch/
x chef-client/templates/default/
x chef-client/templates/windows/
x chef-client/templates/default/debian/
x chef-client/templates/default/redhat/
x chef-client/templates/default/solaris/
x chef-client/templates/arch/conf.d/
x chef-client/templates/arch/rc.d/
x chef-client/recipes/config.rb
x chef-client/recipes/cron.rb
x chef-client/recipes/default.rb
x chef-client/recipes/delete_validation.rb
x chef-client/recipes/service.rb
x chef-client/attributes/default.rb
```


CHEF

DISCUSSION

Examining the chef-client cookbook

We're going to use one recipe on our node from the chef-client cookbook.

`chef-client::service`
(via `chef-client::default`)

CHEF

View the chef-client::default recipe

`cookbooks/chef-client/recipes/default.rb`

```
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

include_recipe "chef-client::service"
```


CHEF

View the chef-client::service recipe

cookbooks/chef-client/recipes/service.rb

- The recipe supports a number of **service** providers and styles.
- It works on a lot of **platforms**.
- Everything is controllable through **attributes**.

```
supported_init_styles = [
  'arch',
  'bluepill',
  'bsd',
  'daemontools',
  'init',
  'launchd',
  'runit',
  'smf',
  'upstart',
  'winsw'
]
init_style = node["chef_client"]["init_style"]
# Services moved to recipes
if supported_init_styles.include? init_style
  include_recipe "chef-client::#{init_style}_service"
else
  log "Could not determine service init style, manual
intervention required to start up the chef-client service."
end
```


CHEF

DISCUSSION

Issue!

We cannot use berks with the cookbook downloaded using knife

It has no metadata.rb!

Stove refuses to upload metadata.rb because it's vulnerable to fairly serious abuse

CHEF

Lab: Download & unpack chef-client cookbook

```
$ cd chef-repo
$ wget https://github.com/opscode-cookbooks/chef-client/archive/master.zip
$ unzip master.zip -d cookbooks
$ mv cookbooks/chef-client-master/ cookbooks/chef-client/
```

So lets download cookbook another way

Or manually create metadata file from here

https://github.com/chef-cookbooks/chef-client/blob/master/metadata.rb

CHEF

Lab: Lets view the dependencies

```
$ cd cookbooks/chef-client
$ cat Berksfile
```

```
source 'https://supermarket.chef.io'

metadata

group :integration do
  cookbook 'runit'
  cookbook 'apt'
  cookbook 'chef-server', '>= 3.0.0'
end
```


CHEF

Lab: Install the Dependencies

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'chef-client' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Installing apt (2.9.2) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
Installing apt-chef (0.2.0) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
Installing chef-ingredient (0.13.1) from https://supermarket.chef.io ([opscode]
https://supermarket.chef.io:443/api/v1)
Using chef-client (4.3.1) from source at .
Installing chef-server (4.1.0) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
Using chef_handler (1.2.0)
Using logrotate (1.9.2)
Installing cron (1.7.0) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
Using packagecloud (0.1.0)
Using runit (1.7.4)
Installing windows (1.38.2) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
Installing yum (3.8.2) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
Installing yum-chef (0.2.1) from https://supermarket.chef.io ([opscode] https://supermarket.chef.io:443/api/v1)
```


CHEF

Whats in the Berksfile.lock?

```
$ cat Berksfile.lock
```

```
DEPENDENCIES
  apt
  chef-client
    path: .
    metadata: true
  chef-server (>= 3.0.0)
  runit

GRAPH
  apt (2.9.2)
  apt-chef (0.2.0)
    apt (>= 0.0.0)
  chef-client (4.3.1)
    cron (>= 1.2.0)
    logrotate (>= 1.2.0)
...
```


CHEF

View the Berkshelf in your home directory

```
$ ls -lt .berkshelf/cookbooks/
```

```
drwxr-xr-x  12 YOU staff  408  2 Dec 15:46 chef_handler-1.1.6
drwxr-xr-x  12 YOU staff  408  2 Dec 15:46 cron-1.6.1
drwxr-xr-x  21 YOU staff  714  2 Dec 15:46 logrotate-1.8.0
drwxr-xr-x  15 YOU staff  510  2 Dec 15:46 windows-1.36.6
drwxr-xr-x   9 YOU staff  306 28 Nov 15:22 build-essential-2.1.3
drwxr-xr-x  25 YOU staff  850  6 Nov 16:27 build-essential-2.1.2
drwxr-xr-x  11 YOU staff  374  6 Nov 16:27 cpu-0.2.0
```


CHEF

Upload cookbooks to Chef Server

```
$ berks upload
```

```
Uploaded apt (2.9.2) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded apt-chef (0.2.2) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded chef-client (4.3.2) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded chef-ingredient (0.11.3) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded chef-server (4.1.0) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded chef_handler (1.2.0) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded cron (1.7.3) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded logrotate (1.9.2) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded packagecloud (0.1.1) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded runit (1.7.6) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded windows (1.39.0) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded yum (3.8.2) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded yum-chef (0.2.2) to: 'https://api.chef.io:443/organizations/ORG'
```


CHEF

View cookbooks on Chef Server

```
$ knife cookbook list
```

```
apache          0.3.0
apt             2.9.2
apt-chef        0.2.0
chef-client     4.3.1
chef_handler    1.2.0
cron            1.7.0
haproxy         0.2.0
logrotate       1.9.2
starter         1.0.0
windows         1.38.2
workstation     0.2.1
...
```


CHEF

Introducing chef-client cookbook

We will use the chef-client community cookbook to configured chef-client on each of our nodes to run periodically

Objective:

- ✓ Download chef-client cookbook & upload to Chef Server
- ❑ Configure each of our nodes to run chef-client as a service

CHEF

DISCUSSION

chef-client as a service

We will add the chef-client default recipe to the roles for each node

CHEF

Lab: Add the recipe to the file web.rb

chef-repo/roles/web.rb

```
name 'web'
description 'Web Server'
run_list 'recipe[chef-client]', 'recipe[apache]'
default_attributes({
  "apache" => {
        "port" => 8181
      }
})
```


CHEF

Lab: Add the recipe to the file proxy.rb

chef-repo/roles/loadbalancer.rb

```
name 'loadbalancer'
description 'loadbalancer'
run_list 'recipe[chef-client]', 'recipe[haproxy]'
default_attributes({
  "apache" => {
        "port" => 8181
      }
})
```


Lab: Upload the roles files

```
$ knife role from file web.rb loadbalancer.rb
```

```
Updated Role web!
Updated Role loadbalancer!
```

Lab: Converge All Nodes

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

```
ec2-514-88-169-195.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-514-88-185-159.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-514-814-233-7.compute-1.amazonaws.com   Starting Chef Client, version 12.4.4
ec2-514-88-185-159.compute-1.amazonaws.com resolving cookbooks for run list:
["apache", "chef-client"]
ec2-514-88-169-195.compute-1.amazonaws.com resolving cookbooks for run list:
["haproxy", "chef-client"]
ec2-514-814-233-7.compute-1.amazonaws.com   resolving cookbooks for run list:
["apache", "chef-client"]
ec2-514-88-185-159.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-514-88-169-195.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-514-814-233-7.compute-1.amazonaws.com   Synchronizing Cookbooks:
...
```


CHEF

Lab: Verify chef-client is running

```
$ knife ssh "*:*" -x USER -P PWD "ps awux | grep chef-
client"
```

```
ec2-514-88-185-159.compute-1.amazonaws.com root     10369  0.0 10.1 266928 61116 ?
Sl   12:54   0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300
...
ec2-514-88-169-195.compute-1.amazonaws.com root     14922  0.0 10.1 267020 61092 ?
Sl   12:54   0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300
...
ec2-514-814-233-7.compute-1.amazonaws.com   root     10202  0.0 10.1 267036 61096 ?
Sl   12:54   0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c
/etc/chef/client.rb -P /var/run/chef/client.pid -i 1800 -s 300
...
```


CHEF

Introducing chef-client cookbook

We will use the chef-client community cookbook to configured chef-client on each of our nodes to run periodically

Objective:

- ✓ Download chef-client cookbook & upload to Chef Server
- ✓ Configure each of our nodes to run chef-client as a service

CHEF

DISCUSSION

Change default settings

Wait!

There has just been a mandate that every node in the infrastructure must run chef-client every 5 minutes

Further this must be the new default setting for anyone downloading the chef-client cookbook

CHEF

DISCUSSION

Well-written cookbooks change behavior based on attributes

- Ideally, you don't have to modify the contents of a cookbook to use it for your specific use case
- Look at the attributes for things you can override through roles to affect behavior of the cookbook
- Of course, well written cookbooks have sane defaults, and a README to describe all this

CHEF

Setting the chef-client run interval

```
...
default['chef_client']['log_file']    = 'client.log'
default['chef_client']['interval']    = '1800'
default['chef_client']['interval']    = '300'
default['chef_client']['splay']       = '300'
default['chef_client']['conf_dir']    = '/etc/chef'
default['chef_client']['bin']         = '/usr/bin/chef-client'
...
```

We need to change the value of the attribute `default['chef_client']['interval']` from to `1800` (seconds) to `300`

CHEF

DISCUSSION

Wrapper Cookbooks

Don't use forked community cookbooks in production, or you will miss out on upstream changes, and will have to rebase

Instead use **wrapper cookbooks** to wrap upstream cookbooks and change their behavior without forking

CHEF

Introducing wrapper cookbooks

We will create a wrapper cookbook to change the default chef-client converge interval to every 5 minutes. If necessary this could still be overwritten via a role

Objective:

- ❑ Create the wrapper cookbook
- ❑ Set the default run interval to 5 minutes
- ❑ Configure each of our nodes to run the default recipe from the wrapper cookbook

CHEF

Lab: Create my-chef-client wrapper cookbook

```
$ cd chef-repo
$ chef generate cookbook cookbooks/my-chef-client
```

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


CHEF

Introducing wrapper cookbooks

We will create a wrapper cookbook to change the default chef-client converge interval to every 5 minutes. If necessary this could still be overwritten via a role

Objective:

- ✓ Create the wrapper cookbook
- ❑ Set the default run interval to 5 minutes
- ❑ Configure each of our nodes to run the default recipe from the wrapper cookbook

CHEF

Lab: Edit my-chef-client default recipe

cookbooks/my-chef-client/recipes/default.rb

```
#
# Cookbook Name:: my-chef-client
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

node.default['chef_client']['interval'] = '300'

include_recipe 'chef-client::default'
```

The recipe just sets the attribute value & calls the recipe `chef-client::default`

CHEF

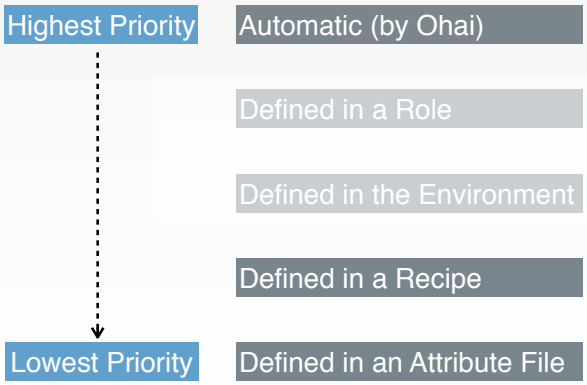
DISCUSSION

Precedence

Attributes set in a recipe trumps the value set in an attributes file!

CHEF

Default Attribute Precedence



Please note this is a simplified diagram, and the precedence shown can be overridden

CHEF

Lab: Update my-chef-client metadata.rb file

cookbooks/my-chef-client/metadata.rb

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

depends   'chef-client'
```


CHEF

Lab: Lets view the dependencies

```
$ cd cookbooks/my-chef-client
```


CHEF

Lab: Install the Dependencies

```
$ cd cookbooks/my-chef-client
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'my-chef-client' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Installing chef-client (4.3.1) from https://supermarket.chef.io ([opscode]
https://supermarket.chef.io:443/api/v1)
Using cron (1.7.0)
Using chef_handler (1.2.0)
Using logrotate (1.9.2)
Using my-chef-client (0.1.0) from source at .
Using windows (1.38.2)
```


CHEF

Upload cookbooks to Chef Server

```
$ berks upload
```

```
Skipping chef-client (4.3.1) (frozen)
Uploaded chef_handler (1.2.0) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded cron (1.7.0) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded logrotate (1.9.2) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded my-chef-client (0.1.0) to: 'https://api.chef.io:443/organizations/ORG'
Uploaded windows (1.38.2) to: 'https://api.chef.io:443/organizations/ORG'
```


Introducing wrapper cookbooks

We will create a wrapper cookbook to change the default chef-client converge interval to every 5 minutes. If necessary this could still be overwritten via a role

Objective:

- ✓ Create the wrapper cookbook
- ✓ Set the default run interval to 5 minutes
- ❑ Configure each of our nodes to run the default recipe from the wrapper cookbook

Lab: Add the recipe to the file web.rb

chef-repo/roles/web.rb

```
name 'web'
description 'Web Server'
run_list 'recipe[chef-client]', 'recipe[apache]'
run_list 'recipe[my-chef-client]', 'recipe[apache]'
default_attributes({
  "apache" => {
        "port" => 8181
      }
})
```


Lab: Add the recipe to the file proxy.rb

chef-repo/roles/loadbalancer.rb

```
name 'loadbalancer'
description 'loadbalancer'
run_list 'recipe[chef-client]', 'recipe[haproxy]'
run_list 'recipe[my-chef-client]', 'recipe[haproxy]'
default_attributes({
  "apache" => {
        "port" => 8181
      }
})
```

Lab: Upload the roles files

```
$ cd chef-repo
$ knife role from file web.rb loadbalancer.rb
```

```
Updated Role web!
Updated Role loadbalancer!
```


Lab: Converge All Nodes

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

```
...
ec2-514-88-169-195.compute-1.amazonaws.com   * service[chef-client] action restart
ec2-514-88-169-195.compute-1.amazonaws.com     - restart service service[chef-
client]
ec2-514-814-233-7.compute-1.amazonaws.com
ec2-514-814-233-7.compute-1.amazonaws.com   Running handlers:
ec2-514-814-233-7.compute-1.amazonaws.com   Running handlers complete
ec2-514-814-233-7.compute-1.amazonaws.com   Chef Client finished, 2/15 resources
updated in 14.158836577 seconds
ec2-514-88-169-195.compute-1.amazonaws.com
ec2-514-88-169-195.compute-1.amazonaws.com Running handlers:
ec2-514-88-169-195.compute-1.amazonaws.com Running handlers complete
ec2-514-88-169-195.compute-1.amazonaws.com Chef Client finished, 2/14 resources
updated in 14.179414969 seconds
```

Lab: Verify chef-client is running

```
$ knife ssh "*:*" -x USER -P PWD "ps awux | grep chef-client|"
```

```
ec2-514-88-169-195.compute-1.amazonaws.com root     15626  0.0 10.1 266936 61120 ?
Sl   13:21   0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c
/etc/chef/client.rb -P /var/run/chef/client.pid -i 300 -s 300
...
ec2-514-88-185-159.compute-1.amazonaws.com root     11209  0.0 10.0 266904 61016 ?
Sl   13:21   0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c
/etc/chef/client.rb -P /var/run/chef/client.pid -i 300 -s 300
...
ec2-514-814-233-7.compute-1.amazonaws.com   root     10906  0.0 10.1 267016 61096 ?
Sl   13:21   0:00 /opt/chefdk/embedded/bin/ruby /usr/bin/chef-client -d -c
/etc/chef/client.rb -P /var/run/chef/client.pid -i 300 -s 300
...
```


CHEF

DISCUSSION

Using Community Cookbooks

Chef Community Cookbooks, can be used as is
But in most cases you will want to use them as a reference as you write your own

Don't use forked community cookbooks in production, or you will miss out on upstream changes, and will have to rebase

CHEF

DISCUSSION

Discussion

What are the benefits and drawbacks of the Chef Supermarket?

Is your team able to leverage community cookbooks? Is the team able to contribute to community cookbooks?

Why do you use a wrapper cookbook? When might you decide to not wrap the cookbook?

DISCUSSION

Q&A

What questions can we help you answer?

- Chef Supermarket
- Wrapper Cookbooks
- chef-client

Environments

Using Environments to Reflect Organization Patterns and Workflow

CHEF

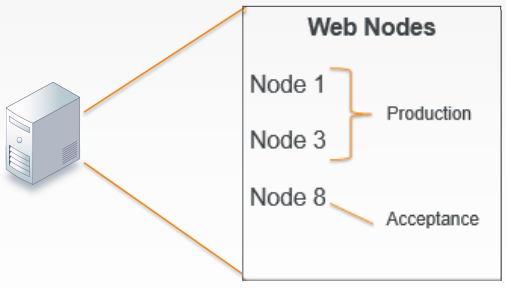
Objectives

After completing this module, you should be able to

- Create an environment
- Deploy a node to an environment
- Update a search query to be more exact

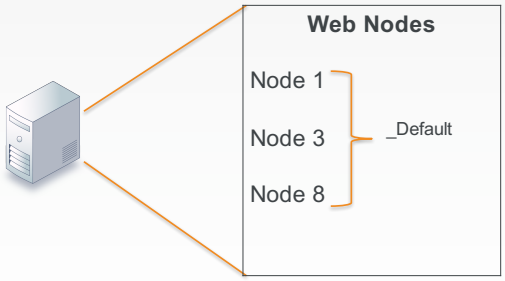
Environments

Environments can define different functions of nodes that live on the same system.



Environments

Every organization or infrastructure starts with the _default environment.



Lab: Production

Let's create a reliable environment for our nodes.

Objective:

- ❑ Deploy Our Site to Production

Lab: Searching All of Our Nodes

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

```
3 items found

Node Name:   node1
Environment: _default
FQDN:        ip-172-31-29-218.ec2.internal
IP:          54.88.185.159
Run List:    role[web]
Roles:       web
Recipes:     apache::default, my-chef-client::default, apache::server, chef-
client::default, chef-client::service, chef-client::init_service
Platform:    centos 6.7
Tags:
```


Lab: Create an environments Directory

```
$ mkdir environments
```

Lab: Create the production.rb

chef-repo/environments/production.rb

```
name 'production'
description 'Where we run production code'

cookbook 'apache', '= 0.3.0'
override_attributes({
  "apache" => {
        "port" => 8181
      }
})
```


CHEF

Lab: Upload the production.rb File

```
$ knife environment from file production.rb
```

```
Updated Environment production
```

```
$ knife environment show production
```

```
chef_type:          environment
cookbook_versions:
  apache: = 0.3.0
default_attributes:
description:        Where we run production code
json_class:         Chef::Environment
name:               production
override_attributes:
  apache:
    port: 8181
```


CHEF

Lab: Set node1 to production environment

```
$ knife node environment set node1 production
node1:
 chef_environment: production
```

```
$ knife node show node1
Node Name:   node1
Environment: production
FQDN:        ip-172-31-8-68.ec2.internal
IP:          54.175.46.24
Run List:    role[web]
Roles:       web
Recipes:     apache, apache::default, apache::server
Platform:    centos 6.7
Tags:
```


Lab: Set More Nodes to Production

- Set node2 & node3's environment to production

Lab: Set node2's Environment to Production

```
$ knife node environment set node2 production
```

```
node2:
  chef_environment: production
```

```
$ knife node environment set node3 production
```

```
node3:
  chef_environment: production
```


Lab: Verify nodes are in Production Environment

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

```
Node Name:   node1
Environment: production
FQDN:        ip-172-31-29-218.ec2.internal
IP:          54.88.185.159
Run List:    role[web]
Roles:       web
Recipes:     apache::default, my-chef-client::default, apache::server, chef-client::default,
chef-client::service, chef-client::init_service
Platform:    centos 6.7
Tags:

Node Name:   node2
Environment: production
FQDN:        ip-172-31-29-217.ec2.internal
```


Lab: Converge All Web Nodes

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

```
ec2-54-88-169-195.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-54-88-185-159.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-54-84-233-7.compute-1.amazonaws.com   Starting Chef Client, version 12.4.4
ec2-54-88-169-195.compute-1.amazonaws.com resolving cookbooks for run list:
["haproxy", "my-chef-client"]
ec2-54-88-185-159.compute-1.amazonaws.com resolving cookbooks for run list:
["apache", "my-chef-client"]
ec2-54-88-169-195.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-88-169-195.compute-1.amazonaws.com   - windows
ec2-54-88-169-195.compute-1.amazonaws.com   - logrotate
ec2-54-88-169-195.compute-1.amazonaws.com   - chef-client
...
```


Production

Let's create a reliable environment for our nodes.

Objective:

- ✓ Deploy our site to Production

DISCUSSION

Acceptance Environment

There is a new mandate requirement for the **Public IP**, the **Public Hostname** and the **Environment** to be displayed on the homepage

This needs to be tested fully before going into production

Lab: Acceptance Environment

- ❑ Bump the Apache cookbook version and update the homepage accordingly
- ❑ Create an environment named "acceptance"
- ❑ Bootstrap a new web node into the acceptance environment using the new apache cookbook
- ❑ Run chef-client on all the nodes

Lab: Bump the cookbook version number

cookbooks/apache/metadata.rb

```
name              'apache'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures apache'
long_description  'Installs/Configures apache'
version           '0.3.1'
```


CHEF

Lab: Update index.html.erb

chef-repo/cookbooks/apache/templates/default/index.html.erb

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>ipaddress: <%= node['cloud']['public_ipv4'] %></h2>
    <h2>hostname: <%= node['cloud']['public_hostname'] %></h2>
    <h2>Environment: <%= "#{node.chef_environment}" %></h2>
</body>
</html>
```


CHEF

Lab: Upload the Cookbook

```
$ cd cookbooks/apache
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'apache' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using apache (0.3.1) from source at .
```

```
$ berks upload
```

```
Uploaded apache (0.3.1) to: 'https://api.opscode.com:443/organizations/ORGNAME'
```


Lab: Acceptance Environment

- ✓ Bump the Apache cookbook version and update the homepage accordingly
- ❑ Create an environment named "acceptance"
- ❑ Bootstrap a new web node into the acceptance environment using the new apache cookbook
- ❑ Run chef-client on all the nodes

Lab: Create a New Environment File

chef-repo/environments/acceptance.rb

```
name 'acceptance'
description 'Where code and applications are tested'

cookbook 'apache', '= 0.3.1'
override_attributes({
  "apache" => {
        "port" => 8181
      }
})
```


Lab: Upload the .rb File

```
$ knife environment from file acceptance.rb
```

```
Updated Environment acceptance
```

```
$ knife environment list
```

```
_default
production
acceptance
```

Lab: Acceptance Environment

- ✓ Bump the Apache cookbook version and update the homepage accordingly
- ✓ Create an environment named "acceptance"
- ❑ Bootstrap a new web node into the acceptance environment using the new apache cookbook
- ❑ Run chef-client on all the nodes

Lab: Bootstrap a new node into the Acceptance Environment

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node4 -r 'role[web]' -E acceptance
```

```
Connecting to 52.23.183.92
52.23.183.92 -----> Existing Chef installation detected
52.23.183.92 Starting first Chef Client run...
52.23.183.92 Starting Chef Client, version 12.4.4
52.23.183.92 resolving cookbooks for run list: ["apache", "my-chef-client"]
52.23.183.92 Synchronizing Cookbooks:
52.23.183.92   - logrotate
52.23.183.92   - chef_handler
52.23.183.92   - chef-client
52.23.183.92   - windows
52.23.183.92   - my-chef-client
...
```

Lab: Verify that the Environment Was Set

```
$ knife node show node4
```

```
Node Name:   node4
Environment: acceptance
FQDN:        ip-172-31-19-80.ec2.internal
IP:          52.23.183.92
Run List:    role[web]
Roles:       web
Recipes:     apache::default, my-chef-client::default, apache::server, chef-client::default, chef-client::service, chef-client::init_service
Platform:    centos 6.7
Tags:
```


Lab: Acceptance Environment

- ✓ Bump the Apache cookbook version and update the homepage accordingly
- ✓ Create an environment named "test"
- ✓ Bootstrap a new web node into the test environment using the new apache cookbook
- ❑ Run chef-client on all the nodes

Lab: Update Load balancing pool

```
$ knife ssh "role:loadbalancer" -x chef -P chef "sudo chef-client"
```

```
ec2-54-88-169-195.compute-1.amazonaws.com Starting Chef Client, version 12.4.4
ec2-54-88-169-195.compute-1.amazonaws.com resolving cookbooks for run list:
["haproxy", "my-chef-client"]
ec2-54-88-169-195.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-88-169-195.compute-1.amazonaws.com   - windows
ec2-54-88-169-195.compute-1.amazonaws.com   - logrotate
ec2-54-88-169-195.compute-1.amazonaws.com   - chef-client
ec2-54-88-169-195.compute-1.amazonaws.com   - chef_handler
ec2-54-88-169-195.compute-1.amazonaws.com   - my-chef-client
ec2-54-88-169-195.compute-1.amazonaws.com   - apache
```

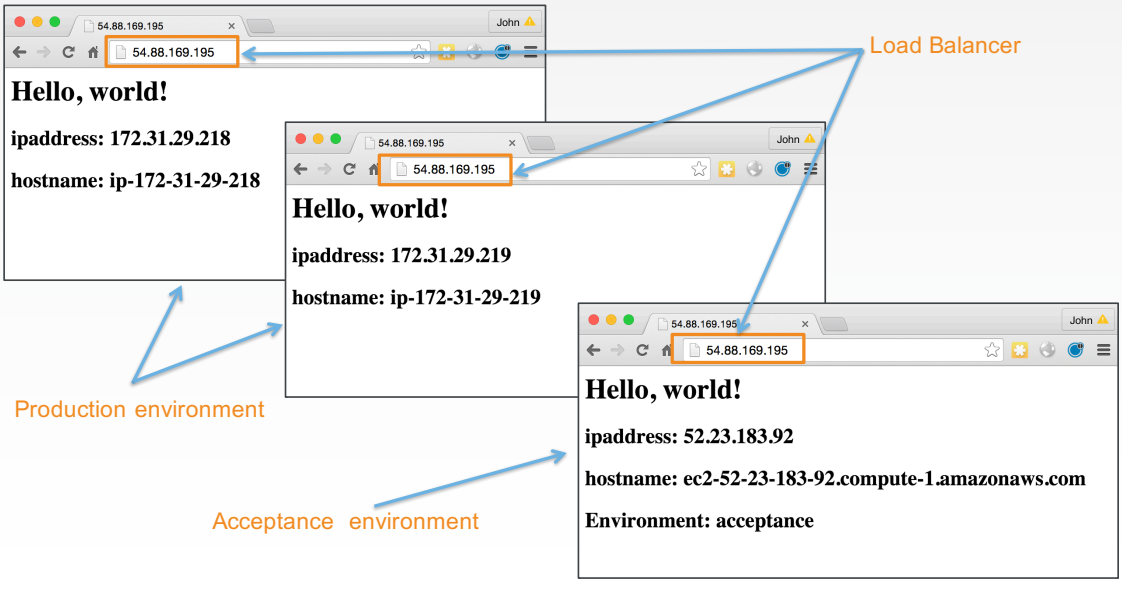

Lab: Acceptance Environment

- ✓ Bump the Apache cookbook version and update the homepage accordingly
- ✓ Create an environment named "acceptance"
- ✓ Bootstrap a new web node into the acceptance environment using the new apache cookbook
- ✓ Run chef-client on all the nodes

DISCUSSION

Separating Environments

Load balancer is distributing load across all environments



Separating Environments

Objective:

- Use Search to separate out the environments

DISCUSSION

Balancing Nodes

Which cookbook handles balancing the requests between web nodes?

Which recipe within that cookbook sets up the request balancing between the two nodes?

Search Criteria

chef-repo/cookbooks/haproxy/recipes/default.rb

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
#...
```


Lab: Modify the myhaproxy default.rb

chef-repo/cookbooks/haproxy/recipes/default.rb

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2015 The Authors, All Rights Reserved.

webservers = search('node', "role:web AND chef_environment:#{node.chef_environment}")

template '/etc/haproxy/haproxy.cfg' do
#...
```

Lab: Run 'berks install'

```
$ cd cookbooks/haproxy
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'apache' from source at ../apache
Fetching 'haproxy' from source at .
Using apache (0.3.1) from source at ../apache
Using apache2 (3.1.0)
Using haproxy (0.3.0) from source at .
```

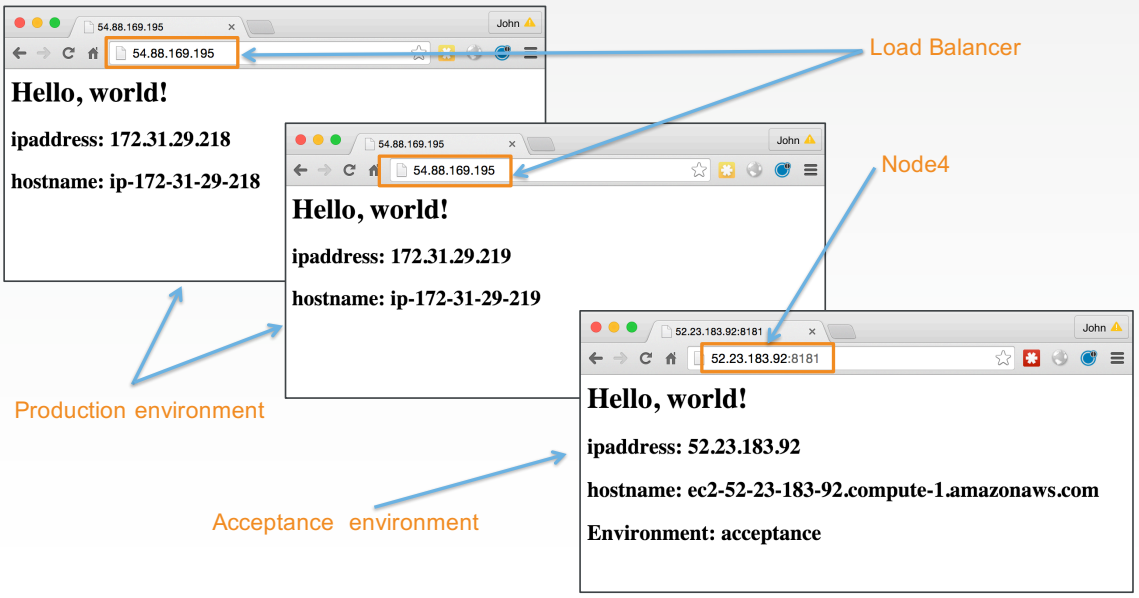
```
$ berks upload
```

```
Skipping apache (0.3.1) (frozen)
Uploaded haproxy (0.3.0) to: 'https://api.chef.io:443/organizations/ORGNAME'
```


Lab: Run chef-client on all nodes

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

```
...
ec2-54-88-169-195.compute-1.amazonaws.com   * directory[/var/log/chef] action create (up to date)
ec2-54-88-169-195.compute-1.amazonaws.com   * directory[/etc/chef] action create (up to date)
ec2-54-88-169-195.compute-1.amazonaws.com   * template[/etc/init.d/chef-client] action create (up to
date)
ec2-54-88-169-195.compute-1.amazonaws.com   * template[/etc/sysconfig/chef-client] action create (up to
date)
ec2-54-88-169-195.compute-1.amazonaws.com   * service[chef-client] action enable (up to date)
ec2-54-88-169-195.compute-1.amazonaws.com   * service[chef-client] action start (up to date)
ec2-54-88-169-195.compute-1.amazonaws.com
ec2-54-88-169-195.compute-1.amazonaws.com Running handlers:
ec2-54-88-169-195.compute-1.amazonaws.com Running handlers complete
ec2-54-88-169-195.compute-1.amazonaws.com Chef Client finished, 0/13 resources updated in 28.735845396
seconds
```



Lab: Separate Environments

✓ Use Search to separate out the environments

DISCUSSION

A Brief Recap

We restricted the production environment to specific cookbook version.

We created an acceptance environment with no cookbook restrictions.

We set specific nodes to each of these environments.

We updated the haproxy's default recipe to include environment search criteria.

And we changed the version number in the myhaproxy metadata.rb file.

DISCUSSION

Discussion

What is the benefit of constraining cookbooks to a particular environment?

When defining search criteria what happens when you **AND** in a query?

What happens when you use an **OR** in a query?

DISCUSSION

Q&A

What questions can we help you answer?

CHEF

©2015 Chef Software Inc.

Further Resources

Other Places to Talk About, Practice, and Learn Chef

CHEF

Going Forward

There are many Chef resources available to you outside this class. During this module we will talk about just a few of those resources.

But…remember what we said at the beginning of this class:

The best way to learn Chef is to use Chef

Practice Chef

First, let's talk about stuff you can read to help you learn Chef.

learnchef.com

Interactive learning for those new to Chef.

Resources You Can Read

A lot of people in the Chef community have written about Chef.

Here are just a few of those resources.

DOCS

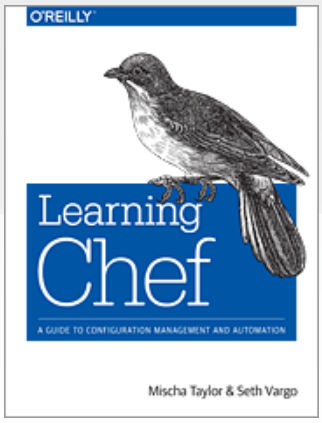
docs.chef.io

Docs are available to you, 24 hours a day, 7 days a week.

Any question you have, you probably will find the answer for on our Docs site.

Learning Chef

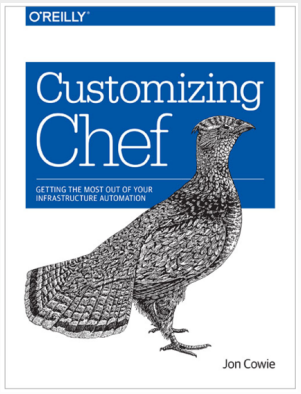
A Guide to Configuration Management and Automation



CHEF

Customizing Chef

Getting the Most Out of Your Infrastructure Automation

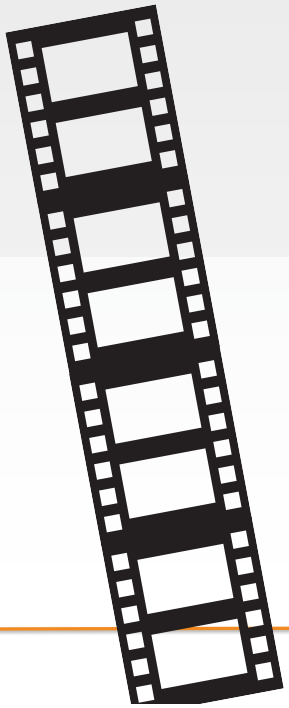


CHEF

YouTube Channel

- ChefConf Talks
- Training Videos

https://www.youtube.com/user/getchef/playlists



CHEF

theshipshow.com

CHEF

foodfightshow.org

The Podcast where DevOps chefs do battle

Chef Developers' IRC Meeting

https://github.com/chef/chef-community-irc-meetings

Chef Community Summit

https://www.chef.io/summit/

CHEF™

Thank You!