

# Chef Fundamentals

[training@getchef.com](mailto:training@getchef.com)

Copyright (C) 2014 Chef Software, Inc.

# Introductions

v2.0.3

# Instructor Introduction

- **Name:** Franklin Webber
- **Current job role:** Developer / Instructor
- **Previous job roles/background:** ARMY SGT, QA, SDET, Developer
- **Experience with Chef/Config Management:** Deployments with Ruby on Rails Applications
- **Favorite Text Editor:** Sublime Text

# Introduce Yourselves

- Name
- Current job role
- Previous job roles/background
- Experience with Chef and/or config management
- Favorite Text Editor
- What are you hoping to get out of the class?

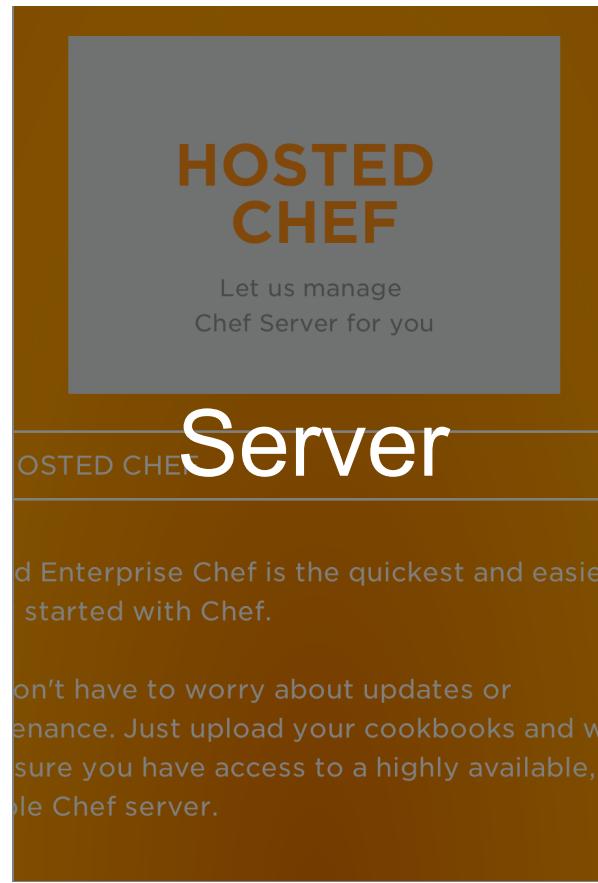
# Agenda

v2.0.3

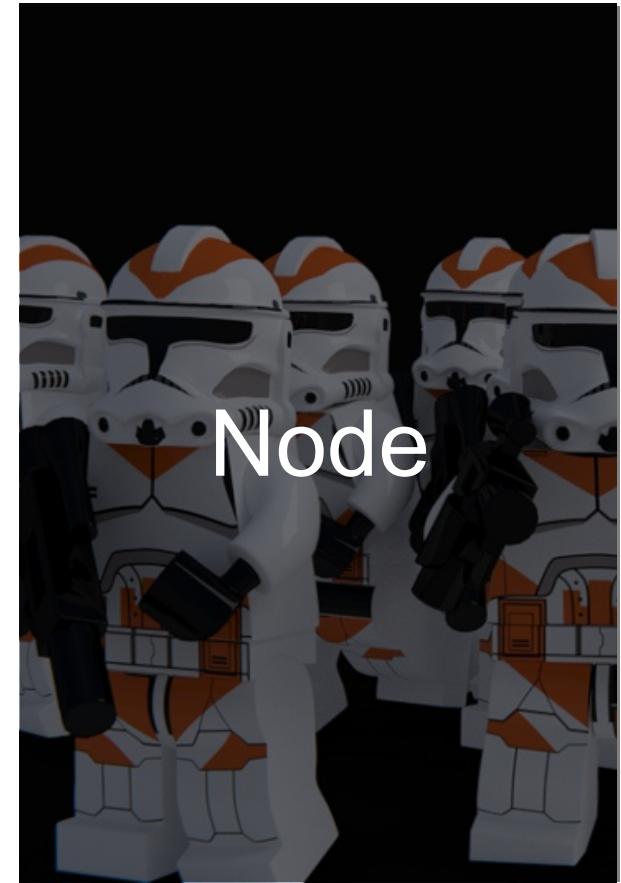
# Preparation



<https://www.flickr.com/photos/rafiqs/14626611136>



6



<https://www.flickr.com/photos/hjmediastudios/7883634326>

# First Course

- Writing a Cookbook
  - Resources
  - Recipes
  - Cookbook files
- Review
  - Dissecting chef-client run
  - Examining the node



[www.flickr.com/photos/shutterhacks/4474421855/](http://www.flickr.com/photos/shutterhacks/4474421855/)

# Second Course

- More with Cookbooks

Attributes

Templates

Dependencies

Template variables

Notifications



# Third Course

- Data Bags
- Search
- Roles
- Environments



# Dessert

- Community Cookbooks



# Apéritif

- Additional Topics

We'll discuss off-topic questions, suggestions, and requests.

...if time permits



[www.flickr.com/photos/shutterhacks/4474421855/](http://www.flickr.com/photos/shutterhacks/4474421855/)

# Full Menu

## Day One

### Preparation

workstation, server, and node

### Write a Cookbook

### Review

### Cookbook

attributes, dependencies,  
templates, notifications

## Day Two

### Cookbook

template variables, notifications,  
and controlling idempotency

### Data Bags

### Search

### Roles

### Environments

### Community Cookbooks

### Additional Topics

# Course Objectives and Style

v2.0.3

# Learning Format

Each section will have an ***objective*** with ***exercises*** that reinforce the topics. When we are done we will ***review*** what we accomplished and answer some ***questions***.

# Breaks!

after each section

OR

every hour

Lunch (~ 11:30AM)

Finished (~ 5:00PM)

# Expectations

I understand if you are late or absent due to obligations. **I will do my best to catch you up** when I can during the breaks or after the class.

# Chef is a Like a Language

The best way to **learn** Chef is to **use** Chef. We will be doing things the **hard way**. We're going to do **a lot** of typing

# Stages of Learning

守破離

Shu

Ha

Ri

“obey”

“detach”

“separate”

# Training is really a discussion

- Ask questions when they come to you
- Ask for help when you need it
- We will troubleshoot and fix bugs on the spot

# Exploration

If everything works for you the first time, take a moment to enjoy your success. But then get back to it and **break it**. Challenge the information that I have shared with you.

# Questions

What questions do you have?

# Workstation Setup

Getting started

v2.0.3

# Lesson Objectives

After completing the lesson, you will be able to

- Login to Enterprise Chef
- View your Organization in Enterprise Chef
- Describe Knife, the Chef command line utility
- Use Knife on your Workstation

# Legend

v2.0.3

## Legend: Do I run that command on my workstation?

This is an example of a command to run on your workstation

```
$ whoami  
i-am-a-workstation
```

This is an example of a command to run on your target node via SSH.

```
user@hostname:~$ whoami  
i-am-a-chef-node
```

# Legend: Example Terminal Command and Output

```
$ ifconfig
```

```
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
        inet 127.0.0.1 netmask 0xff000000
            inet6 ::1 prefixlen 128
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    ether 28:cf:e9:1f:79:a3
    inet6 fe80::2acf:e9ff:fef1f:79a3%en0 prefixlen 64 scopeid 0x4
        inet 10.100.0.84 netmask 0xffffffff broadcast 10.100.0.255
            media: autoselect
            status: active
p2p0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 2304
    ether 0a:cf:e9:1f:79:a3
    media: autoselect
    status: inactive
```

# Legend: Example of editing a file on your workstation



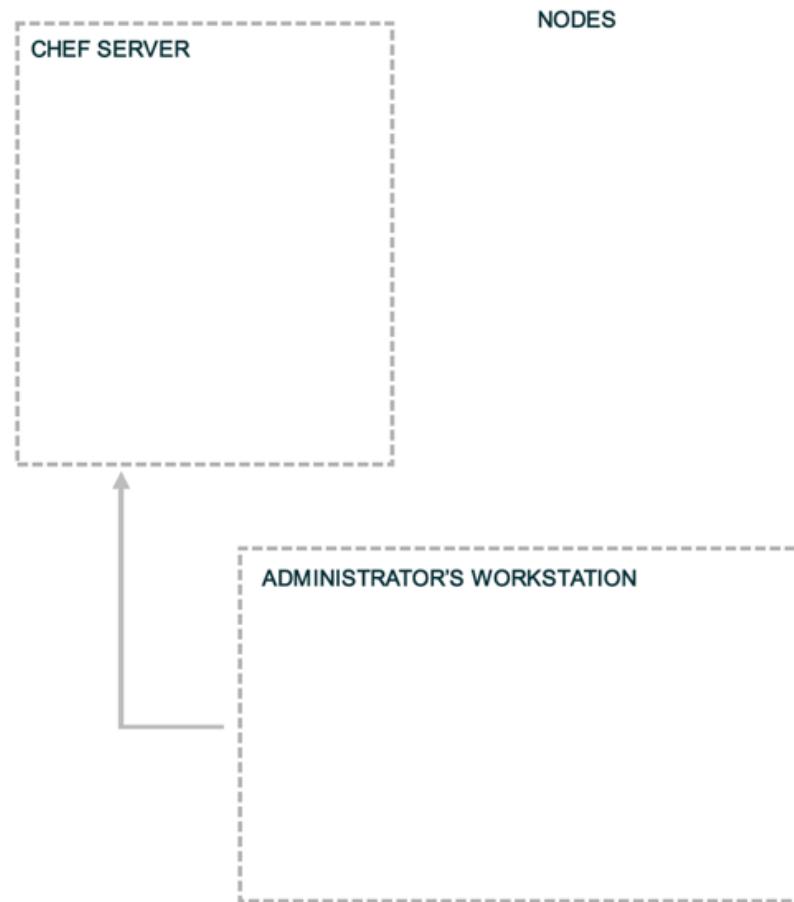
**OPEN IN EDITOR:** ~/hello\_world

Hi!

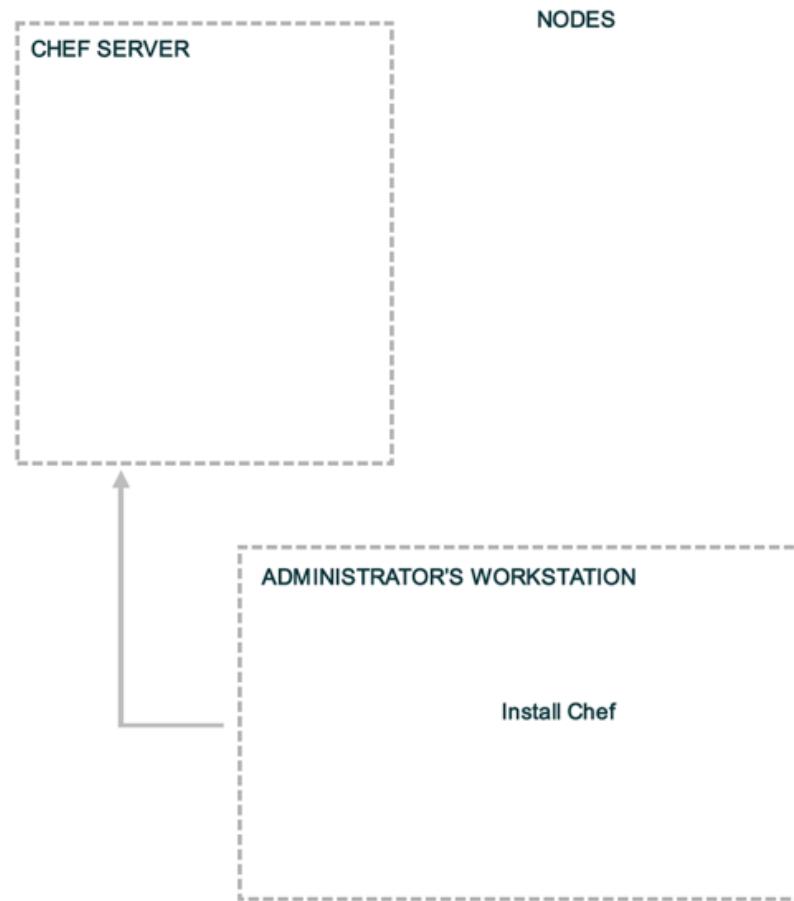
I am a friendly file.

**SAVE FILE!**

# Landscape of a Chef-managed Infrastructure



# Landscape of a Chef-managed Infrastructure



# Install Chef

- Install Chef (if not already installed)
- <http://www.getchef.com/chef/install>

# Managing Your Own Ruby Environment

- If you already have Ruby installed you can just install Chef as a gem

```
$ gem install chef
```

- If you are using a ruby version manager, such as rvm, rbenv, or chruby, then we recommend creating a new gemset or ruby install and installing Chef (and any other tools designed to work with Chef) there

# Install Chef

## Download options

---

[Chef Client](#) [Chef Server](#)

### Installing the Chef Client

Select the kind of system you would like to install the Chef Client on.

The versions listed have been tested and are supported.

[Select an Operating System] ▾

[Select a Version] ▾

[Select an Architecture] ▾

# Install on Mac OSX

## Download options

Chef Client

Chef Server

### Installing the Chef Client

Select the kind of system you would like to install the Chef Client on.

The versions listed have been tested and are supported.

OS X

10.7

x86\_64

### Quick Install Instructions

Open a shell on the target system and run the following command to download and install the latest version of the Chef client:

```
curl -L https://www.opscode.com/chef/install.sh | sudo bash
```

### Downloads

You can install manually by downloading the package below after you have selected a Chef version. For more information about manual installation, [please read the documentation](#).

11.8.2-1

[chef-11.8.2-1.mac\\_os\\_x.10.7.2.sh](#)

33

# Install on Enterprise Linux

## Download options

Chef Client

Chef Server

### Installing the Chef Client

Select the kind of system you would like to install the Chef Client on. The versions listed have been tested and are supported.

Enterprise Linux

6

x86\_64

### Quick Installation Instructions

Open a root shell on the target system and run the following command to download and install the latest version of the Chef client:

```
curl -L https://www.opscode.com/chef/install.sh | bash
```

### Downloads

You can install manually by downloading the package below after you have selected a Chef version. For more information about manual installation, [please read the documentation](#).

11.8.2-1

[chef-11.8.2-1.el6.x86\\_64.rpm](#)

# Workstation Setup - Mac OS X / Linux

```
$ curl -L http://www.opscode.com/chef/install.sh | sudo bash
```

```
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent   Left  Speed
100 14100  100 14100    0      0  8814       0  0:00:01  0:00:01  --:--:--  8812
Downloading Chef  for mac_os_x...
downloading http://www.getchef.com/chef/metadata?v=&prerelease=false&p=mac_os_x&pv=10.7&m=x86_64
  to file /tmp/install.sh.79770/metadata.txt
trying curl...
url http://opscode-omnibus-packages.s3.amazonaws.com/mac_os_x/10.7/x86_64/chef-11.8.2_1.mac_os_x.
10.7.2.sh
md5 af157c6ef941e52f69a9dd6a3b57f597
sha256 c003c0951d80245b1f02c4588f9157a55e7b94a00dd6ac163aed8ef2e854619a
downloaded metadata file looks valid...
downloading http://opscode-omnibus-packages.s3.amazonaws.com/mac_os_x/10.7/x86_64/
chef-11.8.2_1.mac_os_x.10.7.2.sh
  to file /tmp/install.sh.79770/chef--mac_os_x-10.7-x86_64.sh
trying curl...
.
.
.
Thank you for installing Chef!
```

# Workstation Setup - Windows

- Windows
- 2008 (Windows 7) or 2012 (Windows 8)
- i686 (32-bit) or x86\_64 (64-bit)
- 11.8.2

## Download options

Chef Client

Chef Server

### Installing the Chef Client

Select the kind of system you would like to install the Chef Client on. The versions listed have been tested and are supported.

Windows

2008r2

x86\_64

### Downloads

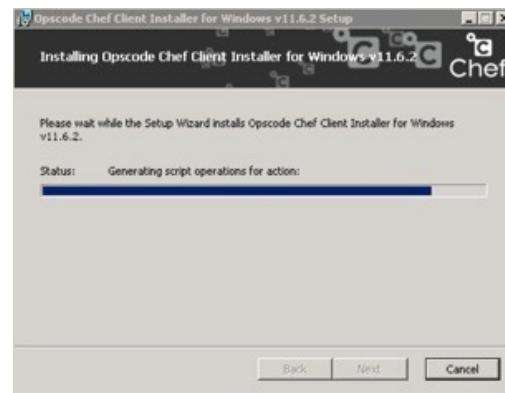
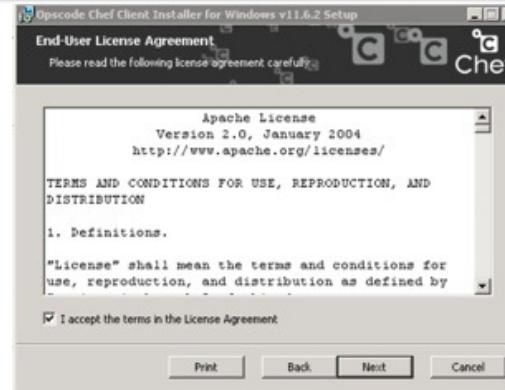
You can install manually by downloading the p information about manual installation, [please r](#)

11.8.2-1

[chef-client-11.8.2-1.windows.msi](#)

Download and install this file

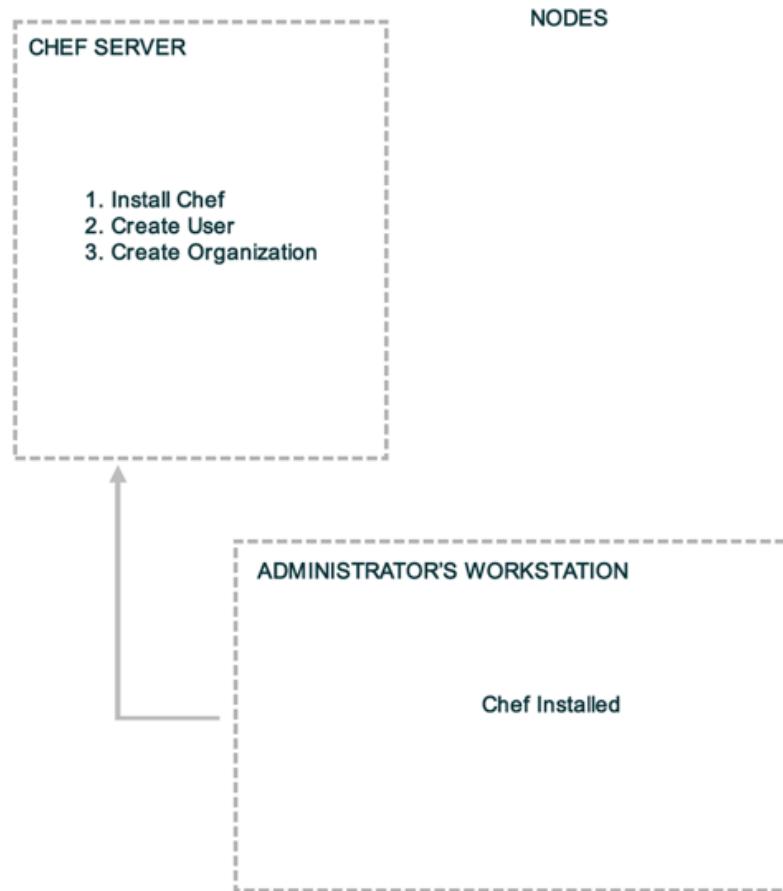
# Install on Windows



# What just happened?

- Chef and all of its dependencies installed via an operating system-specific package ("omnibus installer")
- Installation includes
  - The Ruby language - used by Chef
  - knife - Command line tool for administrators
  - chef-client - Client application
  - ohai - System profiler
  - ...and more

# Landscape of a Chef-managed Infrastructure



# Your Chef Server for this class...

- Hosted Enterprise Chef <http://www.getchef.com>

The image shows two screenshots of the GetChef website. The top screenshot is a navigation bar with links: CUSTOMER LOGIN, SIGN UP, ACCOUNT MANAGEMENT, Products, Solutions, Learn Chef, About, Community, and a prominent orange 'Get Chef' button. An orange oval highlights the 'Get Chef' button. The bottom screenshot is a landing page with the text 'You're almost ready to start your free trial of Enterprise Chef'. It says 'All you need to do to get started is choose how to deploy Chef:' and features two large buttons: 'Hosted Chef' (which is highlighted with an orange oval and has an arrow pointing from the top screenshot) and 'On Premises'.

CUSTOMER LOGIN SIGN UP ACCOUNT MANAGEMENT

Products Solutions Learn Chef About Community Get Chef

You're almost ready to start your free trial of Enterprise Chef

All you need to do to get started is choose how to deploy Chef:

Hosted Chef

On Premises

Install and manage your own Chef Server

If you need to run Chef behind a firewall, set up and manage your own Chef server, install Enterprise Chef on premises.

# Create new account

- Sign up for a new account
- Chef Organization
  - provides multi-tenancy
  - name must be globally unique

## Start your free trial of Enterprise Chef

You're one step away from access to all the power and flexibility of Chef, hosted and supported by Opscode. Get ready to automate your infrastructure to accelerate your time to market, manage complexity, and safeguard your systems. Just complete the form to get started.

Full Name

Username

Email

Password

Company  (Optional)

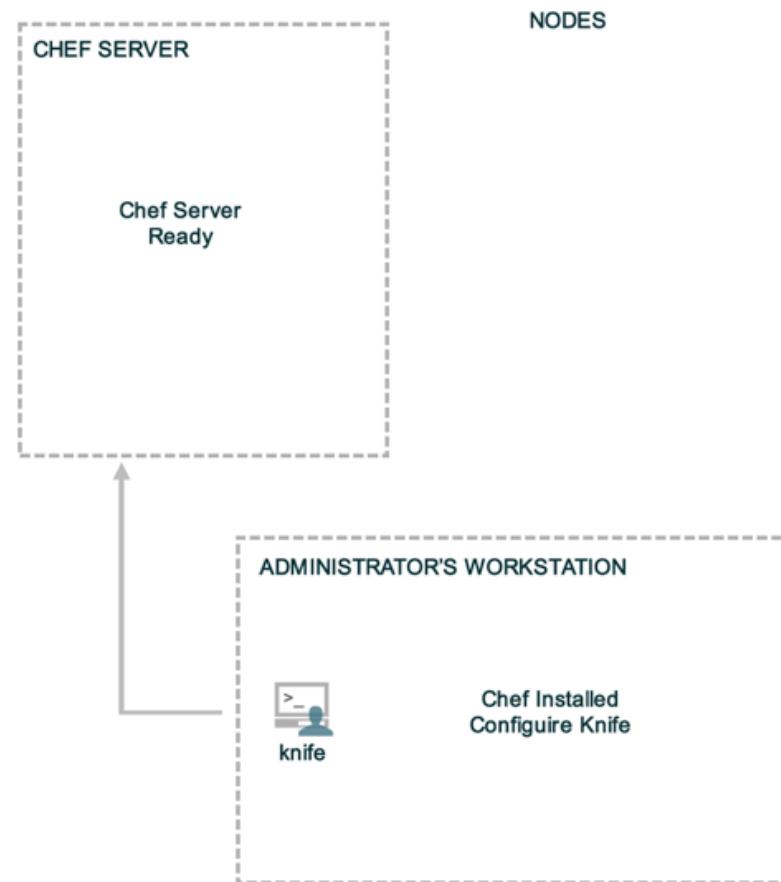
Chef Organization

Organization is the name of your instance of Enterprise Chef.

I agree to the [Terms of Service](#) and the [Master License and Services Agreement](#).

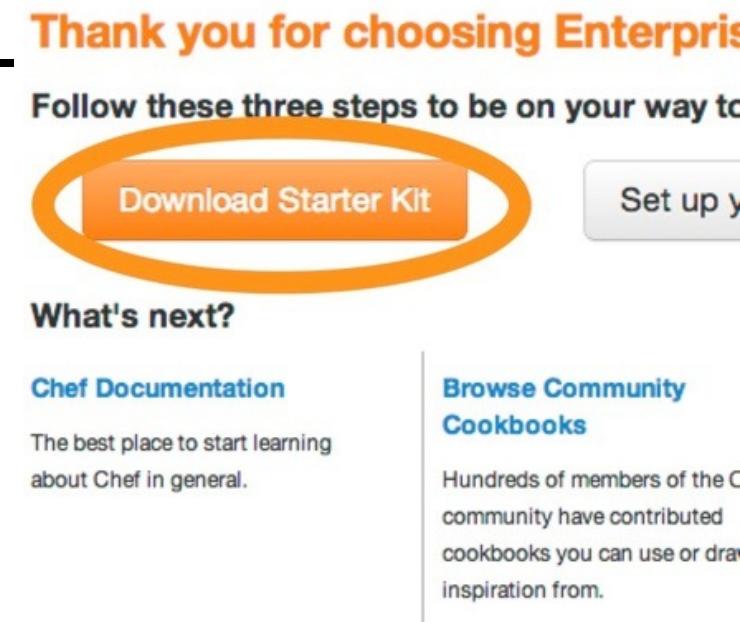
**Get Started**

# Landscape of a Chef-managed Infrastructure



# Download "Starter Kit" for your Org

- You get a .zip file from clicking this
- Unzip the zipfile - you'll get a "chef-repo"
- Put the "chef-repo" somewhere, e.g.:
  - C:\Users\you\chef-repo (Win)
  - /Users/you/chef-repo (Mac)
  - /home/you/chef-repo (Linux)



# Tour the chef-repo

```
$ cd chef-repo
```

```
[ ~/chef-repo ] $
```

# A quick tour of the chef-repo

- Every infrastructure managed with Chef has a Chef Repository ("chef-repo")
- Type all commands in this class from the chef-repo directory
- Let's see what's inside the chef-repo...

# Tour the chef-repo

```
$ ls -al
```

```
total 40
drwxr-xr-x@ 11 opscode  opscode   374 Dec 15 09:42 .
drwxr-xr-x+ 92 opscode  opscode  3128 Dec 15 09:43 ..
drwxr-xr-x@  3 opscode  opscode   102 Dec 15 2013 .berkshelf
drwxr-xr-x@  5 opscode  opscode   170 Dec 15 2013 .chef
-rw-r--r--@  1 opscode  opscode   495 Dec 15 2013 .gitignore
-rw-r--r--@  1 opscode  opscode  1433 Dec 15 2013 Berksfile
-rw-r--r--@  1 opscode  opscode  2416 Dec 15 2013 README.md
-rw-r--r--@  1 opscode  opscode  3567 Dec 15 2013 Vagrantfile
-rw-r--r--@  1 opscode  opscode   588 Dec 15 2013 chefignore
drwxr-xr-x@  3 opscode  opscode   102 Dec 15 2013 cookbooks
drwxr-xr-x@  3 opscode  opscode   102 Dec 15 2013 roles
```

# What's inside the .chef directory?

```
$ ls .chef
```

```
ORGNAME-validator.pem  
USERNAME.pem  
knife.rb
```

# What's inside the .chef directory?

- `knife.rb` is the configuration file for Knife.
- The other two files are certificates for authentication with the Chef Server
  - We'll talk more about that later.

# Knife is the command-line tool for Chef

- Knife provides an API interface between a local Chef repository and the Chef Server, and lets you manage:
  - Nodes
  - Cookbooks and recipes
  - Roles
  - Stores of JSON data (data bags), including encrypted data
  - Environments
  - Cloud resources, including provisioning
  - The installation of Chef on management workstations
  - Searching of indexed data on the Chef Server

# knife.rb



**OPEN IN EDITOR:** chef-repo/.chef/knife.rb

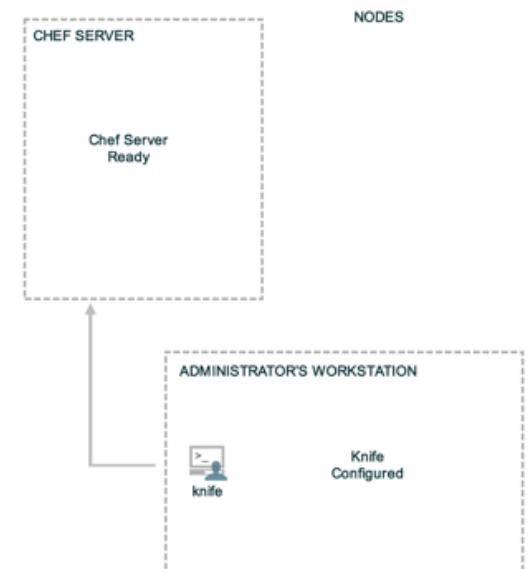
```
current_dir = File.dirname(__FILE__)
log_level           :info
log_location        STDOUT
node_name           "USERNAME"
client_key          "#{current_dir}/USERNAME.pem"
validation_client_name "ORGNAME-validator"
validation_key       "#{current_dir}/ORGNAME-validator.pem"
chef_server_url     "https://api.opscode.com/organizations/ORGNAME"
cache_type           'BasicFile'
cache_options( :path => "#{ENV['HOME']}/.chef/checksums" )
cookbook_path        ["#{current_dir}/../cookbooks"]
```

# Verify Knife

```
$ knife --version  
Chef: 11.8.2
```

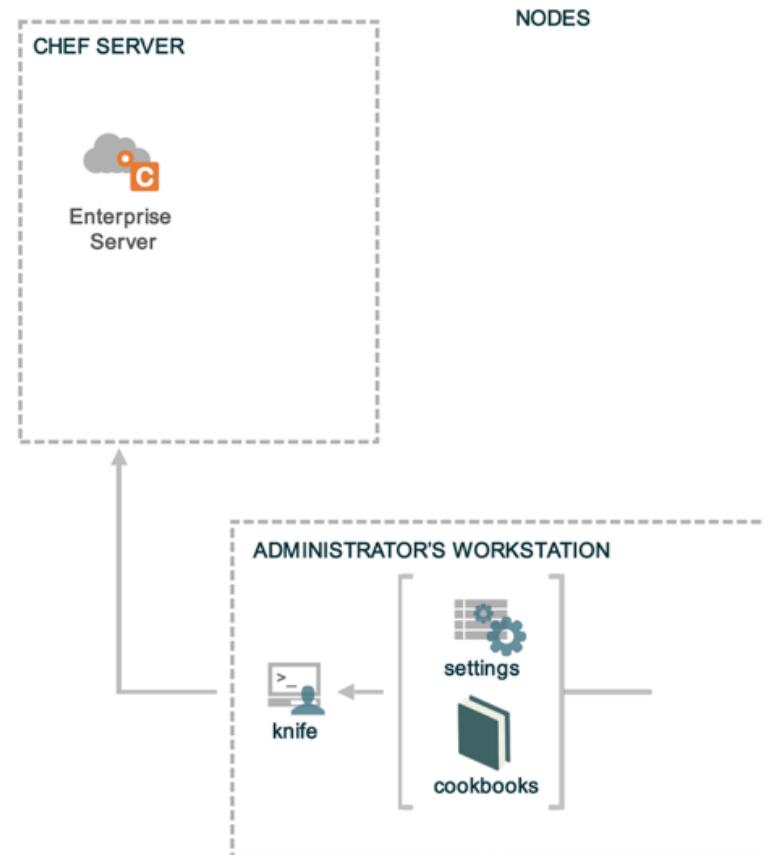
```
$ knife client list  
ORGNAME-validator
```

- Your version may be different, that's ok!



# knife client list

1. Reads the `chef_server_url` from `knife.rb`
2. Invokes HTTP GET to `#{{chef_server_url}}/clients`
3. Displays the result



# Exercise: Run ‘knife help list’

```
$ knife help list
```

```
Available help topics are:
```

```
bootstrap
chef-shell
client
configure
cookbook
cookbook-site
data-bag
delete
deps
diff
download
edit
environment
exec
list
```

## Best Practice: Use a text editor with a project drawer, or equivalent

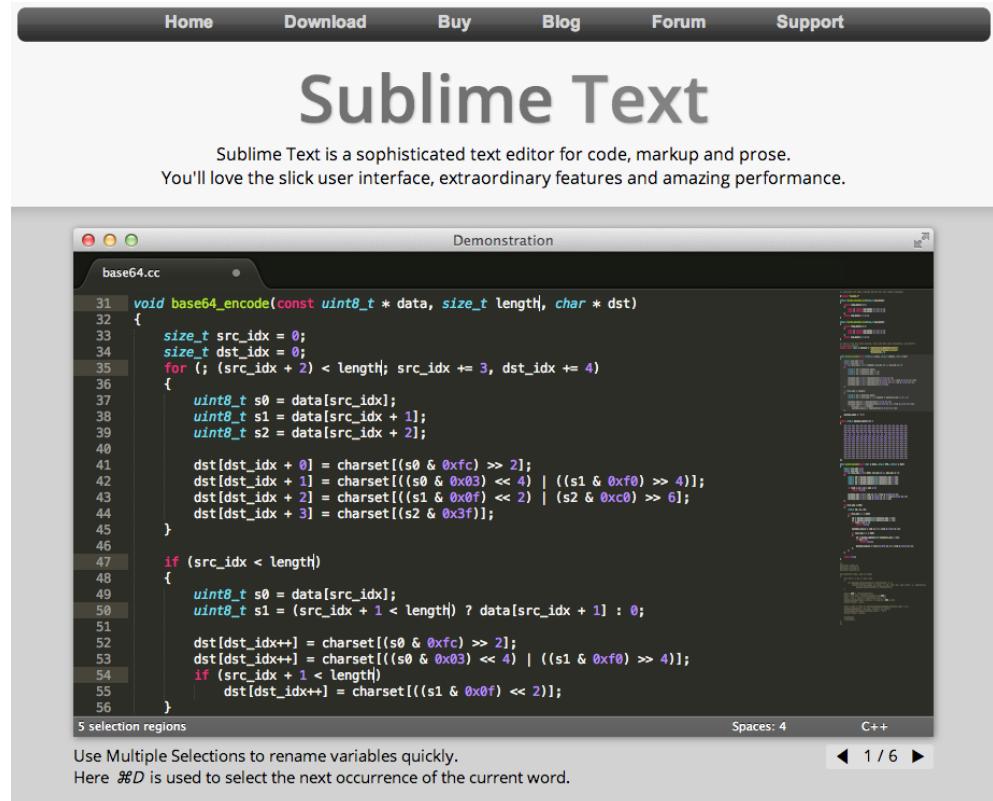
- Chef is about Infrastructure as Code
- People who code for a living use text editors that are designed for the task
  - Vim, Emacs, Sublime Text, Notepad++, etc.

# Benefits of a good text editor

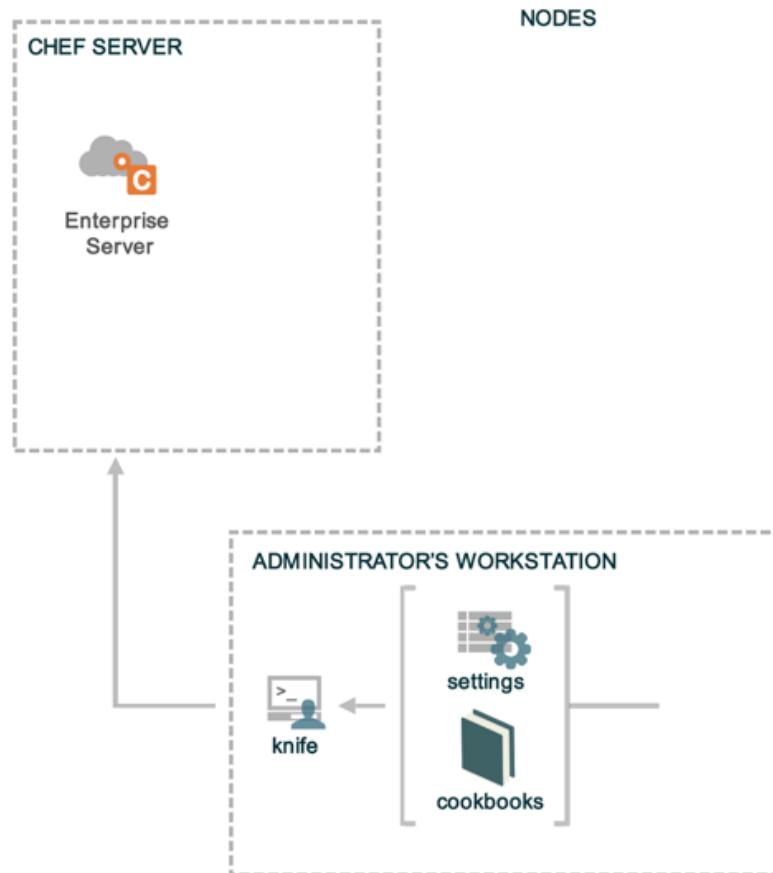
- A good editor will
  - Show line numbers
  - Highlight syntax
  - Autocomplete commands
  - Allow you to open multiple files

If you do not have a preferred text editor on your workstation already...

- Download Sublime Text
- Free trial, not time bound
- Works on every platform
- [sublimetext.com](http://sublimetext.com)



# Checkpoint



# Review Questions

- What is the chef-repo?
- What is knife?
- What is name of the knife configuration file?
- Where is the knife configuration file located?

# Organization Setup

Setup an Organization

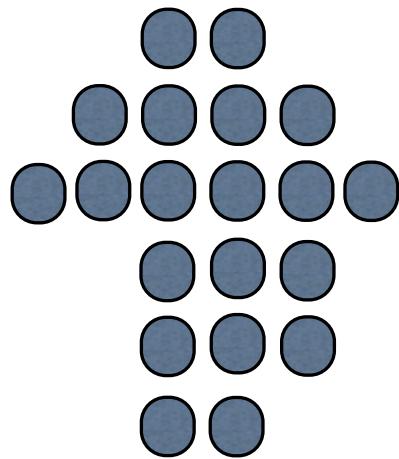
v2.0.3

# Lesson Objectives

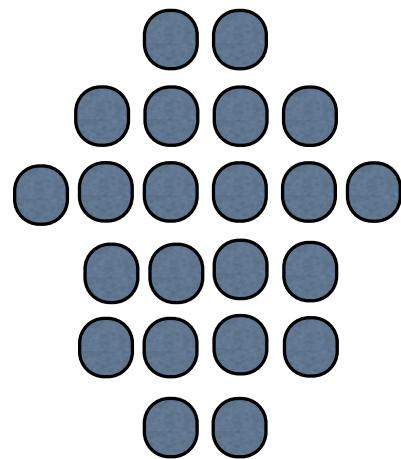
- After completing the lesson, you will be able to
  - Explain the purpose of Organizations
  - Manage your Chef Organization
  - Invite users into your organization
  - Accept invitations into other organizations

# Organizations

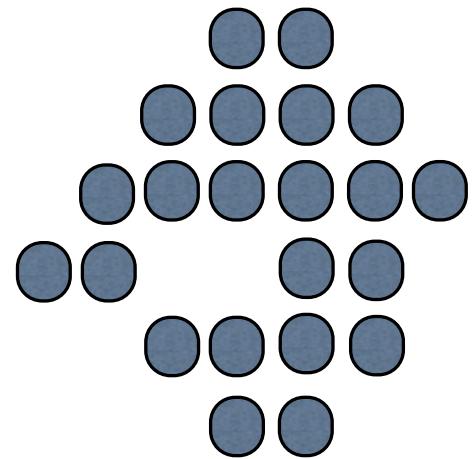
My Infrastructure



Your Infrastructure



Their Infrastructure

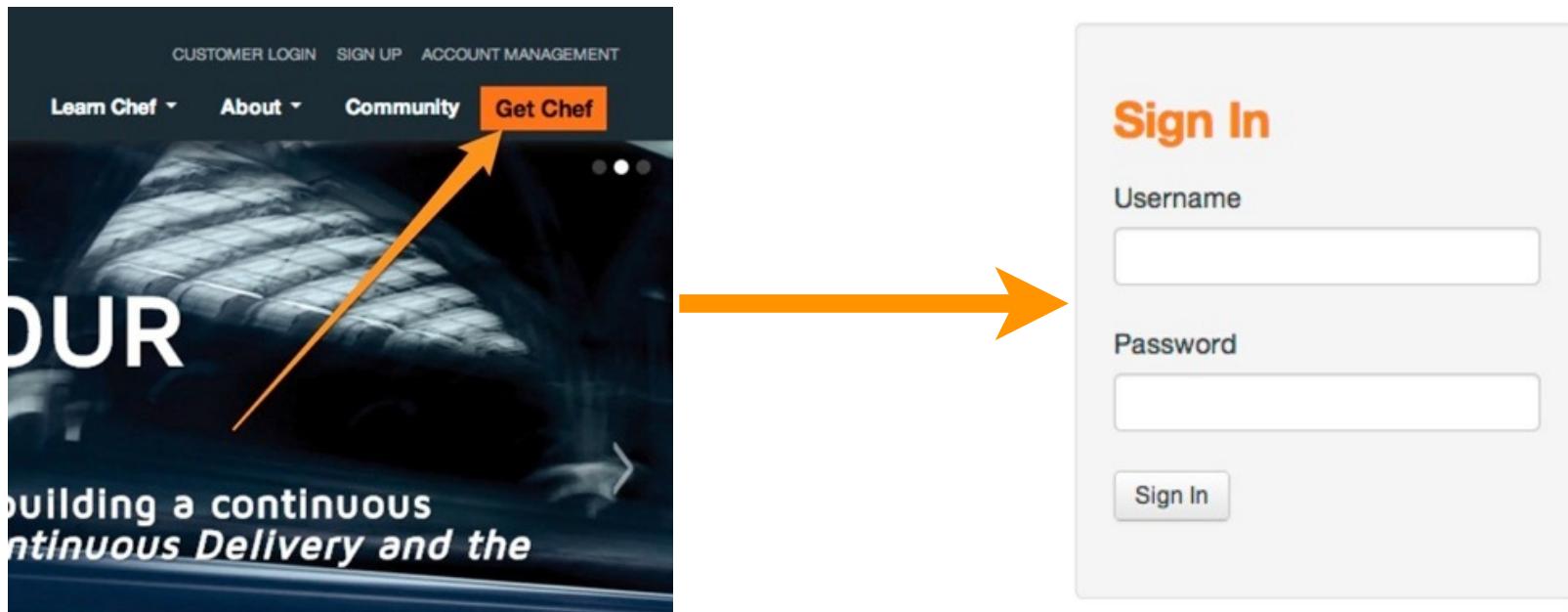


# Organizations

- Provide multi-tenancy in Enterprise Chef
- Nothing is shared between Organizations - they're completely independent
- May represent different
  - Companies
  - Business Units
  - Departments

# Manage Organizations

- Login to your Hosted Enterprise Chef



# Organizations

The screenshot shows the CHEF MANAGE web interface. The top navigation bar includes links for Nodes, Policies, and Administrative, with the Administrative link being highlighted by an orange box. On the left, a sidebar menu lists options: > Organizations (which is also highlighted with an orange box), Create, Reset Validation Key, Generate Knife Config, Leave Organization, and Starter Kit. Below these are sections for Users, Groups, and Global Permissions. The main content area is titled "Showing All Organizations" and displays a single entry: "nhcompany".

CHEF  
MANAGE

> Organizations

Create

Reset Validation Key

Generate Knife Config

Leave Organization

Starter Kit

Users

Groups

Global Permissions

Showing All Organizations

Organization

nhcompany

# Manage Organizations

- Reset Validation Key
- Generate Knife Config
- Leave Organization
- Download the 'Starter Kit'

The screenshot shows the Chef Manage web interface. At the top, there's a navigation bar with tabs for 'Nodes', 'Policies', and 'Administrative'. On the far right of the top bar, it says 'Organization nhcompany' and 'Logged in as Nathan Harvey'. Below the top bar, there's a sidebar with a 'CHEF MANAGE' logo and links for 'Organizations' (which is currently selected), 'Create', 'Reset Validation Key', 'Generate Knife Config', 'Leave Organization', and 'Starter Kit'. The main content area is titled 'Showing All Organizations' and lists one organization: 'nhcompany'. To the right of the organization list is a 'Actions' button containing four options: 'Reset Validation Key', 'Generate Knife Config', 'Leave Organization', and 'Starter Kit'. A gear icon is also visible at the bottom right of the actions area.

# Manage Organizations

- Each Organization may have multiple Users
- Manage an Organization's Users via the Enterprise Server interface
  - <http://manage.opscode.com>

# Exercise: Invite users into your org

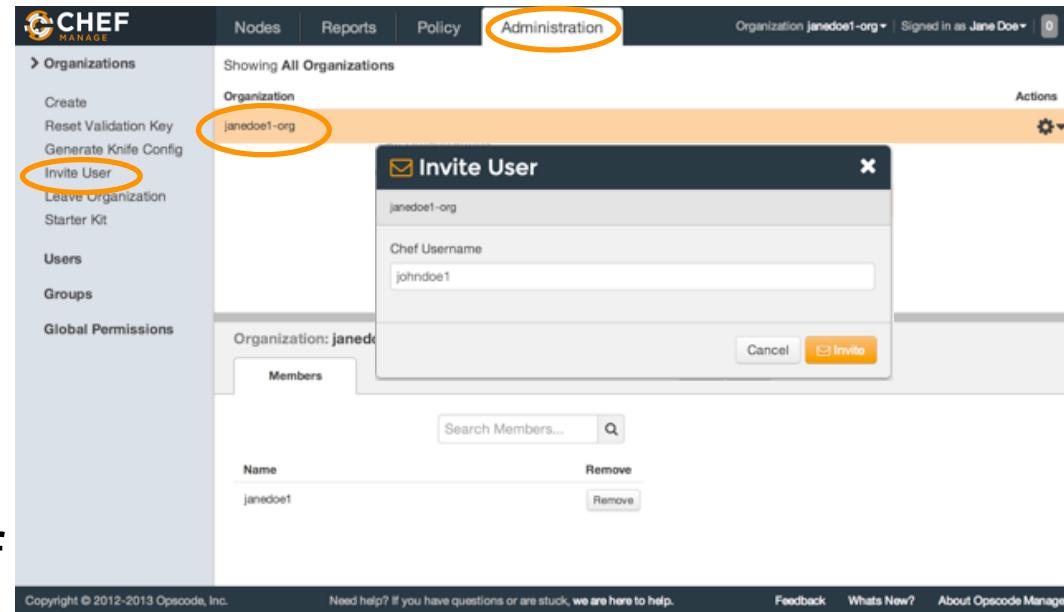
- **The Problem:** You need assistance from a colleague on one of your chef projects
- **Proposed Solution:** Invite the colleague into your organization so they can log into the chef server with **their own** credentials, but can see **your** organization

# Exercise: Invite Users into your Org

- For the Lab Exercise you can team up with one other person in the class, then you will
  1. Invite your classmate into your organization
  2. Accept your classmate's invite into their org
  3. Look around your classmate's organization while they look around yours
  4. Finally remove your classmate's access from your account

# Exercise: Invite a classmate into your org

- Navigate to <http://manage.opscode.com>
- Click the "Administration" tab,
- Select the appropriate Organization
- Click "Invite User" from the left menu
- Enter your classmate's 'Chef Username' and click Invite

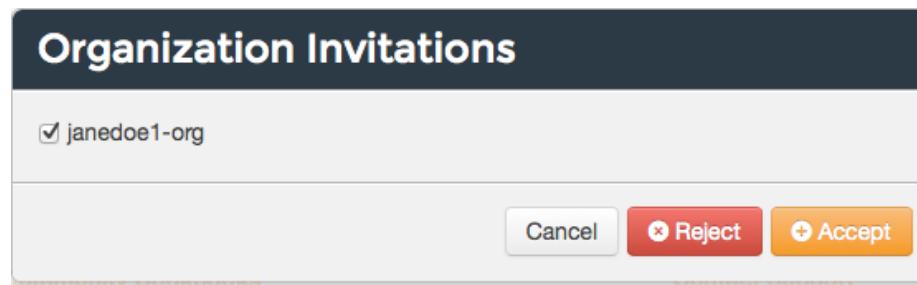


# Exercise: Accept an invite

- You will get a notification once your classmate has invited you into their account

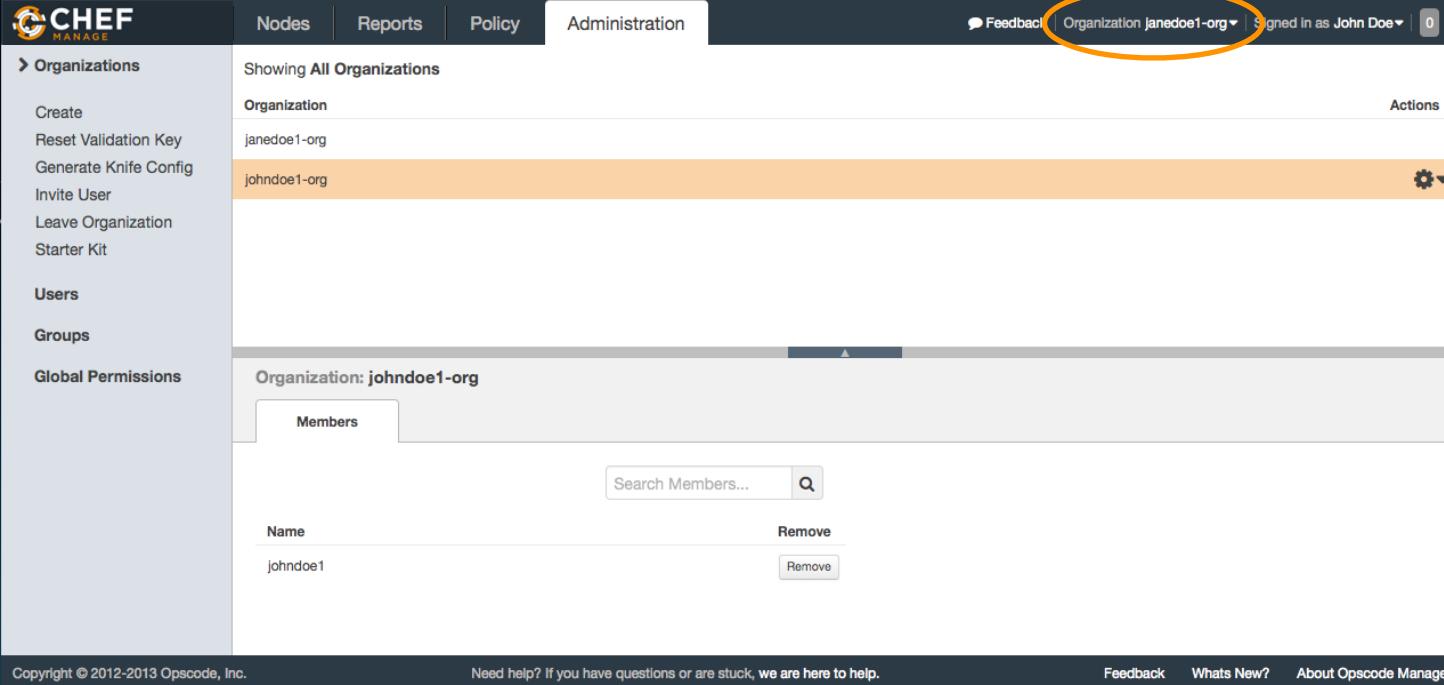


- Click the notification, select the Organization and click 'Accept'



# Exercise: View classmate's org

- Select your classmate's organization from the drop down list and peruse their org



The screenshot shows the Opscode Manage interface. On the left is a sidebar with links for 'Organizations', 'Create', 'Reset Validation Key', 'Generate Knife Config', 'Invite User', 'Leave Organization', and 'Starter Kit'. Below that are 'Users' and 'Groups' sections. At the bottom is a 'Global Permissions' section. The main content area is titled 'Showing All Organizations' and lists two organizations: 'janedoe1-org' and 'johndoe1-org'. The 'johndoe1-org' row is highlighted with a light orange background. A dropdown menu is open over this row, showing the options 'Organization: janedoe1-org' and 'Organization: johndoe1-org'. The 'johndoe1-org' option is circled in orange. Below the dropdown, a modal window is open for the 'johndoe1-org' organization, showing a 'Members' tab with a search bar and a table with one member: 'johndoe1'. The table has columns for 'Name' and 'Remove'. At the bottom of the page, there are copyright information, help links, and navigation links for 'Feedback', 'Whats New?', and 'About Opscode Manage'.

# Exercise: Revoke access

- Now either 'Leave Organization' you've been invited into, or remove your classmate from your organization

The screenshot shows the Opscode Manage web interface. The left sidebar has sections for Organizations, Users, Groups, and Global Permissions. Under Organizations, there are links for Create, Reset Validation Key, Generate Knife Config, Invite User, and Leave Organization. The 'Leave Organization' link is circled in orange. The main content area shows 'Showing All Organizations' with a table. One row in the table is highlighted with an orange background and contains the organization name 'johndoe1-org'. To the right of the table is a column labeled 'Actions' with a gear icon. Below the table, a modal window is open for the organization 'johndoe1-org'. The modal has tabs for 'Members' and 'Pending Invitations'. The 'Members' tab is selected, showing a search bar with 'Search Members...' and a magnifying glass icon. Below the search bar is a table with one row containing the name 'johndoe1'. To the right of this row is a 'Remove' button, which is also circled in orange. At the bottom of the modal are 'Feedback', 'What's New?', and 'About Opscode Manage' links.

# Review Questions

- What is an Organization?
- How do you regenerate the Starter Kit for your Organization?

# Node Setup

Setup a Node to manage

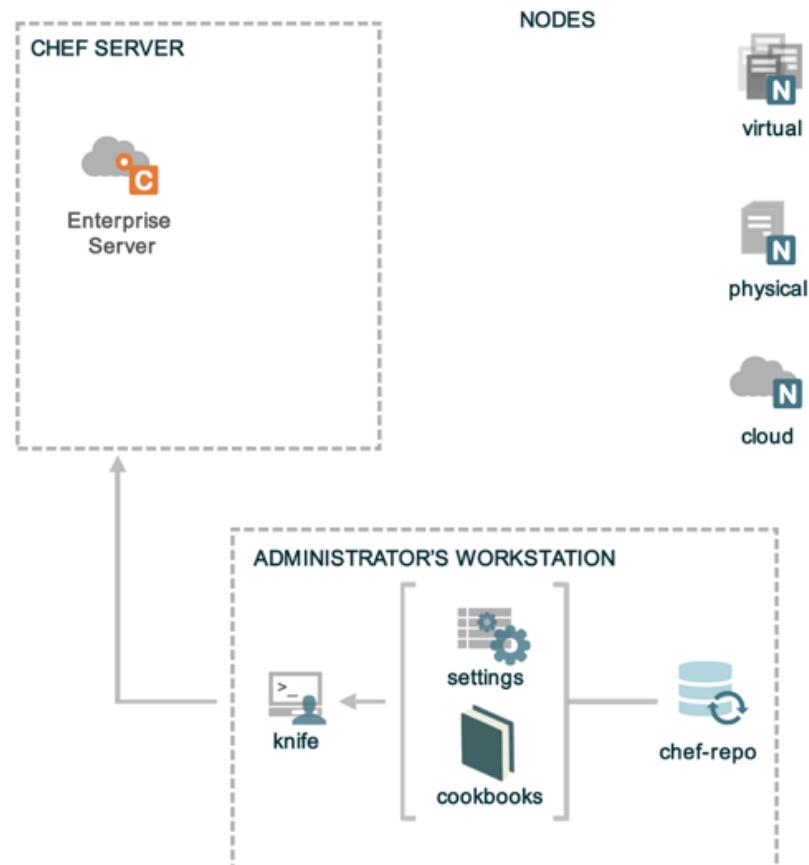
v2.0.3

# Lesson Objectives

After completing the lesson, you will be able to

- Install Chef nodes using "knife bootstrap"
- Explain how knife bootstrap configures a node to use the Organization created in the previous section
- Explain the basic configuration needed to run chef-client

# Nodes



# Nodes

- Nodes represent the servers in your infrastructure  
these may be
  - Physical or virtual servers
  - Hardware that you own
  - Compute instances in a public or private cloud
- Could also be network hardware - switches, routers, etc

# Nodes

- Nodes represent the servers in your infrastructure
  - Could be physical servers or virtual servers
  - May represent hardware that you own or compute instances in a public or private cloud
- Could also be network hardware - switches, routers, etc

# We Have No Nodes Yet

The screenshot shows the Chef Manage web application. At the top, there is a dark header bar with the "CHEF MANAGE" logo on the left. To the right of the logo is a navigation menu with three items: "Nodes", "Policies", and "Administrative". The "Nodes" item is highlighted with a light gray background. Below the header, the main content area has a light gray background. On the left side, there is a sidebar with a dark gray background containing a list of actions: "Delete", "Manage Tags", "Reset Key", "Edit Run List", and "Edit Attributes". The main content area has a heading "Showing All Nodes" and a message "There are no items to display." with an information icon.

CHEF  
MANAGE

Nodes Policies Administrative

> Nodes

Delete  
Manage Tags  
Reset Key  
Edit Run List  
Edit Attributes

Showing All Nodes

There are no items to display.

# Training Node

- The labs require a node to be managed
- We allow for two different options
  - Bring your own Node
  - Use the CloudShare training environment

# Bring Your Own Node

- Use your own Virtual Machine (VM) or Server
- Required for the labs:
  - CentOS 6+
  - 512 MB RAM
  - 15 GB Disk
  - sudo or root level permissions

# CloudShare Node

- <http://bit.ly/1xMT88H>
- Password: learn chef with me

## Chef Fundamentals Webinar

Your dedicated hands-on environment is just a click away.

We believe you shouldn't have to waste time copying gigabytes of software, shipping machines, or traveling, just to get your IT into people's hands.

When you click the 'Start Using' button to your right, you'll have instant access worldwide to a full, enterprise-grade IT environment, available through your browser, mobile device, or dedicated client and VPN connection. You can connect your existing datacenter servers, change configurations, load new software, and provide this functionality to others. It's just like an on-premises data center, but with more flexibility and none of the headaches of backup, access, replication, logging, SLAs, and other annoyances - we take care of all of that for you.

CloudShare's SaaS platform is a quick and easy way to share copies of your complex IT environments, online. So you can collaborate with customers, partners, and colleagues - for Demos, Proofs-of-Concept, Training, or other enterprise applications.

Questions? Click [here](#) to email this environment's author or click [here](#) for CloudShare support.

# CloudShare Node

Chef Fundamentals 2.x



**CentOS 6.3 Server**

**Description:** OS: CentOS 6.3 x64

Spec: 16 GB HD / 1 GB RAM

**OS:** Linux

**State:** Running

[More details ▶](#)



Reboot VM Revert Delete

# CloudShare Node

Chef Fundamentals 2.x

The screenshot shows a CloudShare interface for a running CentOS 6.3 Server. The VM details include:

- External Address:** uvo1khywgnej7omyxst.vm.cld.sr (highlighted with an orange circle)
- Internal IP:** 10.160.201.90
- Total Memory:** 1024 MB
- Disk Size:** 16 GB
- CPU:** 1

A message at the bottom states: "The machine was prepared in 21 seconds".

**Credentials (show password)**

<b>Auto-login:</b>	root (local user)
<b>Username:</b>	root
<b>Password:</b>	*****

At the bottom right are buttons for: Reboot VM, Revert, and Delete.

# Lab - Login

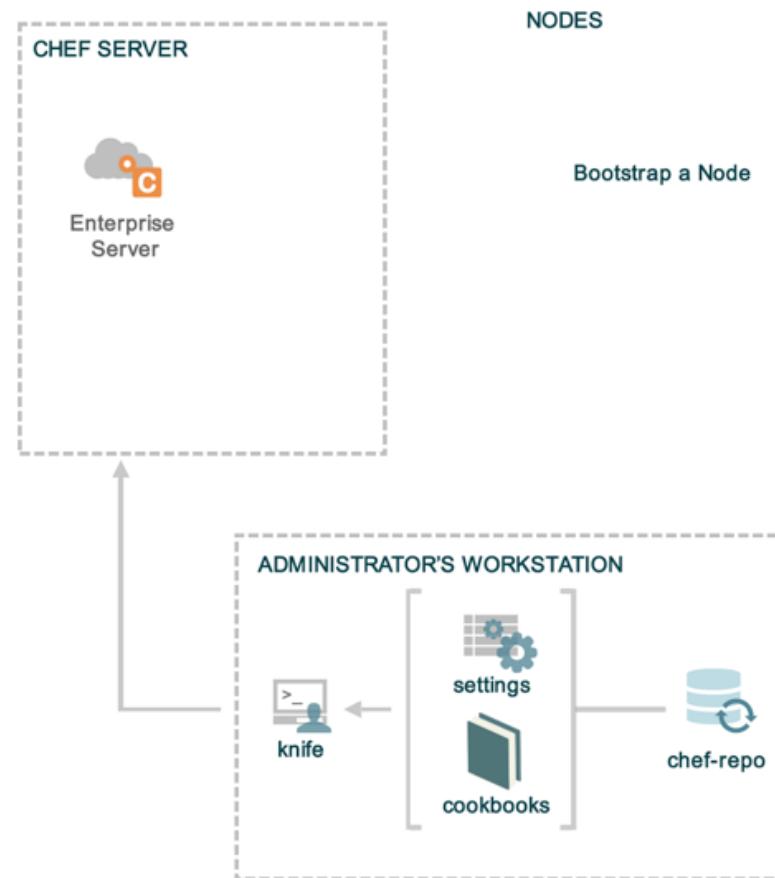
```
$ ssh chef@<EXTERNAL_ADDRESS>
```

```
The authenticity of host 'uvolqrwls0jdgs3blvt.vm.cld.sr  
(69.195.232.110)' can't be established.  
RSA key fingerprint is d9:95:a3:b9:02:27:e9:cd:  
74:e4:a2:34:23:f5:a6:8b.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added 'uvolqrwls0jdgs3blvt.vm.cld.sr,  
69.195.232.110' (RSA) to the list of known hosts.  
chef@uvolqrwls0jdgs3blvt.vm.cld.sr's password:  
Last login: Mon Jan  6 16:26:24 2014 from  
host86-145-117-53.range86-145.btcentralplus.com  
[chef@CentOS63 ~]$
```

# Checkpoint

- At this point you should have
  - One virtual machine (VM) or server that you'll use for the lab exercises
  - The IP address or public hostname
  - An application for establishing an ssh connection
  - 'sudo' or 'root' permissions on the VM

# Checkpoint

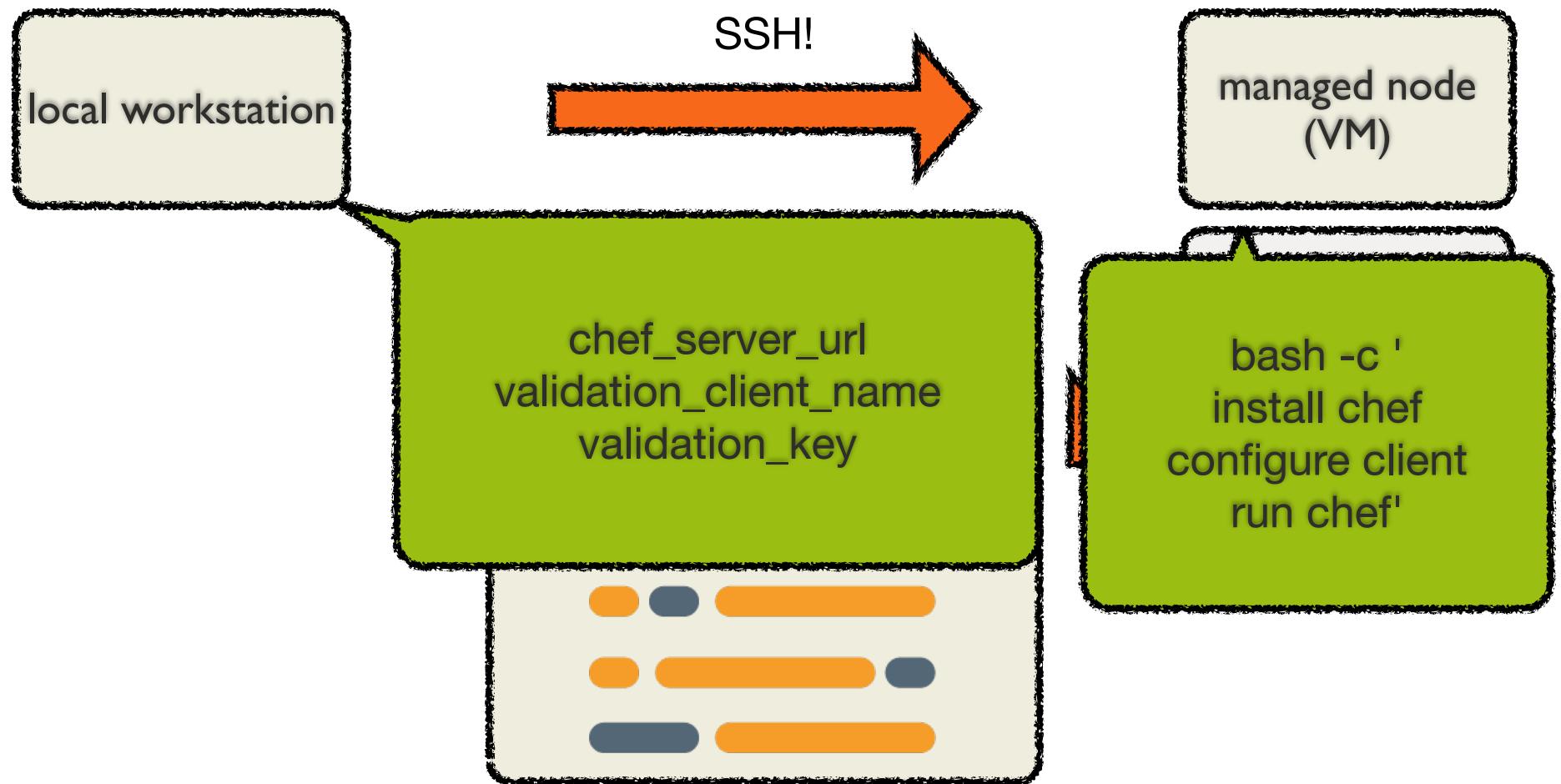


# "Bootstrap" the Target Instance

```
$ knife bootstrap <EXTERNAL_ADDRESS> --sudo -x chef -P chef -N "node1"
```

```
Bootstrapping Chef on uvolqrwls0jdgs3blvt.vm.cld.sr
uvolqrwls0jdgs3blvt.vm.cld.sr knife sudo password:
Enter your password:
...
...
uvolqrwls0jdgs3blvt.vm.cld.sr Creating a new client identity for node1 using the
validator key.
uvolqrwls0jdgs3blvt.vm.cld.sr resolving cookbooks for run list: []
uvolqrwls0jdgs3blvt.vm.cld.sr Synchronizing Cookbooks:
uvolqrwls0jdgs3blvt.vm.cld.sr Compiling Cookbooks...
uvolqrwls0jdgs3blvt.vm.cld.sr [2014-01-28T11:03:14-05:00] WARN: Node node2 has an
empty run list.
uvolqrwls0jdgs3blvt.vm.cld.sr Converging 0 resources
uvolqrwls0jdgs3blvt.vm.cld.sr Chef Client finished, 0 resources updated
```

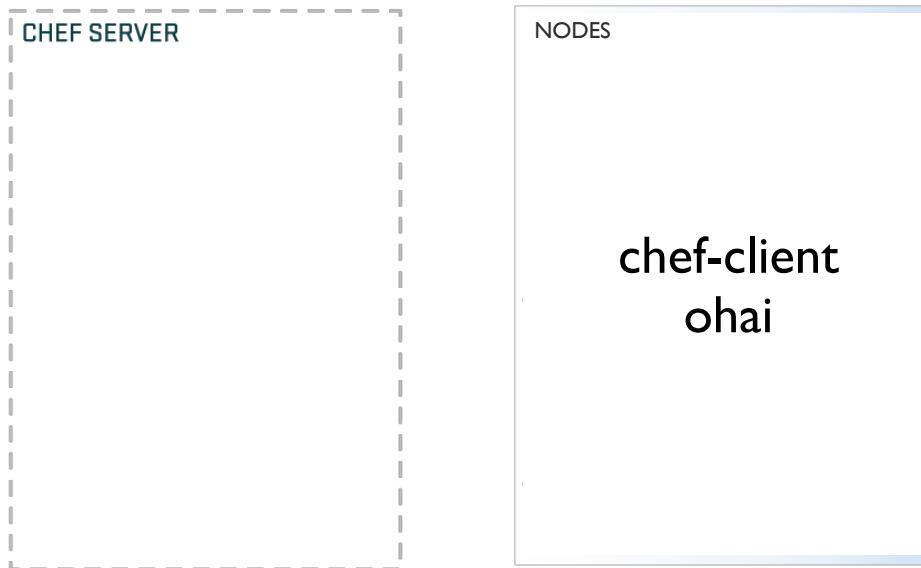
```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N node1
```



# What just happened?

- Chef and all of its dependencies installed via an operating system-specific package ("omnibus installer")
- Installation includes
  - The Ruby language - used by Chef
  - knife - Command line tool for administrators
  - chef-client - Client application
  - ohai - System profiler
  - ...and more

# Workstation or Node?



## Verify Your Target Instance's Chef-Client is Configured Properly

```
$ ssh chef@<EXTERNAL_ADDRESS>

chef@node1:~$ ls /etc/chef
client.pem  client.rb  first-boot.json validation.pem

chef@node1:~$ which chef-client
/usr/bin/chef-client
```

# Examine /etc/chef/client.rb

```
chef@node1:~$ cat /etc/chef/client.rb
```

```
log_level      :auto
log_location    STDOUT
chef_server_url "https://api.opscode.com/organizations/ORGNAME"
validation_client_name "ORGNAME-validator"
node_name       "node1"
```

# Change the log level on your test node

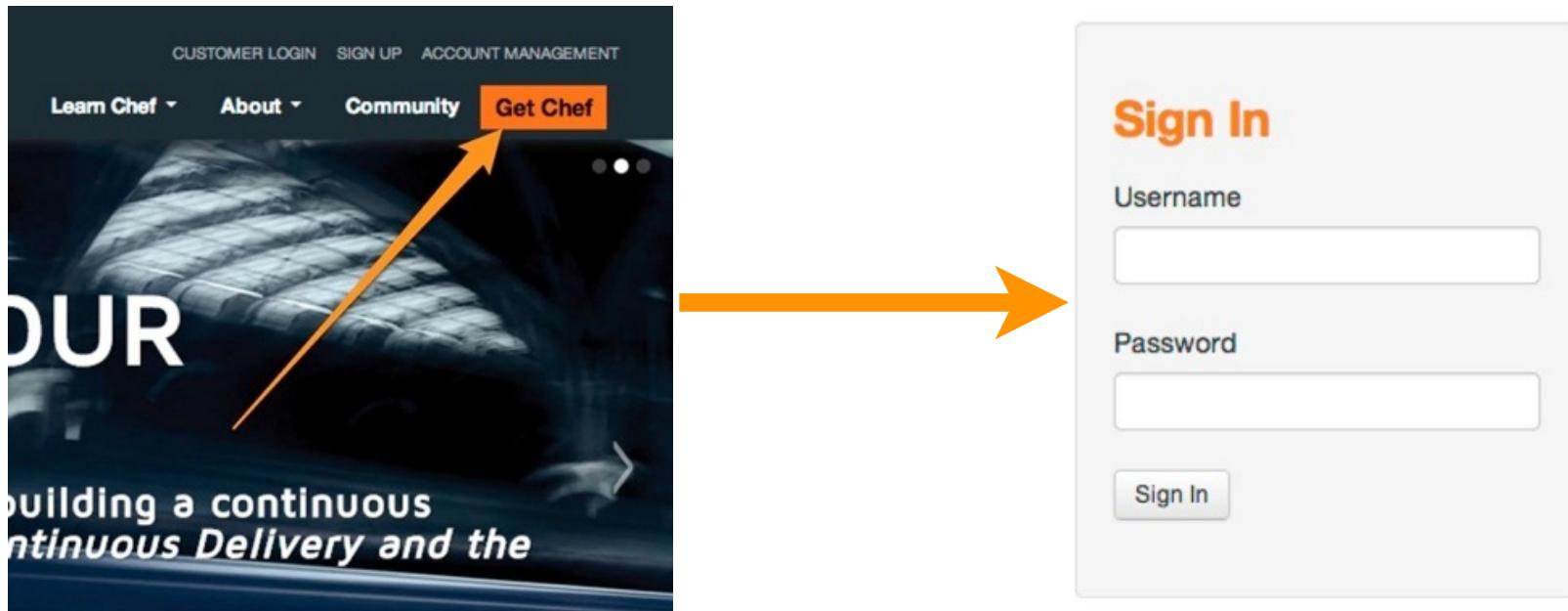
```
chef@node1:~$ sudo vim /etc/chef/client.rb
```

```
log_level      :info
log_location    STDOUT
chef_server_url "https://api.opscode.com/organizations/ORGNAME"
validation_client_name "ORGNAME-validator"
node_name       "node1"
```

- Set the default log level for chef-client to `:info`
- More configuration options can be found on the docs site:  
[http://docs.opscode.com/config\\_rb\\_client.html](http://docs.opscode.com/config_rb_client.html)

# View Node on Chef Server

- Login to your Hosted Enterprise Chef



# View Node on Chef Server

- Click the 'Details' tab

The screenshot shows the Chef Manage web interface. At the top, there's a navigation bar with tabs: Nodes (which is active), Reports, Policy, and Administration. Below the navigation bar, on the left, is a sidebar titled 'Nodes' with options: Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area is titled 'Showing All Nodes' and lists one node: 'node1'. To the right of the node list is a detailed view for 'Node: node1'. This view includes tabs for Details (which is highlighted with an orange circle), Attributes, and Permissions. Below these tabs, there are two boxes: 'Last Check In: 3 Minutes Ago' (2014-01-06 11:36:31 UTC) and 'Uptime: 5 Hours' (Since 2014-01-06 07:08:16 UTC). At the bottom of the node details section, there's a 'Tags' section with a '+ Add' button and a message: 'There are no items to display.'

# View Node on Chef Server

- Click the 'Attributes' tab

The screenshot shows the Chef Manage web interface. At the top, there's a navigation bar with tabs for Nodes, Reports, Policy, and Administration. Below that is a sidebar with options like Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area shows a table titled "Showing All Nodes" with one row for "node1". The table has columns for Node Name, Platform, FQDN, and IP Address. Below the table, a modal window is open for the node named "node1". The modal has three tabs: Details, Attributes, and Permissions. The "Attributes" tab is highlighted with an orange circle. Inside the Attributes section, there's a "Attributes" heading and two buttons: "Expand All" and "Collapse All". Underneath, there's a list of node attributes with plus signs next to some of them:

Node Name	Platform	FQDN	IP Address
node1	centos	centos63.example.com	10.160.201.90

Node: node1

Attributes

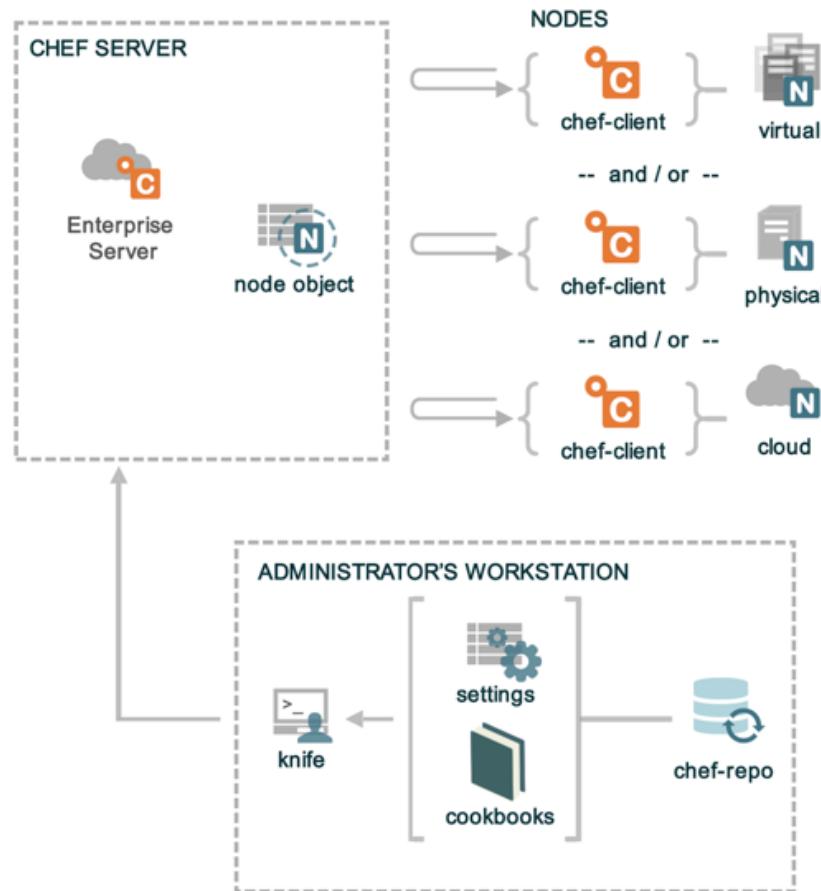
Expand All   Collapse All

tags:  
+ languages  
+ kernel  
os: linux  
os\_version: 2.6.32-358.23.2.el6.x86\_64  
hostname: CentOS63  
fqdn: centos63.example.com  
domain: example.com  
+ network  
+ counters  
ipaddress: 10.160.201.90  
macaddress: 00:50:56:00:00:90

# Node

- The node is registered with Chef Server
- The Chef Server displays information about the node
- This information comes from Ohai - we'll see Ohai later....

# Checkpoint



# Review Questions

- Where is the chef-client configuration file?
- What is the command to run chef?
- What does a knife bootstrap do?

# Chef Resources and Recipes

Writing an Apache cookbook

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Describe a **resource**, **recipe**, and **cookbook**
  - Create a new cookbook
  - Describe how to use the resources: **package**, **service**, and **cookbook\_file**.
  - Upload a cookbook to the Chef Server
  - Explain the node's **run list** and how to set it

# Resources

Resources represents a piece of the system and its desired state.

- A **package** that should be **installed**
- A **service** that should be **running**
- A **file** that should be **created**
- A **user** that should be **removed**
- ...

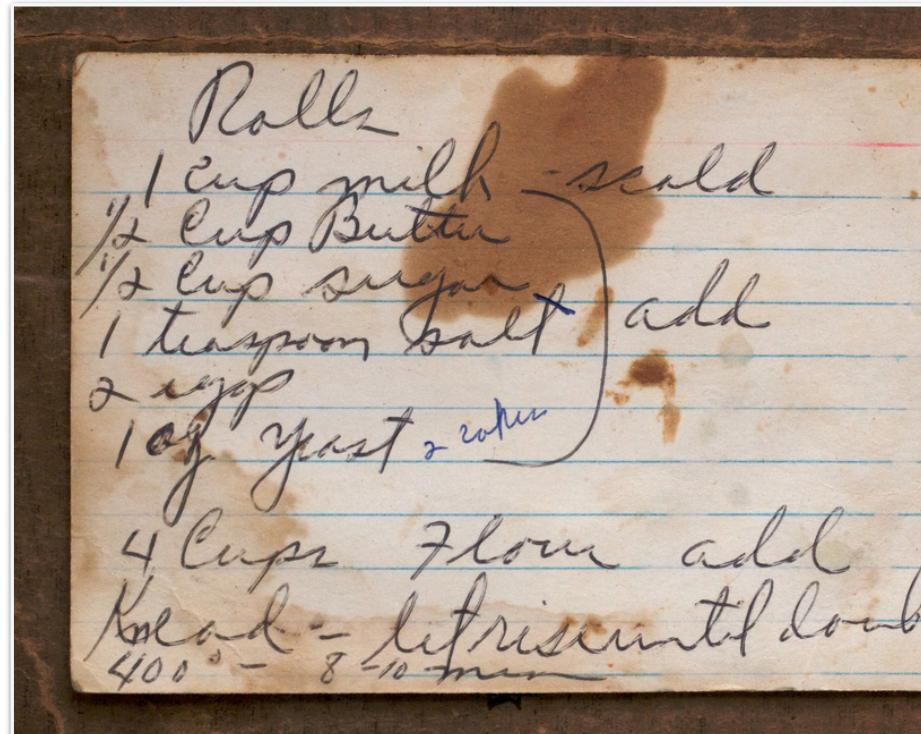
# Example Resources

Our application server requires Apache to be installed.

- **Install** the package **apache**
- **Create** a default home page **file**
- **Enable** the **apache** service
- **Start** the **apache** service

# Resources are the Ingredients

Often several resources act in concert to create a desired state



<https://www.flickr.com/photos/chiotsrun/4457365506>

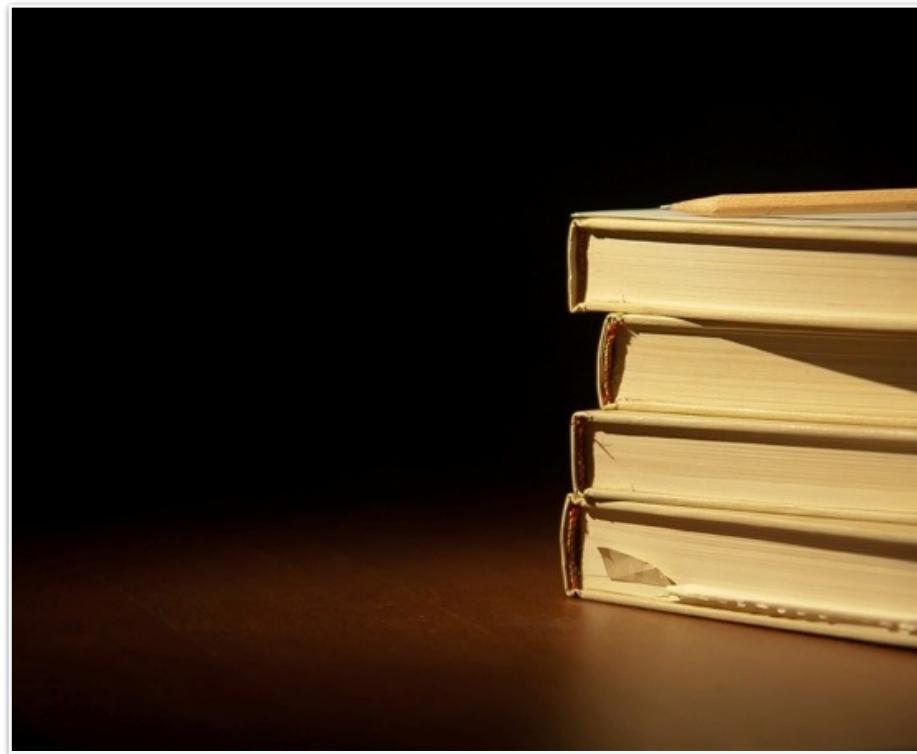
# Recipes

Recipes represent an ordered collection of resources that:

- **Deploy** an application
- **Manage** an existing service
- **Configure** the policy and files for a user
- ...

# Recipes belong in a Cookbook

So we can write down  
**additional instructions**,  
**version** them, **include files**,  
and **share** them with others.



[www.flickr.com/photos/shutterhacks/4474421855/](http://www.flickr.com/photos/shutterhacks/4474421855/)

# What is a Cookbook?

A cookbook is a **collection** of recipes. Typically they map 1:1 to a piece of software or functionality.



<https://www.flickr.com/photos/sackton/7842721042>

# The Problem and the Success Criteria

- **The Problem:** We need a web server configured to serve up our home page.
- **Success Criteria:** We can see the homepage in a web browser.

# Required steps

- Install Apache
- Start the service, and make sure it will start when the machine boots
- Write out the home page

# Chef Resources

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# Chef Resources

- Have a type

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# Chef Resources

- Have a type
- Have a **name**

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# Chef Resources

- Have a type
- Have a name
- Have **parameters**

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# Chef Resources

- Have a type
- Have a name
- Have parameters
- Take **action** to put the resource into the desired state

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# Exercise: Create a new Cookbook

```
$ knife cookbook create apache
```

```
** Creating cookbook apache
** Creating README for cookbook: apache
** Creating CHANGELOG for cookbook: apache
** Creating metadata for cookbook: apache
```

# Exercise: Explore the cookbook

```
$ ls -la cookbooks/apache
```

```
total 24
drwxr-xr-x 13 opscode opscode 442 Jan 24 21:25 .
drwxr-xr-x  5 opscode opscode 170 Jan 24 21:25 ..
-rw-r--r--  1 opscode opscode 412 Jan 24 21:25 CHANGELOG.md
-rw-r--r--  1 opscode opscode 1447 Jan 24 21:25 README.md
drwxr-xr-x  2 opscode opscode   68 Jan 24 21:25 attributes
drwxr-xr-x  2 opscode opscode   68 Jan 24 21:25 definitions
drwxr-xr-x  3 opscode opscode 102 Jan 24 21:25 files
drwxr-xr-x  2 opscode opscode   68 Jan 24 21:25 libraries
-rw-r--r--  1 opscode opscode 276 Jan 24 21:25 metadata.rb
drwxr-xr-x  2 opscode opscode   68 Jan 24 21:25 providers
drwxr-xr-x  3 opscode opscode 102 Jan 24 21:25 recipes
drwxr-xr-x  2 opscode opscode   68 Jan 24 21:25 resources
drwxr-xr-x  3 opscode opscode 102 Jan 24 21:25 templates
```

# Exercise: Edit the default recipe



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
#  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

## Exercise: Add a package resource to install Apache to the default recipe



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
#  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

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

**SAVE FILE!**

# So the resource we just wrote...

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

# So the resource we just wrote...

- Is a **package** resource

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

# So the resource we just wrote...

- Is a package resource
- Whose **name** is *httpd*

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

# So the resource we just wrote...

- Is a package resource
- Whose name is *httpd*
- With an install **action**

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

# Notice we didn't say how to install the package

- Resources are **declarative** - that means we say *what* we want to have happen, rather than *how*
- Resources take action through **Providers** - providers perform the how
- Chef uses the **platform** the node is running to determine the correct **provider** for a resource

# Package Resource

```
package "httpd"
```



```
yum install httpd
```

```
apt-get install httpd
```

```
pacman sync httpd
```

```
pkg_add -r httpd
```

***Providers are  
determined  
by node's platform***

## Exercise: Add a service resource to ensure the service is started and enabled at boot



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

```
...
# All rights reserved - Do Not Redistribute
#
package "httpd" do
  action :install
end

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

**SAVE FILE!**

# So the resource we just wrote...

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

# So the resource we just wrote...

- Is a **service** resource

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

# So the resource we just wrote...

- Is a service resource
- Whose **name** is *httpd*

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

# So the resource we just wrote...

- Is a service resource
- Whose **name** is *httpd*
- With two **actions**:
  - enable
  - start

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

# Order Matters

- Resources are executed in order

1st  
2nd  
3rd

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

## Exercise: Add a cookbook\_file resource to copy the home page in place

 **OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

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

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end
```

**SAVE FILE!**

# So the resource we just wrote...

```
cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# So the resource we just wrote...

- Is a **cookbook\_file** resource

```
cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# So the resource we just wrote...

- Is a `cookbook_file` resource
- Whose **name** is:  
`/var/www/html/index.html`

```
cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# So the resource we just wrote...

- Is a `cookbook_file` resource
- Whose **name** is: `/var/www/html/index.html`
- With two **parameters**:
  - **source** of `index.html`
  - **mode** of “0644”

```
cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# So the resource we just wrote...

- Is a `cookbook_file` resource
- Whose **name** is: `/var/www/html/index.html`
- With two **parameters**:
  - **source** of `index.html`
  - **mode** of “0644”
- With an `create` **action**

```
cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
  action :create
end
```

# Full contents of the apache recipe

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
#  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#  
  
package "httpd" do  
  action :install  
end  
  
service "httpd" do  
  action [ :enable, :start ]  
end  
  
cookbook_file "/var/www/html/index.html" do  
  source "index.html"  
  mode "0644"  
  action :create  
end
```

## Exercise: Add index.html to your cookbook's files/default directory



**OPEN IN EDITOR:** cookbooks/apache/files/default/index.html

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

**SAVE FILE!**

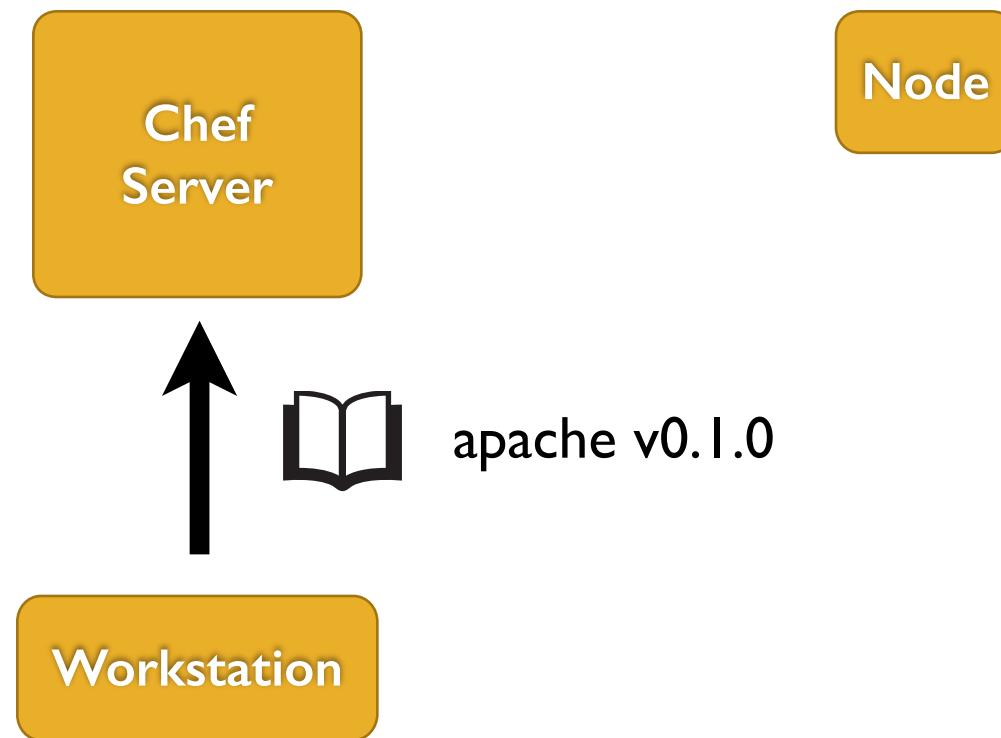
# What's with the 'default' subdirectory?

Chef allows you to select the most appropriate file within a cookbook according to the node's platform:

- host node name (node1)
- platform-version (centos-6.4)
- platform-version\_components (centos-6)
- platform (centos)
- default

99% of the time, you will just use **default**

# Upload the Cookbook



# Exercise: Upload the cookbook

```
$ knife cookbook upload apache
```

```
Uploading apache [ 0.1.0 ]  
Uploaded 1 cookbook.
```

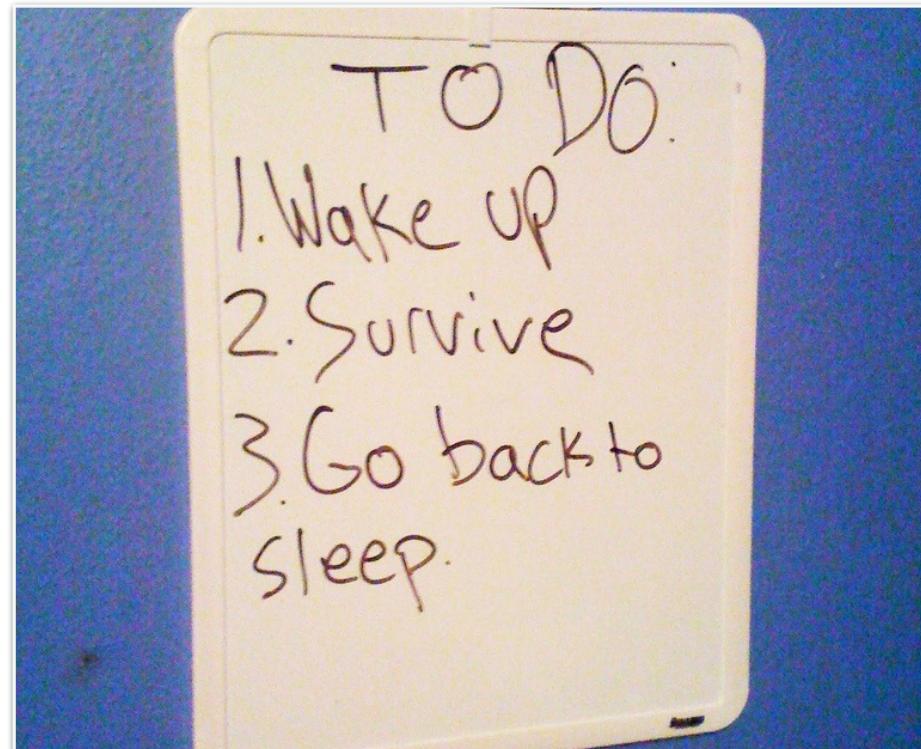
# Exercise: Upload the cookbook

```
$ knife cookbook list
```

```
apache      0.1.0
```

# How do nodes know what to do?

The chef server keeps a todo list for each node called their run list. The run list is an ordered list of recipes the node will execute.



<https://www.flickr.com/photos/tmray02/4415770896>

# An example run list

```
recipe[apache::default]
recipe[motd::default]
recipe[users::default]
recipe[ntp::default]
```

# An example run list

- A recipe (or role)

```
recipe[apache::default]
recipe[motd::default]
recipe[users::default]
recipe[ntp::default]
```

# An example run list

- A recipe (or role)
- Cookbook name

```
recipe[apache::default]
recipe[motd::default]
recipe[users::default]
recipe[ntp::default]
```

# An example run list

- A recipe (or role)
- Cookbook name
- Recipe name

```
recipe[apache::default]
recipe[motd::default]
recipe[users::groups]
recipe[ntp::service]
```

# An example run list

- A recipe (or role)
- Cookbook name
- Recipe name

*When using the default recipe  
you can leave it off.*

```
recipe[apache]
recipe[motd]
recipe[users::groups]
recipe[ntp::service]
```

## Exercise: Add apache recipe to test node's run list

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

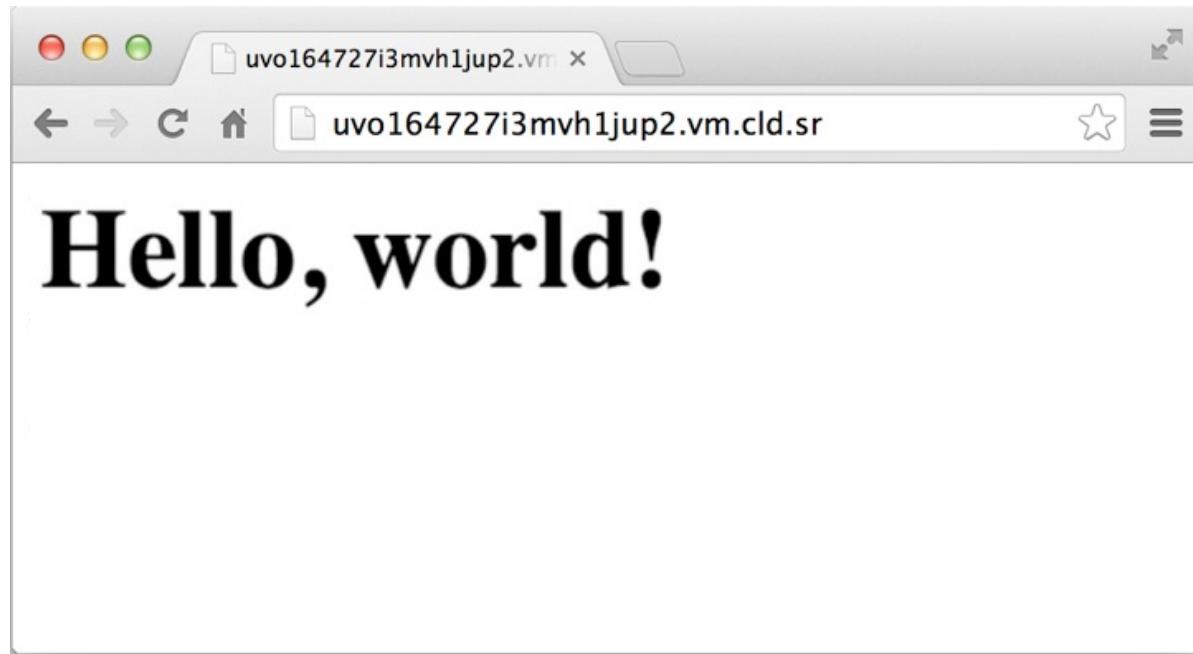
```
node1:  
  run_list: recipe[apache]
```

# Exercise: Run Chef Client

```
chef@node1:~$ sudo chef-client
```

```
[2014-01-21T12:22:40-05:00] INFO: Forking chef instance to converge...
Starting Chef Client, version 11.8.2
[2014-01-21T12:22:41-05:00] INFO: *** Chef 11.8.2 ***
[2014-01-21T12:22:41-05:00] INFO: Chef-client pid: 16346
[2014-01-21T12:22:41-05:00] INFO: Run List is [recipe[apache]]
[2014-01-21T12:22:41-05:00] INFO: Run List expands to [apache]
[2014-01-21T12:22:41-05:00] INFO: Starting Chef Run for node1
[2014-01-21T12:22:41-05:00] INFO: Running start handlers
[2014-01-21T12:22:41-05:00] INFO: Start handlers complete.
[2014-01-21T12:22:42-05:00] INFO: HTTP Request Returned 404 Object Not Found:
resolving cookbooks for run list: ["apache"]
[2014-01-21T12:22:42-05:00] INFO: Loading cookbooks [apache]
Synchronizing Cookbooks:
[2014-01-21T12:22:42-05:00] INFO: Storing updated
cookbooks/apache/recipes/default.rb in the cache.
[2014-01-21T12:22:42-05:00] INFO: Storing updated
cookbooks/apache/CHANGELOG.md in the cache.
[2014-01-21T12:22:43-05:00] INFO: Storing updated
cookbooks/apache/metadata.rb in the cache.
[2014-01-21T12:22:43-05:00] INFO: Storing updated
cookbooks/apache/README.md in the cache.
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
httpd-2.2.15-29.el6.centos from base repository
...
```

# Exercise: Verify that the home page works





# S U C C E S S

The first steps are the hardest.

<https://www.flickr.com/photos/lupus63/102179341>

# Reading the output of a chef-client run

```
Starting Chef Client, version 11.8.2
[2014-01-06T07:06:00-05:00] INFO: *** Chef 11.8.2 ***
[2014-01-06T07:06:00-05:00] INFO: Chef-client pid: 10781
[2014-01-06T07:06:01-05:00] INFO: Run List is [recipe[apache]]
[2014-01-06T07:06:01-05:00] INFO: Run List expands to [apache]
[2014-01-06T07:06:01-05:00] INFO: Starting Chef Run for node1
[2014-01-06T07:06:01-05:00] INFO: Running start handlers
[2014-01-06T07:06:01-05:00] INFO: Start handlers complete.
```

- Chef Client Version
- Run List and Run List Expansion

# Reading the output of a chef-client run

```
resolving cookbooks for run list: ["apache"]
[2014-01-06T07:06:02-05:00] INFO: Loading cookbooks [apache]
Synchronizing Cookbooks:
[2014-01-06T07:06:02-05:00] INFO: Storing updated cookbooks/apache/recipes/default.rb in the
cache.
[2014-01-06T07:06:02-05:00] INFO: Storing updated cookbooks/apache/metadata.rb in the cache.
 - apache
Compiling Cookbooks...
```

- Loads the cookbooks in the order specified by the run list
- Downloads any files that are missing from the server

# Reading the output of a chef-client run

```
Converging 3 resources
Recipe: apache::default
 * package[httpd] action install[2014-01-06T07:51:48-05:00] INFO: Processing
package[httpd] action install (apache::default line 9)
[2014-01-06T07:51:55-05:00] INFO: package[httpd] installing
httpd-2.2.15-29.el6.centos from base repository
      - install version 2.2.15-29.el6.centos of package httpd
```

- Checks to see if the package httpd is installed

# Reading the output of a chef-client run

```
* service[httpd] action enable[2014-01-06T07:52:06-05:00] INFO: Processing
service[httpd] action enable (apache::default line 13)
[2014-01-06T07:52:06-05:00] INFO: service[httpd] enabled
  - enable service service[httpd]

* service[httpd] action start[2014-01-06T07:52:06-05:00] INFO: Processing
service[httpd] action start (apache::default line 13)
[2014-01-06T07:52:07-05:00] INFO: service[httpd] started
  - start service service[httpd]
```

- Checks to see if httpd is already enabled to run at boot - it is, take no further action
- Checks to see if httpd is already started - it is, take no further action

# Reading the output of a chef-client run

```
* cookbook_file[/var/www/html/index.html] action create[2014-01-06T07:52:07-05:00] INFO: Processing
cookbook_file[/var/www/html/index.html] action create (apache::default line 17)
[2014-01-06T07:52:07-05:00] INFO: cookbook_file[/var/www/html/index.html] created file /var/www/html/index.html

- create new file /var/www/html/index.html[2014-01-06T07:52:07-05:00] INFO: cookbook_file[/var/www/html/
index.html] updated file contents /var/www/html/index.html

- update content in file /var/www/html/index.html from none to 03fb1d
  --- /var/www/html/index.html      2014-01-06 07:52:07.285214202 -0500
  +++ /tmp/.index.html20140106-10796-1kxknbg    2014-01-06 07:52:07.868365963 -0500
  @@ -1 +1,6 @@
  +<html>
  +<body>
  +  <h1>Hello, world!</h1>
  +</body>
  +</html>[2014-01-06T07:52:07-05:00] INFO: cookbook_file[/var/www/html/index.html] mode changed to 644

- change mode from '' to '0644'
- restore selinux security context
```

- Checks for an index.html file
- There is already one in place, backup the file
- Set permissions on the file
- A diff of the written file is shown with the modified lines called out

# Reading the output of a chef-client run

```
[2014-01-06T07:52:08-05:00] INFO: Chef Run complete in 23.477837576 seconds
[2014-01-06T07:52:08-05:00] INFO: Running report handlers
[2014-01-06T07:52:08-05:00] INFO: Report handlers complete
Chef Client finished, 4 resources updated
[2014-01-06T07:52:08-05:00] INFO: Sending resource update report (run-id:
952a8431-0994-468e-836c-0f7de7aa656e)
```

- Notice that a complete Chef-Run displays:
  - The time the client took to complete convergence
  - Status of report and exception handlers

# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
...
resolving cookbooks for run list: ["apache"]
[2014-01-06T07:57:13-05:00] INFO: Loading cookbooks [apache]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[httpd] action install[2014-01-06T07:57:13-05:00] INFO: Processing package[httpd] action install (apache::default line 9)
    (up to date)
  * service[httpd] action enable[2014-01-06T07:57:17-05:00] INFO: Processing service[httpd] action enable (apache::default line 13)
    (up to date)
  * service[httpd] action start[2014-01-06T07:57:17-05:00] INFO: Processing service[httpd] action start (apache::default line 13)
    (up to date)
  * cookbook_file[/var/www/html/index.html] action create[2014-01-06T07:57:17-05:00] INFO: Processing cookbook_file[/var/www/html/index.html] action create (apache::default line 17)
    (up to date)
[2014-01-06T07:57:17-05:00] INFO: Chef Run complete in 4.707139533 seconds
[2014-01-06T07:57:17-05:00] INFO: Removing cookbooks/apache/files/default/index.html from the cache; it is no longer needed by chef-client.
[2014-01-06T07:57:17-05:00] INFO: Running report handlers
[2014-01-06T07:57:17-05:00] INFO: Report handlers complete
Chef Client finished, 0 resources updated
[2014-01-06T07:57:17-05:00] INFO: Sending resource update report (run-id: 1c201d2c-797f-43ab-a904-1ba329b9aab8)
```

# Desired State Configuration

Chef is a "desired state configuration" system - if a resource is already configured, no action is taken. This is called ***convergence***

# Recap - So resources are grouped into recipes

```
recipe[apache::default] {  
    package "httpd" do  
        action :install  
    end  
  
    service "httpd" do  
        action [ :enable, :start ]  
    end  
  
    cookbook_file "/var/www/html/index.html" do  
        source "index.html"  
        mode "0644"  
        action :create  
    end  
}
```

# Cookbooks contain recipes and supporting files

```
& pwd  
/Users/you/chef-repo  
$ tree  
...  
└── cookbooks  
    └── apache  
        ├── recipes  
        │   └── default.rb  
        ├── files  
        │   └── default  
        │       └── index.html  
        ├── metadata.rb  
        └── attributes  
            └── default  
...  
...
```

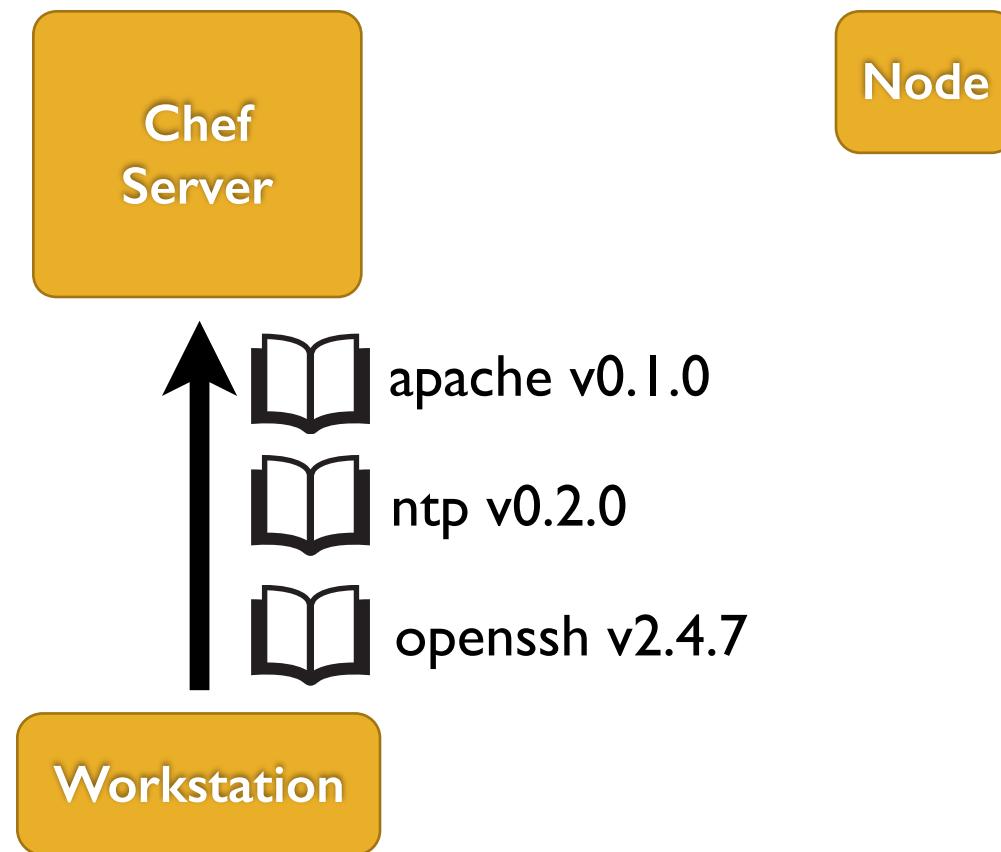


Recipes



Supporting File

# Upload the Cookbook



# Cookbooks are installed as artifacts on Chef Server

Chef Server

## Cookbooks



apache v0.1.0



ntp v0.2.0



openssh v2.4.7

# Chef Server maintains a `run_list` for each node

## Chef Server

### Cookbooks



apache v0.1.0



ntp v0.2.0



openssh v2.4.7

### Nodes



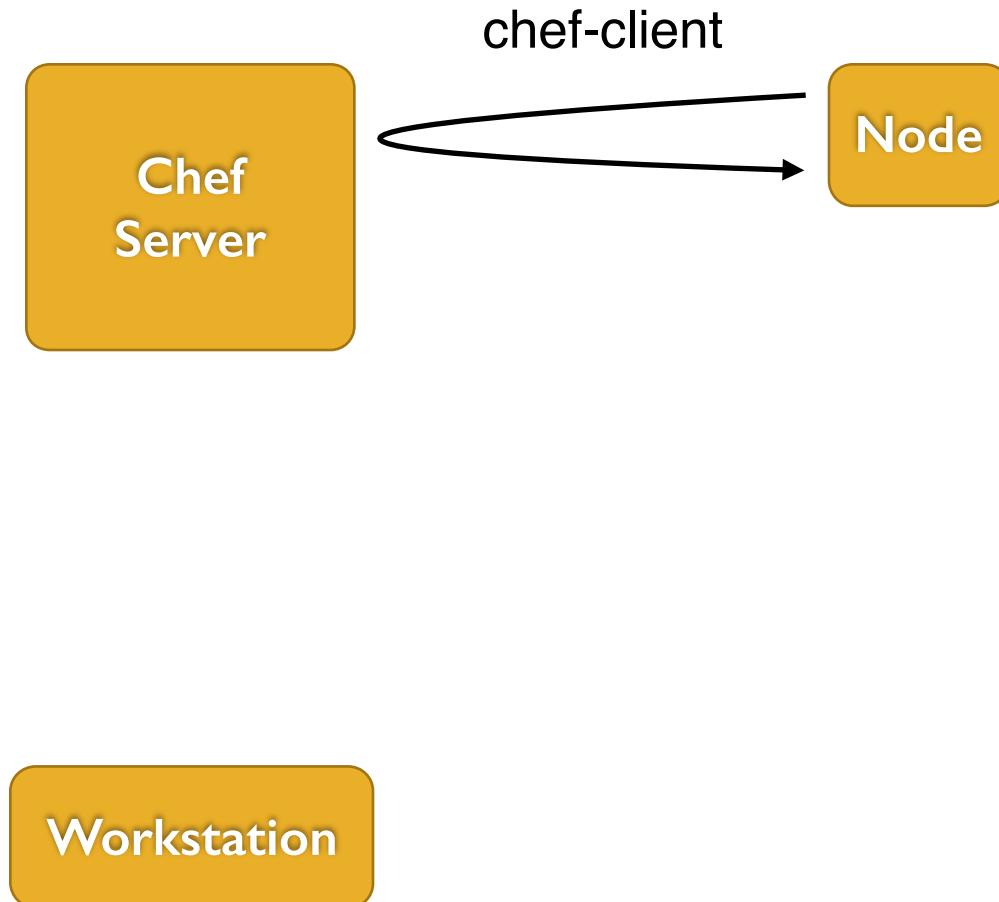
node1

#### run list:

`recipe[apache::default]`

`recipe[ntp::client]`

# Nodes make requests from the Chef Server



# Questions

- What was the name of the cookbook we created?
- What recipe did we add our three resources?
- What three resources did we use in our cookbook?
- What other file did we create in the cookbook?
- How did we upload the cookbook?
- How did we tell the node which recipes to run?

# Abstract Questions

- What is a run list?
- Who tells the node to initiate an update?
- What is a resource?

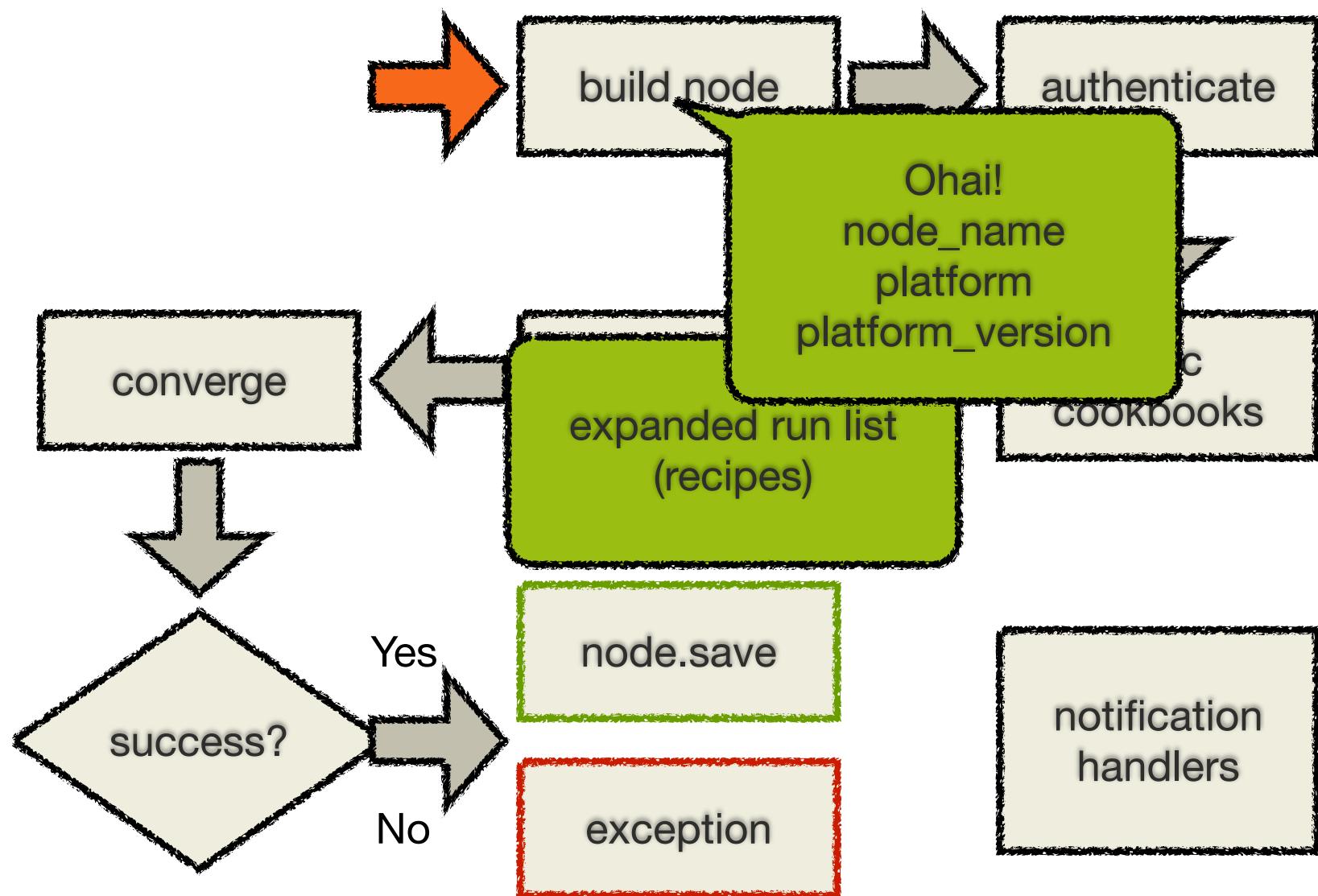
# Dissecting Your First chef-client Run

The Anatomy of a Chef run

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - List all the steps taken by a chef-client during a run
  - Explain the basic security model of Chef
  - Explain the concepts of the Resource Collection



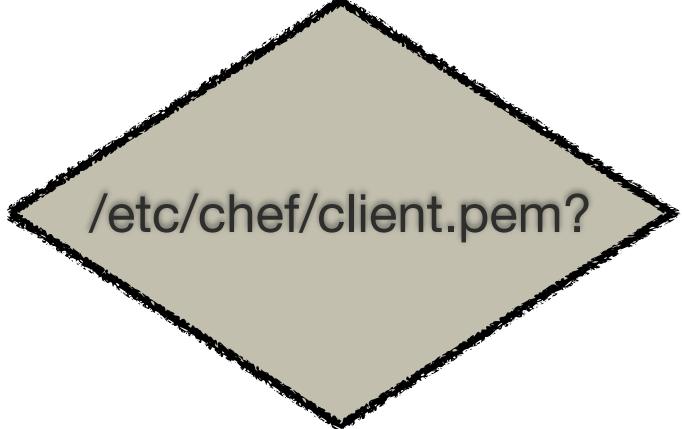
# Build Node with Ohai

```
"languages": {
  "ruby": {

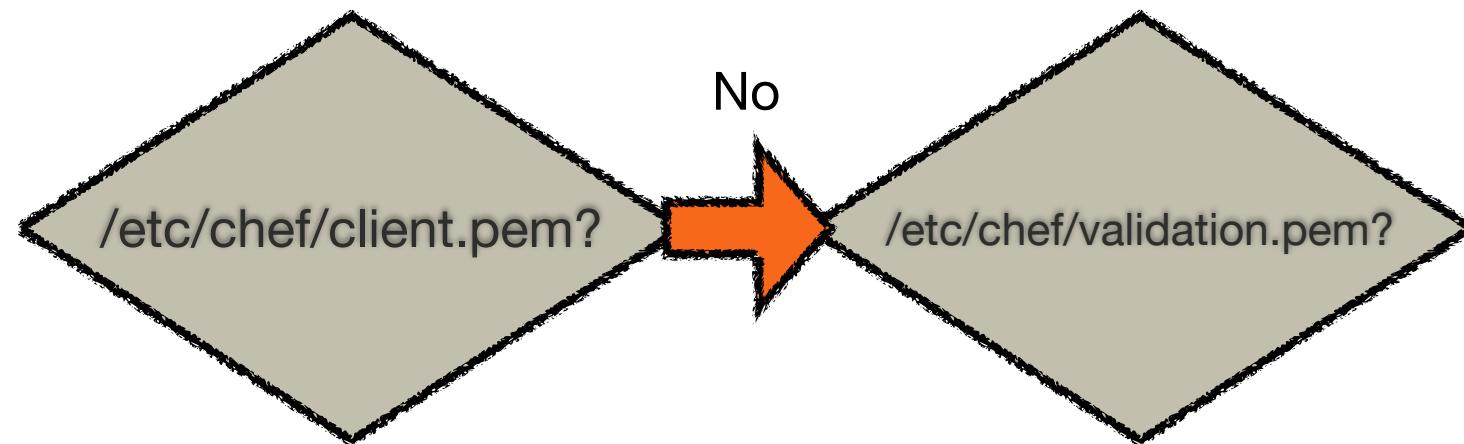
  },
  "perl": {
    "version": "5.14.2",
    "archname": "x86_64-linux-gnu-thread-multi"
  },
  "python": {
    "version": "2.6.6",
    "builddate": "Jul 10 2013, 22:48:45"
  },
  "perl": {
    "version": "version",
    "archname": "x86_64-linux-thread-multi"
  },
  "lua": {
    "version": "5.1.4"
  }
},
"kernel": {
  "name": "Linux",
  "release": "3.2.0-32-virtual",
  "version": "#1 SMP Wed Oct 16 18:37:12 UTC 2013",
  "machine": "x86_64",
  "modules": {
    "isofs": {
      "size": "70066",
      "refcount": "2"
    },
    "des_generic": {
      "size": "16604",
      "refcount": "0"
    }
  },
  "os": "GNU/Linux"
},
"os": "linux",
"os_version": "2.6.32-358.23.2.el6.x86_64",
"ohai_time": 1389105685.7735305,
"network": {
  "interfaces": {
    "lo": {
      "mtu": "16436",
      "flags": [
        "LOOPBACK", "UP", "LOWER_UP"
      ],
      "encapsulation": "Loopback",
      "addresses": {
        "127.0.0.1": {
          "family": "inet",
          "netmask": "255.0.0.0",
          "scope": "Node"
        },
        "::1": {
          "family": "inet6",
          "scope": "Node"
        }
      }
    },
    "eth0": {
      "type": "eth",
      "number": "0",
      "mac": "00:0C:29:4D:4A:00"
    }
  }
}
```

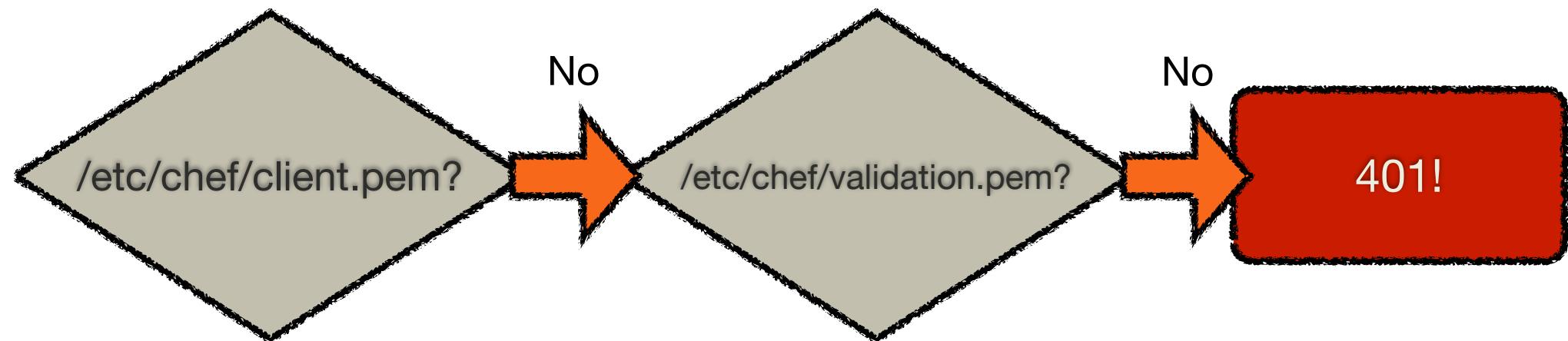
# Authentication

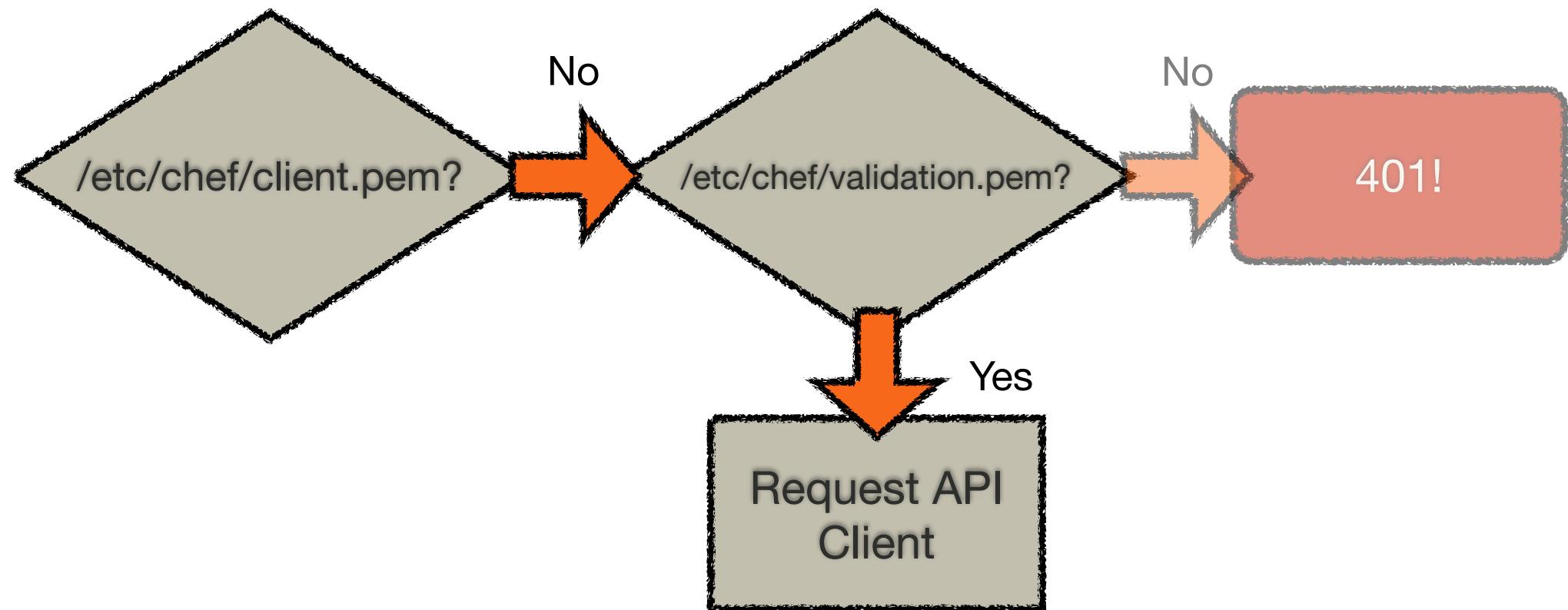
- Chef Server requires keys to authenticate.
  - `client.pem` - private key for API client
  - `validation.pem` - private key for ORGNAME-validator
- Next, let's see how those are used...

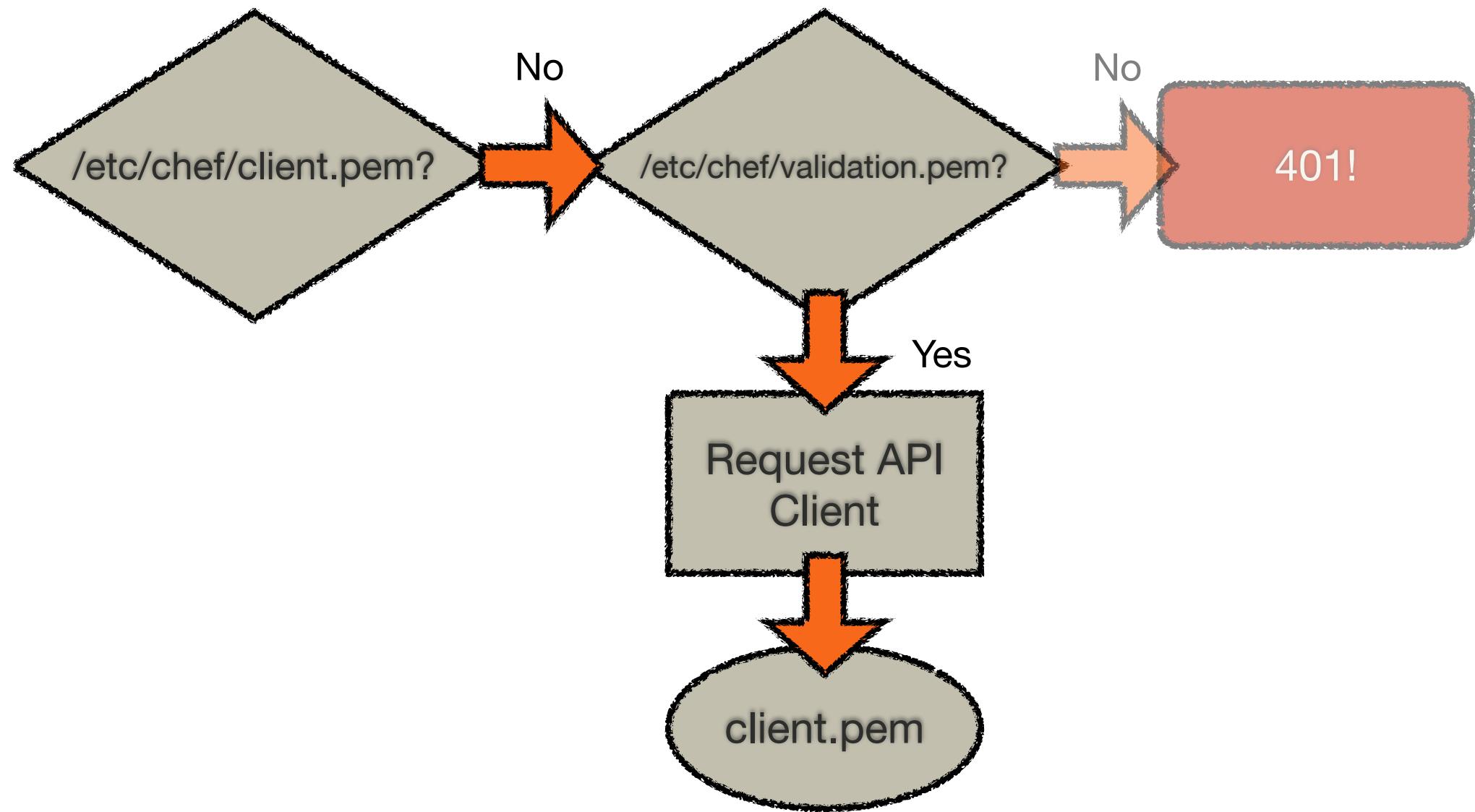


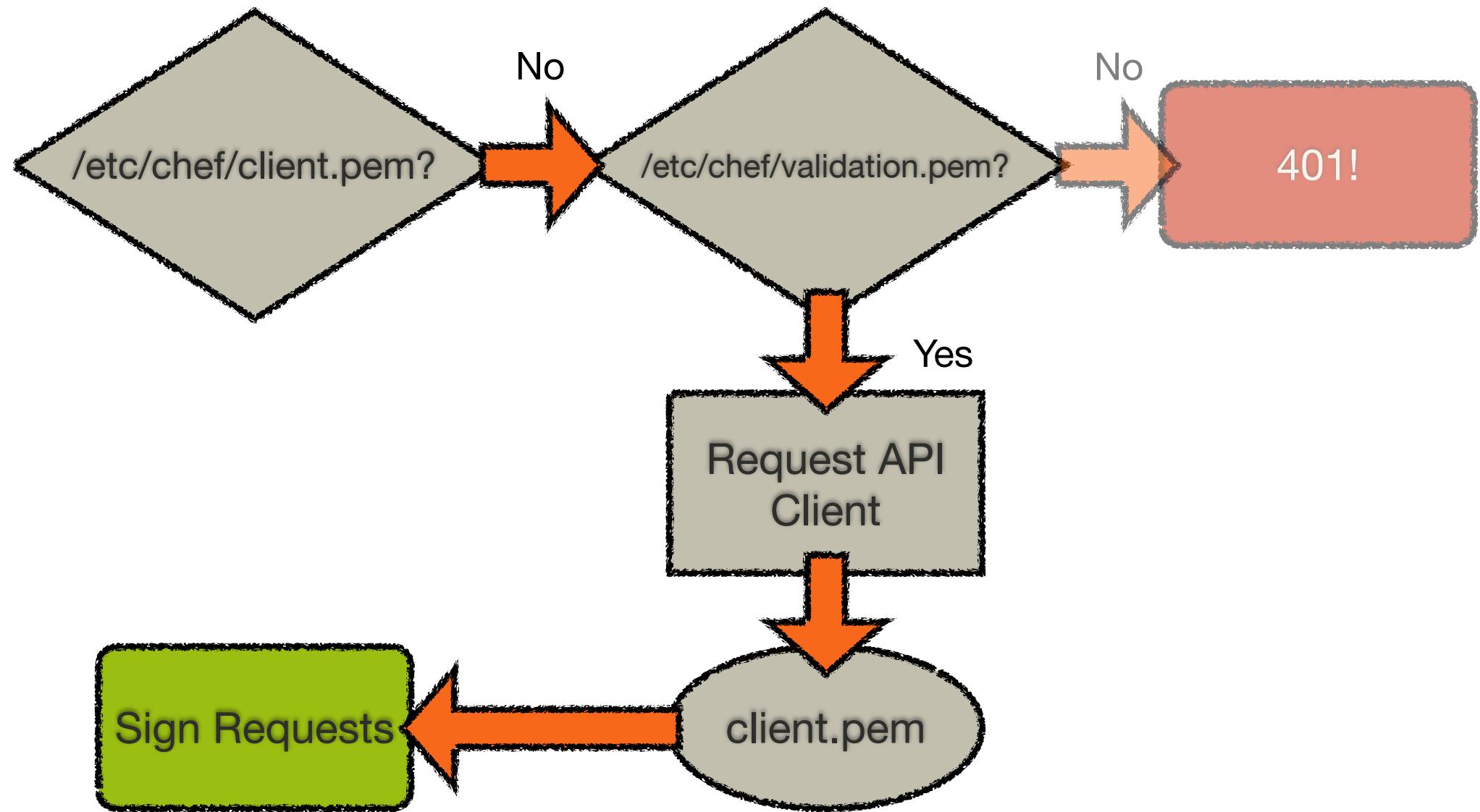
/etc/chef/client.pem?

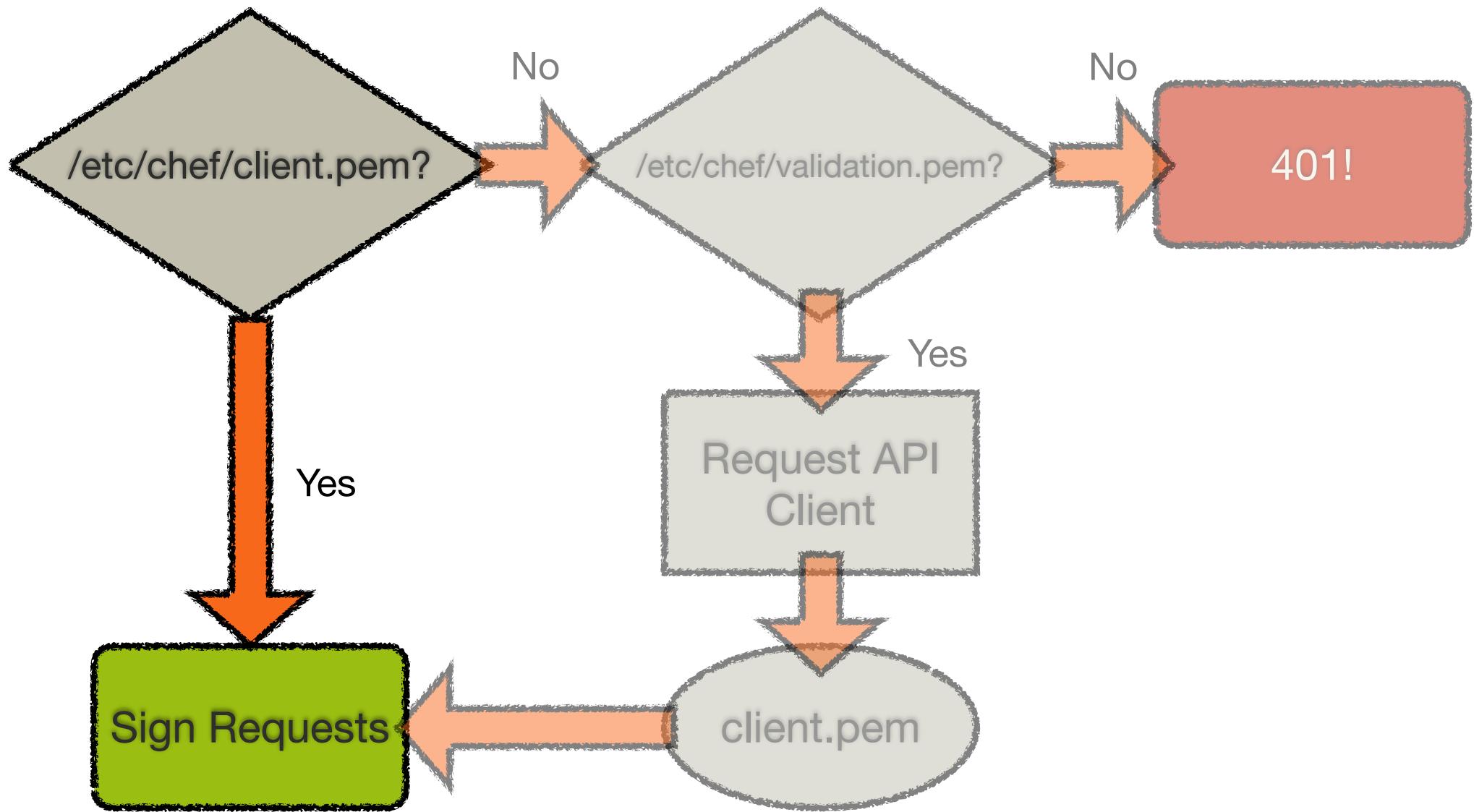












# About the resource collection

# Multiphase Execution - Compile Phase

- During the compile phase, Chef
  1. Loads all cookbooks from the run list
  2. Reads each recipe to build the resource collection

# Multiphase Execution - Execute Phase

- During the execute phase chef takes the resource collection and for each resource it will
  1. Check if the resource is in the required state
    - If 'yes' - do nothing
    - If 'no' - bring resource in line with required state
  2. Move on to next resource

# Resource Collection Phase 1 - Compile Phase

## Recipe

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

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

## Resource Collection

```
resource_collection = [
  package["httpd"],
  cookbook_file["/var/www/html/index.html"],
  service ["httpd"]
]
```

# Resource Collection Phase 2 - Execute Phase

## Recipe

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

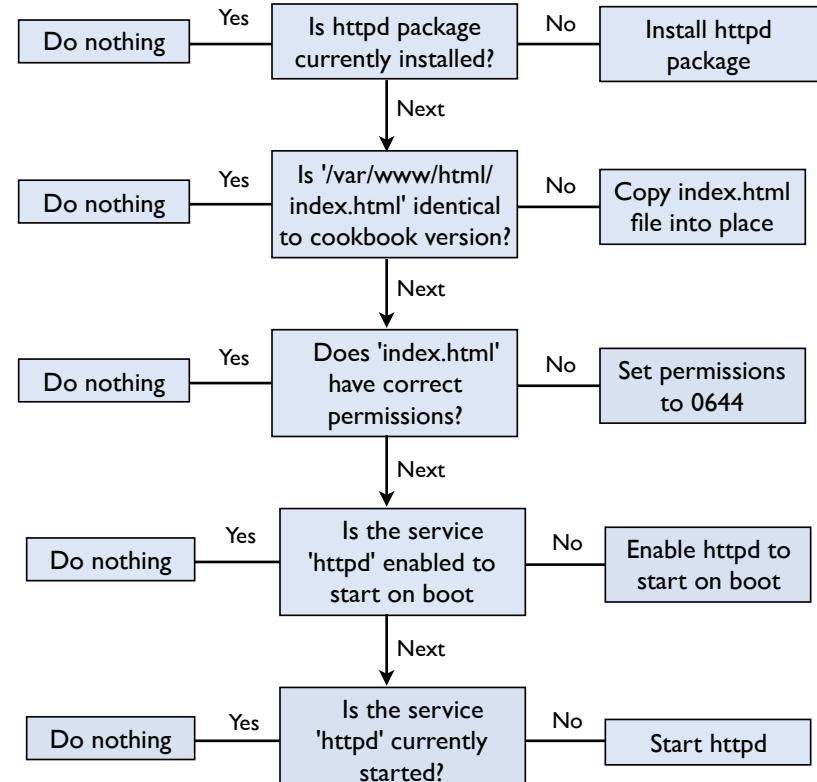
cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end

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

## Resource Collection

```
resource_collection = [
  package["httpd"],
  cookbook_file["/var/www/html/index.html"],
  service ["httpd"]
]
```

## Execution



# Recipe order is important!

- Recipes are executed in the order they appear in the run list

```
Run List: recipe[ntp::client], recipe[openssh::server], recipe[apache::server]
```

- These recipes are invoked in the following order
  1. recipe[ntp::client]
  2. recipe[openssh::server]
  3. recipe[apache::server]

# Resource Collection - Multiple Recipes

## 1. recipe[ntp::client]

```
package "ntp" do
  action :install
end

template "/etc/ntp.conf" do
  source "ntp.conf.erb"
  owner "root"
  mode "0644"
end

service "ntp" do
  action :start
end
```

## Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp]
```

# Resource Collection - Multiple Recipes

## 2. recipe[openssh::client]

```
package "openssh" do
  action :install
end

template "/etc/sshd/sshd_config" do
  source "sshd_config.erb"
  owner "root"
  mode "0644"
end

service "openssh" do
  action :start
end
```

## Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp],
  package[openssh],
  template[/etc/sshd/sshd_config],
  service[openssh]
```

# Resource Collection - Multiple Recipes

## 3. recipe[httpd::server]

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

service "httpd" do
  action [ :enable, :start ]
end

cookbook_file "/var/www/html/index.html" do
  source "index.html"
  mode "0644"
end
```

## Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp],
  package[openssh],
  template[/etc/sshd/sshd_config],
  service[openssh],
  package[httpd],
  service[httpd],
  cookbook_file[/var/www/html/index.html]
]
```

# The final resource collection

- So the resources are invoked in the following order during the execute phase

```
package[ntp]
template[/etc/ntp.conf]
service[ntp]
package[openssh]
template[/etc/sshd/sshd_config]
service[openssh]
package[httpd]
service[httpd]
cookbook_file[/var/www/html/index.html]
```

# Multiphase Execution

- Plain ruby is executed in the compile phase
- Chef DSL is executed in the execute phase

Recipe

```
[ "sites-available", "sites-enabled", "mods-available" ].each do |dir|
  directory "/var/www/#{dir}" do
    action :create
    mode  '0755'
    owner 'root'
    group node[ "apache" ][ "root_group" ]
  end
end
```

Resource Collection

```
resource_collection [
  directory[ "/var/www/sites-available" ],
  directory[ "/var/www/sites-enabled" ],
  directory[ "/var/www/mods-available" ],
]
```

# Remember - Resource order is important!

- Resources are invoked in the order they appear in the recipe

recipe-1	recipe-2	recipe-3
$\begin{matrix} \text{test} \\ \{O_1, O_2, O_3\} \\ \text{action} \end{matrix}$	$\begin{matrix} \text{test} \\ \{O_1, O_2, O_3\} \\ \text{action} \end{matrix}$	$\begin{matrix} \text{test} \\ \{O_1, O_2, O_3\} \\ \text{action} \end{matrix}$

# [----- resource collection -----]

# Review Questions

- What are the steps in a Chef Client run?
- How does a new machine get a private key with which to authenticate requests?
- If you have the right credentials in place, why else might you not be able to authenticate?
- In which phase of a chef-client run do plain Ruby statements get evaluated?

# Introducing the Node object

Attributes & Search

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Explain what the Node object represents in Chef
  - List the Nodes in an organization
  - Show details about a Node
  - Describe what Node Attributes are
  - Retrieve a node attribute directly, and via search

# What is the Node object

- A node is any physical, virtual, or cloud machines that is configured to be maintained by a Chef
- The 'node object' is the representation of that physical node within Chef (e.g. in JSON)
- When you are writing Recipes, the Node object is always available to you.

# Node

- The node is registered with Chef Server
- The Chef Server displays information about the node
- This information comes from Ohai

# View Node on Chef Server

The screenshot shows the Chef Manage web interface. The top navigation bar includes links for Nodes, Reports, Policy, and Administration. On the left, a sidebar titled 'Nodes' provides options for Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area displays a table titled 'Showing All Nodes' with columns for Node Name, Platform, FQDN, and IP Address. A single node, 'node1', is listed with the details: Platform 'centos', FQDN 'centos63.example.com', and IP Address '10.160.201.90'. Below the table, a detailed view for 'Node: node1' is shown with tabs for Details, Attributes, and Permissions. Under 'Details', it shows 'Last Check In: 3 Minutes Ago' (2014-01-06 11:36:31 UTC) and 'Uptime: 5 Hours' (Since 2014-01-06 07:08:16 UTC). A 'Tags' section at the bottom indicates there are no items to display.

Node Name	Platform	FQDN	IP Address
node1	centos	centos63.example.com	10.160.201.90

**Node: node1**

Last Check In: **3 Minutes Ago**  
2014-01-06 11:36:31 UTC

Uptime: **5 Hours**  
Since 2014-01-06 07:08:16 UTC

**Tags**

+ Add

There are no items to display.

# View Node on Chef Server

The screenshot shows the Chef Manage web interface. The top navigation bar includes links for Nodes, Reports, Policy, and Administration. On the left, a sidebar titled 'Nodes' provides options for Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area displays a table titled 'Showing All Nodes' with columns for Node Name, Platform, FQDN, and IP Address. A single row is selected for 'node1', which has a platform of 'centos', a FQDN of 'centos63.example.com', and an IP address of '10.160.201.90'. Below this, a detailed view for 'Node: node1' is shown, with tabs for Details, Attributes, and Permissions. The 'Attributes' tab is active, showing expandable sections for tags, languages, kernel, os, os\_version, hostname, fqdn, domain, network, and counters. Under 'tags', it lists 'languages' and 'kernel'. Under 'os', it lists 'linux', 'os\_version: 2.6.32-358.23.2.el6.x86\_64', 'hostname: CentOS63', 'fqdn: centos63.example.com', and 'domain: example.com'. Under 'network', it lists 'ipaddress: 10.160.201.90' and 'macaddress: 00:50:56:00:00:90'.

Node Name	Platform	FQDN	IP Address
node1	centos	centos63.example.com	10.160.201.90

**Node: node1**

Attributes

Expand All Collapse All

tags:

- + languages
- + kernel

os: linux

os\_version: 2.6.32-358.23.2.el6.x86\_64

hostname: CentOS63

fqdn: centos63.example.com

domain: example.com

- + network
- + counters

ipaddress: 10.160.201.90

macaddress: 00:50:56:00:00:90

# Exercise: List nodes

```
$ knife node list
```

```
node1
```

# Exercise: List clients

```
$ knife client list
```

```
ORGNAME-validator  
node1
```

# Each node must have a unique name

- Every node must have a unique name within an organization
- Chef defaults to the *Fully Qualified Domain Name* of the server, i.e. in the format `server.domain.com`
- We overrode it to "node1" to make typing easier

# Exercise: Show node details

```
$ knife node show node1
```

```
Node Name:    node1
Environment:  _default
FQDN:        centos63.example.com
IP:          10.160.201.90
Run List:    recipe[apache]
Roles:
Recipes:     apache
Platform:   centos 6.4
Tags:
```

# What is the Node object

- Nodes are made up of Attributes
  - Many are discovered **automatically** (platform, ip address, number of CPUs)
  - Many other objects in Chef can also add Node attributes (Cookbooks, Roles and Environments, Recipes, Attribute Files)
- Nodes are stored and indexed on the Chef Server

# Ohai

```
"languages": {
  "ruby": {

  },
  "perl": {
    "version": "5.14.2",
    "archname": "x86_64-linux-gnu-thread-multi"
  },
  "python": {
    "version": "2.6.6",
    "builddate": "Jul 10 2013, 22:48:45"
  },
  "perl": {
    "version": "version",
    "archname": "x86_64-linux-thread-multi"
  },
  "lua": {
    "version": "5.1.4"
  },
  "kernel": {
    "name": "Linux",
    "release": "3.2.0-32-virtual",
    "version": "#1 SMP Wed Oct 16 18:37:12 UTC 2013",
    "machine": "x86_64",
    "modules": {
      "isofs": {
        "size": "70066",
        "refcount": "2"
      },
      "des_generic": {
        "size": "16604",
        "refcount": "0"
      }
    },
    "os": "GNU/Linux"
  },
  "os": "linux",
  "os_version": "2.6.32-358.23.2.el6.x86_64",
  "ohai_time": 1389105685.7735305,
  "network": {
    "interfaces": {
      "lo": {
        "mtu": "16436",
        "flags": [
          "LOOPBACK", "UP", "LOWER_UP"
        ],
        "encapsulation": "Loopback",
        "addresses": {
          "127.0.0.1": {
            "family": "inet",
            "netmask": "255.0.0.0",
            "scope": "Node"
          },
          "::1": {
            "family": "inet6",
            "scope": "Node"
          }
        }
      },
      "eth0": {
        "type": "eth",
        "number": "0",
        "mac": "00:0c:29:1d:01:00"
      }
    }
  }
}
```

# Exercise: Run Ohai on node

```
chef@node1:~$ sudo ohai | less
```

```
{
  "languages": {
    "ruby": {

    },
    "java": {
      "version": "1.6.0_24",
      "runtime": {
        "name": "OpenJDK Runtime Environment (IcedTea6 1.11.13)",
        "build": "rhel-1.65.1.11.13.el6_4-x86_64"
      },
      "hotspot": {
        "name": "OpenJDK 64-Bit Server VM",
        "build": "20.0-b12, mixed mode"
      }
    }
  }
}
```

# Exercise: Show all the node attributes

```
$ knife node show node1 -l
```

```
Node Name:    node1
Environment:  _default
FQDN:        centos63.example.com
IP:          10.160.201.90
Run List:    recipe[apache]
Roles:
Recipes:     apache
Platform:   centos 6.4
Tags:
Attributes:
tags:

Default Attributes:

Override Attributes:

Automatic Attributes (Ohai Data):
block_device:
dm-0:
removable: 0
size:      28393472
```

# Exercise: Show the raw node object

```
$ knife node show node1 -Fj
```

```
{  
  "name": "node1",  
  "chef_environment": "_default",  
  "run_list": ["recipe[apache]"],  
  "normal": {"tags": []}  
}
```

# Exercise: Show only the fqdn attribute

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

```
node1:  
fqdn: centos63.example.com
```

# Exercise: Use search to find the same data

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

```
1 items found
```

```
node1:
```

```
fqdn: centos63.example.com
```

# Review Questions

- What is the Node object?
- What is a Node Attribute?
- How do you display all the attributes of a Node?
- Can you search for the **cpu** attribute of your node?

# Attributes, Templates, and Cookbook Dependencies

Writing an MOTD Cookbook

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Describe Cookbook Attribute files
  - Use ERB Templates in Chef
  - Explain Attribute Precedence
  - Describe Cookbook Metadata
  - Specify cookbook dependencies
  - Perform the cookbook creation, upload, and test loop

# The Problem and the Success Criteria

- **The Problem:** We need to add a message that appears at login that states:
  - "This server is property of COMPANY"
  - "This server is in-scope for PCI compliance" if the server is, in fact, in scope.
- **Success Criteria:** We see the message when we log in to the test node

# Possible Solutions

- We could use an MOTD cookbook to supply the 'Company' name as a node attribute and write out the message
- The line on PCI compliance only gets written if a 'PCI Compliance' attribute is set to 'true' for that Company

# We have a small problem...

- We don't have a node attribute for 'Company', nor if the server is in scope for 'PCI Compliance'
- Also, well factored cookbooks only contain the information relevant to their domain
  - We know we will likely have other things related to PCI (security settings, for example)
- So the best thing to do is create a separate PCI cookbook, and add our attribute there

# Solution

- We'll create an MOTD cookbook to write out the MOTD - this cookbook configures 'Company' name as a node attribute
- We'll also create a PCI cookbook which configures a 'PCI compliance' attribute - this attribute is then read by MOTD cookbook
- We'll create the MOTD cookbook first, then we'll create the PCI cookbook

# Exercise: Create a cookbook named ‘motd’

```
$ knife cookbook create motd
```

```
** Creating cookbook motd
** Creating README for cookbook: motd
** Creating CHANGELOG for cookbook: motd
** Creating metadata for cookbook: motd
```

# Exercise: Create a default.rb attribute file



**OPEN IN EDITOR:** cookbooks/motd/attributes/default.rb

```
default[ "motd" ][ "company" ] = "Chef"
```

**SAVE FILE!**

- Creates a new Node attribute: node[ "motd" ][ "company" ]
- Sets the values to the string "Chef"
- Note: there is no space between 'default' and '['
- The 'motd' in the attribute is just convention so you know the cookbook the attribute was set in. This is not an enforced syntax

# Exercise: Open the default recipe in your editor



**OPEN IN EDITOR:** cookbooks/motd/recipes/default.rb

```
#  
# Cookbook Name:: motd  
# Recipe:: default  
#  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

# What resource should we use?

- We could try and use a `cookbook_file` here, and rely on the file copy rules
  - Create a file per server, basically
- Obviously, that's dramatically inefficient
- Instead, we will render a **template** - a file that is a mixture of the contents we want, and embedded Ruby code

# Exercise: Add template resource for /etc/motd

- Use a **template** resource
- The **name** is "/etc/motd"
- The resource has two parameters
  - **source** is "motd.erb"
  - **mode** is "0644"

# The template[/etc/motd] resource



**OPEN IN EDITOR:** cookbooks/motd/recipes/default.rb

```
#  
# Cookbook Name:: motd  
# Recipe:: default  
#  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
template "/etc/motd" do  
  source "motd.erb"  
  mode "0644"  
end
```

**SAVE FILE!**

# Exercise: Open motd.erb in your Editor



**OPEN IN EDITOR:** cookbooks/motd/templates/default/motd.erb

```
This server is property of <%= node["motd"]["company"] %>
<% if node["pci"]["in_scope"] -%>
This server is in-scope for PCI compliance
<% end -%>
```

**SAVE FILE!**

- **"erb"** stands for "Embedded Ruby"

# Exercise: Open motd.erb in your Editor



**OPEN IN EDITOR:** cookbooks/motd/templates/default/motd.erb

```
This server is property of <%= node["motd"]["company"] %>
<% if node["pci"]["in_scope"] -%>
This server is in-scope for PCI compliance
<% end -%>
```

- To embed a value within an ERB template:
  - Start with <%=
  - Write your Ruby expression - most commonly a node attribute
  - End with %>

# Exercise: Open motd.erb in your Editor



**OPEN IN EDITOR:** cookbooks/motd/templates/default/motd.erb

```
This server is property of <%= node["motd"]["company"] %>
<% if node["pci"]["in_scope"] -%>
This server is in-scope for PCI compliance
<% end -%>
```

- You can use any Ruby construct in a template
  - Starting with <% will evaluate the expression, but not insert the result
  - Ending with -%> will not insert a line in the resulting file
  - Note there are no spaces in 'node[ "pci" ][ "in\_scope" ]'

## Templates Are Used For Almost All Configuration Files

- Templates are very flexible ways to create your configuration files
- Coupled with Chef's attribute precedence rules, you can create very effective, data-driven cookbooks

## Best Practice: Recipes contain the pattern, attributes supply the details

- Recipes contain the pattern for how to do something. ("How we deploy Tomcat")
- Attributes contain the details. ("What port do we run tomcat on?")

# Exercise: Upload the motd cookbook

```
$ knife cookbook upload motd
```

```
Uploading motd [ 0.1.0 ]  
Uploaded 1 cookbook.
```

# Exercise: Create a cookbook named ‘pci’

```
$ knife cookbook create pci
```

```
** Creating cookbook pci
** Creating README for cookbook: pci
** Creating CHANGELOG for cookbook: pci
** Creating metadata for cookbook: pci
```

# Exercise: Create a default.rb attribute file



**OPEN IN EDITOR:** cookbooks/pci/attributes/default.rb

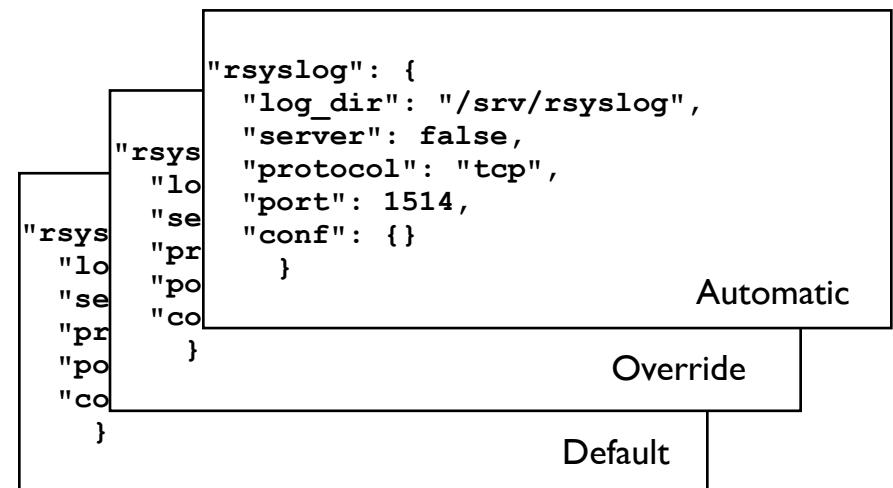
```
default[ "pci" ][ "in_scope" ] = true
```

**SAVE FILE!**

- Creates a new Node attribute: node[ "pci" ][ "in\_scope" ]
- Sets the value to the Ruby 'true' literal

# Node Attributes have several levels of precedence

- **Automatic** attributes are those discovered by Ohai
- **Override** attributes are the strongest way to set an attribute - use sparingly
- **Default** attributes are typically set in Cookbooks, Roles and Environments



# Best Practice: Always use ‘default’ attributes in your cookbooks

- When setting an attribute in a cookbook, it should (almost) always be a **default** attribute

# Exercise: Upload the PCI cookbook

```
$ knife cookbook upload pci
```

```
Uploading pci [0.1.0]
Uploaded 1 cookbook.
```

## Exercise: Add the motd recipe to your test node's run list

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

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

## Exercise: Add the motd recipe to your test node's run list

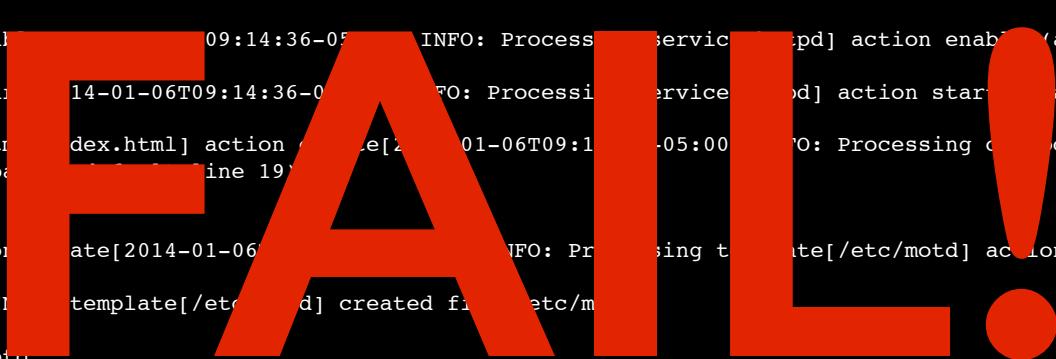
```
$ knife node show node1
```

```
Node Name:    node1
Environment:  _default
FQDN:        centos63.example.com
IP:          10.160.201.90
Run List:    recipe[apache], recipe[motd]
Roles:
Recipes:     apache
Platform:   centos 6.4
Tags:
```

# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
Starting Chef Client, version 11.8.2
...
- motd
Compiling Cookbooks...
Converging 4 resources
Recipe: apache::default
 * package[httpd] action install[2014-01-06T09:14:33-05:00] INFO: Processing package[httpd] action install (apache::default line 9)
 (up to date)
 * service[httpd] action enable[2014-01-06T09:14:36-05:00] INFO: Processing service[httpd] action enable (apache::default line 13)
 (up to date)
 * service[httpd] action start[2014-01-06T09:14:36-05:00] INFO: Processing service[httpd] action start (apache::default line 13)
 (up to date)
 * cookbook_file[/var/www/html/index.html] action create (apache::default line 19)
 (up to date)
Recipe: motd::default
 * template[/etc/motd] action create[2014-01-06T09:14:36-05:00] INFO: Processing template[/etc/motd] action create (motd::default line 9)
 [2014-01-06T09:14:37-05:00] INFO: template[/etc/motd] created file /etc/motd
 - create new file /etc/motd
=====
Error executing action `create` on resource 'template[/etc/motd]'
=====
Chef::Mixin::Template::TemplateError
-----
undefined method `[]' for nil:NilClass
```



# You probably see this at the bottom of your screen...

```
Template Context:  
-----  
on line #2  
1: This server is property of <%= node["motd"]["company"] %>  
2: <% if node["pci"]["in_scope"] -%>  
3: This server is in-scope for PCI compliance  
4: <% end -%>  
  
[2014-01-06T09:14:37-05:00] INFO: Running queued delayed notifications before re-raising exception  
[2014-01-06T09:14:37-05:00] ERROR: Running exception handlers  
[2014-01-06T09:14:37-05:00] ERROR: Exception handlers complete  
[2014-01-06T09:14:37-05:00] FATAL: Stacktrace dumped to /var/chef/cache/chef-stacktrace.out  
Chef Client failed. 0 resources updated  
[2014-01-06T09:14:37-05:00] INFO: Sending resource update report (run-id: c001b9d5-c2f6-4026-9cc6-ff0901aa563c)  
[2014-01-06T09:14:37-05:00] ERROR: "\xE2" from ASCII-8BIT to UTF-8  
[2014-01-06T09:14:37-05:00] FATAL: Chef::Exceptions::ChildConvergeError: Chef run process exited unsuccessfully (exit code 1)
```

# Stack Traces

- A stack trace tells you where in a program an error occurred
- They can (obviously) be very detailed
- They can also be intensely useful, as they supply the data you need to find a problem

# Scroll up

- In this case, Chef actually knows exactly what went wrong.
- Scroll up to find out.

```
=====
Error executing action `create` on resource 'template[/etc/motd]'
=====

Chef::Mixin::Template::TemplateError
-----
undefined method `[]' for nil:NilClass

Resource Declaration:
-----
# In /var/chef/cache/cookbooks/motd/recipes/default.rb

10: template "/etc/motd" do
11:   source "motd.erb"
12:   mode "0644"
13: end
```

# We do not have the attribute we are using in the conditional

```
INFO: Run List is [recipe[apache], recipe[motd]]  
INFO: Run List expands to [apache, motd]  
INFO: Starting Chef Run for node1.local  
INFO: Running start handlers  
INFO: Start handlers complete.  
resolving cookbooks for run list: ["apache", "motd"]  
INFO: Loading cookbooks [apache, motd]
```

- Can anyone guess why?
- We did not load the PCI cookbook!

# nil:NilClass error

- The bottom line is when you see this error

```
undefined method `[]' for nil:NilClass
```

- It most often means you tried to look up a node attribute that does not exist!

## Best Practice: Make sure cookbooks have default values

- If a cookbook needs an attribute to exist, it should
  1. Define a default value for it in an attribute file, or
  2. Depend on another cookbook that does
- **Never rely on an attribute being created manually**

## Exercise: Add a dependency on the PCI cookbook to the MOTD cookbook



**OPEN IN EDITOR:** cookbooks/motd/metadata.rb

```
maintainer          "YOUR_COMPANY_NAME"
maintainer_email    "YOUR_EMAIL"
license             "All rights reserved"
description         "Installs/Configures motd"
long_description   IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version             "0.1.0"
```

## Exercise: Add a dependency on the PCI cookbook to the MOTD cookbook



**OPEN IN EDITOR:** cookbooks/motd/metadata.rb

```
maintainer          "YOUR_COMPANY_NAME"
maintainer_email    "YOUR_EMAIL"
license             "All rights reserved"
description         "Installs/Configures motd"
long_description   IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version             "0.1.0"
depends "pci"
```

- Cookbooks that depend on other cookbooks will cause the dependent cookbook to be downloaded to the client, and evaluated

# Cookbook Metadata



**OPEN IN EDITOR:** cookbooks/motd/metadata.rb

```
maintainer          "YOUR_COMPANY_NAME"
maintainer_email    "YOUR_EMAIL"
license             "All rights reserved"
description         "Installs/Configures motd"
long_description   IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version             "0.1.0"
depends             "pci"
```

## Cookbook Attributes are applied for all downloaded cookbooks!

- Cookbooks downloaded as dependencies will have their attribute files evaluated
- Even if there is no recipe from the cookbook in the run-list

# Exercise: Upload the motd cookbook

```
$ knife cookbook upload motd
```

```
Uploading motd [ 0.1.0 ]  
Uploaded 1 cookbook.
```

# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
Forking chef instance to converge...
Starting Chef Client, version 11.8.2
*** Chef 11.8.2 ***
Chef-client pid: 15152
Run List is [recipe[apache], recipe[motd]]
Run List expands to apache, motd
Starting Chef Run for node1
...
- update content    file /etc/motd   e3b0c725428062e
  (new content is in /etc/motd, did not update)
- restore selinux security context
  (new context is in /etc/selinux/default, did not update)
Chef Run complete in 6.81877 seconds
Removing cookbooks/apache/files/default/index2.html from cache; it is no longer needed by chef-client.
Running report handlers
Report handlers complete
Chef Client finished, 1 resources updated
Sending resource update report (run-id: f923449e-cc7b-45f7-a9fb-ac1da3653557)
```



# Exercise: Check your work

```
chef@node1:~$ cat /etc/motd
```

This server is property of Chef  
This server is in-scope for PCI compliance

## Exercise: Show your test node's pci attribute

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

```
1 items found
```

```
node1:
```

```
  pci:
```

```
    in_scope: true
```

# Exercise: Set attribute to false



**OPEN IN EDITOR:** cookbooks/pci/attributes/default.rb

```
default[ "pci" ][ "in_scope" ] = false
```

**SAVE FILE!**

- Set the attribute to true

# Exercise: Upload the PCI cookbook

```
$ knife cookbook upload pci
```

```
Uploading pci [0.1.0]
Uploaded 1 cookbook.
```

# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
Forking chef instance to converge...
Starting Chef Client, version 11.8.2
*** Chef 11.8.2 ***
Chef-client pid: 15350
Run List is [recipe[apache], recipe[motd]]
Run List expands to [apache, motd]
Starting Chef Run for node1
...
template[/etc/motd] updated file contents /etc/motd
  - update content in file /etc/motd from 62ebb9 to d36e1f
    (current file is binary, diff output suppressed)
  - restore selinux security context

Chef Run complete in 5.634796214 seconds
Running report handlers
Report handlers complete
Chef Client finished, 1 resources updated
Sending resource update report (run-id: 06efc909-6740-49bb-971d-82657e066236)
```

# Exercise: Check your work

```
chef@node1:~$ cat /etc/motd
```

This server is property of Chef

## Exercise: Show your test node's pci attribute

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

```
1 items found
```

```
node1:  
  pci:  
    in_scope: false
```

# Congratulations!

- You now know the 3 most important resources in the history of configuration management
  - Package
  - Template
  - Service

# Review Questions

- What goes in a cookbook's attribute files?
- What are the 3 different levels of precedence?
- When do you need to specify a cookbook dependency?
- What does <%= mean, and where will you encounter it?
- What are the 3 most important resources in configuration management?

# Full Menu

## Day One

### Preparation

workstation, server, and node

### Write a Cookbook

### Review

### Cookbook

attributes, dependencies,  
templates, notifications

## Day Two

### Cookbook

template variables, notifications,  
and controlling idempotency

### Data Bags

### Search

### Roles

### Environments

### Community Cookbooks

### Additional Topics

# Attributes, Templates, and Cookbook Dependencies

Refactoring the Apache Cookbook

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Use the **execute** resource
  - Control idempotence manually with **not\_if** and **only\_if**
  - Navigate the Resources page on [docs.opscode.com](https://docs.opscode.com)
  - Describe the **directory** resource
  - Implement resource notifications
  - Explain what Template Variables are, and how to use them
  - Use Ruby variables, loops, and string expansion

# The Problem and the Success Criteria

- **The Problem:** We need to deploy multiple custom home pages running on different ports
- **Success Criteria:** Be able to view our custom home page

## Exercise: Change the cookbook's version number in the metadata



**OPEN IN EDITOR:** cookbooks/apache/metadata.rb

```
maintainer          "YOUR_COMPANY_NAME"
maintainer_email    "YOUR_EMAIL"
license             "All rights reserved"
description         "Installs/Configures apache"
long_description IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version            "0.2.0"
```

**SAVE FILE!**

- Major, Minor, Patch
- Semantic Versioning Policy: <http://semver.org/>

# Exercise: Create a default.rb attribute file



**OPEN IN EDITOR:** cookbooks/apache/attributes/default.rb

```
default[ "apache" ][ "sites" ][ "clowns" ] = { "port" => 80 }
default[ "apache" ][ "sites" ][ "bears" ] = { "port" => 81 }
```

**SAVE FILE!**

- We add information about the sites we need to deploy
  - One about Clowns, running on port 80
  - One about Bears, running on port 81

# Exercise: Open the default apache recipe in your editor



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

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

service "httpd" do
  action [ :enable, :start ]
end

node.default["apache"]["indexfile"] = "index2.html"
cookbook_file "/var/www/html/index.html" do
  source node["apache"]["indexfile"]
  mode "0644"
end

# Tasks:-
# Disable the default virtual host
# Iterate over the apache sites
# Set the document root
# Add a template for Apache virtual host configuration
# Add a directory resource to create the document_root
# Add a template resource for the virtual host's index.html
```

## Exercise: Use the 'execute' resource to disable the default Apache virtual host



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

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

# Disable the default virtual host
execute "mv /etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf.disabled" do
  only_if do
    File.exist?("/etc/httpd/conf.d/welcome.conf")
  end
  notifies :restart, "service[httpd]"
end

cookbook_file "/var/www/html/index.html" do
```

**SAVE FILE!**

- Renames the default site config file, but only if the file exists
- If the action succeeds, restart Apache

# Execute resources are generally not idempotent

- Chef will stop your run if a resource fails
- Most command line utilities are not idempotent and can only be run once - they assume a human being is interacting with, and understands, the state of the system
  - e.g. 'mv /foo/file1 /bar' will work the first time its run, but will fail the second time
- The result is - it's up to you to make execute resources idempotent

# Enter the `not_if` and `only_if` metaparameters

```
only_if do
  File.exist?("/etc/httpd/conf.d/welcome.conf")
end
```

- The `only_if` parameter causes the resources actions to be taken only if its argument returns true
- The `not_if` parameter is the opposite of `only_if` - the actions are taken only if its argument returns false

# Building the resource collection - revisited

- It is important to understand the difference between these two pieces of code

```
execute "start-apache" do
  not_if <code to check if apache is running>
  notifies :start, "service[httpd]"
end
```

This code is placed in the resource collection during the 'compile' phase and executed during the 'execute' phase

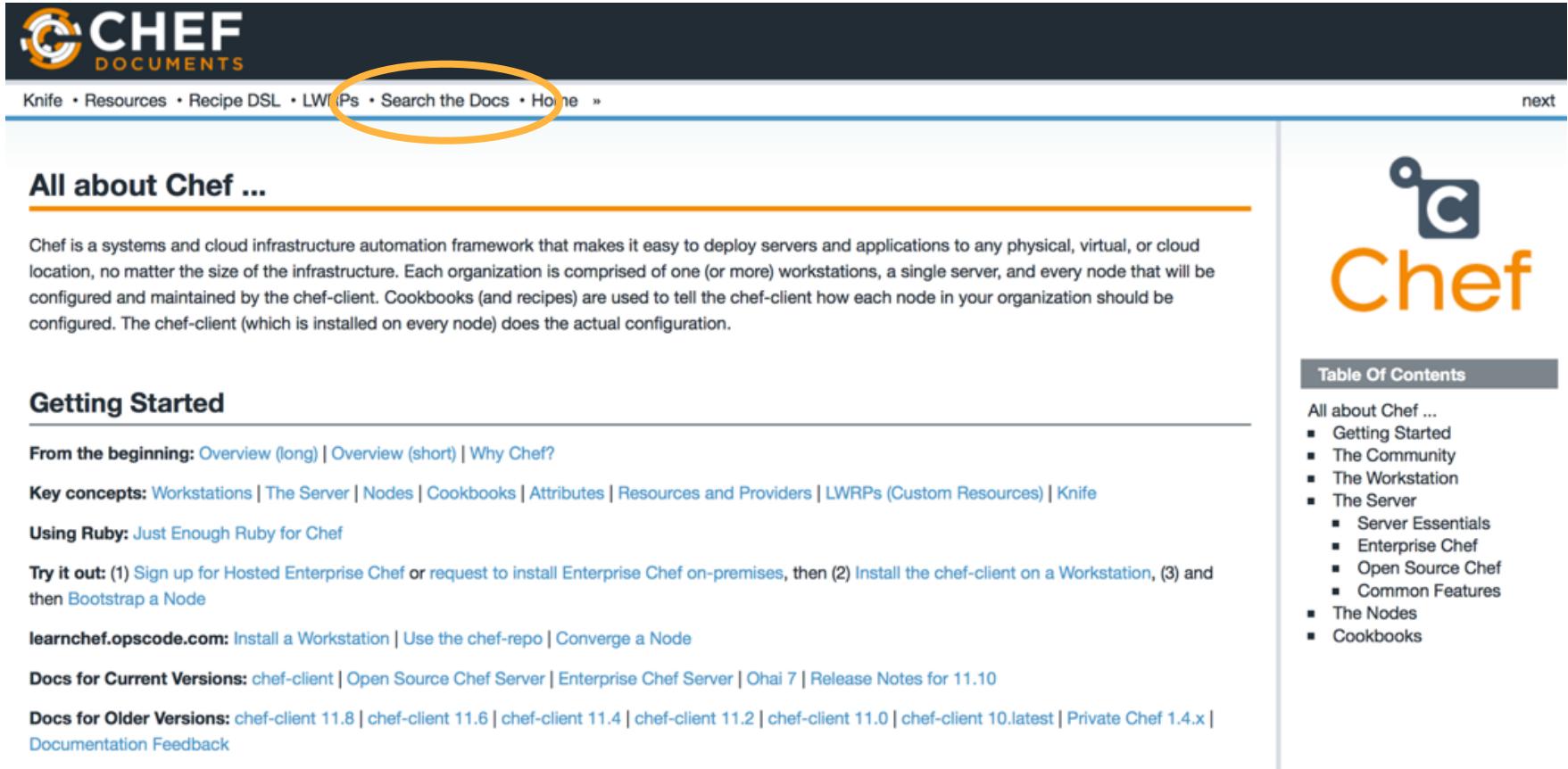
```
if !<code to check if apache is running>
  execute "start-apache" do
    notifies :start, "service[httpd]"
  end
end
```

This code is executed during the 'compile' phase before any Chef DSL  
Executing code during the compile phase could potentially prevent the resource from getting added to the resource collection

# Best Practice: The Chef Docs Site

- The Chef Docs Site is the home for all of the documentation about Chef.
  - It is very comprehensive
  - It has a page on every topic
- <http://docs.opscode.com>
- Let's use the docs to learn more about **not\_if** and **only\_if**

# Exercise: Search for more information about Resources



The screenshot shows the Chef Documentation website. At the top, there's a navigation bar with links: Knife, Resources, Recipe DSL, LWRPs, Search the Docs (which is highlighted with a yellow circle), and Home. Below the navigation bar, the page title is "All about Chef ...". The main content area contains several sections of text and links related to Chef's features and setup. On the right side, there's a sidebar titled "Table Of Contents" which lists various sections of the documentation.

**CHEF DOCUMENTS**

Knife • Resources • Recipe DSL • LWRPs • **Search the Docs** • Home »

All about Chef ...

Chef is a systems and cloud infrastructure automation framework that makes it easy to deploy servers and applications to any physical, virtual, or cloud location, no matter the size of the infrastructure. Each organization is comprised of one (or more) workstations, a single server, and every node that will be configured and maintained by the chef-client. Cookbooks (and recipes) are used to tell the chef-client how each node in your organization should be configured. The chef-client (which is installed on every node) does the actual configuration.

## Getting Started

**From the beginning:** [Overview \(long\)](#) | [Overview \(short\)](#) | [Why Chef?](#)

**Key concepts:** [Workstations](#) | [The Server](#) | [Nodes](#) | [Cookbooks](#) | [Attributes](#) | [Resources and Providers](#) | [LWRPs \(Custom Resources\)](#) | [Knife](#)

**Using Ruby:** [Just Enough Ruby for Chef](#)

**Try it out:** (1) [Sign up for Hosted Enterprise Chef](#) or [request to install Enterprise Chef on-premises](#), then (2) [Install the chef-client on a Workstation](#), (3) and then [Bootstrap a Node](#)

**learnchef.opscode.com:** [Install a Workstation](#) | [Use the chef-repo](#) | [Converge a Node](#)

**Docs for Current Versions:** [chef-client](#) | [Open Source Chef Server](#) | [Enterprise Chef Server](#) | [Ohai 7](#) | [Release Notes for 11.10](#)

**Docs for Older Versions:** [chef-client 11.8](#) | [chef-client 11.6](#) | [chef-client 11.4](#) | [chef-client 11.2](#) | [chef-client 11.0](#) | [chef-client 10.latest](#) | [Private Chef 1.4.x](#) | [Documentation Feedback](#)

## Table Of Contents

- All about Chef ...
- Getting Started
- The Community
- The Workstation
- The Server
  - Server Essentials
  - Enterprise Chef
  - Open Source Chef
  - Common Features
- The Nodes
- Cookbooks

- Search for "Resources"

# Exercise: Search for more information about Resources



## Search the Documentation for Chef

From here you can use a scoped Google search query to search all of the documentation about Chef that is located at docs.opscode.com. (This page requires JavaScript be enabled to view the search box.)

resources x Search

All    **Chef Documentation**    Cookbooks

About 706 results (0.32 seconds)    Sort by: Relevance

### [About Resources and Providers — Chef Docs](#)

[docs.opscode.com/resource.html](http://docs.opscode.com/resource.html)

If you want to see all of the information about **resources** in a single document, see : <http://docs.opscode.com/chef/resources.html>. Keep reading this page for ...

Labeled Chef ...

### [Resources and Providers Reference — Chef Single-page Topics](#)

[docs.opscode.com/chef/resources.html](http://docs.opscode.com/chef/resources.html)

A **resource** is a key part of a recipe. A resource defines the actions that can be taken, such as when a package should be installed, whether a service should be

...

Labeled Chef ...

- Find "Resources and Providers Reference"

# The Resources Page

## Resources and Providers Reference

A resource is a key part of a recipe. A resource defines the actions that can be taken, such as when a package should be installed, whether a service should be enabled or restarted, which groups, users, or groups of users should be created, where to put a collection of files, what the name of a new directory should be, and so on. During a chef-client run, each resource is identified and then associated with a provider. The provider then does the work to complete the action defined by the resource. Each resource is processed in the same order as they appear in a recipe. The chef-client ensures that the same actions are taken the same way everywhere and that actions produce the same result every time. A resource is implemented within a recipe using Ruby.

Where a resource represents a piece of the system (and its desired state), a provider defines the steps that are needed to bring that piece of the system from its current state into the desired state. These steps are de-coupled from the request itself. The request is made in a recipe and is defined by a lightweight resource. The steps are then defined by a lightweight provider.

The `Chef::Platform` class maps providers to platforms (and platform versions). Ohai, as part of every chef-client run, verifies the `platform` and `platform_version` attributes on each node. The chef-client then uses those values to identify the correct provider, build an instance of that provider, identify the current state of the resource, do the specified action, and then mark the resource as updated (if changes were made). For example, given the following resource:

```
directory "/tmp/folder" do
  owner "root"
  group "root"
  mode 0755
  action :create
end
```



### Table Of Contents

#### Resources and Providers

##### Reference

- Common Functionality for all Resources
  - Actions
  - Examples
  - Attributes
  - Examples
- Guards
  - Attributes
  - Arguments
  - not\_if Examples
  - only\_if Examples
- Lazy Attribute Evaluation
- Notifications
  - Notifications Timers
  - Notifies Syntax
    - Examples
  - Subscribes Syntax

# Notifications

```
notifies :restart, "service[httpd]"
```

- Resource Notifications in Chef are used to trigger an action on a resource when the current resources actions are successful.
- "If we delete the site, restart apache"
- The first argument is an action, and the second argument is the string representation of a given resource
- Like `not_if` and `only_if`, **notifies** is a resource metaparameter - any resource can notify any other

# Exercise: Iterate over each apache site



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

```
execute "mv /etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf.disabled" do
  only_if do
    File.exist?("/etc/httpd/conf.d/welcome.conf")
  end
  notifies :restart, "service[httpd]"
end

node.default["apache"]["indexfile"] = "index2.html"
cookbook_file "/var/www/index.html" do
  source node["apache"]["indexfile"]
  mode "0644"
end
```

- Delete the cookbook\_file resource

## Exercise: Iterate over each apache site and set document\_root



**OPEN IN EDITOR:** cookbooks/apache/recipes/default.rb

```
# Disable the default virtual host
execute "mv /etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf.disabled" do
  only_if do
    File.exist?("/etc/httpd/conf.d/welcome.conf")
  end
  notifies :restart, "service[httpd]"
end

# Iterate over the apache sites
node["apache"]["sites"].each do |site_name, site_data|
  # Set the document root
  document_root = "/srv/apache/#{site_name}"
```

**SAVE FILE!**

- `node["apache"]["sites"]` is a ruby hash, with keys and values

# Exercise: Iterate over each apache site

```
node["apache"]["sites"].each do |site_name, site_data|
  document_root = "/srv/apache/#{site_name}"
```

- Calling .each loops over each site

```
default["apache"]["sites"]["clowns"] = { "port" => 80 }
default["apache"]["sites"]["bears"] = { "port" => 81 }
```

- First pass
  - site\_name = 'clowns'
  - site\_data = { "port" => 80 }
- Second pass
  - site\_name = 'bears'
  - site\_data = { "port" => 81 }

# Exercise: Iterate over each apache site

```
node["apache"]["sites"].each do |site_name, site_data|
  document_root = "/srv/apache/#{site_name}"
```

- Create a variable called `document_root`
- `#{site_name}` means "insert the value of `site_name` here"
- First pass
  - The value is the string `"/srv/apache/clowns"`
- Second pass
  - The value is the string `"/srv/apache/bears"`

## Exercise: Add a template for Apache virtual host configuration

```
# Iterate over the apache sites
node["apache"]["sites"].each do |site_name, site_data|
# Set the document root
  document_root = "/srv/apache/#{site_name}"

# Add a template for Apache virtual host configuration
  template "/etc/httpd/conf.d/#{site_name}.conf" do
    source "custom.erb"
    mode "0644"
    variables(
      :document_root => document_root,
      :port => site_data["port"]
    )
    notifies :restart, "service[httpd]"
  end
```

# Template Variables

- Not all data you might need in a template is necessarily node attributes
- The **variables** parameter lets you pass in custom data for use in a template

## Exercise: Add a directory resource to create the document\_root

- Use a **directory** resource
- The **name** is `document_root`
- The resource has two parameters
  - **mode** is `"0755"`
  - **recursive** is `true`
- Use the Resources page on the Docs Site to read more about what **recursive** does.

# Exercise: Add the directory resource

```
# Add a template for Apache virtual host configuration
template "/etc/httpd/conf.d/#{site_name}.conf" do
  source "custom.erb"
  mode "0644"
  variables(
    :document_root => document_root,
    :port => site_data["port"]
  )
  notifies :restart, "service[httpd]"
end

# Add a directory resource to create the document_root
directory document_root do
  mode "0755"
  recursive true
end
```

## Exercise: Add a template resource to supply the index.html for the virtual host

```
# Add a directory resource to create the document_root
directory document_root do
  mode "0755"
  recursive true
end

# Add a template resource for the virtual host's index.html
template "#{document_root}/index.html" do
  source "index.html.erb"
  mode "0644"
  variables(
    :site_name => site_name,
    :port => site_data["port"]
  )
end
end
```

# Don't forget the last “end”

```
# Add a template resource for the virtual host's index.html
template "#{document_root}/index.html" do
  source "index.html.erb"
  mode "0644"
  variables(
    :site_name => site_name,
    :port => site_data["port"]
  )
end
end
```

See the correct, whole file at <https://gist.github.com/8954699>

# Exercise: Add custom.erb to your templates directory



**OPEN IN EDITOR:** cookbooks/apache/templates/default/custom.erb

- Note the two **template variables** are prefixed with an @ symbol
- Our first conditional if!
- If you are feeling hardcore, type it
- <https://gist.github.com/8955103>

```
<% if @port != 80 -%>
  Listen <%= @port %>
<% end -%>

<VirtualHost *:<%= @port %>>
  ServerAdmin webmaster@localhost

  DocumentRoot <%= @document_root %>
  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
  <Directory <%= @document_root %>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
</VirtualHost>
```

**SAVE FILE!**

# Exercise: Add index.html.erb to your templates directory



**OPEN IN EDITOR:** cookbooks/apache/templates/default/index.html.erb

```
<html>
  <body>
    <h1>Welcome to <%= node[ "motd" ][ "company" ] %></h1>
    <h2>We love <%= @site_name %></h2>
    <%= node[ "ipaddress" ] %>:<%= @port %>
  </body>
</html>
```

**SAVE FILE!**

- Note the two **template variables** are prefixed with an @ symbol
- <https://gist.github.com/8955080>

# Exercise: Upload the Apache cookbook

```
$ knife cookbook upload apache
```

Uploading apache

[ 0 . 2 . 0 ]

Uploaded 1 cookbook.

# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
...
=====
Error executing action `restart` on resource 'service[httpd]'
=====
```

```
Mixlib::ShellOut::ShellCommandFailed
-----
Expected process to exit with [0] but received '1'
---- Begin output of /sbin/service httpd start ----
STDOUT: Starting httpd: [FAILED]
STDERR: Syntax error on line 1 in /etc/httpd/conf.d/admin.conf:
<Directory> directive missing!
---- End output of /sbin/service httpd start ----
Ran /sbin/service httpd start 1 times, turned 1
```

```
Resource Declaration:
```

```
# In /var/chef/cache/cookbooks/apache/recipes/default.rb

43: service "httpd" do
44:   action [ :enable, :start ]
45: end
...
```

FAIL!

# Exercise: Fix typo in custom.erb



**OPEN IN EDITOR:** cookbooks/apache/templates/default/custom.erb

```
<% if @port != 80 -%>
  Listen <%= @port %>
<% end -%>

<VirtualHost *:<%= @port %>>
  ServerAdmin webmaster@localhost

  DocumentRoot <%= @document_root %>
  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
  <Directory <%= @document_root %>>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride None
    Order allow,deny
    allow from all
  </Directory>
</VirtualHost>
```



**SAVE FILE!**

290

# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
...
=====
Error executing action `restart` on resource 'service[httpd]'

=====
Mixlib::ShellOut::ShellCommandFailed
-----
Expected process to exit with [0], but received '1'
---- Begin output of /sbin/service httpd start ----
STDOUT: Starting httpd: [FAILED]
STDERR: Syntax error on line 1 in /etc/httpd/conf.d/admin.conf:
<Directory> directive missing required 'Path' parameter
---- End output of /sbin/service httpd start ----
Ran /sbin/service httpd start 1

Resource Declaration:
-----
# In /var/chef/cache/cookbooks/apache/recipes/default.rb

43: service "httpd" do
44:   action [ :enable, :start ]
45: end
...
...
```

FAIL!

# Exercise: Move the service resource to end of recipe

```
# Add a template resource for the virtual host's index.html
template "#{document_root}/index.html" do
  source "index.html.erb"
  mode "0644"
  variables(
    :site_name => site_name,
    :port => site_data["port"]
  )
end
end

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

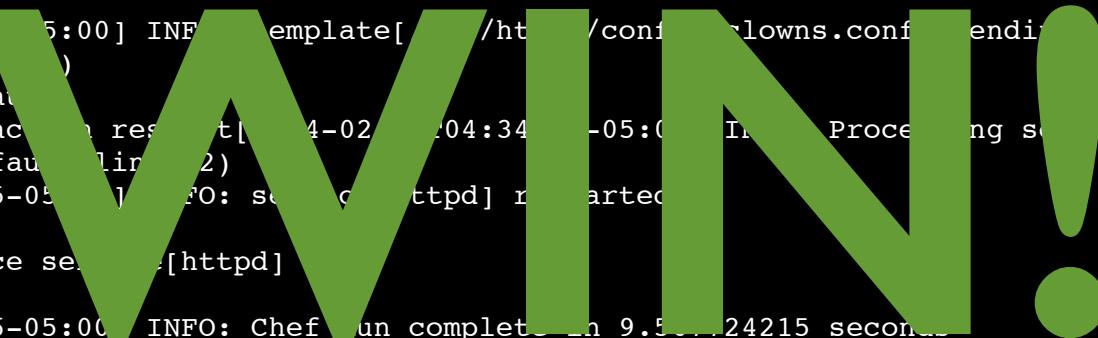
# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
- start service service[httpd]

Recipe: motd::default
 * template[/etc/motd] action create[2014-02-11T04:34:23-05:00] INFO: Processing template[/etc/motd]
action create (motd::default line 9)
 (up to date)
[2014-02-11T04:34:25-05:00] INFO: template[/etc/motd] action create[2014-02-11T04:34:25-05:00] INFO: Processing service[httpd] action restart
service[httpd] (delayed)
Recipe: apache::default
 * service[httpd] action restart[2014-02-11T04:34:25-05:00] INFO: Processing service[httpd] action
restart (apache::default line 2)
[2014-02-11T04:34:25-05:00] INFO: service[httpd] restarted
 - restart service service[httpd]

[2014-02-11T04:34:26-05:00] INFO: Chef run completed in 9.50024215 seconds
[2014-02-11T04:34:26-05:00] INFO: Running report handlers
[2014-02-11T04:34:26-05:00] INFO: Report handlers complete
Chef Client finished, 5 resources updated
[2014-02-11T04:34:26-05:00] INFO: Sending resource update report (run-id: d1e6c73e-9406-46b8-b022-d7bdf0868432)
```



# Exercise: Re-run the Chef Client

```
chef@node1:~$ sudo chef-client
```

```
[2014-01-07T05:13:43-05:00] INFO: execute[mv /etc/httpd/conf.d/welcome.conf /etc/httpd/conf.d/welcome.conf.disabled] sending restart action to service[httpd] (delayed)
Recipe: apache::default
 * service[httpd] action restart[2014-01-07T05:13:43-05:00] INFO: Processing service[httpd] action
restart (apache::default line 13)
[2014-01-07T05:13:45-05:00] INFO: service[httpd] restarted
 - restart service service[httpd]

[2014-01-07T05:13:46-05:00] INFO: Chef Run complete in 10.389166164 seconds
[2014-01-07T05:13:46-05:00] INFO: Removing cookbooks/apache/files/default/index2.html from the cache;
it is no longer needed by chef-client.
[2014-01-07T05:13:46-05:00] INFO: Running report handlers
[2014-01-07T05:13:46-05:00] INFO: Report handlers complete
Chef Client finished, 8 resources updated
[2014-01-07T05:13:46-05:00] INFO: Sending resource update report (run-id: b929bc5f-4465-4669-8027-
b7d7b44741d9)
```

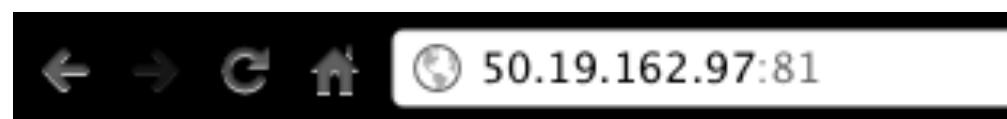
# Exercise: Verify the two sites are working!



Welcome to opscode

We love clowns

10.4.25.155:80



Welcome to opscode

We love bears

10.4.25.155:81

## Best Practice: Recipes contain the pattern, attributes supply the details

- Recipes contain the pattern for how to do something. ("How we deploy apache virtual hosts")
- Attributes contain the details. ("What virtual hosts should we deploy?")

# Review Questions

- How do you control the idempotence of an Execute resource?
- Where can you learn the details about all the core resources in Chef?
- What is a notification?
- What is a template variable?
- What does `#{}{foo}` do in a Ruby string?

# Recipe Inclusion, Data Bags, and Search

Writing a Users cookbook

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Explain what Data Bags are, and how they are used
  - Describe the User and Group resources
  - Use the `include_recipe` directive
  - Describe the role Search plays in recipes

# The Problem and the Success Criteria

- **The Problem:** Employees should have local user accounts created on servers, along with custom groups
- **Success Criteria:** We can add new employees and groups to servers dynamically

# Where should we store the user data?

- As we've seen, we could start by storing information about users as Node Attributes
- This is sort of a bummer, because we would be duplicating a lot of information - every user in the company would be stored in every Node object!
- Additionally, it would be very hard to integrate such a solution with another source of truth about users

# Introducing Data Bags

- A data bag is a **container** for **items** that represent information about your infrastructure that is not tied to a single node
- Examples
  - Users
  - Groups
  - Application Release Information

# Make a data\_bags directory

```
$ mkdir -p data_bags/users
```

( No output )

# Exercise: Create a data bag named users

```
$ knife data_bag create users
```

```
Created data_bag[users]
```

# Exercise: Create a user item in the users data bag



**OPEN IN EDITOR:** data\_bags/users/bobo.json

```
{  
  "id": "bobo",  
  "comment": "Bobo T. Clown",  
  "uid": 2000,  
  "gid": 0,  
  "home": "/home/bobo",  
  "shell": "/bin/bash"  
}
```

**SAVE FILE!**

# Exercise: Create the data bag item

```
$ knife data_bag from file users bobo.json
```

```
Updated data_bag_item[users::bobo]
```

# Exercise: Create another user in the users data bag



**OPEN IN EDITOR:** data\_bags/users/frank.json

```
{  
  "id": "frank",  
  "comment": "Frank Belson",  
  "uid": 2001,  
  "gid": 0,  
  "home": "/home/frank",  
  "shell": "/bin/bash"  
}
```

**SAVE FILE!**

# Exercise: Create the data bag item

```
$ knife data_bag from file users frank.json
```

```
Updated data_bag_item[users::frank]
```

# Exercise: Show all the items in users data bag

```
$ knife search users "*:*"
```

```
2 items found

chef_type: data_bag_item
comment: Frank Belson
data_bag: users
gid: 0
home: /home/frank
id: frank
shell: /bin/bash
uid: 2001

chef_type: data_bag_item
comment: Bobo T. Clown
data_bag: users
gid: 0
home: /home/bobo
id: bobo
shell: /bin/bash
uid: 2000
```

# Exercise: Find Bobo's shell in Chef

```
$ knife search users "id:bobo" -a shell
```

```
1 items found
```

```
data_bag_item_users_bobo:  
  shell: /bin/bash
```

# Exercise: Create a data bag named groups

```
$ mkdir data_bags/groups  
$ knife data_bag create groups
```

```
Created data_bag[groups]
```

# Exercise: Create a group item in the group data bag



**OPEN IN EDITOR:** data\_bags/groups/clowns.json

```
{  
  "id": "clowns",  
  "gid": 3000,  
  "members": [ "bobo", "frank" ]  
}
```

**SAVE FILE!**

# Exercise: Create the data bag item

```
$ knife data_bag from file groups clowns.json
```

```
Updated data_bag_item[groups::clowns]
```

# Exercise: Show all the groups in Chef

```
$ knife search groups "*:*"
```

```
1 items found

chef_type: data_bag_item
data_bag: groups
gid: 3000
id: clowns
members:
  bobo
  frank
```

# Exercise: Create a cookbook named ‘users’

```
$ knife cookbook create users
```

```
** Creating cookbook users
** Creating README for cookbook: users
** Creating CHANGELOG for cookbook: users
** Creating metadata for cookbook: users
```

# Exercise: Open the default recipe in your editor



**OPEN IN EDITOR:** cookbooks/users/recipes/default.rb

```
#  
# Cookbook Name:: users  
# Recipe:: default  
#  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

**SAVE FILE!**

# Exercise: Open the default recipe in your editor

```
search( "users", "*:*" ).each do |user_data|
  user user_data[ "id" ] do
    comment user_data[ "comment" ]
    uid user_data[ "uid" ]
    gid user_data[ "gid" ]
    home user_data[ "home" ]
    shell user_data[ "shell" ]
  end
end

include_recipe "users::groups"
```

- We use the same search we just tried with Knife in the recipe
- Each item is bound to `user_data`
- Use the Chef Docs for information about the `user` resource

# Exercise: Open the default recipe in your editor

```
search("users", "*:*").each do |user_data|
  user user_data["id"] do
    comment user_data["comment"]
    uid user_data["uid"]
    gid user_data["gid"]
    home user_data["home"]
    shell user_data["shell"]
  end
end

include_recipe "users::groups"
```

- **include\_recipe** ensures another recipes resources are complete before we continue
- Chef will only include each recipe once

# Best Practice: Use `include_recipe` liberally

- If there is a pre-requisite for your recipe that resides in another recipe (the JVM existing for your Java application, for example)
  - Always use `include_recipe` to include it specifically, even if you put it in a run list

## Exercise: Open the users::groups recipe in your editor



**OPEN IN EDITOR:** cookbooks/users/recipes/groups.rb

```
search( "groups", "*:*" ).each do |group_data|
  group group_data[ "id" ] do
    gid group_data[ "gid" ]
    members group_data[ "members" ]
  end
end
```

**SAVE FILE!**

- This file follows the same pattern as the default users recipe
- Use the Chef Docs for information about the **group** resource

# Exercise: Upload the users cookbook

```
$ knife cookbook upload users
```

```
Uploading users [ 0.1.0 ]
Uploaded 1 cookbook.
```

## Exercise: Add the users recipe to your test node's run list

```
$ knife node run_list add node1 'recipe[users]'
```

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

# Exercise: Re-run the Chef Client

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

```
...
Recipe: users::default
 * user[bobo] action create[2014-01-07T06:16:26-05:00] INFO: Processing user[bobo] action create
(users::default line 10)
[2014-01-07T06:16:26-05:00] INFO: user[bobo] created
      - create user user[bobo]

 * user[frank] action create[2014-01-07T06:16:26-05:00] INFO: Processing user[frank] action create
(users::default line 10)
[2014-01-07T06:16:27-05:00] INFO: user[frank] created
      - create user user[frank]

Recipe: users::groups
 * group[circus] action create[2014-01-07T06:16:27-05:00] INFO: Processing group[circus] action
create (users::groups line 2)
[2014-01-07T06:16:27-05:00] INFO: group[circus] created
      - create group[circus]
...
```

# Exercise: Verify the users and groups exist

```
chef@node1:~$ cat /etc/passwd
```

```
frank:x:2001:0:Frank Belson:/home/frank:/bin/bash
bobo:x:2000:0:Bobo T. Clown:/home/bobo:/bin/bash
```

```
chef@node1:~$ cat /etc/group
```

```
clowns:x:3000:bobo,frank
```

# Let's review real quick..

- We just created a centralized user and group repository, from scratch
  - (That's kind of like what LDAP and Active Directory do, only they are, well, fancier.)
- Between Data Bags and Node Attribute precedence, Chef provides a plethora of ways to inform the patterns you use to configure your infrastructure

# Review Questions

- What are Data Bags?
- How are they used?
- What does the User resource do?
- What is `include_recipe`, and why is it useful?
- How does search work inside a recipe?
- What other applications do you see for search?
- How could we have used Data Bags in the refactored Apache recipe?
- Where would you go to find out more?

# Roles

Role-based Attributes and Merge Order Precedence

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Explain what Roles are, and how they are used to provide clarity
  - Discuss the Role Ruby DSL
  - Show a Role with Knife
  - Explain how merge order affects the precedence hierarchy
  - Describe nested Roles

# What is a Role?

- So far, we've been just adding recipes directly to a single node
- But that's not how your infrastructure works - think about how you refer to servers
  - "***It's a web server***"
  - "***It's a database server***"
  - "***It's a monitoring server***"

# What is a Role?

- Roles allow you to conveniently encapsulate the run lists and attributes required for a server to "be" what you already think it is
- In practice, Roles make it **easy to configure many nodes identically** without repeating yourself each time

# Best Practice: Roles live in your chef-repo

- Like Data Bags, you have options with how to create a Role
- The best practice is that all of your Roles live in the roles directory of your chef-repo
- They can be created via the API and Knife, but it's nice to be able to see them evolve in your source control history

# Exercise: Create the webserver role



**OPEN IN EDITOR:** roles/webserver.rb

- A Role has a:
  - name
  - description
  - run\_list

```
name "webserver"
description "Web Server"
run_list "recipe[apache]"
default_attributes({
  "apache" => {
    "sites" => {
      "admin" => {
        "port" => 8000
      }
    }
  }
})
```

**SAVE FILE!**

# Exercise: Create the webserver role



**OPEN IN EDITOR:** roles/webserver.rb

- You can set default node attributes within a role.

```
name "webserver"
description "Web Server"
run_list "recipe[apache]"
default_attributes({
  "apache" => {
    "sites" => {
      "admin" => {
        "port" => 8000
      }
    }
  }
})
```

**SAVE FILE!**

# Exercise: Create the role

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

Updated Role webserver!

# Exercise: Show the role with knife

```
$ knife role show webserver
```

```
chef_type:           role
default_attributes:
  apache:
    sites:
      admin:
        port: 8000
description:        Web Server
env_run_lists:
json_class:         Chef::Role
name:               webserver
override_attributes:
run_list:           recipe[apache]
```

## Exercise: Search for roles with recipe[apache] in their run list

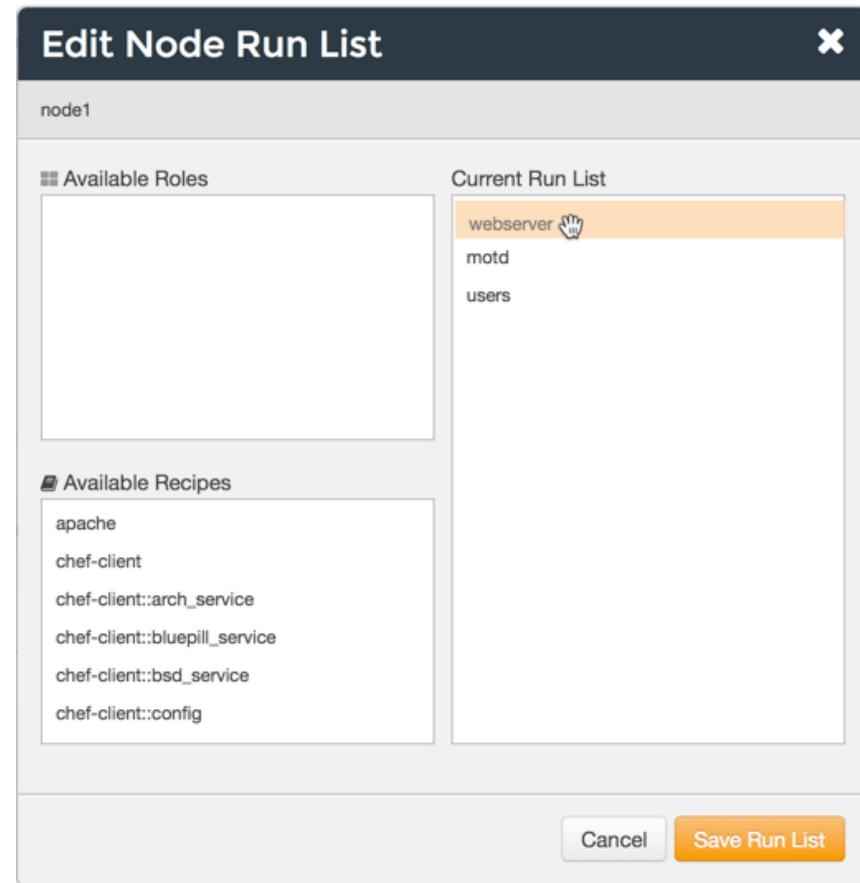
```
$ knife search role "run_list:recipe\[apache\]"
```

```
1 items found

chef_type:          role
default_attributes:
  apache:
    sites:
      admin:
        port:  8000
description:        Web Server
env_run_lists:
json_class:         Chef::Role
name:               webserver
override_attributes:
run_list:
```

## Exercise: Replace recipe[apache] with role[webserver] in run list

- Click the ‘Nodes’ tab then select node ‘node1’
- Click ‘Edit Run List’ from left navigation bar
- Drag ‘Apache’ over from ‘Current Run List’ to ‘Available Recipes’
- Drag ‘webserver’ over from ‘Available Roles’ to the top of ‘Current Run List’
- Click ‘Save Run List’



# Exercise: Re-run the Chef Client

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

```
INFO: *** Chef 11.8.2 ***
INFO: Chef-client pid: 5634
INFO: Run List is [role[webserver], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
```

# Exercise: Re-run the Chef Client

```
* directory[/srv/apache/admin] action create[2014-01-07T06:39:51-05:00] INFO: Processing directory[/srv/apache/admin] action
create (apache::default line 53)
[2014-01-07T06:39:51-05:00] INFO: directory[/srv/apache/admin] created directory /srv/apache/admin

- create new directory /srv/apache/admin[2014-01-07T06:39:51-05:00] INFO: directory[/srv/apache/admin] mode changed to 755

- change mode from '' to '0755'
- restore selinux security context

* template[/srv/apache/admin/index.html] action create[2014-01-07T06:39:51-05:00] INFO: Processing template[/srv/apache/
admin/index.html] action create (apache::default line 58)
[2014-01-07T06:39:51-05:00] INFO: template[/srv/apache/admin/index.html] created file /srv/apache/admin/index.html

- create new file /srv/apache/admin/index.html[2014-01-07T06:39:51-05:00] INFO: template[/srv/apache/admin/index.html]
updated file contents /srv/apache/admin/index.html

- update content in file /srv/apache/admin/index.html from none to 8abbb3
  --- /srv/apache/admin/index.html      2014-01-07 06:39:51.762119942 -0500
  +++ /tmp/chef-rendered-template20140107-17953-lsuqbi      2014-01-07 06:39:51.764120008 -0500
  @@ -1,8 @@
  +<html>
  +  <body>
  +    <h1>Welcome to Chef</h1>
  +    <h2>We love admin</h2>
  +    10.160.201.90:8000
  +  </body>
  +</html>[2014-01-07T06:39:51-05:00] INFO: template[/srv/apache/admin/index.html] mode changed to 644

- change mode from '' to '0644'
```

# Node Attributes that are hashes are merged

- The apache cookbooks attribute file contains:

```
default[ "apache" ][ "sites" ][ "clowns" ] = { "port" => 80 }
default[ "apache" ][ "sites" ][ "bears" ] = { "port" => 81 }
```

- While our role has...

```
default_attributes( {
  "apache" => {
    "sites" => {
      "admin" => {
        "port" => 8000
      }
    }
  }
})
```

## Exercise: Display the apache.sites attribute on all nodes with webserver role

```
$ knife search node "role:webserver" -a apache.sites
```

```
1 items found
```

```
node1:  
  apache.sites:  
    admin:  
      port: 8000  
    bears:  
      port: 81  
    clowns:  
      port: 80
```

# Exercise: Edit the webserver role



**OPEN IN EDITOR:** roles/webserver.rb

- Do not forget the **comma** after the admin site
- Change the value of the bears site to be 8081

```
default_attributes({  
  "apache" => {  
    "sites" => {  
      "admin" => {  
        "port" => 8000  
      },  
      "bears" => {  
        "port" => 8081  
      }  
    }  
  }  
})
```

**SAVE FILE!**

# Exercise: Create the role

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

Updated Role webserver!

# Exercise: Re-run the Chef Client

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

```
...
[2014-01-07T06:57:48-05:00] INFO: template[/etc/httpd/conf.d/bears.conf] updated file contents /etc/
httpd/conf.d/bears.conf

- update content in file /etc/httpd/conf.d/bears.conf from 30610b to 92e1e4
  --- /etc/httpd/conf.d/bears.conf2014-01-07 05:13:42.450113970 -0500
  +++ /tmp/chef-rendered-template20140107-18263-5563t5 2014-01-07 06:57:48.091397610 -0500
  @@ -1,6 +1,6 @@
- Listen 81
+ Listen 8081

-<VirtualHost *:81>
+<VirtualHost *:8081>
  ServerAdmin webmaster@localhost

  DocumentRoot /srv/apache/bears
...
```

## Exercise: Display the apache.sites attribute on all nodes with the webserver role

```
$ knife search node 'role:webserver' -a apache.sites
```

```
1 items found
```

```
node1:  
  apache.sites:  
    admin:  
      port: 8000  
    bears:  
      port: 8081  
    clowns:  
      port: 80
```



When you combine merge  
order and precedence  
rules, you get this:

# Merge Order and Precedence

	Attribute Files	Node / Recipe	Environment	Role
default	1	2	3	4
force_default	5	6		
normal	7	8		
override	9	10	12	11
force_override	13	14		
automatic			15	

# Best Practice: Roles get default attributes

- While it is awesome that you can use overrides, in practice there is little need
- If you always set **default** node attributes in your cookbook attribute files
- You can almost **always** set default node attributes in your role, and let merge order do the rest

# Best Practice: Have "base" roles

- In addition to obvious roles, such as "webserver", it is a common practice to group any functionality that "goes together" in a role
- The most common example here is a **base** role, where you include all the recipes that should be run on every node

# Exercise: Create the base role



**OPEN IN EDITOR:** roles/base.rb

```
name "base"
description "Base Server Role"
run_list "recipe[motd]", "recipe[users]"
```

**SAVE FILE!**

# Exercise: Create the role

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

Updated Role base!

## Exercise: Add the base role to the webserver role's run list



**OPEN IN EDITOR:** roles/webserver.rb

```
name "webserver"
description "Web Server"
run_list "role[base]", "recipe[apache]"
default_attributes(
  "apache" => {
```

- Put `role[base]` at the front of the `run_list`

**SAVE FILE!**

# Exercise: Update the role

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

Updated Role webserver!

# Exercise: Re-run the Chef Client

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

```
INFO: *** Chef 11.8.2 ***
Chef-client pid: 5634
INFO: Run List is [role[webserver], recipe[motd],
recipe[users]]
INFO: Run List expands to [motd, users, apache]
```

## Best Practice: Be explicit about what you need or expect

- Chef will only execute a recipe the first time it appears in the run list
- So be explicit about your needs and expectations - either by nesting roles or using `include_recipe`

## Exercise: Set the run list to just role[webserver]

- Remove all the entries in the run list other than role[ webserver ]

# Review Questions

- What is a Role?
- What makes for a "good" role?
- How do you search for roles with a given recipe in their run list?
- How many times will Chef execute a recipe in the same run?

# Environments

Cookbook Version Constraints and Override Attributes

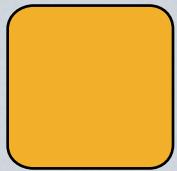
v2.0.3

# Lesson Objectives

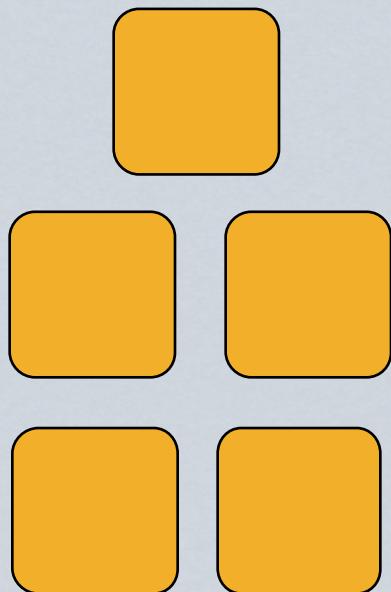
- After completing the lesson, you will be able to
  - Describe what an Environment is, and how it is different from an Organization
  - Set cookbook version constraints
  - Explain when to set attributes in an environment

# Environments

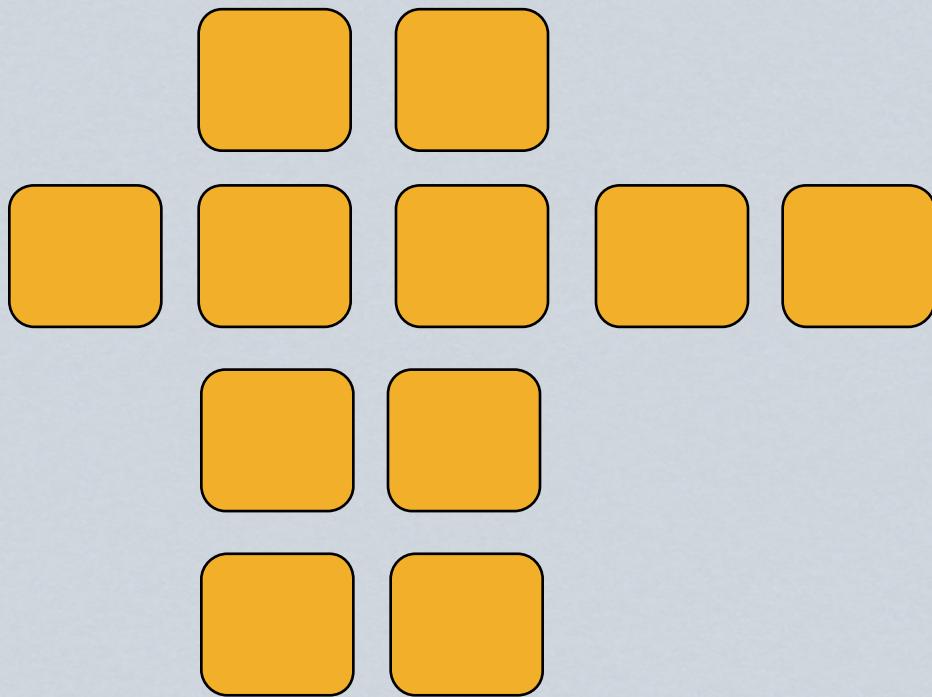
Development



Staging



Production



Organization

# Environments

- Every Organization starts with a single environment
- Environments reflect your patterns and workflow
  - Development
  - Test
  - Staging
  - Production
  - etc.

# Environments Define Policy

- Each environment may include attributes necessary for configuring the infrastructure in that environment
  - Production needs certain Yum repos
  - QA needs different Yum repos
  - The version of the Chef cookbooks to be used

# Environment Best Practice

- We cannot share cookbooks between organizations
- Best Practice: If you need to share cookbooks or roles, you likely want an Environment rather than an organization
- Environments allow for isolating resources within a single organization

## Exercise: Use knife to show the available cookbook versions

```
$ knife cookbook show apache
```

```
apache      0.2.0      0.1.0
```

# Exercise: List current environments

```
$ knife environment list
```

```
_default
```

- The `_default` environment is read-only, and sets no policy at all

# Make an environments directory

```
$ mkdir environments
```

(No output)

# Exercise: Create a dev environment



**OPEN IN EDITOR:** environments/dev.rb

```
name "dev"  
description "For developers!"  
cookbook "apache", "= 0.2.0"
```

**SAVE FILE!**

- Environments have **names**
- Environments have a **description**
- Environments can have one or more **cookbook** constraints

# Cookbook Version Constraints

- = Equal to
- There are other options but equality is the recommended practice.
- Learn more at [http://docs.opscode.com/chef/essentials\\_cookbook\\_versions.html](http://docs.opscode.com/chef/essentials_cookbook_versions.html)

# Exercise: Create the dev environment

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

```
Updated Environment dev
```

# Exercise: Show your dev environment

```
$ knife environment show dev
```

```
chef_type:          environment
cookbook_versions:
  apache:  = 0.2.0
default_attributes:
description:       For developers!
json_class:        Chef::Environment
name:              dev
override_attributes:
```

# Exercise: Change your node's environment to "dev"

- Click the 'Nodes' tab then select node 'node1'
- Select dev from the 'Environments' drop-down list
- Click 'Save'

The screenshot shows the Chef Manage web interface. The top navigation bar includes 'Feedback', 'Organization johnf241213', and a user sign-in message. Below the navigation is a search bar labeled 'Search Nodes...'. The main content area is titled 'Showing All Nodes' and lists one node: 'node1' (Platform: centos, FQDN: centos63.example.com, IP Address: 10.160.201.90, Uptime: 9 hours, Last Check-In: 10 minutes ago, Environment: \_default). On the left sidebar under 'Nodes', there are options: Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The 'Details' tab is selected for node1. A callout box highlights the 'Environment:' dropdown, which has been changed from '\_default' to 'dev'. The message in the callout box states: 'Node environment changed from \_default to dev.' with 'Cancel' and 'Save' buttons.

# Exercise: Re-run the Chef Client

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

```
INFO: Chef Run complete in 1.587776095 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

# Exercise: Create a production environment



**OPEN IN EDITOR:** environments/production.rb

- Make sure the apache cookbook is set to version 0.1.0
- Set an override attribute for being in scope - no matter what, you are in scope

```
name "production"
description "For Prods!"
cookbook "apache", "= 0.1.0"
override_attributes({
  "pci" => {
    "in_scope" => true
  }
})
```

**SAVE FILE!**

# Exercise: Create the production environment

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

Updated Environment production

## Exercise: Change your node's environment to "production"

- Click the 'Nodes' tab then select node 'node1'
- Select production from the 'Environments' drop-down list
- Click 'Save'

The screenshot shows the Chef Manage web interface. The top navigation bar includes 'Nodes', 'Reports', 'Policy', and 'Administration' tabs, with 'Nodes' being the active tab. The top right corner shows 'Feedback | Organization johnf241213 | Signed in as John Fitzpatrick | 0'. Below the navigation is a search bar labeled 'Search Nodes...' with a magnifying glass icon.

The main content area displays a table titled 'Showing All Nodes' with columns: Node Name, Platform, FQDN, IP Address, Uptime, Last Check-In, Environment, and Actions. A single row is selected for 'node1', which has a yellow background. The details for 'node1' are shown in a modal window at the bottom:

- Node: node1
- Details tab is selected, showing:
  - Last Check In: 4 Minutes Ago (2014-01-07 13:32:58 UTC)
  - Uptime: 9 Hours (Since 2014-01-07 04:38:23 UTC)
- Attributes tab is visible.
- Permissions tab is visible.
- Environment: A dropdown menu is open, showing 'production' as the selected option. A yellow status message below the dropdown says 'Node environment changed from dev to production.' with 'Cancel' and 'Save' buttons.
- Platforms: centos
- FQDN: centos63.example.com
- IP Address: 10.160.201.90

# Exercise: Re-run the Chef Client

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

```
INFO: Loading cookbooks [apache, motd, pci, users]
Synchronizing Cookbooks:
...
Recipe: motd::default
 * template[/etc/motd] action create[2014-01-07T08:40:00-05:00] INFO: Processing template[/etc/motd] action create
(motd::default line 9)
[2014-01-07T08:40:00-05:00] INFO: template[/etc/motd] backed up to /var/chef/backup/etc/motd.chef-20140107084000.070961
[2014-01-07T08:40:00-05:00] INFO: template[/etc/motd] updated file contents /etc/motd

 - update content in file /etc/motd from d36e1f to 62ebb9
   (current file is binary, diff output suppressed)
...
 * cookbook_file[/var/www/index.html] action create[2014-01-07T08:40:05-05:00] INFO: Processing cookbook_file[/var/
www/index.html] action create (apache::default line 18)
  (up to date)
[2014-01-07T08:40:06-05:00] INFO: Chef Run complete in 8.048307322 seconds
[2014-01-07T08:40:06-05:00] INFO: Removing cookbooks/apache/templates/default/index.html.erb from the cache; it is no
longer needed by chef-client.
[2014-01-07T08:40:06-05:00] INFO: Removing cookbooks/apache/templates/default/custom.erb from the cache; it is no
longer needed by chef-client
```

# Rollbacks and Desired State Best Practice

- Chef is not magic - it manages state for **declared** resources
- We just rolled back to an earlier version of the apache cookbook
- While the recipe applied fine, investigating the system will reveal Apache is still configured as it was in the 0.2.0 cookbook
- A better way to ensure a smooth rollback: write contra-resources to clean up, and have a new version of the cookbook.

# Review Questions

- What is an Environment?
- How is it different from an Organization?
- What kind of node attributes do you typically set from an Environment?

# Using Community Cookbooks

A trip to the Supermarket.

v2.0.3

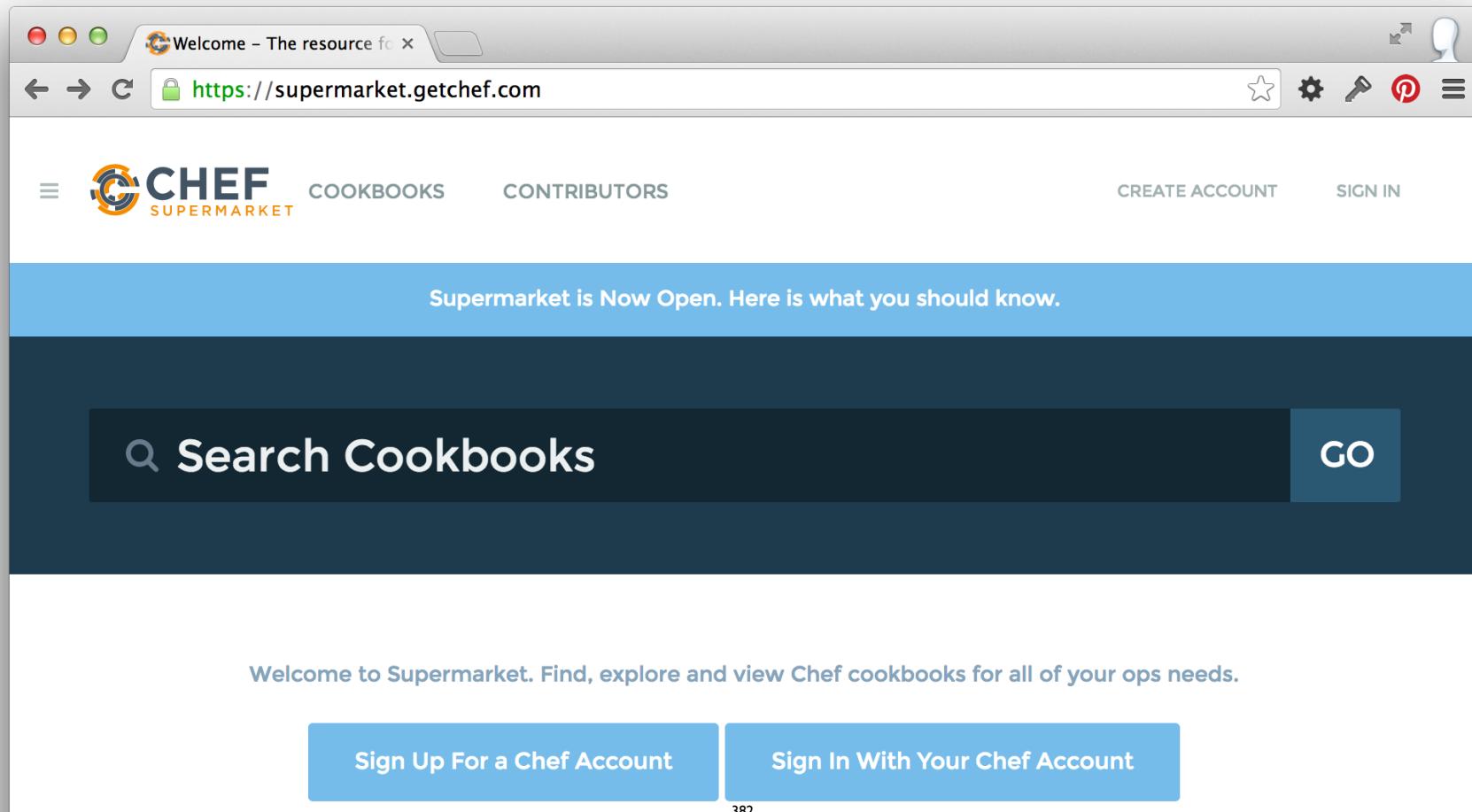
# Lesson Objectives

- After completing the lesson, you will be able to
  - Find, preview and download cookbooks from the Chef Supermarket
  - Use knife to work with the Supermarket API
  - Download, extract, examine and implement cookbooks from the Supermarket

# The easy way...

- We've been writing some cookbooks so far...
- Hundreds already exist for a large number of use cases and purposes. Many (but only a fraction) are maintained by Opscode.
- Think of it like RubyGems.org, CPAN.org, or other focused plugin-style distribution sites.

# Exercise: Find and preview cookbooks in the supermarket



The screenshot shows a web browser window with the following details:

- Address Bar:** https://supermarket.getchef.com
- Header:** Welcome – The resource fc
- Navigation:** Back, Forward, Stop, Refresh, Home, Favorites, Settings, Help.
- Logo:** CHEF SUPERMARKET logo.
- Menu:** COOKBOOKS, CONTRIBUTORS.
- User Options:** CREATE ACCOUNT, SIGN IN.
- Middle Section:** A blue banner states "Supermarket is Now Open. Here is what you should know." Below it is a search bar with a magnifying glass icon and the text "Search Cookbooks". To the right of the search bar is a blue "GO" button.
- Welcome Message:** "Welcome to Supermarket. Find, explore and view Chef cookbooks for all of your ops needs."
- Buttons:** "Sign Up For a Chef Account" and "Sign In With Your Chef Account".
- Page Number:** 382 at the bottom center.

# Exercise: Search for a chef-client cookbook

Supermarket is Now Open. Here is what you should know.

chef-client GO

160 Cookbooks [RSS](#) Sort by [Most Downloaded](#) [Most Followed](#)

**chef-client**  3.6.0  Updated June 8, 2014  opscode

Manages client.rb configuration and chef-client service

`cookbook 'chef-client', '~> 3.6.0'`

SUPPORTED PLATFORMS 

 2860816 Downloads  123 Followers Follow

383

# Search Results...

chef-client

160 Cookbooks [RSS](#)

Sort by [Most Downloaded](#) [Most Followed](#)

chef-client  Updated June 8, 2013

We're probably looking for this one

Manages client.rb configuration and chef-client service

cookbook 'chef-client', '~> 3.6.0'

SUPPORTED PLATFORMS

2860816 Downloads 123 Followers [Follow](#)

chef-client\_syslog 0.1.0 Updated December 17, 2012

sawanoboly

chef-client log to syslog

# Viewing a cookbook

The image shows two side-by-side screenshots. The left screenshot is from the Chef Supermarket for the 'chef-client' cookbook. It features a dark header with 'RSS' and a light header with '(42) Versions 3.6.0'. Below the header is a blue 'Follow' button with '123' next to it. A dark banner at the top contains the command 'knife cookbook site install chef-client'. Below this, there's a section with 'README' and 'Dependencies' tabs, where 'README' is highlighted with an orange arrow. A dashed orange box surrounds the text 'README displayed on the page (if it has one)'. The main content area is titled 'chef-client Cookbook' and describes it as a tool for configuring a system as a Chef Client. The right screenshot is from the Opscode website. It shows the 'opscode' logo and 'Opscode Inc.' text. Below are three user profile icons. A horizontal line separates this from the 'DETAILS' section, which includes a 'View Source' button with an orange arrow pointing to it, and a 'Browse Source Code' button enclosed in a dashed orange box.

knife cookbook site install chef-client

(42) Versions 3.6.0

Follow 123

README Dependencies

README displayed on the page (if it has one)

# chef-client Cookbook

This cookbook is used to configure a system as a Chef Client.

## Requirements

- Chef 0.10.14+
- Ohai 0.6.12+

RSS

Follow 123

## DETAILS

View Source

Browse Source Code

UPDATED JUNE 8, 2014

Created on December 16, 2010

### PLATFORMS

Apache 2.0

68268 LATEST VERSION DOWNLOADS

2860856 Total Downloads

## You can download cookbooks directly from the site...

- You can download cookbooks directly from the Supermarket:
  - It doesn't put them in your Chef Repository
  - It isn't fast if you know what you're looking for (click, click...)
  - It isn't necessarily fast if you **don't** know what you're looking for.
  - You're already using knife for managing cookbooks and other things in your Chef Repository.

# Introducing Knife Cookbook Site plugin

- Knife can interact with the Supermarket some sub-commands:
  - knife cookbook site search ...
  - knife cookbook site show...
  - knife cookbook site download ...
  - ... and more!

# Download and use ntp cookbook

v2.0.3

# NTP Cookbook

- Network time protocol - keeps system clocks in sync
- Chef Server authentication is time sensitive!

# Exercise: Download the ntp cookbook

```
$ knife cookbook site download ntp
```

Downloading ntp from the cookbooks  
site at version 1.5.4 to /Users/YOU/  
chef-repo/ntp-1.5.4.tar.gz

Cookbook saved: /Users/YOU/chef-repo/  
ntp-1.5.4.tar.gz

# Exercise: Extract the ntp cookbook

```
$ tar -zxvf ntp*.tar.gz -C cookbooks/
```

```
x ntp/
x ntp/CHANGELOG.md
x ntp/README.md
x ntp/attributes
x ntp/attributes/default.rb
x ntp/files
x ntp/files/default
x ntp/files/default/ntp.ini
x ntp/files/default/ntp.leapseconds
x ntp/files/default/tests
x ntp/files/default/tests/minitest
x ntp/files/default/tests/minitest/default_test.rb
x ntp/files/default/tests/minitest/support
x ntp/files/default/tests/minitest/support/helpers.rb
x ntp/files/default/tests/minitest/undo_test.rb
x ntp/files/default/usr.sbin.ntpd.apparmor
x ntp/metadata.json
x ntp/metadata.rb
x ntp/recipes
```

# What we just did...

- Cookbooks are distributed as a versioned .tar.gz archive.
- The latest version is downloaded by default (you can specify the version).
- Extract the cookbook into the "cookbooks" directory with tar.
- Next, let's examine the contents.

## Best practice: well written cookbooks have a README!

- Documentation for cookbooks doesn't need to be extensive, but a README should describe some important aspects of a cookbook:
  - Expectations (cookbooks, platform, data)
  - Recipes and their purpose
  - LWRPs, Libraries, etc.
  - Usage notes
- Read the README first!

# Best Practice: This runs as root!

- So, you just downloaded source code from the internet.
- As root.
- To load in the magic machine that:
  - **Makes your computers run code**
- Read the *entire* cookbook first!

# Examining the ntp cookbook

- The cookbook is quite flexible, but for this exercise we're just interested in the most basic use, an NTP client.
  - default recipe

# Exercise: View the ntp::default recipe

```
node['ntp']['packages'].each do |ntppkg|
  package ntppkg
end
```

```
template node['ntp']['conffile'] do
  source  'ntp.conf.erb'
  owner   node['ntp']['conf_owner']
  group   node['ntp']['conf_group']
  mode    '0644'
  notifies :restart,
"service[#{node['ntp']['service']}]"
end
```

```
service node['ntp']['service'] do
  supports :status => true, :restart => true
  action [:enable, :start]
end
```

- All **packages** are installed
- The **service** is enabled and started
- The **template** notifies the service

# Exercise: Upload the ntp cookbook

```
$ knife cookbook upload ntp
```

```
Uploading ntp [1.5.4]  
Uploaded 1 cookbook.
```

# Exercise: Add the ntp recipe to the base role



**OPEN IN EDITOR:** roles/base.rb

```
name "base"
description "Base Server Role"
run_list "recipe[ntp]", "recipe[motd]", "recipe[users]"
```

**SAVE FILE!**

# Exercise: Upload the base role

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

Updated Role base!

# Exercise: Re-run the Chef Client

```
Recipe: ntp::default
  * package[ntp] action install[2014-01-07T09:27:01-05:00] INFO: Processing package[ntp] action install
(ntp::default line 25)
  (up to date)
  * package[ntpdate] action install[2014-01-07T09:27:08-05:00] INFO: Processing package[ntpdate] action
install (ntp::default line 25)
  (up to date)
...
  * template[/etc/ntp.conf] action create[2014-01-07T09:27:09-05:00] INFO: Processing template[/etc/
ntp.conf] action create (ntp::default line 71)
[2014-01-07T09:27:09-05:00] INFO: template[/etc/ntp.conf] backed up to /var/chef/backup/etc/
ntp.conf.chef-20140107092709.254951
[2014-01-07T09:27:09-05:00] INFO: template[/etc/ntp.conf] updated file contents /etc/ntp.conf

  - update content in file /etc/ntp.conf from ba8f03 to c381bb
    --- /etc/ntp.conf      2013-10-27 09:07:05.541644862 -0400
    +++ /tmp/chef-rendered-template20140107-20557-ttvqaw      2014-01-07 09:27:09.248975286 -0500
    @@ -1,58 +1,30 @@
...
[2014-01-07T09:27:10-05:00] INFO: template[/etc/ntp.conf] sending restart action to service[ntpd] (delayed)
Recipe: ntp::default
  * service[ntpd] action restart[2014-01-07T09:27:10-05:00] INFO: Processing service[ntpd] action restart
(ntp::default line 100)
[2014-01-07T09:27:11-05:00] INFO: service[ntpd] restarted

  - restart service service[ntpd]
```

# Download and use chef-client cookbook

v2.0.3

# Exercise: Download the chef-client cookbook

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

Downloading chef-client from the cookbooks site at version 3.2.0 to /Users/johnfitzpatrick/cheftraining/fundamentals2.0/chef-repo/chef-client-3.2.0.tar.gz

Cookbook saved: /Users/YOU/chef-repo/chef-client-3.2.0.tar.gz

# Exercise: 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
```

# Examining the chef-client cookbook

- We're going to use two recipes on the node from the chef-client cookbook.
  - `delete_validation`
  - `service` (via default)

# Exercise: View the chef-client::delete\_validation recipe



**OPEN IN EDITOR:** cookbooks/chef-client/recipes/delete\_validation.rb

```
unless chef server?
  file Chef::Config[:validation_key] do
    action :delete
    backup false
    only_if { ::File.exists?(Chef::Config[:client_key]) }
  end
end
```

**SAVE FILE!**

## Exercise: Add `chef-client::delete_validation` to your base role



**OPEN IN EDITOR:** `roles/base.rb`

```
name "base"
description "Base Server Role"
run_list "recipe[chef-client::delete_validation]", "recipe[motd]", "recipe[users]"
```

**SAVE FILE!**

- Add the `delete_validation` recipe

## Best Practice: Delete the validation certificate when it isn't required

- Once Chef enters the actual run, synchronizing cookbooks, it has registered its own API client with the validation certificate
- That certificate is no longer required. We do this first because in case the run fails for another reason, we know at least the validation certificate is gone

# Exercise: View the chef-client::default recipe



**OPEN IN EDITOR:** 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"
```

**SAVE FILE!**

## Best Practice: Sane defaults do "pretty much" what you expect

- The main point of the "chef-client" cookbook is managing the "chef-client" program. It is designed that it can run as a daemonized service.
- The least surprising thing for most users is that the default recipe starts the service.
- You can manage the service in a number of ways, see the cookbook's README.md.

# Exercise: View the chef-client::service recipe



**OPEN IN EDITOR:** 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',
  'winst'
]
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
```

## Best Practice: 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.

# Exercise: Upload the chef-client cookbook

```
$ knife cookbook upload chef-client
```

Uploading chef-client [3.0.0]  
ERROR: Cookbook 'chef-client' depends on cookbook 'cron' version '>= 1.2.0',  
ERROR: which is not currently being loaded. An older version cannot be found on the  
server.



# Exercise: Download the cron cookbook

```
$ knife cookbook site download cron
```

```
Downloading cron from the cookbooks site at version  
1.2.8 to /Users/YOU/chef-repo/cron-1.2.8.tar.gz  
Cookbook saved: /Users/YOU/chef-repo/  
cron-1.2.8.tar.gz
```

# Exercise: Extract cron cookbook tarball

```
$ tar -zxvf cron*.tar.gz -C cookbooks/
```

```
x cron/
x cron/CHANGELOG.md
x cron/README.md
x cron/metadata.json
x cron/metadata.rb
x cron/providers
x cron/providers/d.rb
x cron/recipes
x cron/recipes/default.rb
x cron/recipes/test.rb
x cron/resources
x cron/resources/d.rb
x cron/templates
x cron/templates/default
x cron/templates/default/cron.d.erb
```

# Exercise: Upload the cron cookbook

```
$ knife cookbook upload cron
```

```
Uploading cron [1.2.8]  
Uploaded 1 cookbook.
```

# Exercise: Upload the chef-client cookbook

```
$ knife cookbook upload chef-client
```

Uploading chef-client [3.1.0]  
ERROR: Cookbook chef-client depends on  
cookbook 'logrotate' version '>= 1.2.0',  
ERROR: which is not currently being  
uploaded and cannot be found on the  
server.



# Exercise: Download the logrotate cookbook

```
$ knife cookbook site download logrotate
```

```
Downloading logrotate from the cookbooks site at
version 1.4.0 to /Users/johnfitzpatrick/
cheftraining/chef-repo/logrotate-1.4.0.tar.gz
Cookbook saved: /Users/YOU/chef-repo/
logrotate-1.4.0.tar.gz
```

# Exercise: Extract logrotate cookbook tarball

```
$ tar -zxvf logrotate*.tar.gz -C cookbooks/
```

```
x logrotate/
x logrotate/CHANGELOG.md
x logrotate/README.md
x logrotate/attributes
x logrotate/attributes/default.rb
x logrotate/definitions
x logrotate/definitions/logrotate_app.rb
x logrotate/libraries
x logrotate/libraries/logrotate_config.rb
x logrotate/metadata.json
x logrotate/metadata.rb
x logrotate/recipes
x logrotate/recipes/default.rb
x logrotate/recipes/global.rb
x logrotate/templates
x logrotate/templates/default
x logrotate/templates/default/logrotate-global.erb
x logrotate/templates/default/logrotate.erb
```

# Exercise: Upload the logrotate cookbook

```
$ knife cookbook upload logrotate
```

```
Uploading logrotate  
Uploaded 1 cookbook.
```

```
[ 1.4.0 ]
```

# Exercise: Upload the chef-client cookbook

```
$ knife cookbook upload chef-client
```

Uploading chef-client [31.0]  
Uploaded chef-client cookbook

A large, bold, green text "WIN!" is overlaid on the terminal output. The letters are slightly slanted to the right. The "W" is positioned above the word "Uploading", the "I" is positioned between "chef-client" and "[31.0]", and the "N" is positioned below the word "Uploaded".

# Exercise: Add chef-client recipe to base role



**OPEN IN EDITOR:** roles/base.rb

```
name "base"
description "Base Server Role"
run_list "recipe[chef-client::delete_validation]", "recipe[chef-client]",
"recipe[motd]", "recipe[users]"
```

**SAVE FILE!**

# Exercise: Upload the base role

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

Updated Role base!

# Exercise: Re-run the Chef Client

```
...
Recipe: chef-client::delete_validation
 * file[/etc/chef/validation.pem] action delete[2014-01-07T09:05:43-05:00] INFO: Processing file[/etc/chef/validation.pem] action
delete (chef-client::delete_validation line 25)
[2014-01-07T09:05:43-05:00] INFO: file[/etc/chef/validation.pem] deleted file at /etc/chef/validation.pem

 - delete file /etc/chef/validation.pem
...
...
 * service[chef-client] action enable[2014-01-07T09:05:46-05:00] INFO: Processing service[chef-client] action enable (chef-
client::init_service line 32)
[2014-01-07T09:05:47-05:00] INFO: service[chef-client] enabled

 - enable service service[chef-client]

 * service[chef-client] action start[2014-01-07T09:05:47-05:00] INFO: Processing service[chef-client] action start (chef-
client::init_service line 32)
[2014-01-07T09:05:48-05:00] INFO: service[chef-client] started

 - start service service[chef-client]
...
[2014-01-07T09:05:55-05:00] INFO: template[/etc/init.d/chef-client] sending restart action to service[chef-client] (delayed)
Recipe: chef-client::init_service
 * service[chef-client] action restart[2014-01-07T09:05:55-05:00] INFO: Processing service[chef-client] action restart (chef-
client::init_service line 32)
[2014-01-07T09:06:01-05:00] INFO: service[chef-client] restarted

 - restart service service[chef-client]
[2014-01-07T09:06:01-05:00] INFO: Chef Run complete in 29.341053545 seconds
```

# Exercise: Verify chef-client is running

```
chef@node1$ ps awux | grep chef-client
```

```
root      8933  0.3  2.2 130400 37816 ?          S1   03:19
0:01 /opt/chef/embedded/bin/ruby /usr/bin/chef-client -d -c /
etc/chef/client.rb -L /var/log/chef/client.log -P /var/run/
chef/client.pid -i 1800 -s 300
```

# Convergent infrastructure

- Our node is now running chef-client as a daemon, and it will converge itself over time on a (by default) 30 minute interval.
- The amount of resources converged may vary with longer intervals, depending on configuration drift on the system.
- Because Chef resources are idempotent, it will only configure what it needs to each run.

# Review Questions

- What is the Chef Supermarket site URL?
- What are two ways to download cookbooks from the Supermarket site?
- What is the first thing you should read when downloading a cookbook?
- Who vets the cookbooks on the Supermarket?
- Who has reads the recipes they download from the Supermarket?

# Further Resources

v2.0.3

# Further Resources

- <http://getchef.com/>
- <http://supermarket.getchef.com/>
- <http://docs.opscode.com/>
- <http://learnchef.com>
- <http://youtube.com/user/Opscode>
- irc.freenode.net #chef
- Twitter @getchef #getchef

# Food Fight Show

- <http://foodfightshow.org>
- The Podcast Where DevOps Chef Do Battle
- Regular updates about new Cookbooks, Knife-plugins, and more
- Best Practices for working with Chef



# Local Meetup Groups

[chefs.meetup.com](https://www.meetup.com/chefs/)



- April 15 - 17 in San Francisco
- Keynotes by Target, GE Capital, Jez Humble, Rachel Chalmers
- 50+ Sessions over 2 days
- Community Concert

## Training Workshops

- Introduction to Chef - Managing Linux
- Introduction to Chef - Managing Windows
- Chef - Intermediate Topics
- Team Workflows with Chef
- Testing Your Automation Code

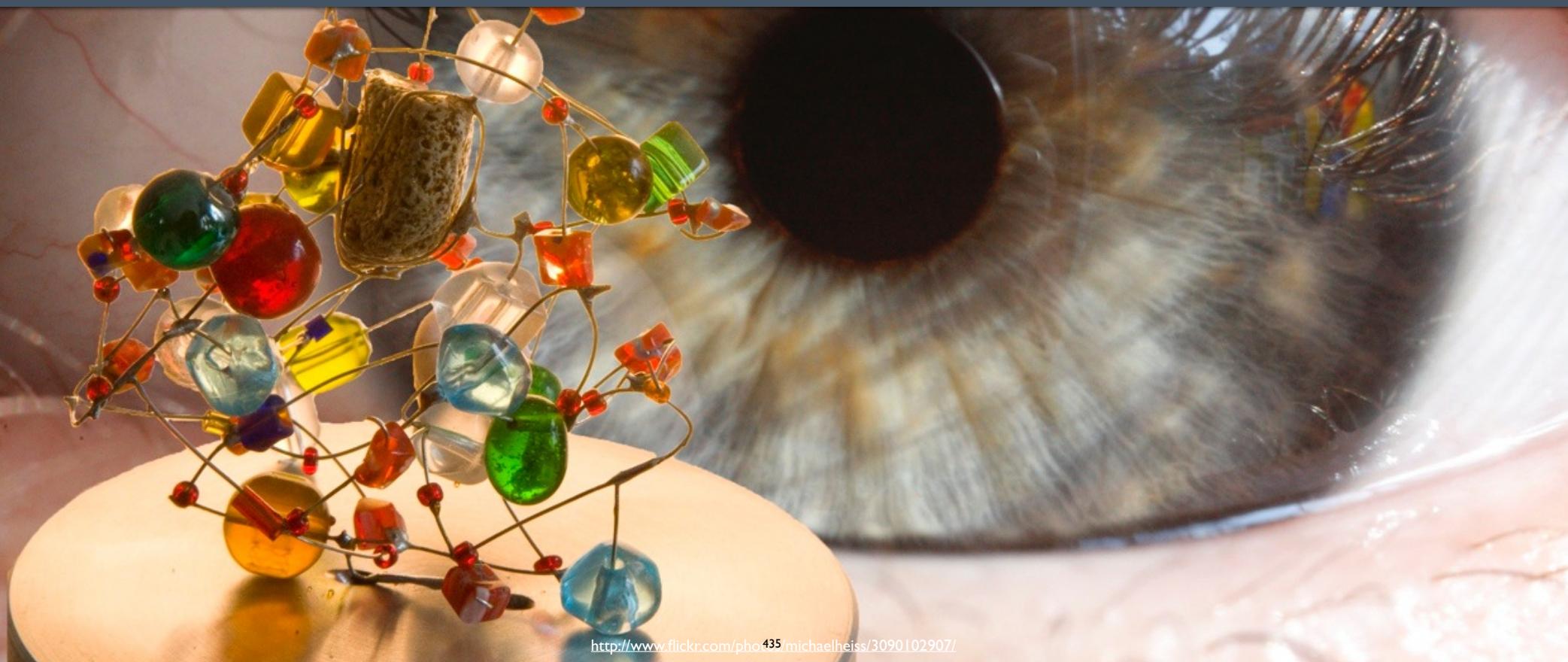
## Training Workshops

- Introduction to Kanban for Operations & DevOps
- Awesome Postmortems
- Git Foundations & Intermediate and Advanced Git and Github Tips and Tricks
- Ruby for Chefs

# Appendix: Why Use Chef?

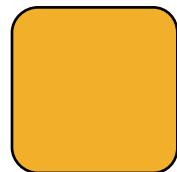
v2.0.3

# Complexity



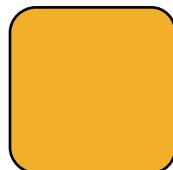
[http://www.flickr.com/photos/435\\_michaelheiss/3090102907/](http://www.flickr.com/photos/435_michaelheiss/3090102907/)

# A tale of growth...

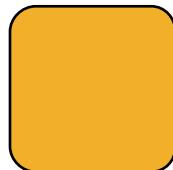


Application

# Add a database

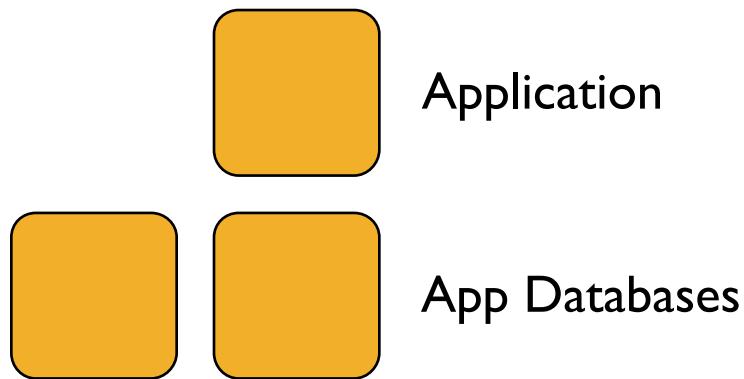


Application

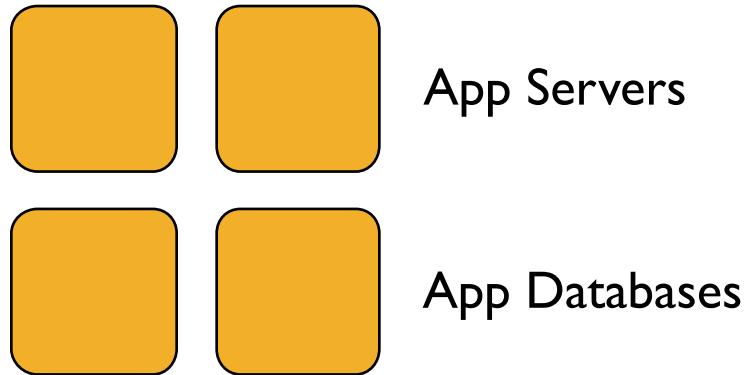


Application Database

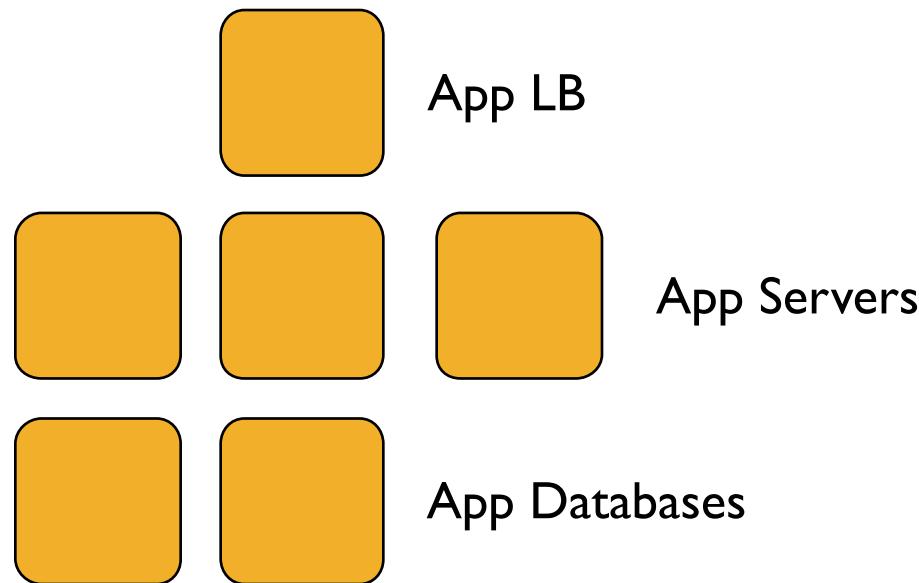
# Make database redundant



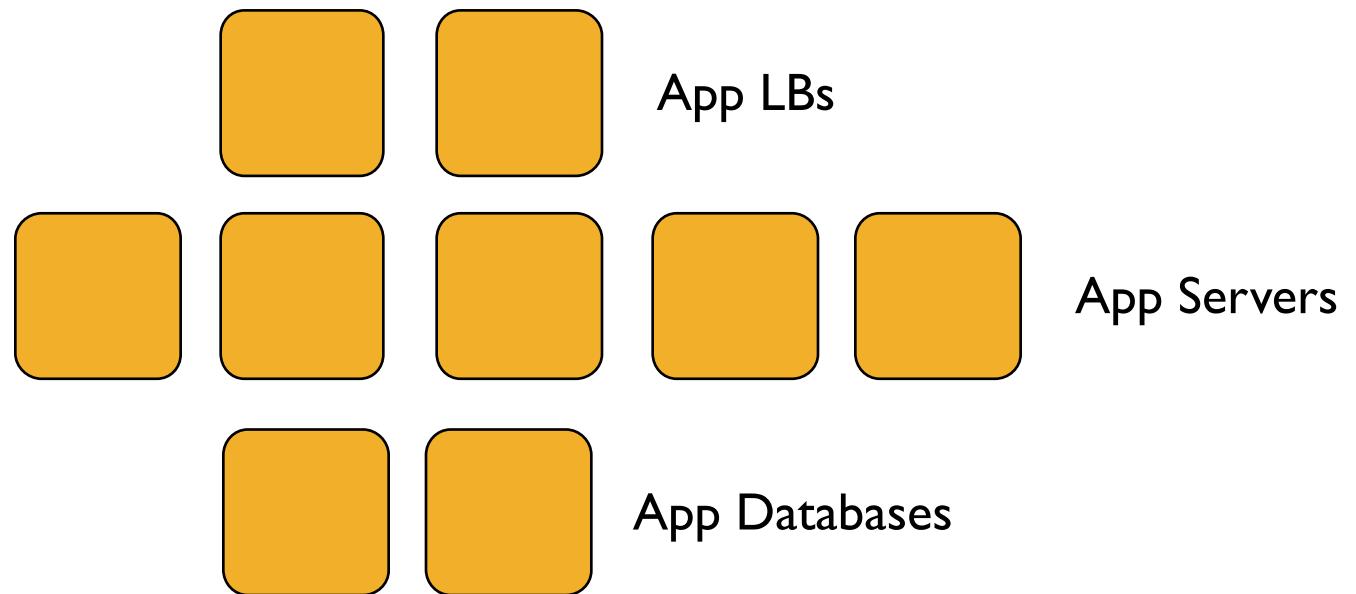
# Application server redundancy



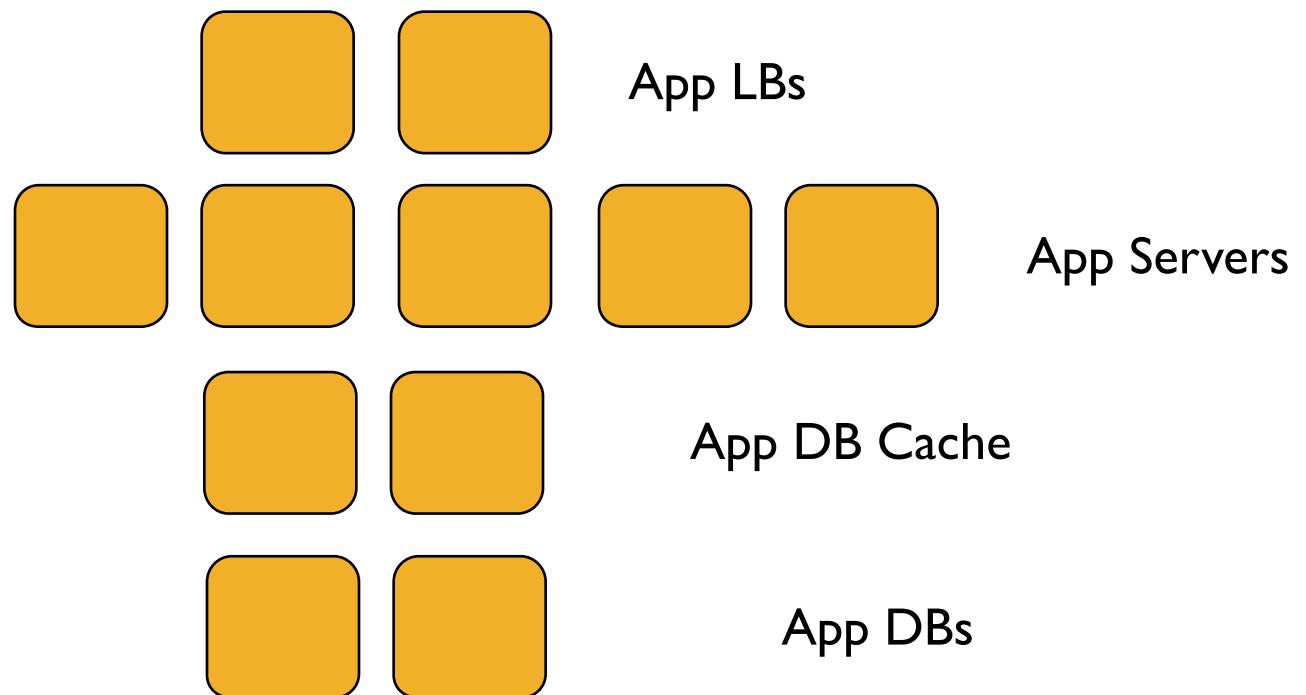
# Add a load balancer



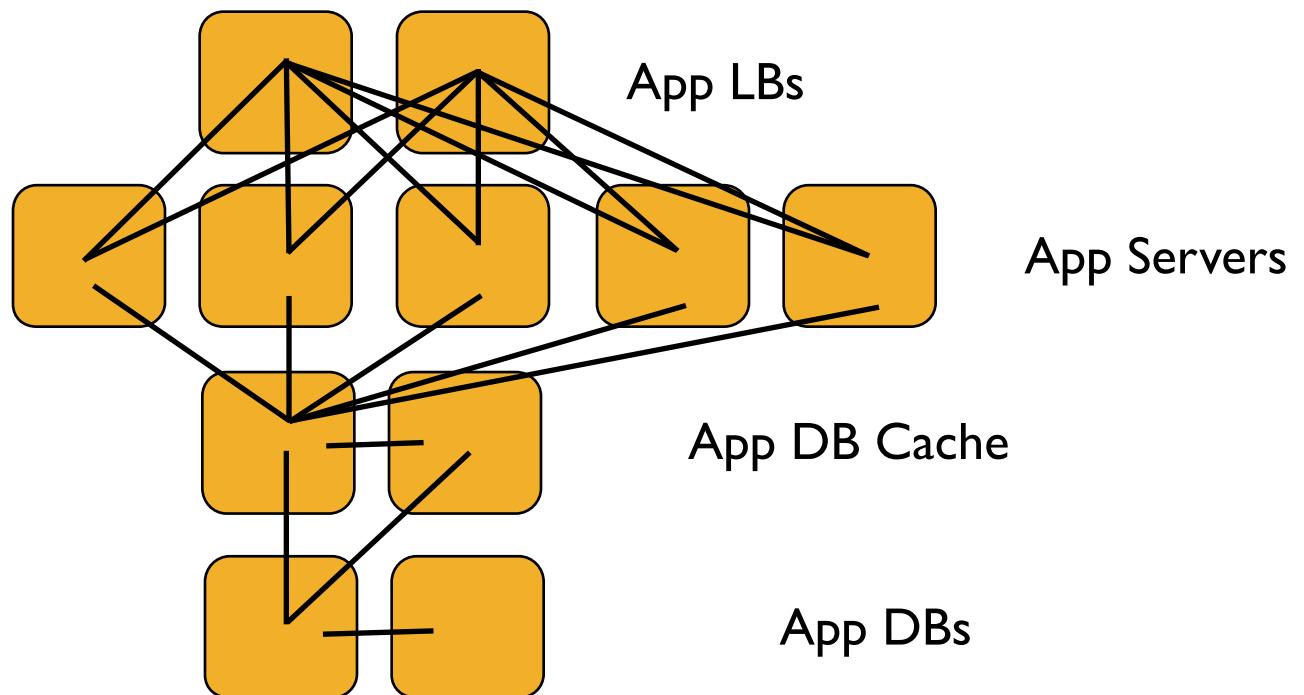
# Webscale!



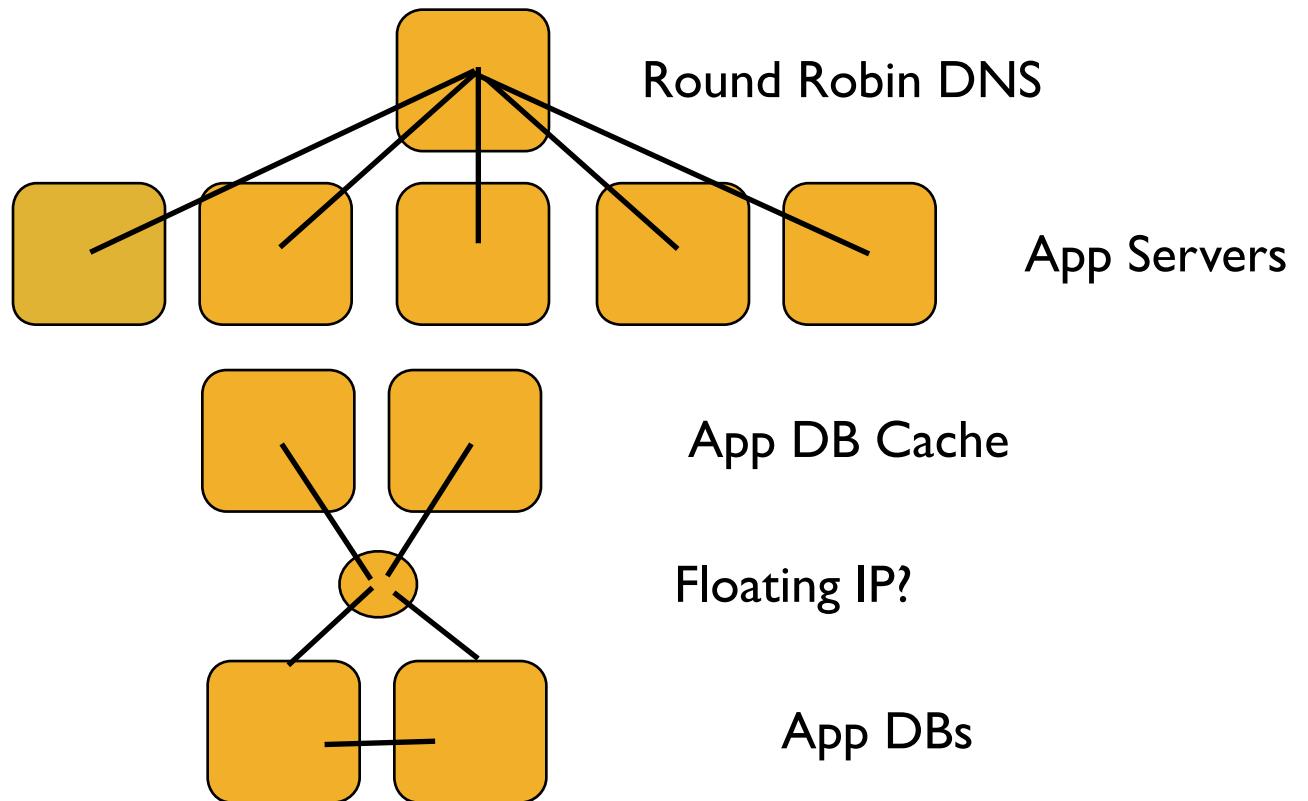
# Now we need a caching layer



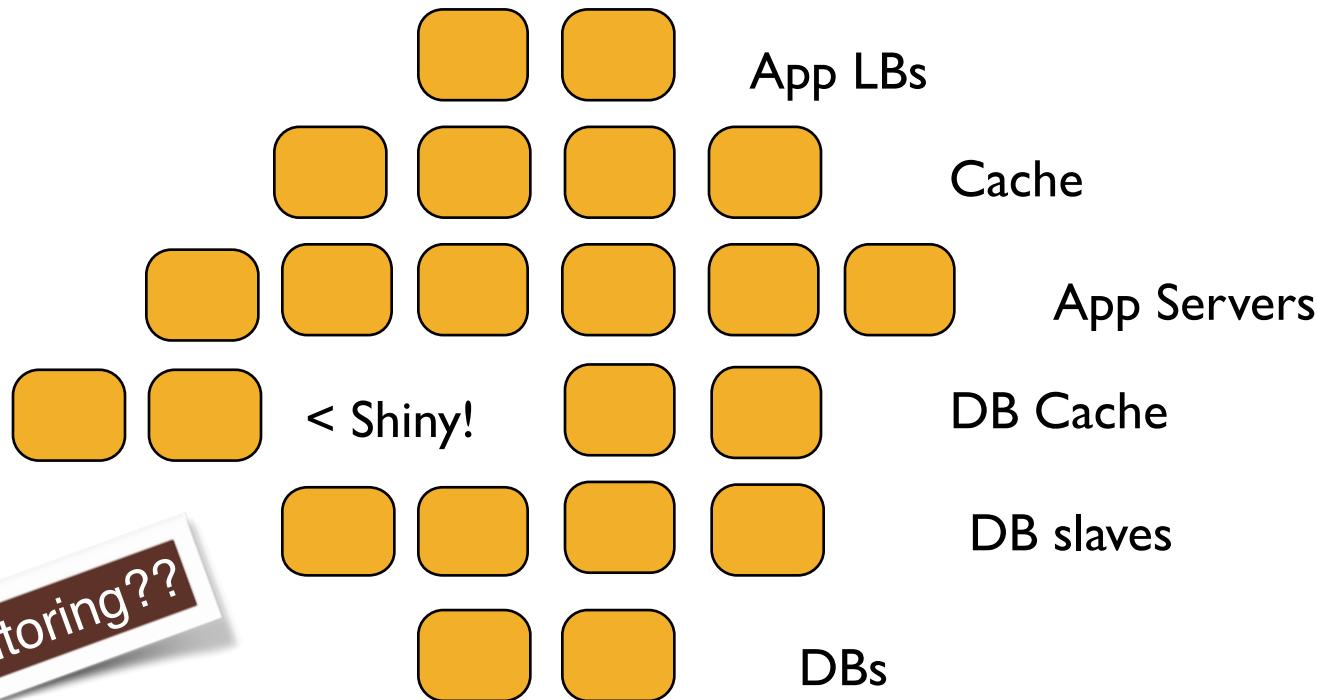
# Infrastructure has a Topology



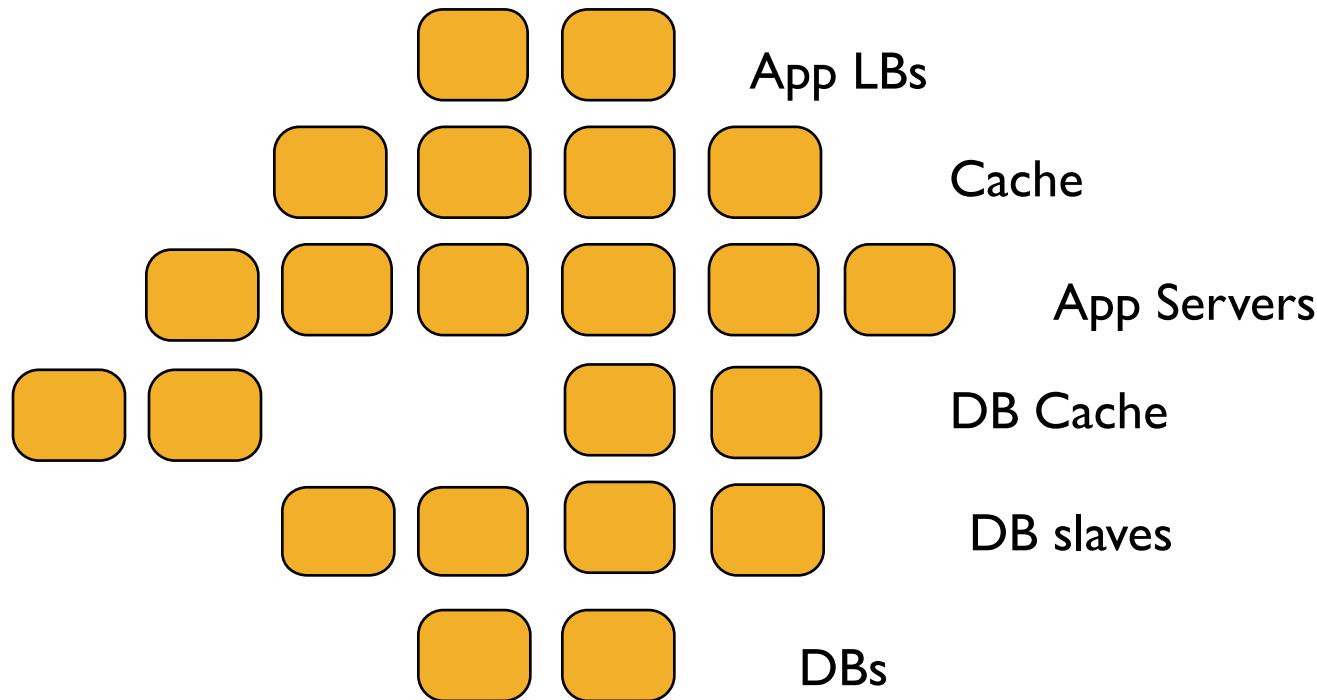
# Your Infrastructure is a Snowflake



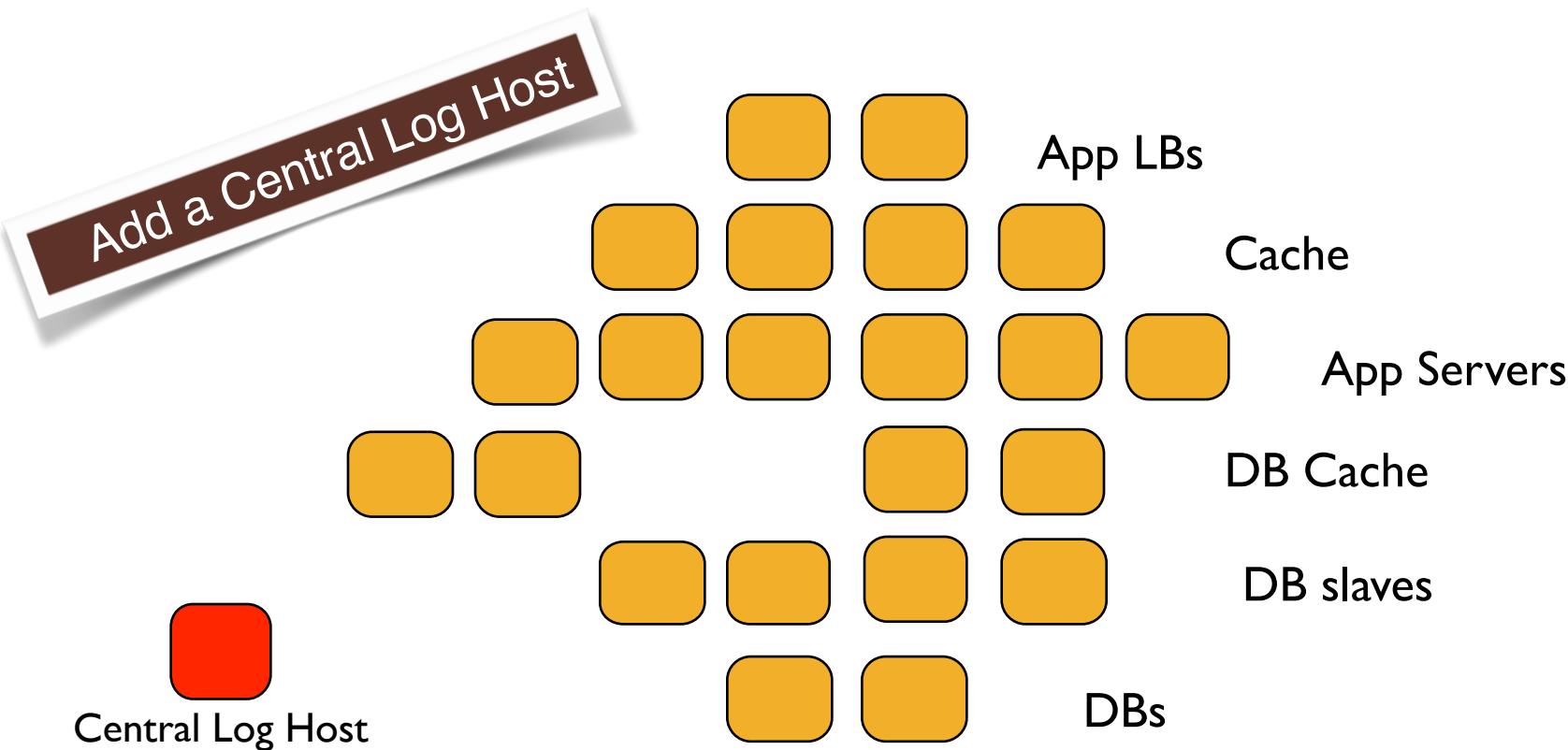
# Complexity Increases Quickly



# ...and change happens!

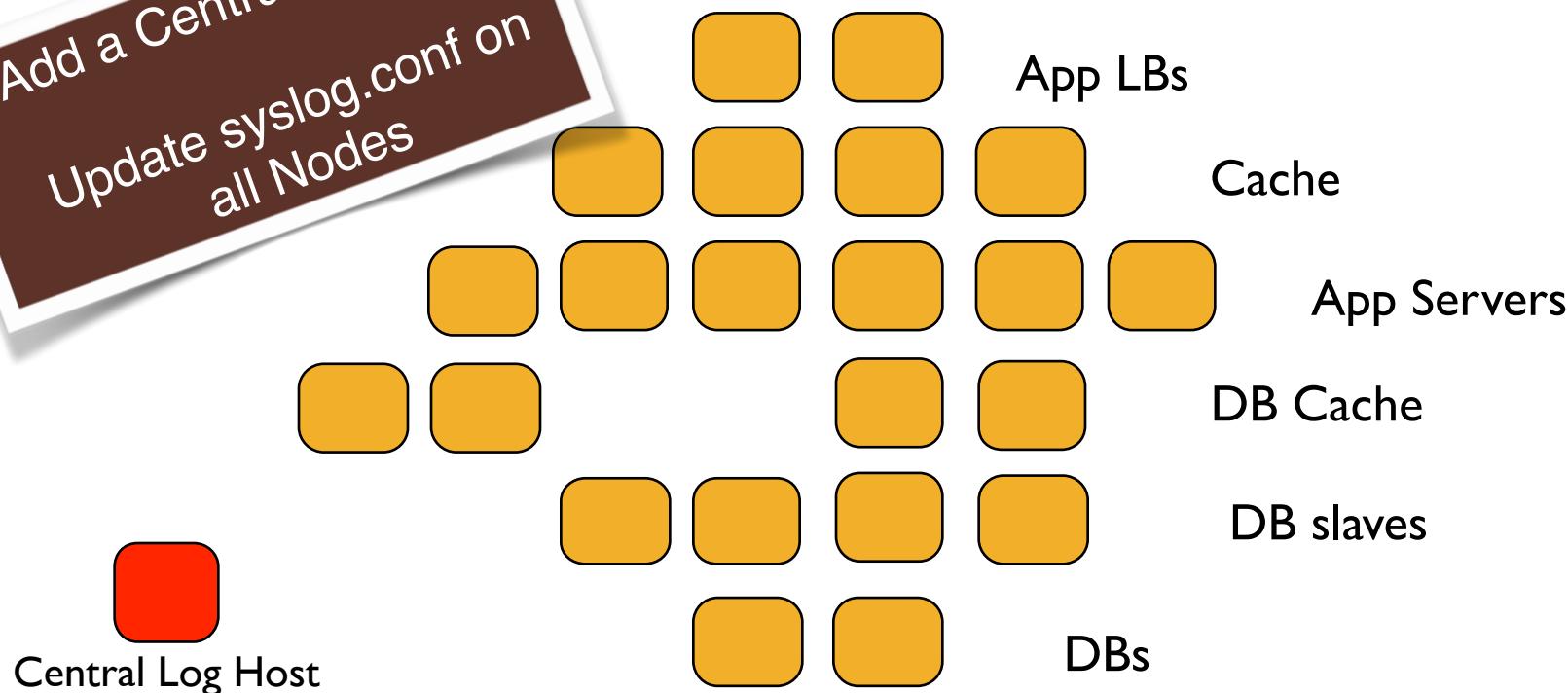


# ...and change happens!



# ...and change happens!

Add a Central Log Host  
Update syslog.conf on  
all Nodes



# Chef Solves This Problem



**CHEF**<sup>TM</sup>  
GETCHEF.COM  
449

# Appendix: Just Enough Ruby for Chef

A quick & dirty crash course

v2.0.3

# Lesson Objectives

- After completing the lesson, you will be able to
  - Explain what irb is, and how to use it.
  - Describe the function of:
    - Variable Assignment
    - Basic Arithmetic
    - Strings
    - Truthiness
    - Operators
    - Hashes
    - Regular Expressions
    - Conditionals
    - Method Declaration
    - Classes
    - Objects

# irb - the interactive ruby shell

- irb is the interactive ruby shell
- It is a REPL for Ruby
  - Read-Eval-Print-Loop
  - LISP, Python, Erlang, Clojure, etc.
- An interactive programming environment
- Super-handy for trying things out

# Exercise: Start irb on your ‘target’ node

```
chef@node1$ /opt/chef/embedded/bin/irb
```

```
irb(main):001:0>
```

# Variable assignment

```
chef@node1$ /opt/chef/embedded/bin/irb
```

```
irb(main):001:0> x = "hello"  
=> "hello"
```

```
irb(main):002:0> puts x  
hello  
=> nil
```

# Arithmetic

```
irb(main):003:0> 1 + 2  
=> 3
```

# Arithmetic

```
irb(main):004:0> 18 - 5  
=> 13
```

# Arithmetic

```
irb(main):005:0> 2 * 7  
=> 14
```

# Arithmetic

```
irb(main):006:0> 5 / 2  
=> 2
```

# Arithmetic

```
irb(main):007:0> 5 / 2.0  
=> 2.5
```

# Arithmetic

```
irb(main):008:0> 5.class  
=> Fixnum
```

```
irb(main):009:0> 5.0.class  
=> Float
```

# Arithmetic

```
irb(main):010:0> 1 + (2 * 3)  
=> 7
```

# Strings

```
irb(main):011:0> 'jungle'  
=> "jungle"
```

# Strings

```
irb(main):012:0> 'it\'s alive'  
=> "it's alive"
```

# Strings

```
irb(main):013:0> "animal"  
=> "animal"
```

# Strings

```
irb(main):014:0> "\\"so easy\\"
=> "\\\"so easy\\\""
```

```
irb(main):015:0> puts "\\"so easy\\"
"so easy"
=> nil
```

# Strings

```
irb(main):016:0> x = "pretty"  
=> "pretty"
```

```
irb(main):017:0> "#{x} nice"  
=> "pretty nice"
```

```
irb(main):018:0> '#{x} nice'  
=> "\#{x} nice"
```

# Truthiness

```
irb(main):019:0> true
=> true
```

```
irb(main):020:0> false
=> false
```

```
irb(main):021:0> nil
=> nil
```

```
irb(main):022:0> !!nil
=> false
```

```
irb(main):023:0> !!0
=> true
```

```
irb(main):024:0> !!x
=> true
```

# Operators

```
irb(main):025:0> 1 == 1  
=> true
```

```
irb(main):026:0> 1 == true  
=> false
```

```
irb(main):027:0> 1 != true  
=> true
```

```
irb(main):028:0> !!1 == true  
=> true
```

# Operators

```
irb(main):029:0> 2 < 1  
=> false
```

```
irb(main):030:0> 2 > 1  
=> true
```

```
irb(main):031:0> 4 >= 3  
=> true
```

```
irb(main):032:0> 4 >= 4  
=> true
```

```
irb(main):033:0> 4 <= 5  
=> true
```

```
irb(main):034:0> 4 <= 3  
=> false
```

# Operators

```
irb(main):035:0> 5 <=> 5  
=> 0
```

```
irb(main):036:0> 5 <=> 6  
=> -1
```

```
irb(main):037:0> 5 <=> 4  
=> 1
```

# Arrays

```
irb(main):038:0> x = ["a", "b", "c"]
=> ["a", "b", "c"]
```

```
irb(main):039:0> x[0]
=> "a"
```

```
irb(main):040:0> x.first
=> "a"
```

```
irb(main):041:0> x.last
=> "c"
```

# Arrays

```
irb(main):042:0> x + ["d"]
=> ["a", "b", "c", "d"]
```

```
irb(main):043:0> x
=> ["a", "b", "c"]
```

```
irb(main):044:0> x = x + ["d"]
=> ["a", "b", "c", "d"]
```

```
irb(main):045:0> x
=> ["a", "b", "c", "d"]
```

# Arrays

```
irb(main):046:0> x << "e"
=> [ "a", "b", "c", "d", "e" ]
```

```
irb(main):047:0> x
=> [ "a", "b", "c", "d", "e" ]
```

# Arrays

```
irb(main):048:0> x.map { |i| "the letter #{i}" }
=> ["the letter a", "the letter b", "the letter
c", "the letter d", "the letter e"]
```

```
irb(main):049:0> x
=> ["a", "b", "c", "d", "e"]
```

# Arrays

```
irb(main):050:0> x.map! { |i| "the letter #{i}" }
=> ["the letter a", "the letter b", "the letter
c", "the letter d", "the letter e"]
```

```
irb(main):051:0> x
=> ["the letter a", "the letter b", "the letter
c", "the letter d", "the letter e"]
```

# Hashes

```
irb(main):052:0> h = {  
irb(main):053:1* "first_name" => "Gary",  
irb(main):054:1* "last_name" => "Gygax"  
irb(main):055:1> }  
=> {"first_name"=>"Gary",  
"last_name"=>"Gygax"}
```

# Hashes

```
irb(main):056:0> h.keys  
=> ["first_name", "last_name"]  
  
irb(main):057:0> h["first_name"]  
=> "Gary"  
  
irb(main):058:0> h["age"] = 33  
=> 33  
  
irb(main):059:0> h.keys  
=> ["first_name", "last_name", "age"]
```

# Hashes

```
irb(main):060:0> h.values  
=> [ "Gary", "Gygax", 33 ]
```

# Hashes

```
irb(main):061:0> h.each { |k, v| puts "#{k}: #{v}" }
first_name: Gary
last_name: Gygax
age: 33
=> {"first_name"=>"Gary", "last_name"=>"Gygax",
"age"=>33}
```

# Regular Expressions

```
irb(main):062:0> x = "I want to believe"  
=> "I want to believe"
```

```
irb(main):063:0> x =~ /I/  
=> 0
```

```
irb(main):064:0> x =~ /lie/  
=> 12
```

```
irb(main):065:0> x =~ /smile/  
=> nil
```

```
irb(main):066:0> x !~ /smile/  
=> true
```

# Regular Expressions

```
irb(main):067:0> x.sub(/t/, "T")
=> "I wanT to believe"
```

```
irb(main):068:0> puts x
I want to believe
=> nil
```

```
irb(main):069:0> x.gsub!(/t/, "T")
=> "I wanT To believe"
```

```
irb(main):070:0> puts x
I wanT To believe
=> nil
```

# Conditionals

```
irb(main):071:0> x = "happy"
=> "happy"

irb(main):072:0> if x == "happy"
irb(main):073:1>   puts "Sure am!"
irb(main):074:1> elsif x == "sad"
irb(main):075:1>   puts "Boo!"
irb(main):076:1> else
irb(main):077:1*>   puts "Therapy?"
irb(main):078:1> end
Sure am!
=> nil
```

# Conditionals

```
irb(main):079:0> case x
irb(main):080:1> when "happy"
irb(main):081:1>   puts "Sure Am!"
irb(main):082:1>   1
irb(main):083:1> when "sad"
irb(main):085:1>   puts "Boo!"
irb(main):086:1>   2
irb(main):087:1> else
irb(main):088:1>   puts "Therapy?"
irb(main):089:1>   3
irb(main):090:1> end
Sure Am!
=> 1
```

# Method Definition

```
irb(main):091:0> def metal(str)
irb(main):092:1>   puts "!!#{str} is metal!!"
irb(main):093:1> end
=> nil

irb(main):094:0> metal("ozzy")
!!ozzy is metal!!
=> nil
```

# Classes

```
irb(main):095:0> class Person
irb(main):096:1>   attr_accessor :name, :is_metal
irb(main):097:1>
irb(main):098:1>   def metal
irb(main):099:2>     if @is_metal
irb(main):100:3>       puts "!!#{@name} is metal!!"
irb(main):101:3>     end
irb(main):102:2>   end
irb(main):103:1> end
=> nil
```

# Classes

```
irb(main):104:0> p = Person.new
=> #<Person:0x891ab4c>

irb(main):105:0> p.name = "Adam Jacob"
=> "Adam Jacob"

irb(main):106:0> p.is_metal = true
=> true

irb(main):107:0> p.metal
!!Adam Jacob is metal!!
=> nil

irb(main):108:0> p.is_metal = false
=> false

irb(main):109:0> p.metal
=> nil
```

# Review Questions

- What is irb?
- What is true in ruby? What is false?
- How do you press for the truth?
- What does `>=` do?
- How do you define a method?
- What is a class?
- What is an object?
- The book you want: "Programming Ruby 1.9" <http://pragprog.com/book/ruby3/programming-ruby-1-9>