

# Chef Fundamentals For Windows and Linux

training@chef.io

Copyright (C) 2014 Chef Software, Inc.

v2.1.0\_DUAL



# Chef Fundamentals For Windows

training@chef.io

Copyright (C) 2014 Chef Software, Inc.

v2.1.0\_DUAL



# Chef Fundamentals For Linux

training@chef.io

Copyright (C) 2014 Chef Software, Inc.

v2.1.0\_DUAL



# Course Logistics

v2.1.0\_DUAL



# Logistics

- Class Start Time:
- We'll take a break about once an hour
- We'll break for lunch:
- Class End Time:

# Agenda

v2.1.0\_DUAL



# Course Objective

**Learn To Use Chef  
To Make Your Life  
More Awesome!**

# Course Objectives

**We are going to learn how to:**

- Automate infrastructure and application deployment
- Read and write Chef Recipes and Cookbooks
- Understand Chef's architecture
- Apply Chef's basic building blocks to solve problems
- Use the multitude of assets already created within the Global Chef Community

# Topics

- Introductions
- Overview of Chef
- Chef Call flow
- Organizations
- Creating a local chef-repo

# Topics

- Bootstrapping a Node
- Secure Communications with the Chef Server
- What are Resources, Recipes & Cookbooks
- Writing a web server Cookbook
- Node Object
- Attributes

# Topics

- Search
- Environments
- Roles
- Data\_bags
- Supermarket

# Training is really a discussion

- Ask questions when they come to you
- Ask for help when you need it
- Ask the other people in the class
  - Letting them help you is the biggest favor you can do for them!
- You will have the slides available to you

# Fundamentals

- This course covers the **fundamentals** of Chef
- We likely will not have time to discuss the latest trends in the Chef ecosystem
- Any discussion items that are too far off topic will be captured

# Introductions

v2.1.0\_DUAL



# Introductions

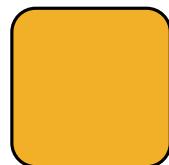
- Name:
- Where are you located?
- Company:
- Current job role:
- Experience with Chef/Config Management:
- Which Workstation / Server platforms do you work with?
- What do you want from / expect from this course:

# Why Chef

v2.1.0\_DUAL

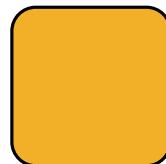


# A tale of growth...

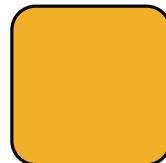


Application

# Add a database

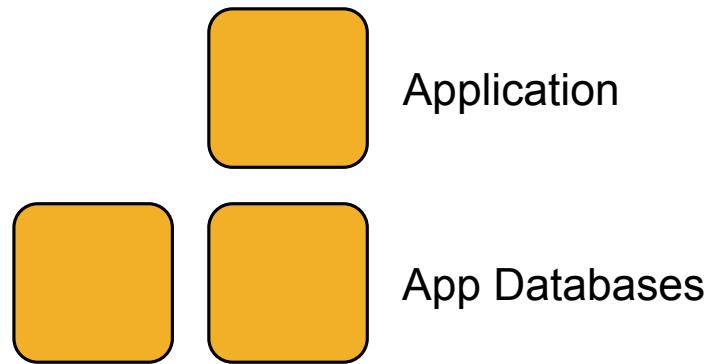


Application

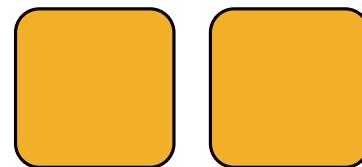


Application Database

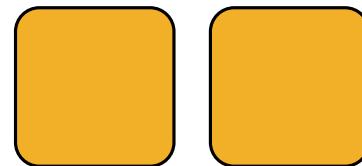
# Make database redundant



# Application server redundancy

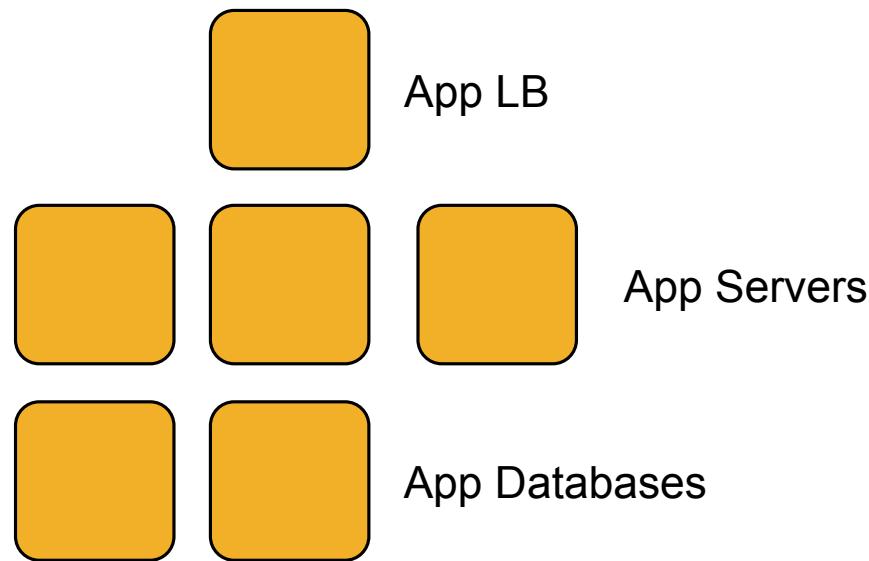


App Servers

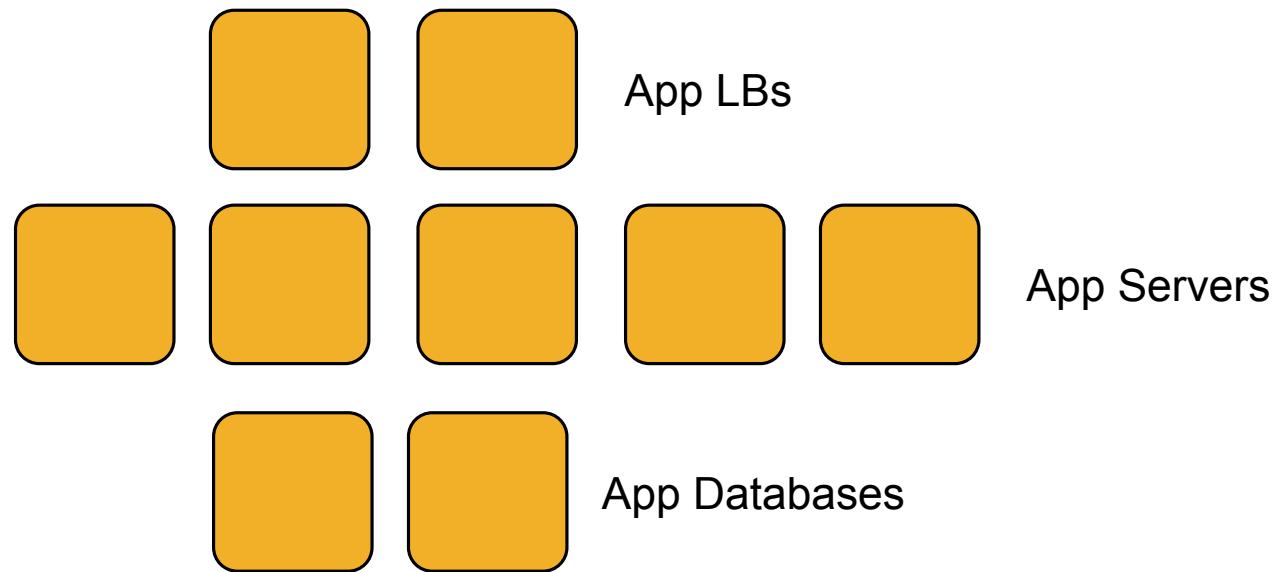


App Databases

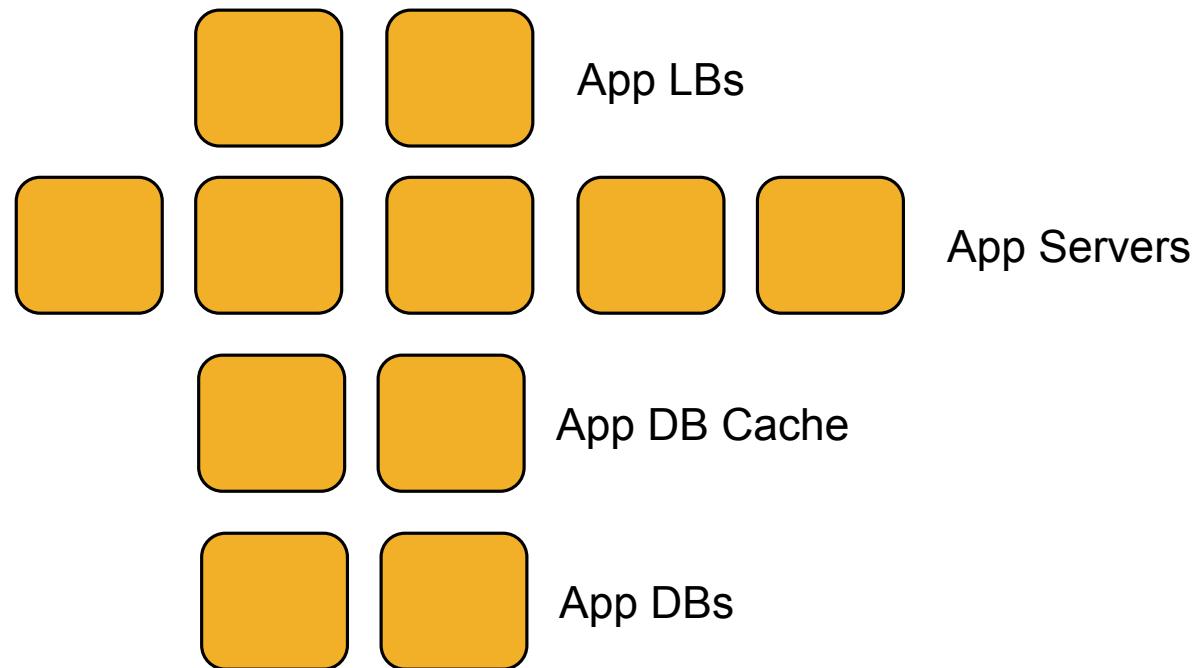
# Add a load balancer



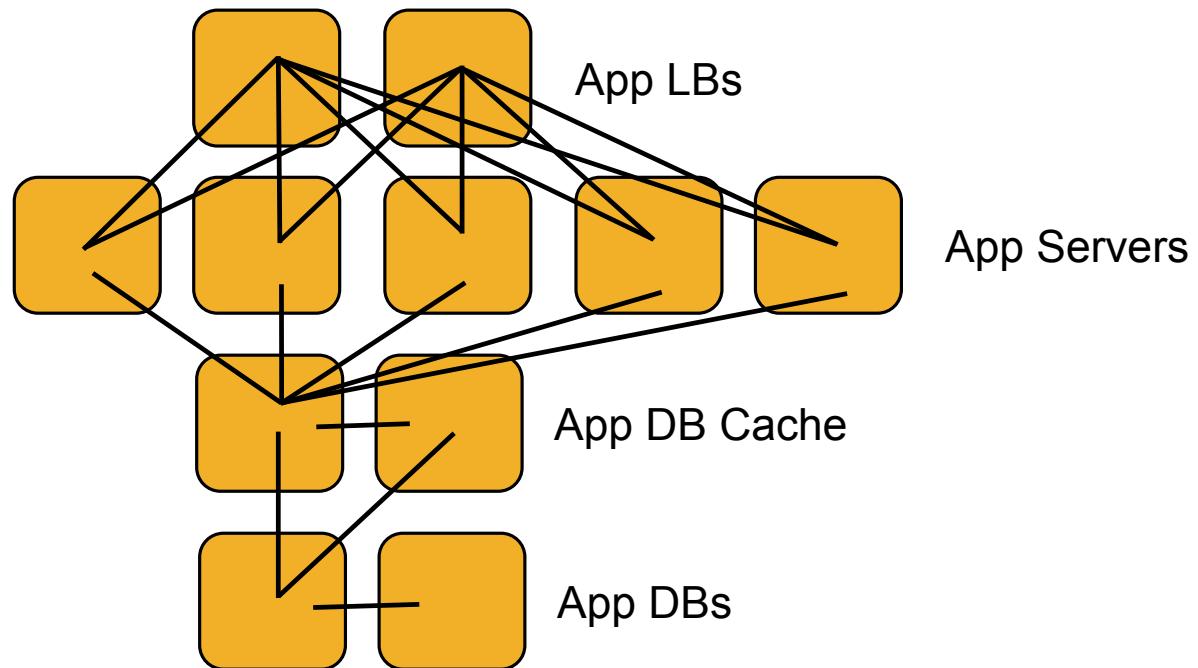
# Webscale!



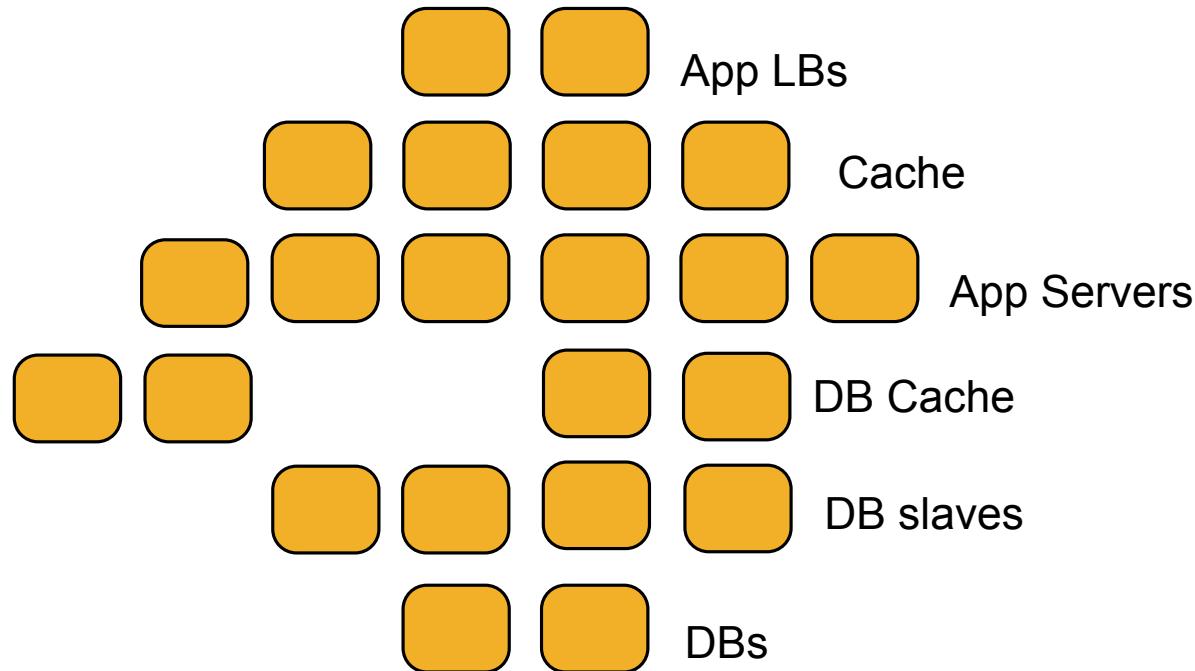
# Now we need a caching layer



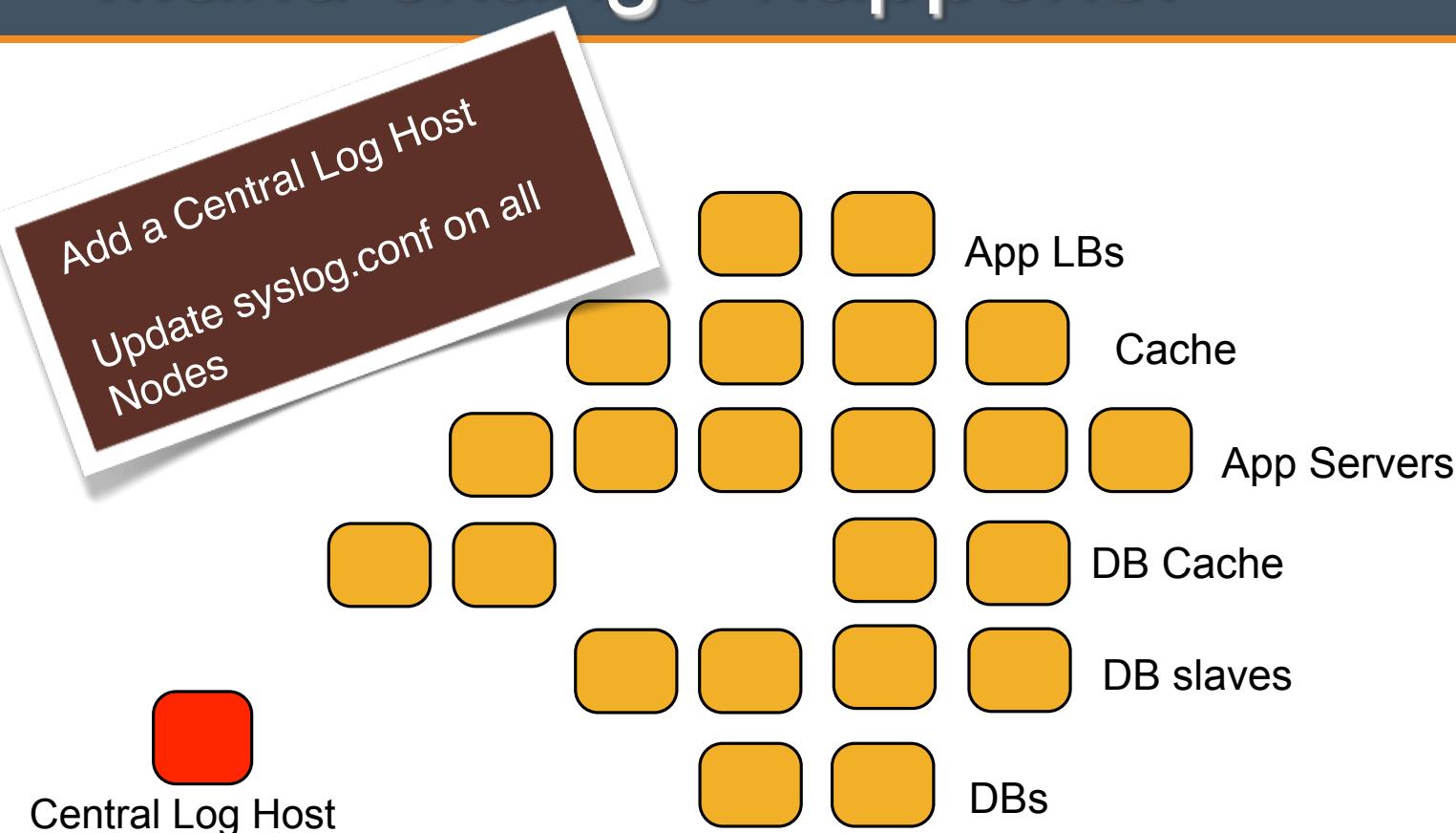
# Infrastructure has a Topology



# ...and change happens!



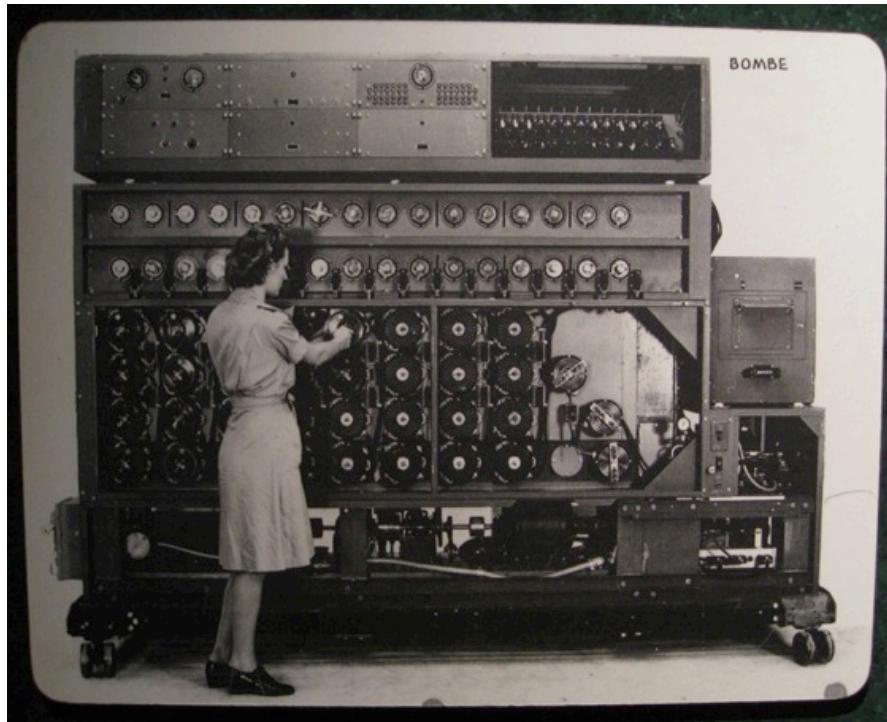
# ...and change happens!



# What did Chef do here?

- **First**, we used **Chef** to create each of the individual servers, including deploying our database and our application code
- **Second**, we used **Chef** to configure the servers to communicate with each other
- **Third**, we used **Chef** to automate any changes (such as modifying syslog.conf, or adding a webserver), making our lives easier

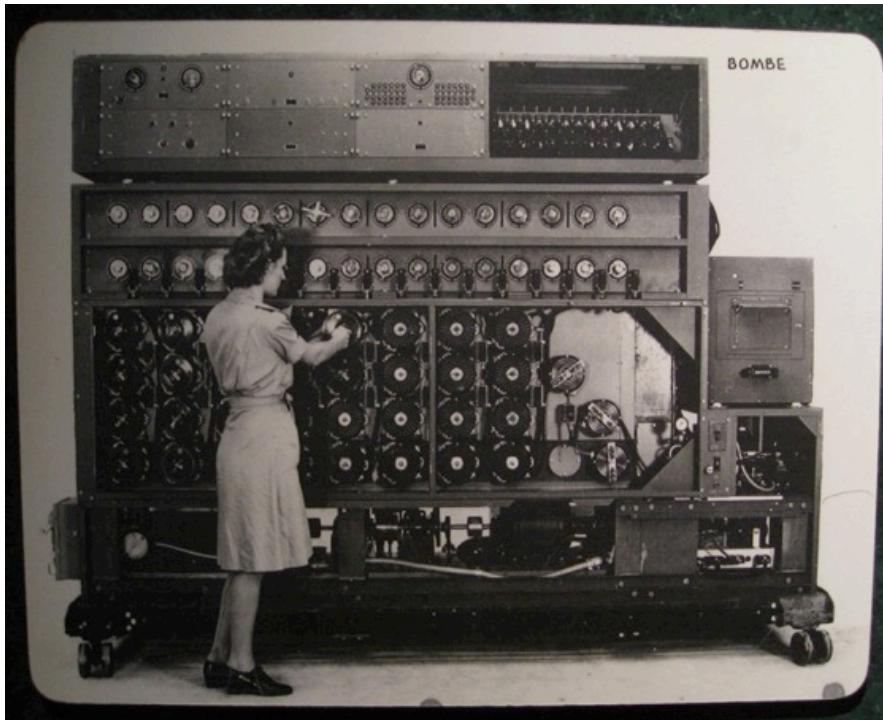
# Chef is Infrastructure as Code



<http://www.flickr.com/photos/louisb/4555295187/>

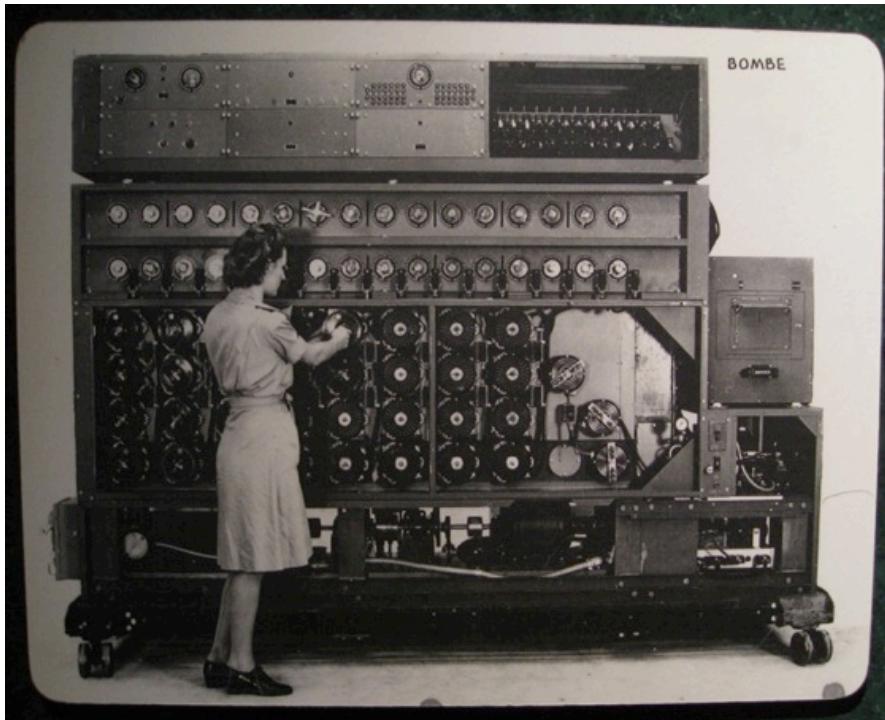
# Chef is Infrastructure as Code

- Programmatically provision and configure components



<http://www.flickr.com/photos/louisb/4555295187/>

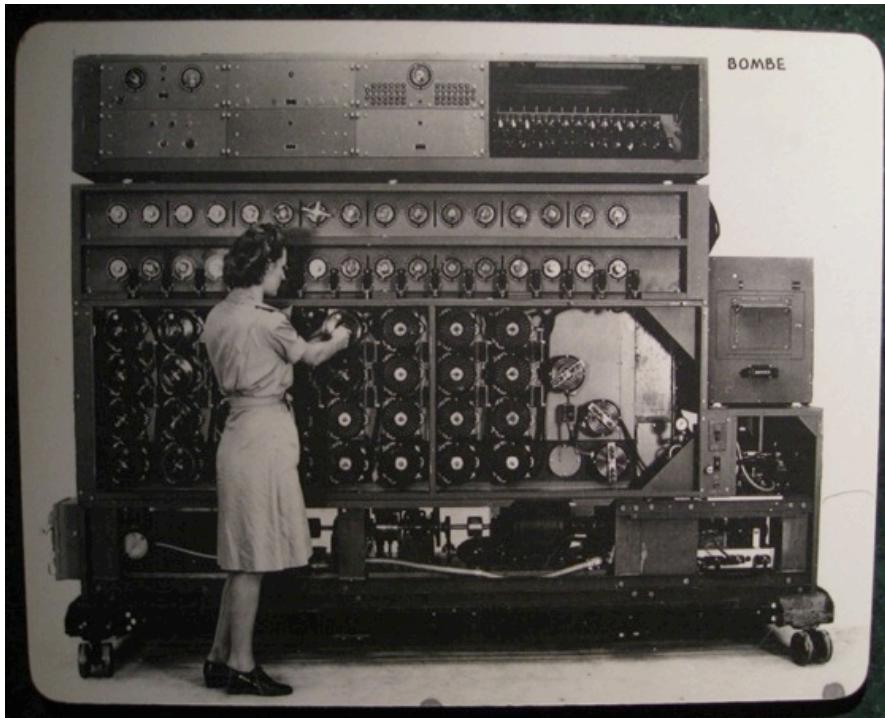
# Chef is Infrastructure as Code



<http://www.flickr.com/photos/louisb/4555295187/>

- Programmatically provision and configure components
- Treat like any other code base

# Chef is Infrastructure as Code



<http://www.flickr.com/photos/louisb/4555295187/>

- Programmatically provision and configure components
- Treat like any other code base
- Reconstruct business from **code repository (chef server), data backup, and compute resources**

# Chef is a Language

- Learning Chef is like learning the basics of a language
  - Chef is based on Ruby, a relatively easy programming language to learn
  - 80% fluency will be reached very quickly
  - The remaining 20% just takes practice
  - The best way to **learn** Chef is to **use** Chef



# Chef Overview – The Call Flow

v2.1.0\_DUAL

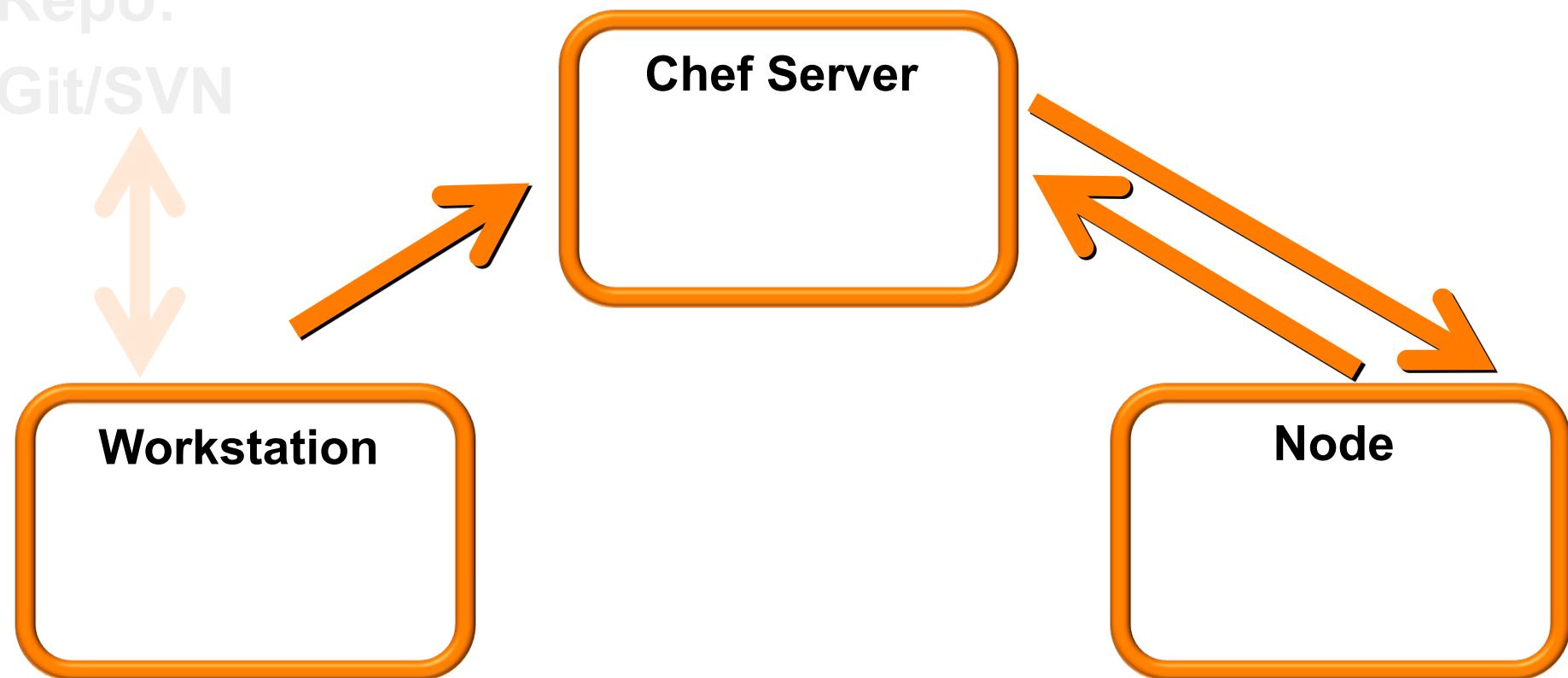
# Lesson Objectives

- Describe how Chef thinks about Infrastructure Automation
- Identify the three major system components in the Chef ecosystem:
  - **Workstation**
  - **Chef Server**
  - **Node**

# The Major System Components

Repo:

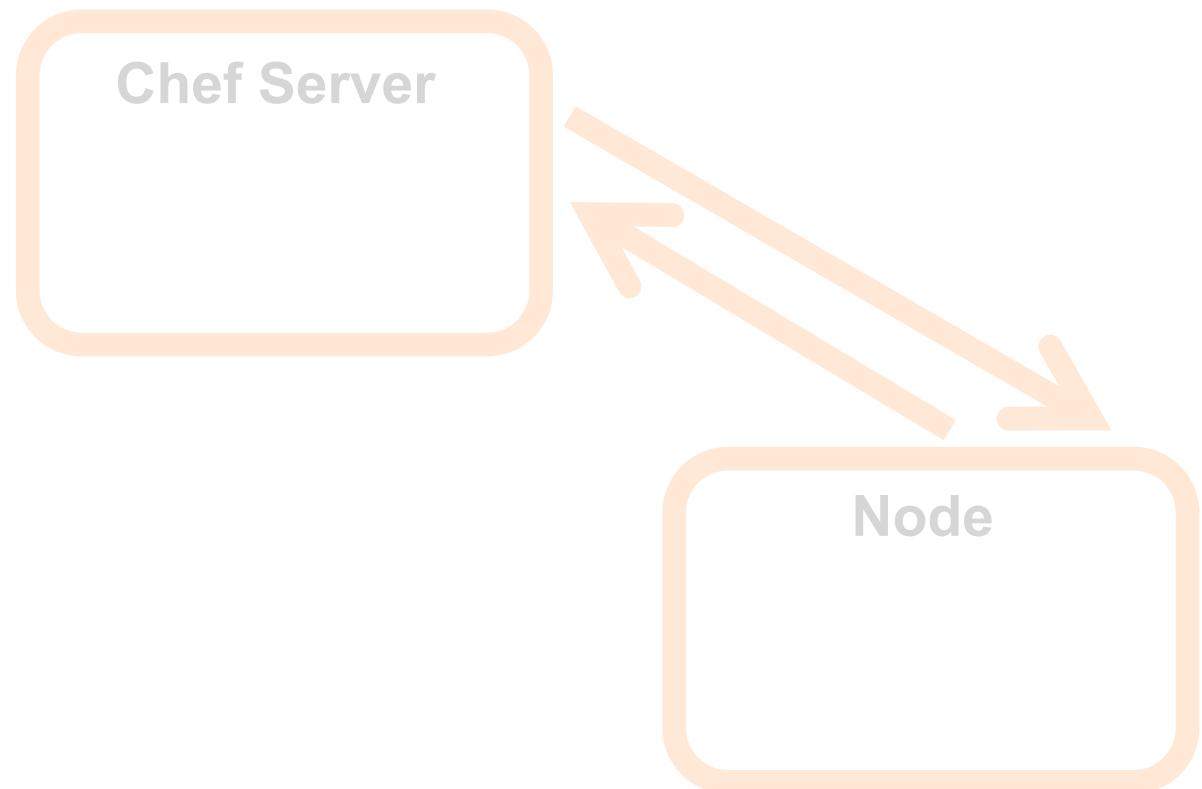
Git/SVN



# Workstation

Repo:

Git/SVN



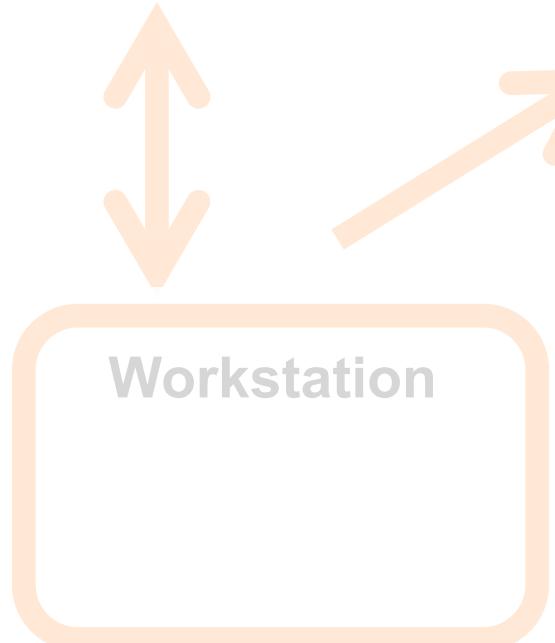
# Workstation

- Write your Chef configuration here
  - Local copy of **all** your Chef configuration files
- Upload your Chef configuration to Chef Server
- Your configuration can be synced with an external repo (such as Github or SVN) for:
  - Team workflows
  - Data backups
  - Version control

# Chef Server

Repo:

Git/SVN



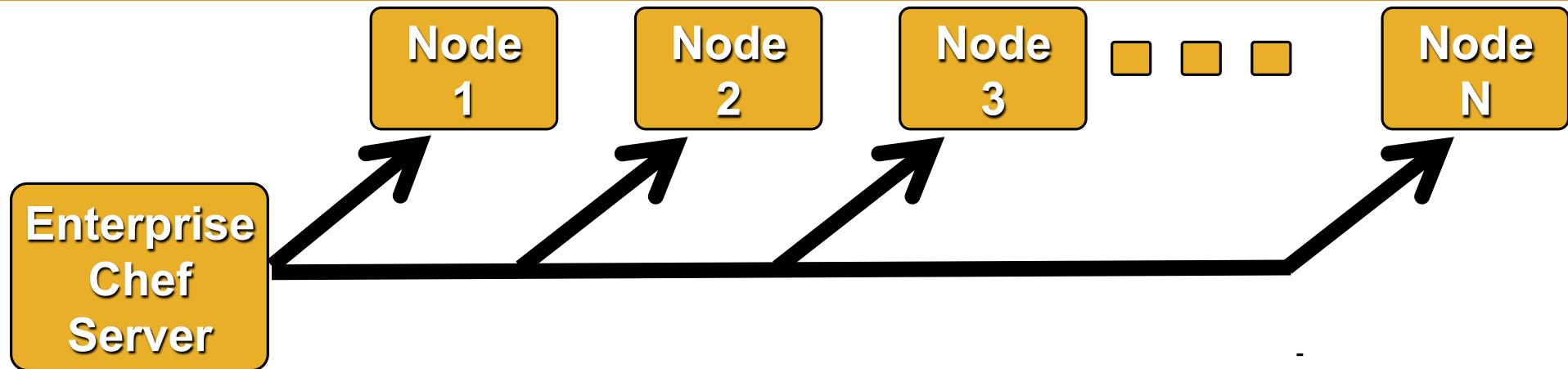
Chef Server

Node

# Chef Server

- Stores your system configuration
- Authenticates workstations and nodes
  - (“nodes” == “servers”)
- Delivers system configuration to nodes
- Scales by design to handle continuous configuration for thousands of nodes

# Scalability



Chef Server **delivers** the configuration to the **nodes**

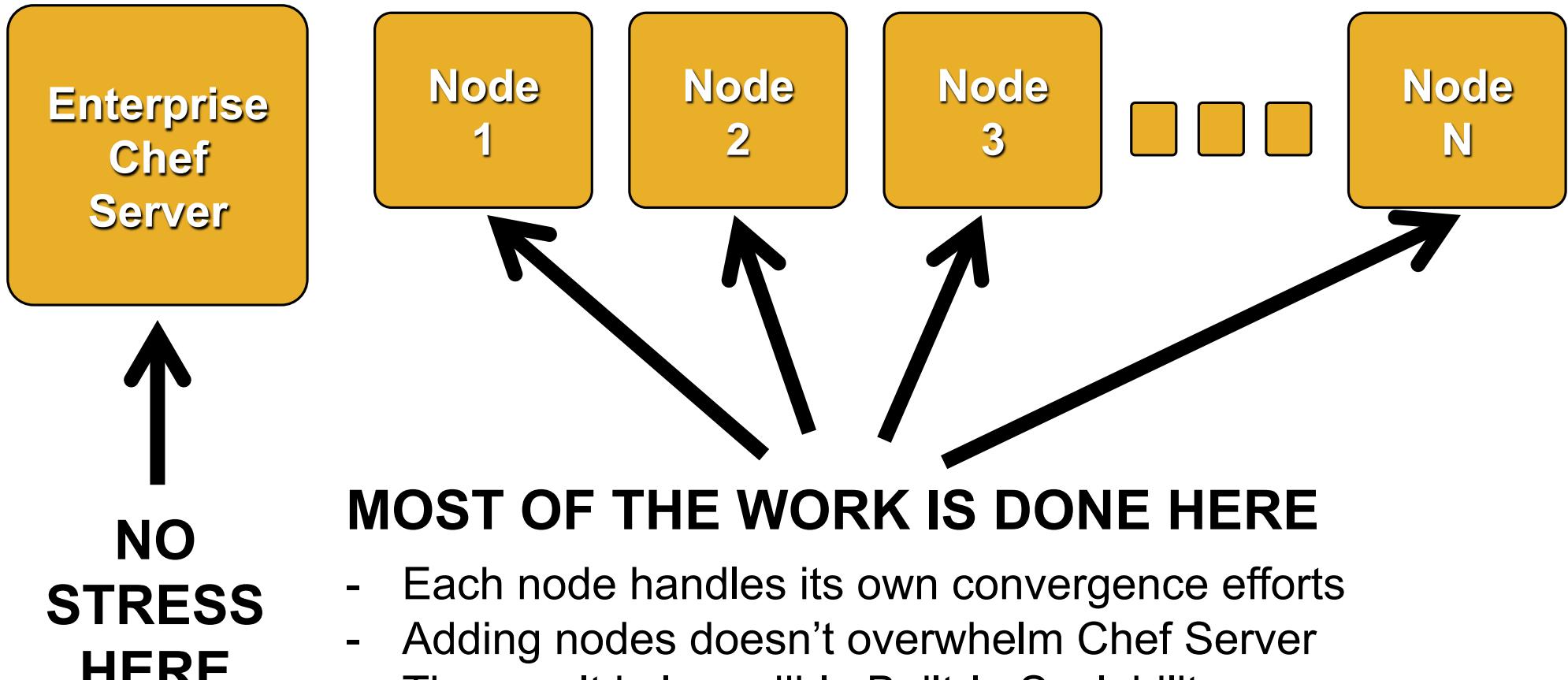
(i.e. “Apache needs to be Installed”)

but all **processing** of that configuration

(i.e. Actually installing Apache) happens **on** the nodes.

These configuration packages (known as **Cookbooks**) might be as small as 50Mb each.

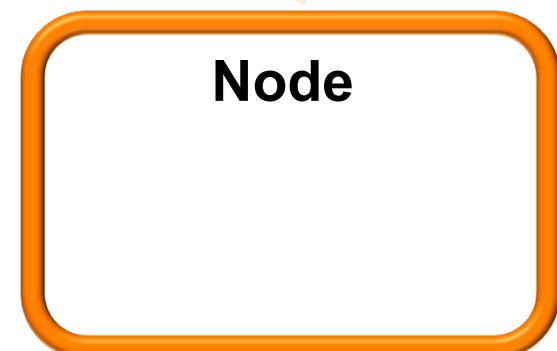
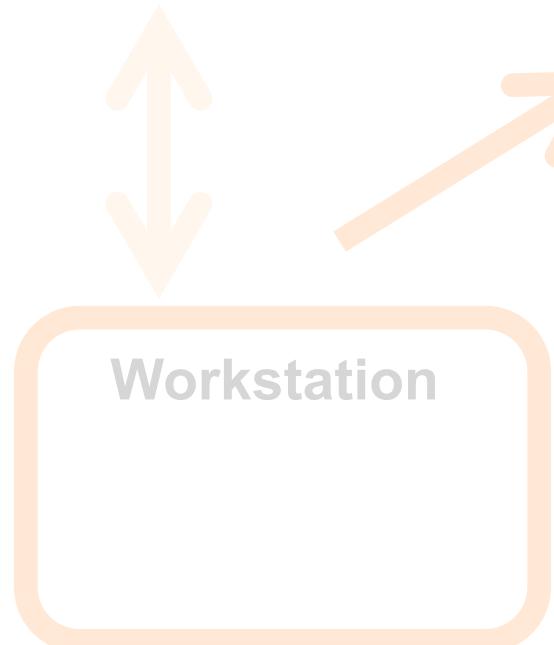
# Scalability



# Nodes

Repo:

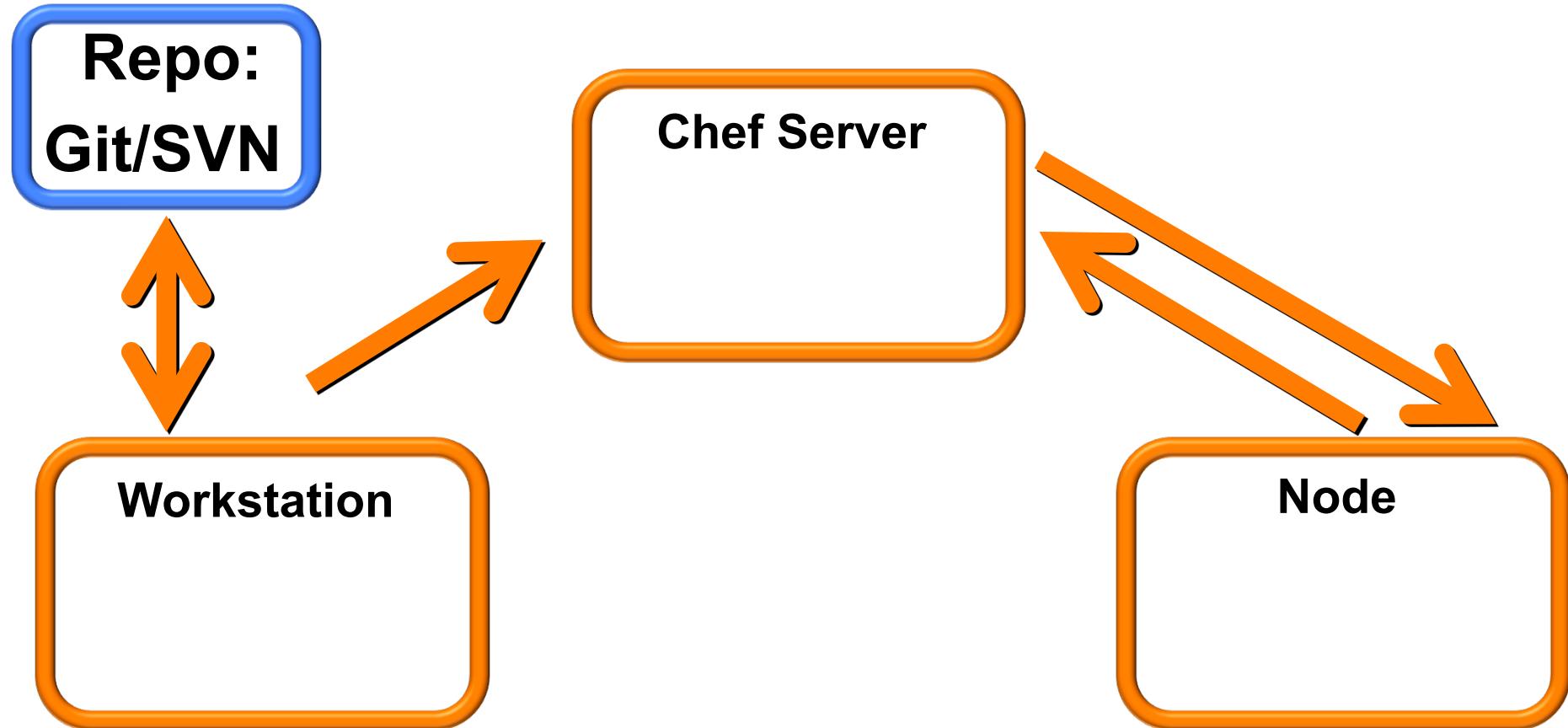
Git/SVN



# Nodes

- The computer you are managing with Chef
- Most any device that can run the **chef-client** application and **ruby**
- Typically a Server
  - Can be virtual, physical, owned or rented from a cloud, either public or private
- Can also be:
  - Network devices, tablets, phones
  - Chef can also manage most any device that has an **API** that allows management from another server

# Team Workflows



# Team Workflows and Backups

- Integrate your Workstation with an external repo
  - Such as: Git, SVN, BitBucket
- Allows version control for infrastructure configuration
- Good method for making Chef code backups
- Offers Team Collaboration

# Legend

v2.1.0\_DUAL



## Legend: Where do I run that command?

Example command to run **on your local workstation (ws)**

```
ws> whoami  
i-am-a-workstation
```

Hint: In this class, ‘knife’ commands will always be run on the local Workstation

Example command to run **on your remote node (rn)** i.e. in CloudShare

```
rn> whoami  
i-am-a-chef-node
```

## Legend: Example Terminal Command and Output

```
ws> ipconfig
```

```
Windows IP Configuration
```

```
Ethernet adapter Local Area Connection:
```

```
Connection-specific DNS Suffix . :  
IPv4 Address . . . . . : 10.38.161.230  
Subnet Mask . . . . . : 255.255.255.0  
Default Gateway . . . . . : 10.38.161.1
```

## Legend: Example of editing a file on your workstation

**OPEN IN EDITOR:** Windows: cookbooks\iis\_demo\recipes\default.rb  
-----  
Mac/Linux: cookbooks/iis\_demo/recipes/default.rb

Hi!

I am a friendly file.

**SAVE FILE!**

# A word about copy/paste

- **Don't Do It!**
  - Our class .pdf has embedded errors, making copy/paste a dangerous proposition
  - Writing code by hand is important for learning

# Workstation Setup

v2.1.0\_DUAL

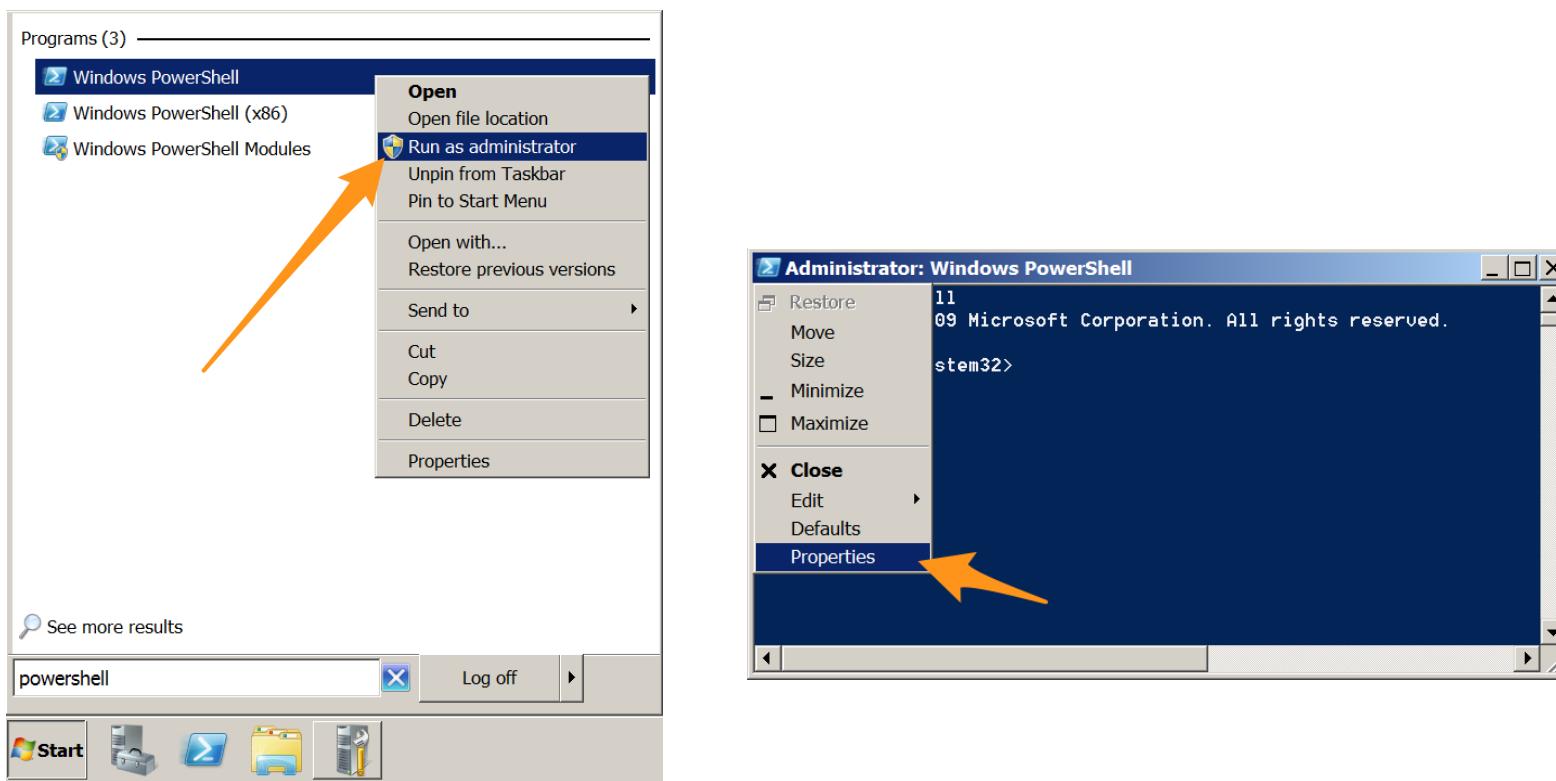


# Lesson Objectives

- After completing the lesson, you will be able to
  - Install Chef on your Workstation
  - Describe how to configure and use Knife, the Chef command line utility

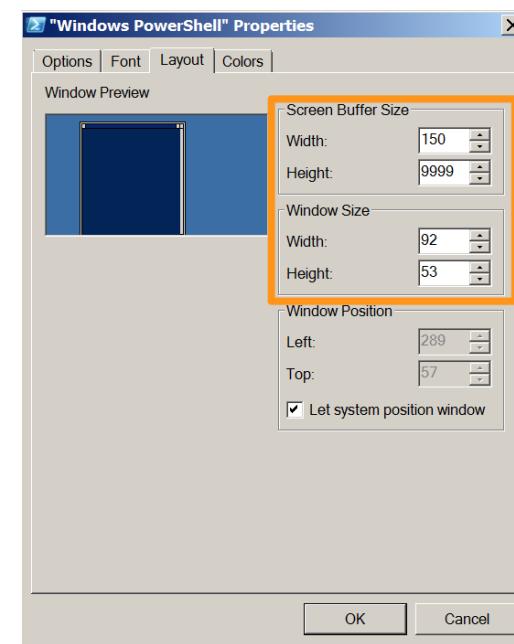
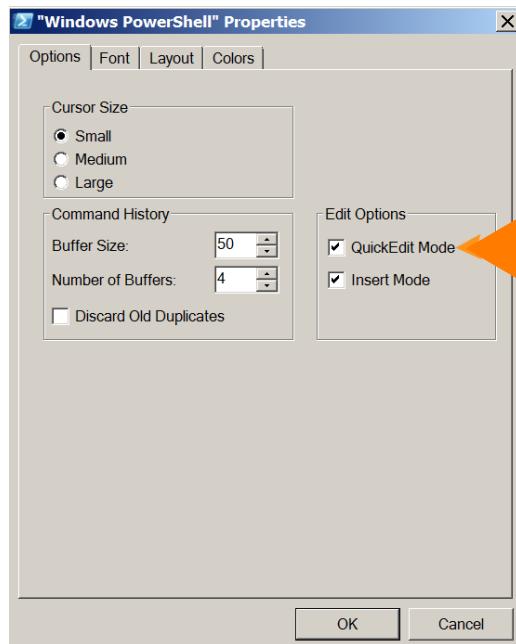
# Configure PowerShell

- Run PowerShell as Administrator and open Properties



# Configure PowerShell

- Enable QuickEdit Mode
- Set Height buffer to 9999



# Workstation Setup

- <http://downloads.chef.io>
- 2008 (Windows 7) or 2012 (Windows 8)
- i686 (32-bit) or x86\_64 (64-bit)

- Or browse to
- <https://www.chef.io/chef/install.msi>

## 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 information about manual installation, pl

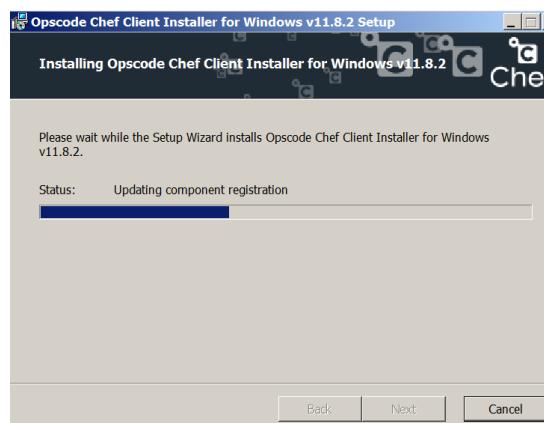
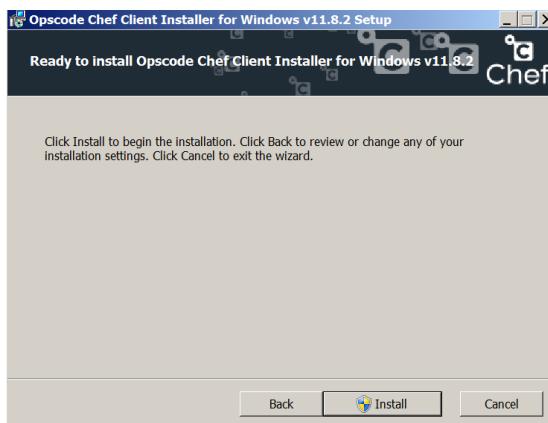
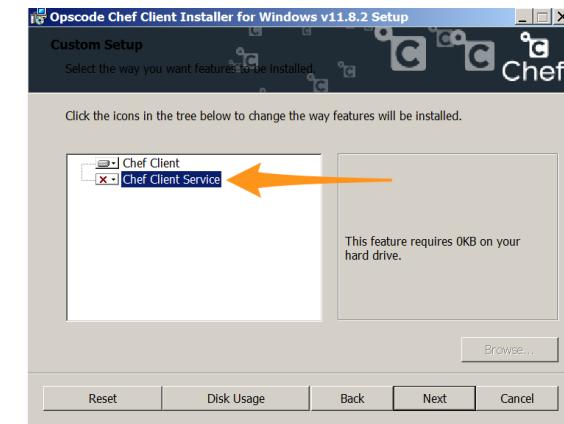
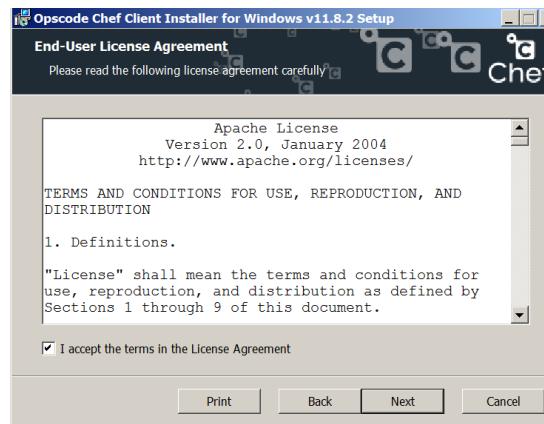
11.8.2-1

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

Download and install this file



# Install on Windows

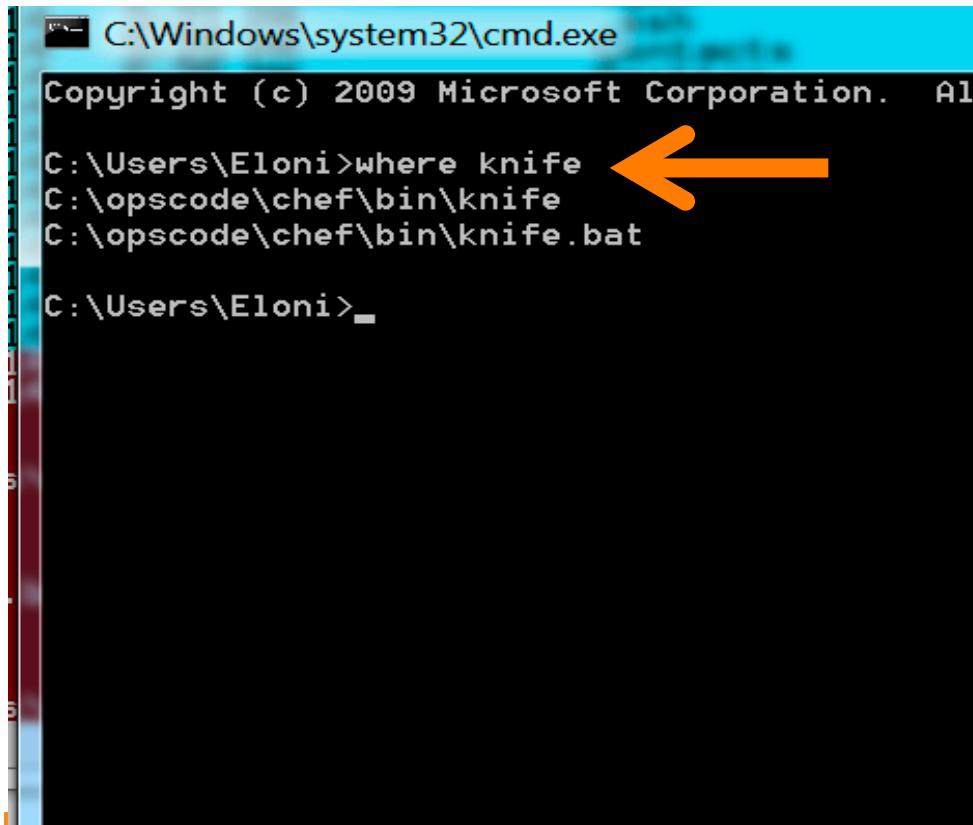


# After install

- Exit and **restart** your CMD and PowerShell windows to read the new Environment variables
- Check your system path (next slides)

# Test your install from CMD and PS

In Windows CMD prompt: where knife



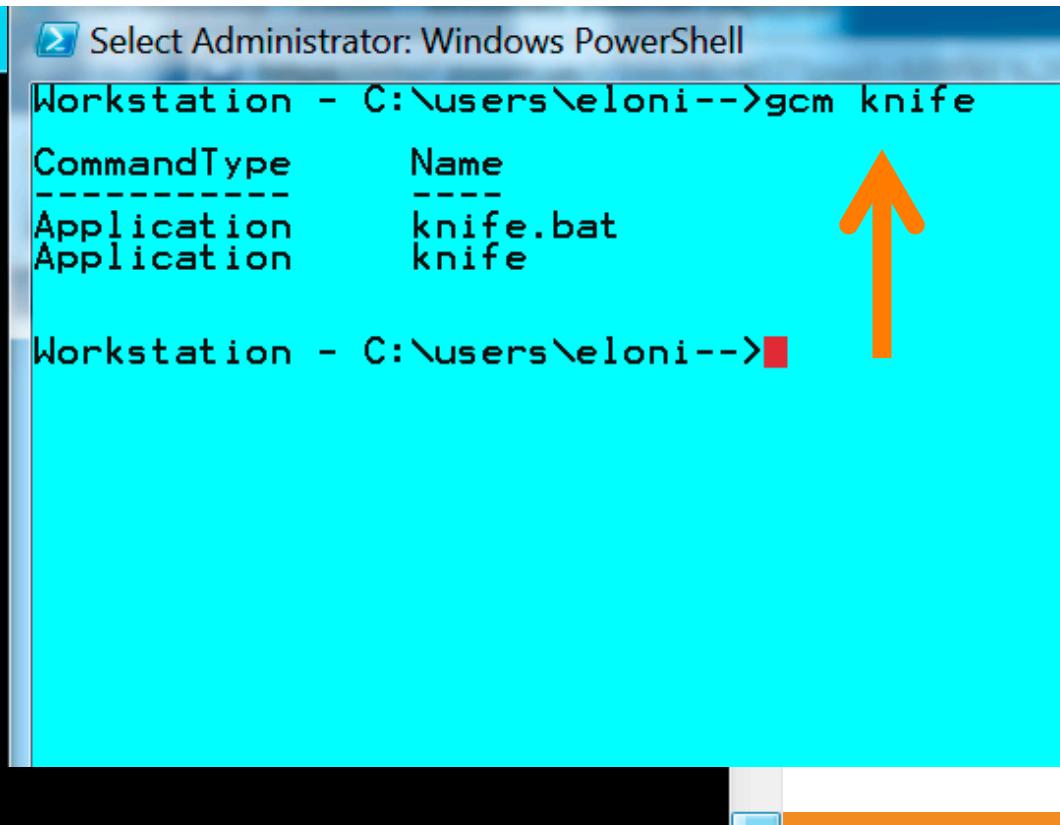
```
C:\Windows\system32\cmd.exe
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Eloni>where knife
C:\opscode\chef\bin\knife
C:\opscode\chef\bin\knife.bat

C:\Users\Eloni>
```

An orange arrow points to the first line of output, "C:\opscode\chef\bin\knife".

In Windows PowerShell: gcm knife



```
Select Administrator: Windows PowerShell
Workstation - C:\users\eloni-->gcm knife

 CommandType      Name
-----      ----
 Application      knife.bat
 Application      knife

Workstation - C:\users\eloni-->
```

An orange arrow points to the second line of output, "knife".

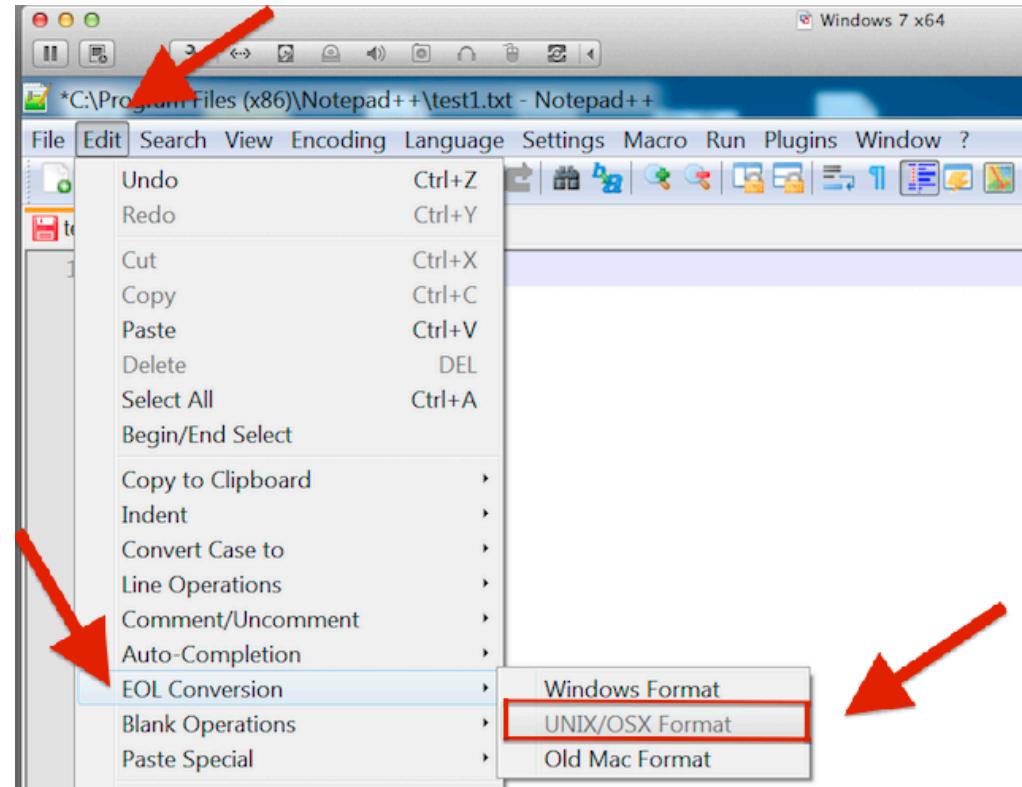
# Fixing your Windows path (if needed)

On your Windows Workstation:

1. Start -> View advanced system settings
2. Advanced -> Environment Variables
3. Find “Path” (under System variables)
4. Add **c:\opscode\chef\bin;**  
**c:\opscode\chef\embedded\bin;**
5. Put this at the **beginning** of the path
6. Re-open all CMD and/or PowerShell windows

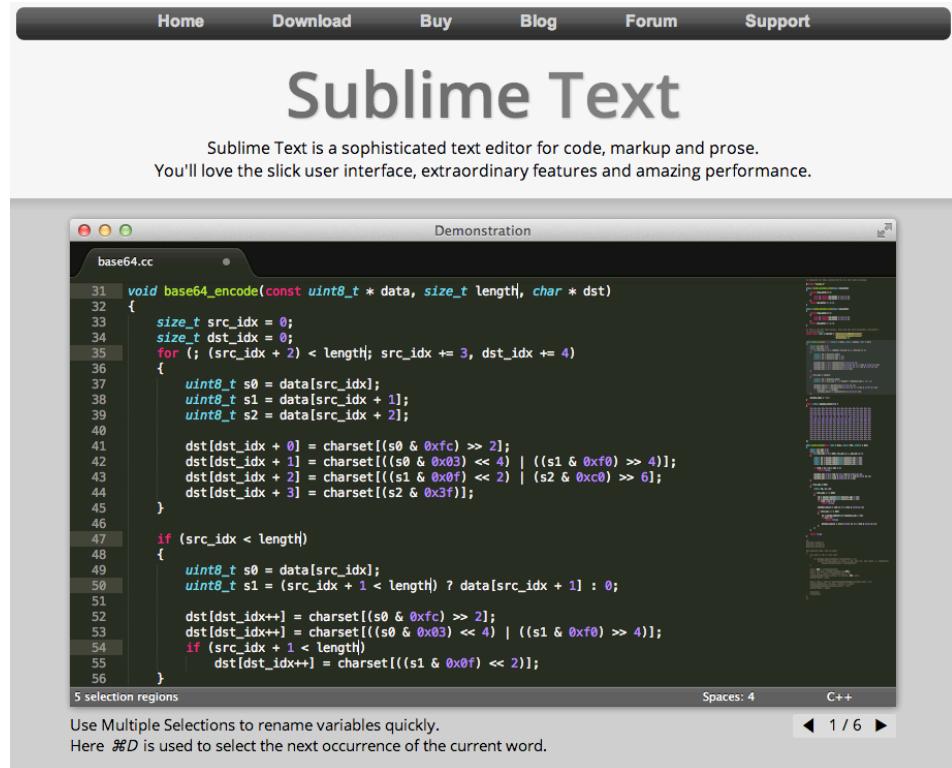
# A Note About Notepad++

In Notepad++, be sure to save your files as UNIX files, otherwise control characters that are undiscoverable will be embedded into your code.



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)



# Workstation Setup - Mac OS X / Linux

```
ws> 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  Speed100  6515  100  6515    0    0
20600      0  --::--  --::--  --::--  31172Downloading Chef for
ubuntu...Installing ChefSelecting previously unselected package chef.(Reading
database ... 47446 files and directories currently installed.)Unpacking chef
(from .../tmp.MqRJP6lz/chef_amd64.deb) ...Setting up chef (11.4.4-2.ubuntu.
11.04) ...Thank you for installing Chef!Processing triggers for initramfs-
tools ...update-initramfs: Generating /boot/initrd.img-3.2.0-48-virtual
```

## Fixing your Mac OSX/Linux path (if needed)

1. From a Terminal Window  
run: **\$> echo \$PATH**
2. Verify this is at the beginning of your path:  
**/opt/chef/embedded/bin**
3. If it needs to be modified:  
**\$> export PATH=/opt/chef/embedded/bin:\$PATH**



**Note: no '\$' in front of "PATH"**

# 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
  - ...and more

# Install Windows Knife plugin (all platforms)

- This step is only required when managing **Windows Nodes**
- You may need to re-run the gem install if you get a failure

```
ws> chef gem install knife-windows --no-ri  
--no-rdoc -V
```

OR

```
ws> sudo chef gem install knife-windows --no-ri  
--no-rdoc -V
```

**Successfully installed knife-windows 0.8.2**  
**19 gems installed**

# Organizations

v2.1.0\_DUAL



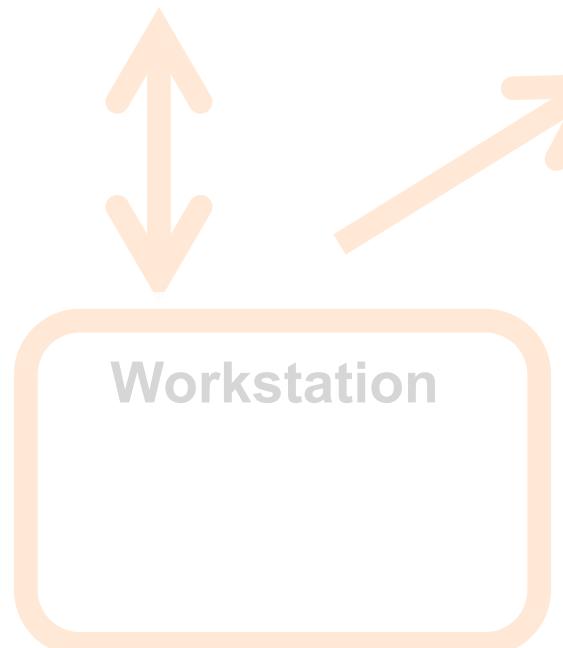
# Lesson Objectives

- After completing the lesson, you will be able to
  - Explain the purpose of Organizations
  - Create an Organization on Hosted Enterprise Chef
  - Login to your Organization
  - View and manage Organization in Enterprise Chef
  - Invite users into your Organization
  - Accept invitations into other Organizations

# Organizations exist on the Chef Server

Repo:

Git/SVN



Node

# Organizations

- The “sandbox” where all your nodes are registered and all your policies (configurations) are stored
- Nothing is shared between Organizations – they are completely independent
- One Enterprise Chef Server can contain many independent Organizations
- Different Organizations can represent different companies, departments, business units or projects

# Your Chef Server for this class...

- Hosted Enterprise Chef <http://www.chef.io>

The screenshot shows the Chef website homepage. At the top, there 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 circle highlights the 'Get Chef' button. Below the navigation, the word 'Home.' is partially visible. In the center, there is a large orange and grey circular logo. To the right of the logo, the word 'CHEF' is written in white. A large orange arrow points from the 'Get Chef' button down to the 'Hosted Chef' button. The 'Hosted Chef' button is also highlighted with an orange circle. The text next to it reads: 'You're almost ready to start your free trial of Enterprise Chef' and 'All you need to do to get started is choose how to deploy Chef:'. Below this, there are two options: 'Hosted Chef' (which is highlighted) and 'On Premises C...' (which is cut off). The 'Hosted Chef' section contains the following text: 'Let us manage Chef Server for you!', 'Hosted Enterprise Chef is the quickest and easiest way to get started with Chef.', and 'You don't have to worry about updates or maintenance. Just upload your cookbooks and we'll make sure you have access to a highly available, highly scalable Chef server.' To the right of the 'Hosted Chef' section, there is a smaller box with the heading 'On Premises C...' and the text 'Install and manage your own Chef Server' followed by a description.

# Create new account

- Sign up for a new account
- Chef Organization
  - provides multi-tenancy
  - name must be globally unique  
(to this particular Chef Server)

## 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	<input type="text"/>
Username	<input type="text"/>
Email	<input type="text"/>
Password	<input type="password"/>
Company	<input type="text"/> (Optional)
Chef Organization	<input type="text"/>

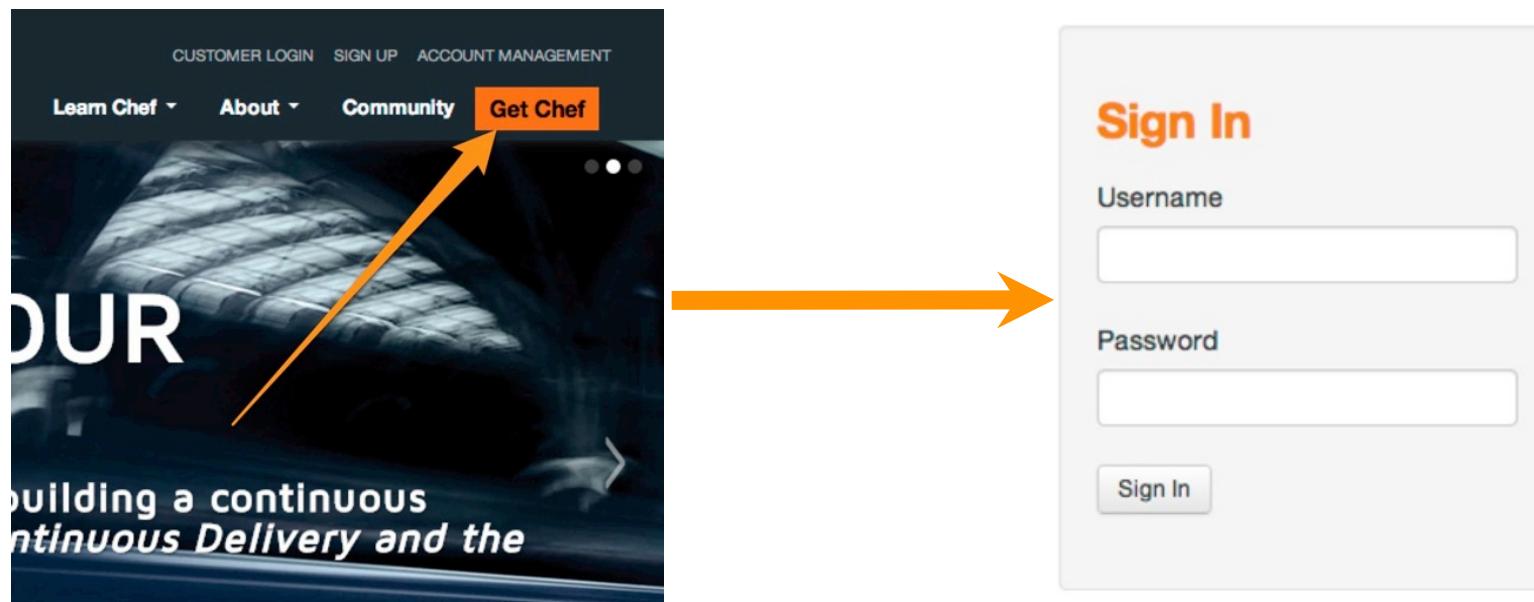
Organization is the name of your instance of Enterprise Chef.

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

[Get Started](#)

# Manage Organizations

- Login to your Hosted Enterprise Chef
- <https://manage.chef.io/login>



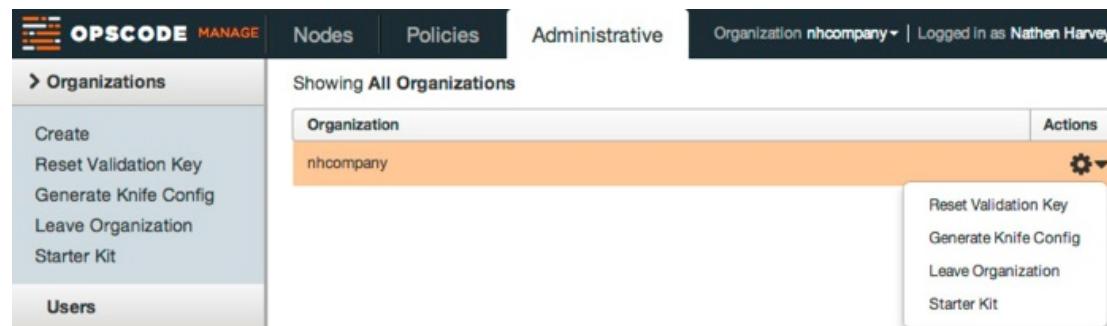
# Organizations

The screenshot shows the Opscode Manage interface with the following details:

- Header:** OPSCODE MANAGE
- Top Navigation:** Nodes, Policies, **Administrative** (highlighted with an orange border)
- Left Sidebar:**
  - Organizations:** Create, Reset Validation Key, Generate Knife Config, Leave Organization, Starter Kit
  - Users**
  - Groups**
  - Global Permissions**
- Main Content:** Showing All Organizations, listing "nhcompany".

# Manage Organizations

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



# Download the "Starter Kit" for your Org

Notes about the Starter Kit:

- Every time you download a **new** Starter Kit, you get a **new** authentication certificate, **invalidating the old one**, so you have to use the **NEW** .pem files, NOT the old .pem files
- **If one person on a team downloads a new Starter Kit, they need to share the new certificate with the rest of the team, or the team won't have access to the Organization**

# Download the "Starter Kit" for your Org

- You'll download **chef-starter.zip** from clicking this button
- When you unzip chef-starter.zip you'll get a directory named "**chef-repo**"
- Put "chef-repo" in your home directory
  - C:\Users\<your\_name>\chef-repo (Win)
  - /Users/<your\_name>/chef-repo (Mac)
  - /home/<your\_name>/chef-repo (Linux)

Thank you for choosing

Follow these three steps to be o

[Download Starter Kit](#)

What's next?

[Chef Documentation](#)

The best place to start learning about Chef in general.

[Browse Co  
Cookbooks](#)

Hundreds of r  
Chef commun  
cookbooks yo  
inspiration fro

# 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
- The policies (configurations) in your chef-repo match the policies in your Organization

# Exercise: View the chef-repo

```
ws> cd SOMEDIR/chef-repo
```

**Note:** You are on your local Workstation

# View the chef-repo

Windows workstation:

```
ws> dir
```

Mac/Linux workstation:

```
ws> ls -al
```

9/24/2013 11:09 PM	.chef
9/24/2013 11:09 PM	cookbooks
9/24/2013 11:09 PM	roles
9/24/2013 11:09 PM	495 .gitignore
9/24/2013 11:09 PM	2292 README.md
9/24/2013 11:09 PM	3572 Vagrantfile

# Inside the .chef directory

Windows workstation:

```
ws> dir .chef
```

Mac/Linux workstation:

```
ws> ls -l .chef
```

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

Note: Contents of the .chef directory are the same for Mac, Linux and Windows workstations

# What's inside the .chef directory?

- `knife.rb` is the configuration file for Knife.
- The other two files in `.chef` are certificates for authentication with the Chef Server (covered later)

# knife.rb sample

**OPEN IN** Windows: C:\Users\You\chef-repo\.chef\knife.rb  
**EDITOR:** Mac/Linux: ~/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" ]
```

# Navigating a proxy from Workstation

[http://docs.chef.io/config\\_rb\\_knife.html](http://docs.chef.io/config_rb_knife.html)

Can set **proxy details** to allow communication between the Workstation and the Chef Server, if needed

# 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 & cloud resources, including provisioning
  - Cookbooks and recipes
  - Roles and Environments
  - Data Bags (stores of JSON data), including encrypted data
  - Searching of indexed data on the Chef Server

# Verify Knife

```
ws> knife --version  
Chef: 11.16.4
```

- Your version may be slightly different, that's ok!
- During this course, the 'knife' command is only run on the workstation

# 'knife client list' command

- Displays a list (from Chef Server) of all **clients** with access to your **Organization**
  - Includes all **Workstations** and **Nodes**
- Reads the **chef\_server\_url** from knife.rb
- Invokes HTTP GET to the Chef Server at:  
`#{{chef_server_url}}/clients`

# Exercise: Run ‘knife client list’

```
ws> knife client list  
ORGNAME-validator
```

Note: **ORGNAME-validator** represents your workstation. When we have nodes configured, they will show up here as well.

If this command works, then the connection from your workstation to the Chef Server is setup correctly!

# Exercise: Run ‘knife help list’

```
ws> knife help list
```

Available help topics are:

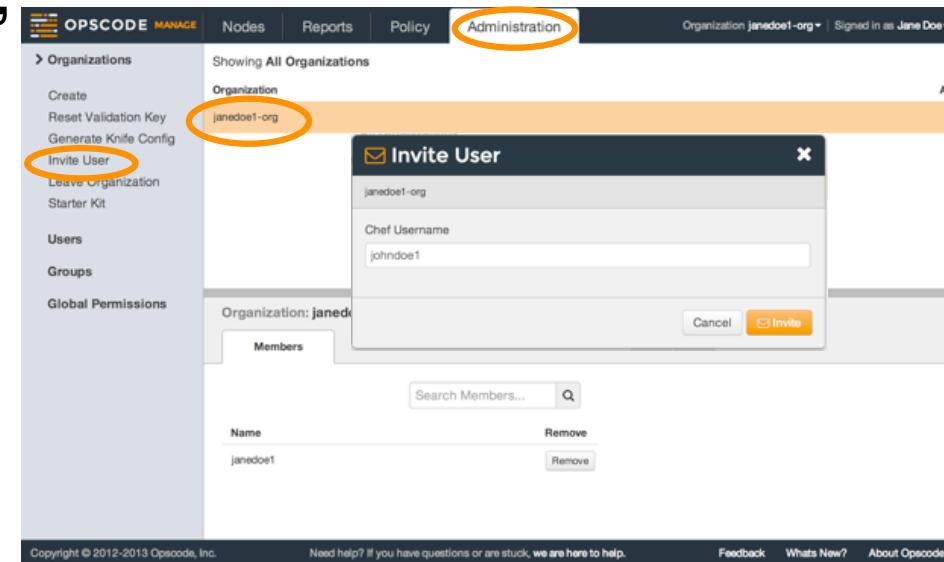
- bootstrap
- chef-shell
- client
- configure
- cookbook
- cookbook-site
- data-bag
- environment
- exec
- index
- Knife
- ...

# Exercise: Invite users into your org

- **The Problem:** You need assistance from the instructor for your chef course
- **Proposed Solution:** Invite the instructor into your organization so they can log into the chef server with their own credentials, but can see your organization

# Exercise: Invite your instructor into your org

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

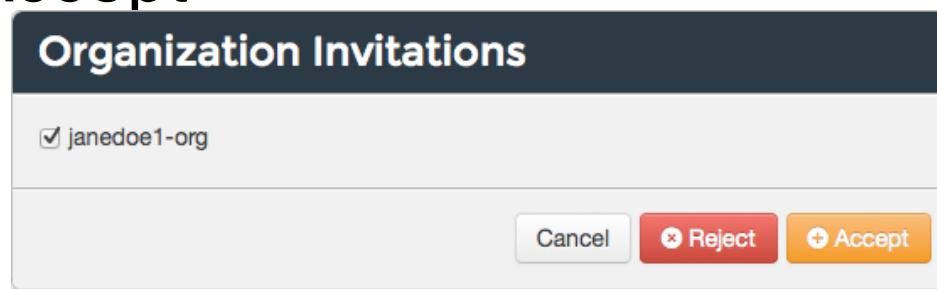


# Exercise: Accepting an invite

- Your instructor will get a notification and an email once you have invited them into your account

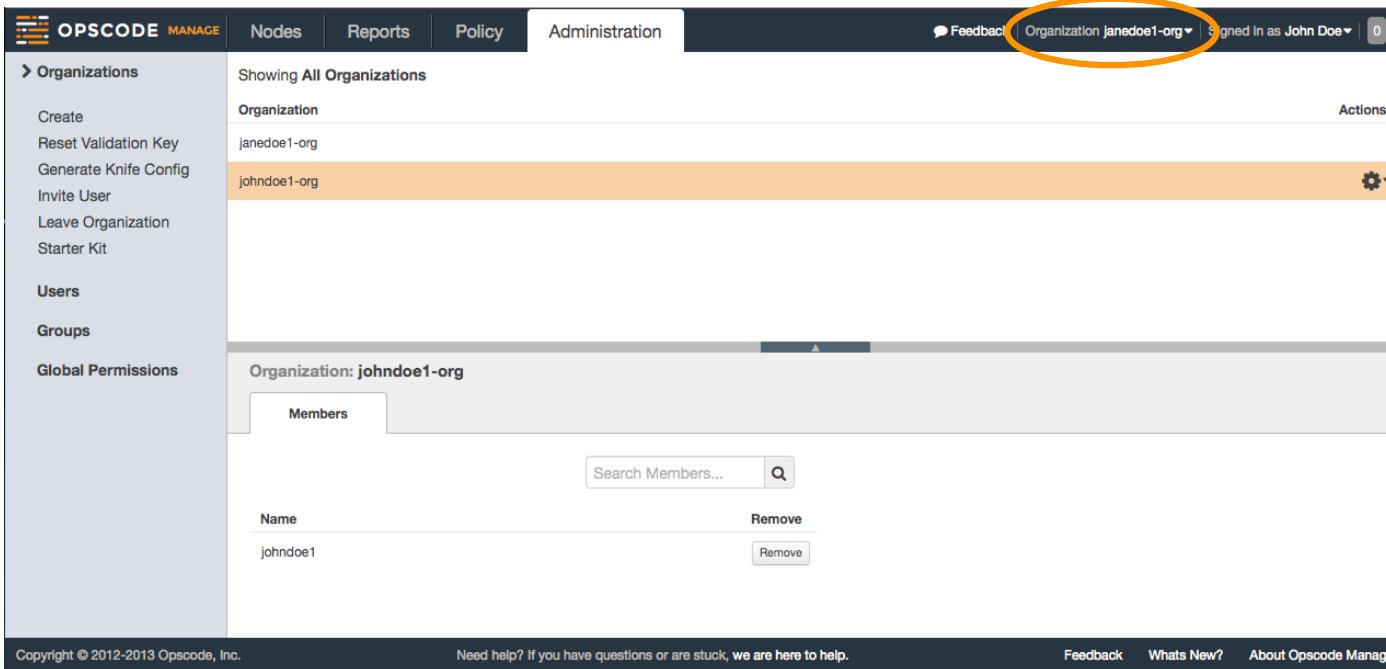


- They will click the notification, select the Organization and click 'Accept'



# Exercise: Viewing an org

- Selecting an organization from the drop down list and perusing that org



The screenshot shows the Opscode Manage web interface. At the top, there's a navigation bar with tabs for Nodes, Reports, Policy, Administration, Feedback, Organization (with a dropdown menu showing 'janedoe1-org'), and Signed in as John Doe. A notification icon with the number '0' is also present. On the left, a sidebar has sections for Organizations (Create, Reset Validation Key, Generate Knife Config, Invite User, Leave Organization, Starter Kit), Users, Groups, and Global Permissions. 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. Below this, a modal window is open for the 'johndoe1-org' organization, specifically the 'Members' tab. It contains a search bar labeled 'Search Members...' with a magnifying glass icon, and a table with one entry: 'Name' (johndoe1) and 'Remove' (a button). At the bottom of the page, there's a footer with links for Copyright © 2012-2013 Opscode, Inc., Need help? (with a link to 'we are here to help.'), Feedback, What's New?, and About Opscode Manage.

# Review Questions

- What is an Organization?
- How do you regenerate the Starter Kit for your Organization?
- What is the chef-repo?
- What is knife?
- What is path and name of the knife configuration file?

# Configuring Nodes

v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - **Chef enable** an instance using "knife bootstrap"
    - Making it a “node”
  - Explain how knife bootstrap configures a node to use the Organization created in the previous section

# 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 logo and the word "MANAGE". Below the header, there is a navigation bar with three tabs: "Nodes" (which is highlighted in white), "Policies", and "Administrative". The main content area has a title "Showing All Nodes" and a message "There are no items to display." with an information icon.

**CHEF**  
MANAGE

Nodes Policies Administrative

Showing All Nodes

There are no items to display.

> Nodes

Delete  
Manage Tags  
Reset Key  
Edit Run List  
Edit Attributes

# Training Nodes For Windows/Linux Labs

- We each have two nodes in the CloudShare training environment
- **Windows Node:** Windows Server 2012 x64
  - 2GB RAM / 40GB HDD
  - Administrator level permissions
- **Linux Node:** Centos 6.X
  - 2GB RAM / 15GB HDD
  - ‘sudo’ or root level permissions

# Training Nodes For Our Windows Labs

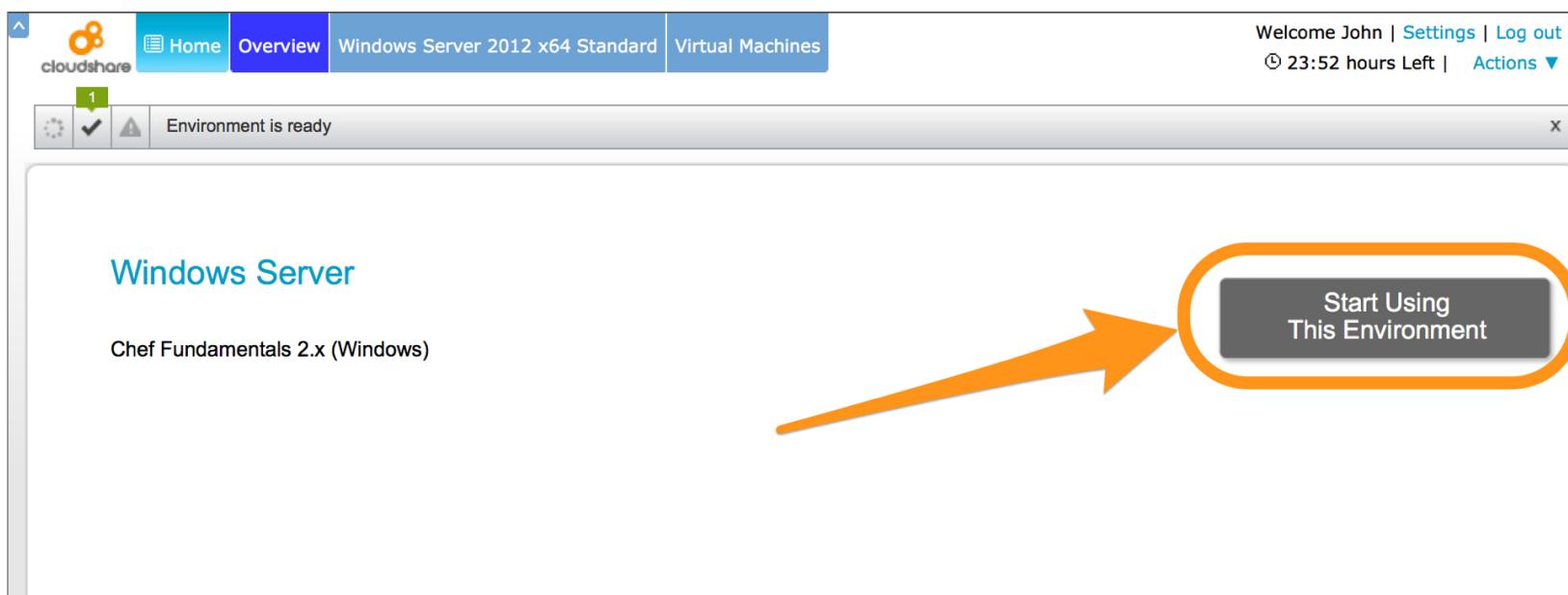
- We each have a node in the CloudShare training environment
- **Windows Node:** Windows Server 2012 x64
  - 2GB RAM / 40GB HDD
  - Administrator level permissions

# Training Nodes For Our Linux Labs

- We each have a node in the CloudShare training environment
- **Linux Node:** Centos 6.X
  - 2GB RAM / 15GB HDD
  - ‘sudo’ or root level permissions

# CloudShare Node

- Register and login to CloudShare (see invite)
- Start Using This Environment



# CloudShare Node

Welcome John | [Settings](#) | [Log out](#)  
⌚ 23:51 hours Left | [Actions ▾](#)

1 Environment is ready X

Windows Server

**Windows Server 2012**

**Windows Server 2012 x64 Standard**

**Description:** OS: Windows Server 2012 x64  
Spec: 40 GB HD/2 GB RAM

**OS:** Windows  
**State:** Running

[More details ►](#)

[View VM](#)

[RDP file](#) [Reboot VM](#) [Revert](#) [Delete](#)

# CloudShare Node Hostname

The screenshot shows a CloudShare interface for a virtual machine named "Windows Server 2012 x64 Standard". The machine is described as running Windows Server 2012 x64 with 40 GB HD/2 GB RAM. It is currently running. The "VM Details" section is expanded, showing the following information:

<b>External Address:</b>	uv01i6gvz7p0mup9j.vm.cld.sr
<b>Internal IP:</b>	10.160.33.236
<b>Total Memory:</b>	2048 MB
<b>Disk Size:</b>	40 GB
<b>CPU:</b>	1
<b>The machine was prepared in 3 minutes</b>	
<b>Credentials (show password)</b>	
<b>Auto-login:</b>	Administrator (local user)
<b>Username:</b>	Administrator

Two orange arrows point to the "External Address" field and the "Credentials (show password)" link.

# Checkpoint – Windows and Linux

- At this point you should have
  - One **Windows** and one **Linux** Virtual Machine (VM)
  - The IP Address or public hostname of each VM
  - An application for establishing an RDP connection
    - 'RDP' permissions on the **Windows** VM
  - An application for establishing an SSH connection
    - 'sudo' or root-level permissions on the **Linux** VM

# Checkpoint - Windows

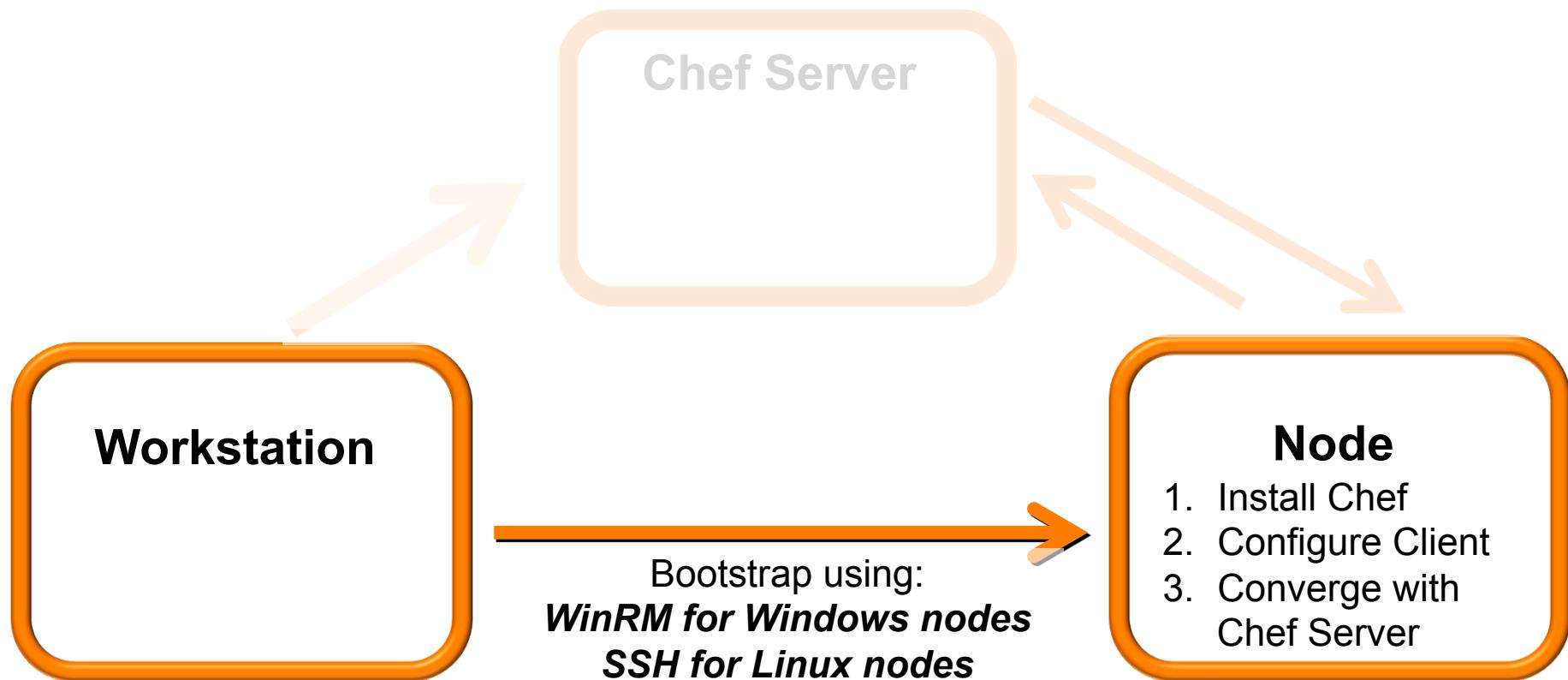
- At this point you should have
  - One **Windows** Virtual Machine (VM)
  - The IP Address or public hostname of the VM
  - An application for establishing an RDP connection
    - 'RDP' permissions on the **Windows** VM

# Checkpoint - Linux

- At this point you should have
  - One **Linux** Virtual Machine (VM)
  - The IP Address or public hostname of the VM
  - An application for establishing an SSH connection
    - 'sudo' or root-level permissions on the **Linux** VM

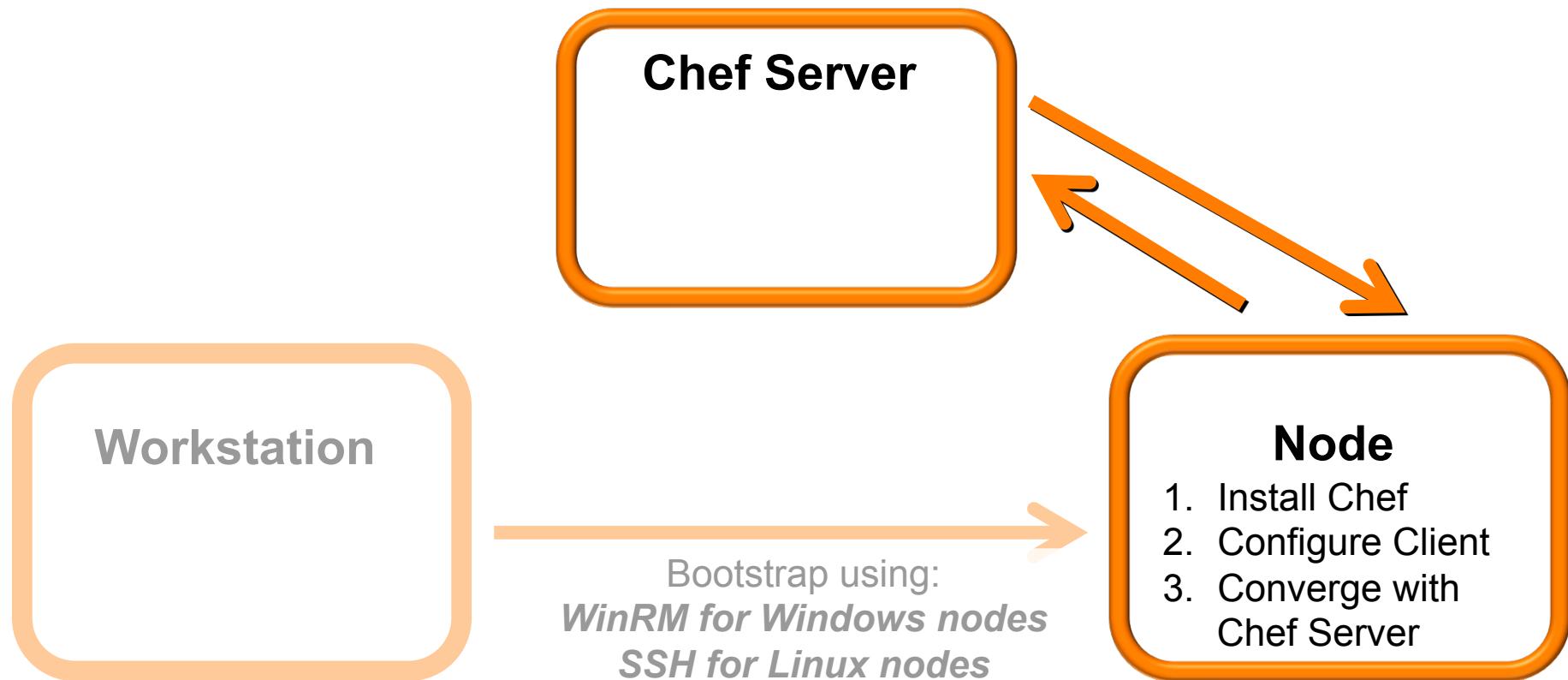
# Bootstrapping a Node

**First:** The Workstation bootstraps the Node



# Bootstrapping a Node

**Second:** The Node contacts Chef Server



# "Bootstrap" command

To Bootstrap a **Linux** node

```
ws> knife bootstrap <hostname or IP Address> -x username -P password -N "linuxnode"
```

-x = username  
-P = password  
-N = rename the node (from the default of FQDN)

To Bootstrap a **Windows** node

```
ws>knife bootstrap windows winrm <hostname or IP> -x username -P password -N "windowsnode"
```

**NOTE: If you are using more then one node within your Organization,  
name them differently or you'll get an error!**

# 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 the FQDN to make typing easier using:
  - –N “windowsnode” or –N “linuxnode”
- In production, let the **default** FQDN be the node name

# Node names

View node names here

Nodes      Reports      Policy      Adminis

Showing All Nodes

Node Name	Platform	F
linuxnode5		
windowsnode2	windows	C
windowsnode3	windows	C
windowsnode5		

# What just happened?

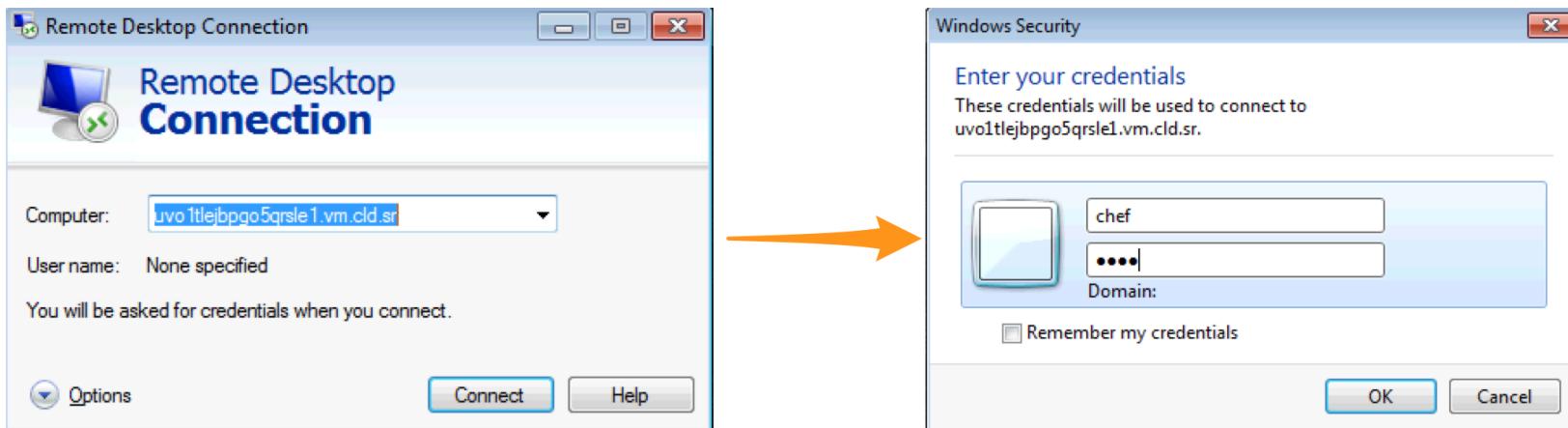
- Chef and its dependencies installed on the Node via an operating system-specific package ("omnibus installer")
- Installation includes
  - The Ruby language - used by Chef
  - Configured the node with correct security certificates
  - chef-client - Client application (same version as your workstation)
  - ohai - System profiler (which creates the node object)

# What just happened?

- The security and configuration files can be found at:
  - Windows node: C:\chef
  - Linux node: /etc/chef
- ORGNAME-validator.pem copied from the Workstation to the Node, and renamed to **validation.pem**
- **client.rb** created on the Node, with the configuration needed by the Node to connect with the Chef Server

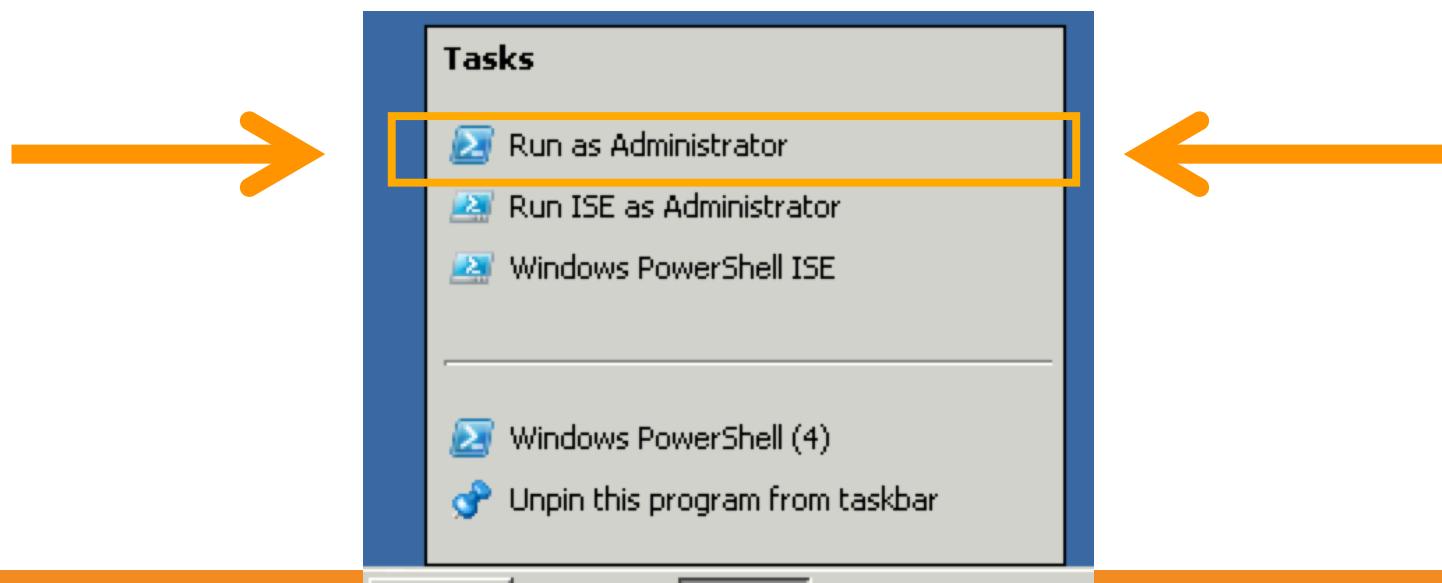
## Windows node: Verify Target Instance is Configured Properly

- Use Remote Desktop to connect to the host.
- User: **Administrator** Password: **<your\_password>**



## Windows node: Verify Target Instance is Configured Properly

- On the VM
  - Close all existing PowerShell windows
  - Open an **Administrator** Powershell window



## Windows node: Verify Target Instance is Configured Properly

```
rn\> dir c:\chef
```

```
Directory: C:\chef
```

Mode	LastWriteTime	Length	Name
d----	10/7/2013 3:41 PM		cache
-a---	10/4/2013 7:18 PM	1702	client.pem
-a---	10/4/2013 7:17 PM	434	client.rb
-a---	10/4/2013 7:17 PM	17	first-boot.json
-a---	10/4/2013 7:17 PM	1706	validation.pem
-a---	10/4/2013 7:16 PM	161	wget.ps1
-a---	10/4/2013 7:16 PM	1273	wget.vbs

# Windows node: Examine C:\chef\client.rb

```
rn> gc c:\chef\client.rb
```

```
log_level          :info
log_location       STDOUT

chef_server_url   "https://api.opscode.com/organizations/ORGNAME"
validation_client_name ORGNAME-validator"
client_key         "c:/chef/client.pem"
validation_key     "c:/chef/validation.pem"

file_cache_path   "c:/chef/cache"
file_backup_path  "c:/chef/backup"
cache_options      ({:path => "c:/chef/cache/checksums", :skip_expires =>
true})

node_name "windowsnode"
```

## Windows node: Examine C:\chef\first-boot.json

```
rn> gc c:\chef\first-boot.json
```

```
{"run_list":[]}
```

- You can bootstrap nodes with recipes/run lists right away
- Disaster recovery!

## Linux node: Verify Target Instance is Configured Properly

```
ws> ssh root@<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: Tue July  8 16:26:24 2014 from  
host86-145-117-53.range86-145.btcentralplus.com  
[chef@CentOS63 ~]$
```

## Linux node: Verify Target Instance is Configured Properly

```
rn> ls /etc/chef  
client.pem  client.rb  first-boot.json  
validation.pem
```

```
rn> which chef-client  
/usr/bin/chef-client
```

## Linux node: Examine /etc/chef/client.rb

```
rn> cat /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       "linuxnode"
```

## Linux node: Examine /etc/chef/first-boot.json

```
rn> cat /etc/chef/first-boot.json
```

```
{"run_list":[]}
```

- You can bootstrap nodes with recipes/run lists right away

# Change log level on the node (all platforms)

**OPEN IN EDITOR:** Windows node: c:\chef\client.rb

Linux node /etc/chef/client.rb

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

- Ensure the default log level for chef-client is :**info**
- This setting gives you more information when you converge with Chef Server
- More configuration options can be found on the docs site:  
[http://docs.chef.io/config\\_rb\\_client.html](http://docs.chef.io/config_rb_client.html)

# Run chef-client on the node

```
rn> chef-client
```

- ‘chef-client’ is the executable to start the convergence on either Windows or Linux nodes

# View Node on Chef Server

- Login to your Hosted Enterprise Chef



# View Node on Chef Server

The screenshot shows the Chef Manage interface. The top navigation bar has tabs for Nodes, Reports, Policy, and Administration. The Nodes tab is highlighted with an orange circle. On the left sidebar, under the Nodes section, there are links for Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area shows a table titled "Showing All Nodes" with columns: Node Name, Platform, FQDN, IP Address, Uptime, Last Check In, Environment, and Actions. A row for "node1" is selected, highlighted with an orange circle. The "Actions" column for node1 contains a gear icon with a dropdown arrow, also circled in orange. Below the table, a modal window is open for "Node: node1". It has tabs for Details, Attributes, and Permissions. The Details tab is active. It displays "Last Check In: 24 Minutes" (2014-02-21 11:15), "Uptime: 44 Minutes" (Since 2014-02-21 15), "Environment: \_default", "Platforms: windows", "FQDN: C1263834251", and "IP Address: 10.160.33.236". The Attributes tab shows a "Tags" section with a "+ Add" button and a message "There are no items to display." The Permissions tab is currently inactive. At the bottom of the modal, there are "Run List" buttons for "Expand All", "Collapse All", and "Edit". The footer of the page includes copyright information ("Copyright © 2012–2014 Chef, Inc."), a help link ("Need help? If you have questions or are stuck, we are here to help."), and links for Feedback, What's New?, and About Chef Manage.

Nodes

Showing All Nodes

Node Name	Platform	FQDN	IP Address	Uptime	Last Check In	Environment	Actions
node1	windows	C1263834251	10.160.33.236	44 minutes	27 minutes ago	_default	▾

Node: node1

Details

Attributes

Permissions

Last Check In: 24 Minutes  
2014-02-21 11:15

Uptime: 44 Minutes  
Since 2014-02-21 15

Environment: \_default

Platforms: windows

FQDN: C1263834251

IP Address: 10.160.33.236

Tags

+ Add

There are no items to display.

Run List

Expand All

Collapse All

Edit

Version Position

Copyright © 2012–2014 Chef, Inc.

Need help? If you have questions or are stuck, we are here to help.

Feedback

What's New?

About Chef Manage

# 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, a table lists nodes, showing one entry: node1 (Platform: windows, FQDN: C1263834251, IP Address: 10.160.33.236, Uptime: 44 minutes, Last check-in: 24 minutes ago). To the right of the table, there's a sidebar with sections for Environment, Platforms, FQDN, and IP Address. Below the table, there are tabs for Details, Attributes, and Permissions, with the Details tab circled in orange. Under the Details tab, there are two boxes: 'Last Check in: 24 Minutes 2014-02-21 11:15' and 'Uptime: 44 Minutes Since 2014-02-21 15:15'. Further down, there are sections for Tags (with a '+ Add' button) and Run List (with 'Expand All' and 'Collapse All' buttons). A message at the bottom of the Tags section says 'There are no items to display.' At the very bottom of the page, there's a footer with copyright information, help links, and a feedback link.

Nodes Reports Policy Administration

Showing All Nodes

Node Name	Platform	FQDN	IP Address	Uptime	Last Check In
node1	windows	C1263834251	10.160.33.236	44 minutes	27 minutes ago

Node: node1

Details Attributes Permissions

Last Check in: 24 Minutes 2014-02-21 11:15

Uptime: 44 Minutes Since 2014-02-21 15:15

Environment: \_dev

Platforms: windows

FQDN: C1263834251

IP Address: 10.160.33.236

Tags + Add

Run List

Expand All Collapse All

Copyright © 2012 – 2014 Chef, Inc.

Need help? If you have questions or are stuck, we are here to help.

Feedback

# Review Questions

- What is the path and name of the chef-client configuration file on the node?
- What is the command to run chef on the node?
- What does a knife bootstrap do?



# Chef Configuration Concepts

v2.1.0\_DUAL



# Lesson Objectives

- Define the purpose and relationship of:
  - **Resources**
  - **Recipes**
  - **Cookbooks**
  - **Run Lists**
- Understand **Configuration Drift**, and now to mitigate this

# Declarative Interface to Resources

- A “Policy” is the collection of system configuration that you define
- You define the policy in your Chef configuration Recipes and Cookbooks
- Your policy declares what **state** each resource **should be in**  
but  
**not how to get there**

# Nodes Adhere to Policy

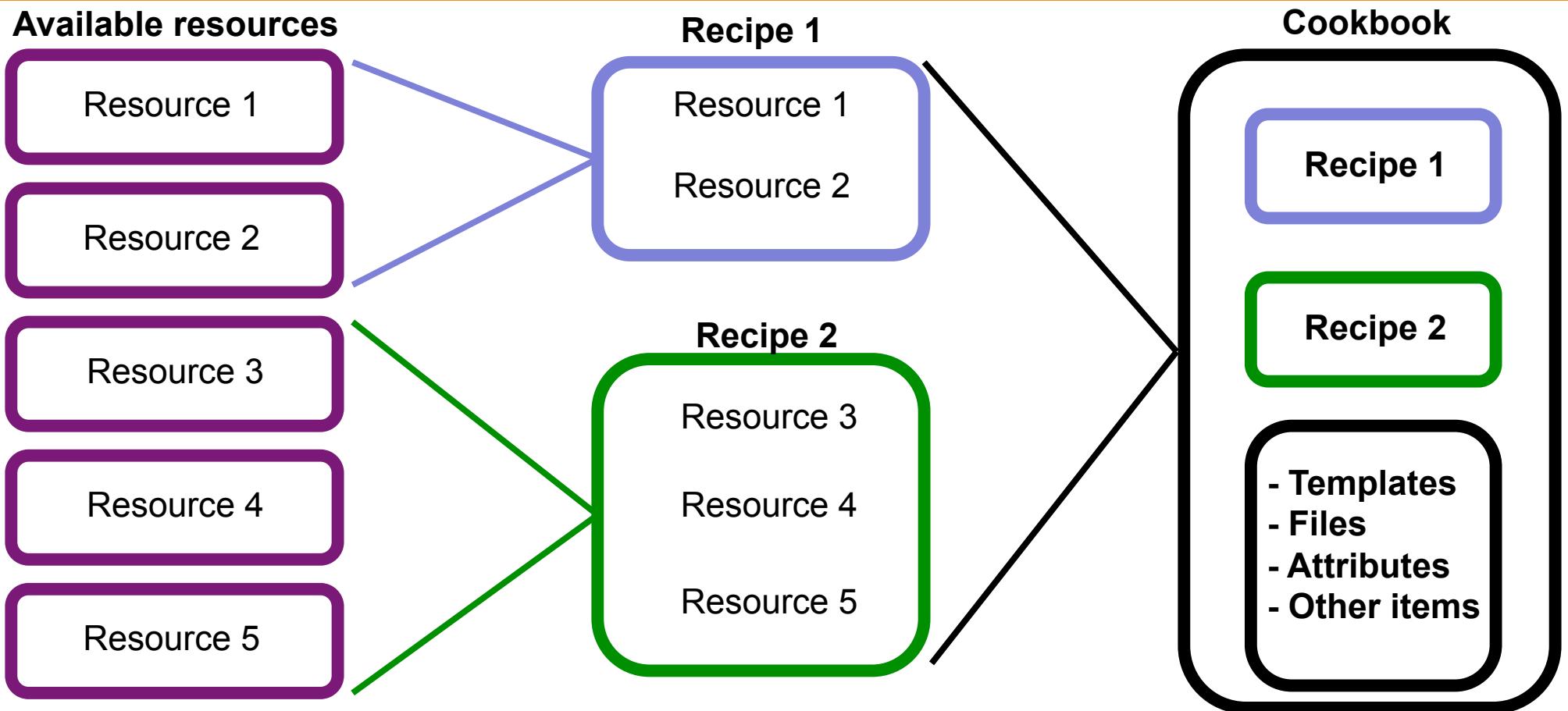
- **Examples:**

- Policy states “IIS should be installed”
  - If IIS is not installed, then node installs it
  - If IIS is already installed, then node does nothing
- Policy states ”/var/www/html/hello\_world.html” should exist
  - If it does not, then create the file
  - If the file already exists (and has the exact same content), then do nothing

# Resources -> Recipes -> Cookbooks

- **Resources** configure individual items on a node
  - Eg: create a directory
- **Resources** are gathered into **Recipes**
- **Recipes** are stored in **Cookbooks**
- **Cookbooks** contain:
  - Recipes and the items that support those recipes

# Resources -> Recipes -> Cookbooks



# Resources

- Represent a piece of the system and its desired state
- Fundamental building blocks of Chef configuration
- Affects a single element of the managed node
- Examples:
  - Installing a package
  - Starting a service
  - Creating a directory, with specific permissions
  - Running a Powershell script on Windows Server

# Windows Resource Example

- Add a key to the registry

```
registry_key "HKEY_LOCAL_MACHINE\SOFTWARE\  
\Microsoft\Windows\CurrentVersion\Policies\System" do  
  values [{  
    :name => "EnableLUA",  
    :type => :dword,  
    :data => 0  
  }]  
  action :create  
end
```

# Linux Resource Example

- Create an index.html file for Apache webserver to serve up

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

# Recipes

- Configuration files that contain one or more **resources**
- Recipes have a larger business purpose then a resource
- Example of a **recipe**:
  - A single configuration file that contains all the **resources** needed to **install**, **configure** and **start** a database for immediate access by application servers

# Example Windows Server Recipe

```
windows_package "Microsoft .NET Framework 4.0" do
  action :install
  source "http://download.microsoft.com/download/dotNetFx40_Full_x86_x64.exe"
  options "/quiet /norestart"
end
```

```
windows_registry "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" do
  values "FdenyTSConnections" => 0
end
```

**Note:** This **recipe** contains **three** resources

```
service "W3SVC" do
  action [:disable,:stop]
end
```

# Example Linux Server Recipe

```
package "net-snmp" do
  action :install
end
```

```
template "/etc/snmpd.conf" do
  source "snmpd.conf.erb"
  owner "root"
  group "root"
  mode 0644
  variables(:community_string => "not_public")
  notifies :restart, "service[snmpd]"
end
```

**Note:** This **recipe** contains **three** resources

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

# What is a cookbook?

- A cookbook is like a “package” for Chef recipes.
  - It contains all the recipes, files, templates, libraries, etc. required to configure a portion of your infrastructure
- Typically they map 1:1 to a piece of software or functionality, such as:
  - MySQL Cookbook
  - IIS Cookbook

# Recipes in Cookbooks (all platforms)

- Denote a recipe using this format:  
**cookbook::recipe**
- Example:
  - `recipe[iis::cleanup]`
  - This points to the “**cleanup**” recipe inside the “**iis**” cookbook
  - The actual file will be called “`cleanup.rb`”
  - `Path='chef-repo/cookbooks/iis/recipes/cleanup.rb'`

# Default Recipes (all platforms)

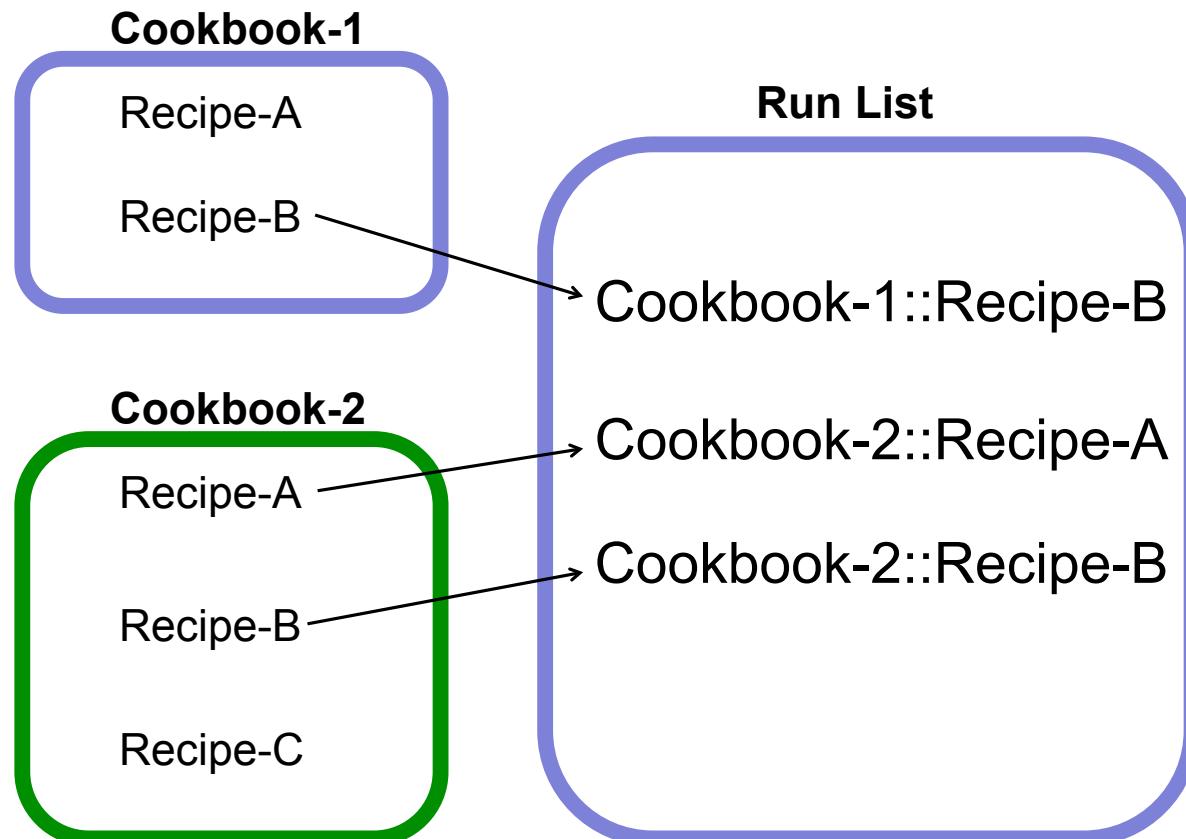
- New cookbooks come with an empty **default recipe**
- Filename is “default.rb”
  - Path: chef-repo/cookbooks/<cookbook\_name>/recipes/default.rb
- Can be referenced as either:  
**recipe[mysql::default]**  
or  
**recipe[mysql]**

# Run List (all platforms)

If a cookbook can contain multiple recipes, how do you specify WHICH recipes to run on the node?

- A **run\_list** is an ORDERED list of which recipes to run, and in which cookbook(s) they can be found
- The order of the recipes is important
- The **run\_list** can contain multiple recipes from multiple cookbooks

# Run List



The following recipes will be **ignored** (they aren't in the run\_list)

- Cookbook-1::Recipe-A
- Cookbook-2::Recipe-C

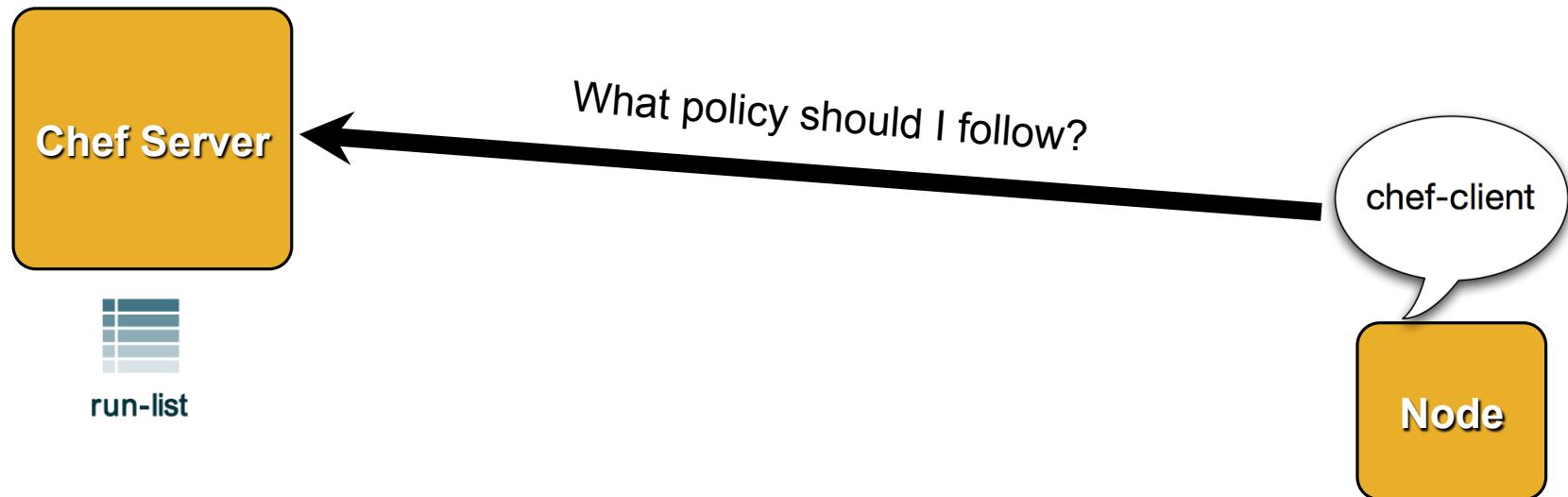
# Duplication in the Run List

If a recipe is repeated **more than once** in a run\_list, all subsequent iterations of the same recipe are ignored

Example: If the same IIS recipe is listed twice in the run\_list, then:

- The first IIS recipe installs IIS
- The second IIS recipe doesn't need to run, since IIS is now installed by the first run of the recipe
- Chef Server makes sure the node only sees **one** of the duplicate IIS recipes

# Run List



# Run List



# Configuration Drift

Configuration Drift happens when:

- Your infrastructure requirements change
- The configuration of a server falls out of policy

**Convergence** and **Chef Search** make it easy to find and correct issues automatically!

# Configuration Drift

Nodes can be configured to automatically check in with Chef Server every X minutes:

- Node can put itself back into compliance
- This is called “Test and Repair”
- See the **chef-client** cookbook to enable this service
  - <https://supermarket.chef.io>
  - You would need to add ‘recipe[chef-client]’ to your run list

# Review Questions

- What is a Node?
- What is a Resource?
- What is a Recipe? How is it different from a Cookbook?
- What is a Run List?
- What is configuration drift, and how does Chef mitigate this issue?

# Chef Documentation

v2.1.0\_DUAL



# Find the Docs

- <http://docs.chef.io>
- All you seek is within!
- Click on ‘Search the Docs’
- Search for “package resource”
  - Which **action** is the **default**?
  - While you might not know what an **action** is yet, just try searching for the requested information

# Writing Webserver Cookbooks

v2.1.0\_DUAL



# Windows/Linux Lesson Objectives

- In this lesson, we will
  - Create new cookbooks
  - Describe how to use the **package (for Linux)**, **powershell\_script (for Windows)**, **service** and **cookbook\_file** resources
  - Write completely functional Apache and IIS webserver recipes
  - **Upload** our cookbooks to the Chef Server
  - Set a **run\_list** for our nodes using ‘knife’
  - Converge our nodes
  - Explain the output of a chef-client run

# The Problem and the Success Criteria

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

# Windows Lesson Objectives

- In this lesson, we will
  - Create new cookbooks
  - Describe how to use the **powershell\_script**(**only for Windows**), **service** and **cookbook\_file** resources
  - Write a completely functional IIS webserver recipe
  - **Upload** our cookbook to the Chef Server
  - Set a **run\_list** for our nodes via knife
  - Converge the node
  - Explain the output of a chef-client run

# Linux Lesson Objectives

- In this lesson, we will
  - Create new cookbooks
  - Describe how to use the **package(only for Linux)**, **service** and **cookbook\_file** resources
  - Write a completely functional Apache webserver recipe
  - **Upload** our cookbook to the Chef Server
  - Set a **run\_list** for our nodes via knife
  - Converge the node
  - Explain the output of a chef-client run

# Problem and the Success Criteria

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

# Writing an Apache cookbook for Linux nodes

v2.1.0\_DUAL



# Linux: Required steps

- 1. Install Apache on the node**
- 2. Start Apache on the node, and configure Apache to start upon reboot**
- 3. Write out the home page, to the location where Apache expects to find it**

# Exercise: Create a new Cookbook

```
ws> cd chef-repo  
ws> 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

Windows workstation

```
ws> dir cookbooks/apache
```

Mac/Linux workstation

```
ws> ls -l cookbooks/apache
```

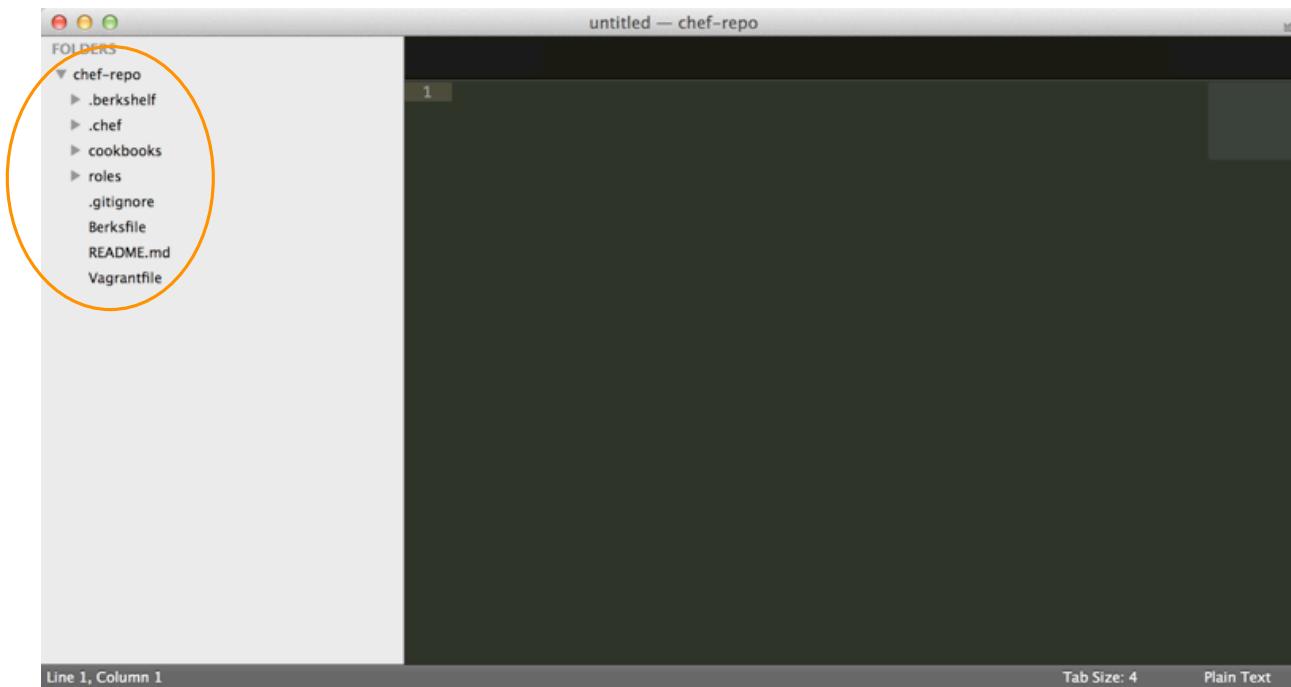
Output

```
-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
-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  3 opscode  opscode   102 Jan 24 21:25 templates
```

## Exercise: Open a project drawer if you're using Sublime Text

- If you're using Sublime, then File>Open the chef-repo directory you created earlier

Access the cookbook files from the left menu



# Chef Resources

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

- Have a **type**

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

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

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

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

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

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

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

- Have a type
- Have a name
- Have parameters
- Take action to put the resource into the desired state
- Can send **notifications** to other resources

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Linux exercise: Edit the default recipe

**OPEN IN EDITOR:** Windows workstation: cookbooks\apache\recipes\default.rb  
-----  
Mac/Linux workstation: cookbooks/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
#  
# Copyright 2014, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

# Linux Exercise: Add a package resource to install Apache

Add the following to: **chef-repo/cookbooks/apache/recipes/default.rb**

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
#  
# Copyright 2014, 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 resource with a type of **package**
- Whose **name** is '**httpd**'
- With the **action** '**install**'

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

## Linux 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!**

## Linux Exercise:

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

Add the following to: **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 !**

# Linux: Full contents of the Apache recipe

```
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
```

# 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"
end
```

## Linux Exercise:

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

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

```
<html>
  <body>
    <h1>Hello, Linux World!</h1>
  </body>
</html>
```

**SAVE FILE !**

# Linux: Full contents of the Apache recipe

apache/recipes/default.rb

```
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
```

apache/files/default/index.html

```
<html>
  <body>
    <h1>Hello, Linux World!</h1>
  </body>
</html>
```

# Linux Exercise: Upload the cookbook

```
ws> knife cookbook upload apache
```

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

# The Run List

- The Run List is the ordered set of recipes and roles that the Chef Client will execute on a node
  - Recipes are specified by “`recipe[ name ]`”
  - Roles are specified by “`role[ name ]`”
  - If you leave off the recipe name, the default recipe is assumed (`default.rb`)

## Linux Exercise: Add the Apache recipe to the Linux test node's run list

```
ws> knife node run_list add <nodename> "recipe[apache]"
```

```
linuxnode:  
  run_list: recipe[apache]
```

# Linux Exercise: Run chef-client on your Linux test node

```
ws> knife node run_list add <nodename> "recipe[apache]"
```

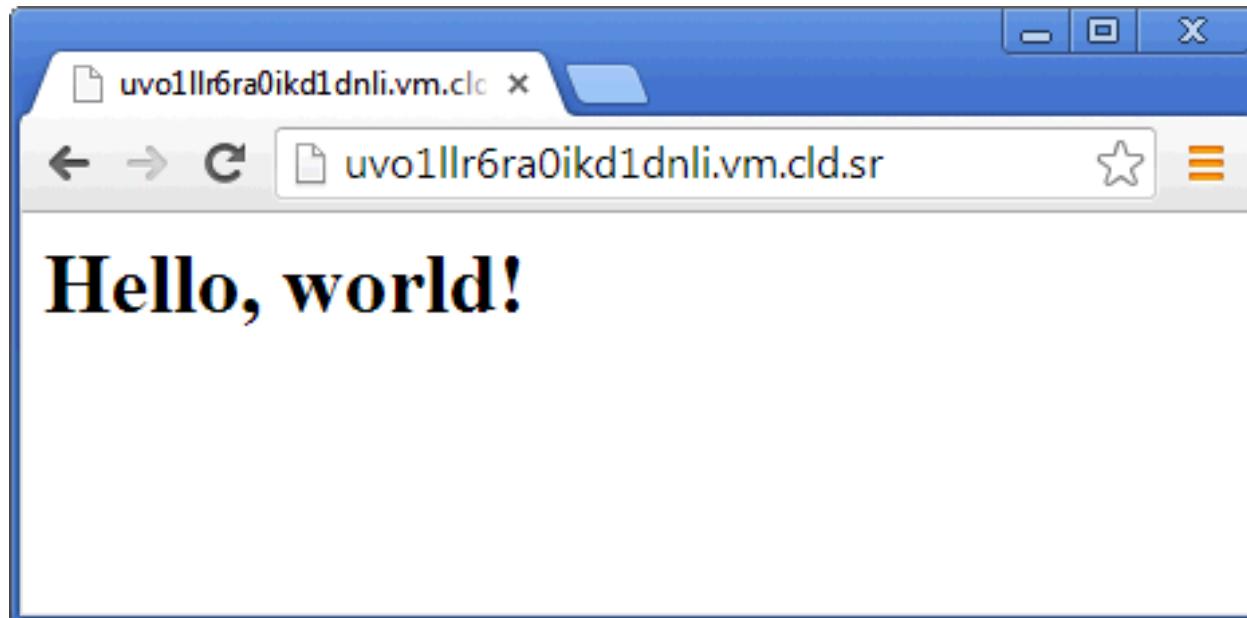
**Use the nodename you defined, Do NOT use the hostname**

```
rn>$ chef-client
```

```
Starting Chef Client, version 11.10.4
[2014-02-24T05:37:54-08:00] INFO: *** Chef 11.10.4 ***
[2014-02-24T05:37:54-08:00] INFO: Chef-client pid: 260
[2014-02-24T05:38:09-08:00] INFO: Run List is [recipe[apache]]
[2014-02-24T05:38:09-08:00] INFO: Run List expands to [apache]
[2014-02-24T05:38:09-08:00] INFO: Starting Chef Run for linuxnode
[2014-02-24T05:38:09-08:00] INFO: Running start handlers
[2014-02-24T05:38:09-08:00] INFO: Start handlers complete.
```

## Exercise: Verify that the home page works

- Open a web browser
- Type in the URL for your test node



# Congratulate yourself!

- You have just written your first Chef cookbook!
- (clap!)

## Linux: 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  
linuxnode[2014-01-06T07:06:01-05:00] INFO: Running start  
handlers[2014-01-06T07:06:01-05:00] INFO: Start handlers complete.
```

- The run list is shown
- The expanded Run List is the complete list, after nested roles are expanded

# Linux: 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

```
[2014-02-24T06:25:34-08:00] INFO: Chef Run complete in 17.432572377 seconds
[2014-02-24T06:25:34-08:00] INFO: Running report handlers
[2014-02-24T06:25:34-08:00] INFO: Report handlers complete
Chef Client finished, 4/4 resources updated
```

- Time to complete the Chef run is displayed
- Report and exception handlers are now run

# Idempotence

- Actions on resources in Chef are designed to be idempotent
- In practical terms, this means they only change the state of the system if they have to
- If a resource in Chef is properly configured, we move on to the next resource

# Linux Exercise: Re-run the Chef Client Run

```
rn> chef-client
```

```
* service[w3svc] action enable
[2014-02-24T06:25:33-08:00] INFO: Processing service[w3svc] action enable (iis_demo::default
line 14)
  (up to date)
* service[w3svc] action start
[2014-02-24T06:25:33-08:00] INFO: Processing service[w3svc] action start (iis_demo::default
line 14)
  (up to date)
```

- Service is idempotent.
- No changes have been made

# Best Practice: Omit the default action

So this:

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

Becomes this:

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

Becomes this:

```
package "httpd"
```

# What's with the 'default' subdirectory?

- Chef allows you to select the most appropriate file (or template) within a cookbook depending on the node's platform
  - Host node name (foo.bar.com)
  - platform-version (centos-6.3)
  - platform-major (centos-6)
  - Platform (centos)
  - default
- **99% of the time, you will just use default**

# Questions

- How do you create a new cookbook?
- How do you upload a cookbook to the Chef Server?
- How do add a recipe to the node's run\_list?

# Writing an IIS cookbook for Windows nodes

v2.1.0\_DUAL



# Required steps

- 1. Install IIS on the node**
- 2. Start IIS on the node, and configure IIS to start upon reboot**
- 3. Write out the home page, to the location where IIS expects to find it**

# Windows Exercise: Create a new Cookbook

```
ws> cd chef-repo  
ws> knife cookbook create iis_demo
```

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

# Exercise: Explore the cookbook

Windows workstation

```
ws> dir cookbooks/iis_demo
```

Mac/Linux workstation

```
ws> ls -l cookbooks/iis_demo
```

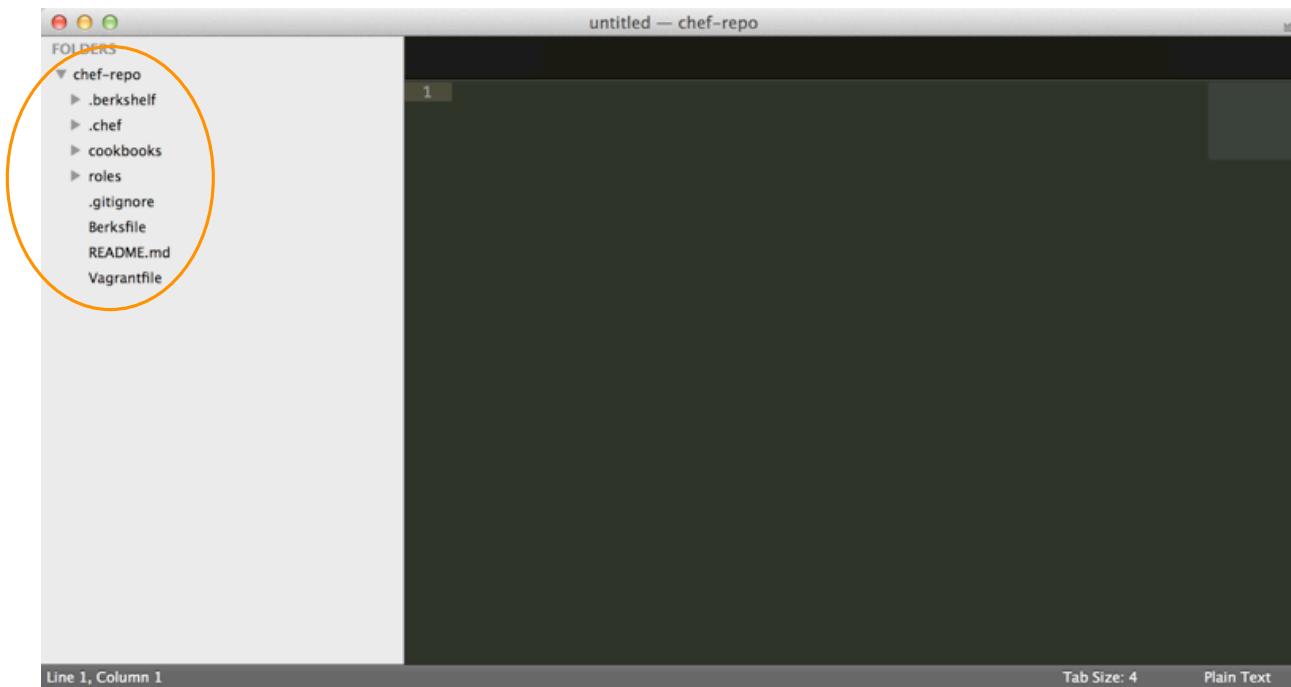
Output

```
-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
-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  3 opscode  opscode   102 Jan 24 21:25 templates
```

## Exercise: Open a project drawer if you're using Sublime Text

- If you're using Sublime, then File>Open the chef-repo directory you created earlier

Access the cookbook files from the left menu



# Chef Resources

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

- Have a **type**

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

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

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

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

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

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

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

- Have a type
- Have a name
- Have parameters
- Take action to put the resource into the desired state
- Can send **notifications** to other resources

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```

# Chef Resources

- Have a type
- Have a name
- Have parameters
- Take action to put the resource into the desired state
- Can send **notifications** to other resources

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

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  owner "root"
  group "root"
  mode "0644"
  notifies :restart, "service[haproxy]"
end

service "haproxy" do
  supports :restart => :true
  action [:enable, :start]
end
```



# Windows: Edit the default recipe

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
#  
# Cookbook Name:: iis_demo  
# Recipe:: default  
#  
# Copyright 2014, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

# Windows Exercise: Add a powershell resource to install IIS

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
#  
# Cookbook Name:: iis_demo  
# Recipe:: default  
#  
# Copyright 2014, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

```
powershell_script "Install IIS" do  
  code "add-windowsfeature Web-Server"  
  action :run  
end
```

**SAVE FILE!**

# So the resource we just wrote...

```
powershell_script "Install IIS" do
  code "add-windowsfeature Web-Server"
  action :run
end
```

# So the resource we just wrote...

- Is a **powershell\_script** resource

```
powershell_script "Install IIS" do
  code "add-windowsfeature Web-Server"
  action :run
end
```

# So the resource we just wrote...

- Is a `powershell_script` resource
- Whose **name** is “`Install IIS`”

```
powershell_script "Install IIS" do
  code "add-windowsfeature Web-Server"
  action :run
end
```

# So the resource we just wrote...

- Is a `powershell_script` resource
- Whose name is `Install IIS`
- With the **parameter** “`code`”

```
powershell_script "Install IIS" do
  code "add-windowsfeature Web-Server"
  action :run
end
```

# So the resource we just wrote...

- Is a `powershell_script` resource
- Whose name is `Install IIS`
- With the parameter “code”
- With an `action` “`run`”

```
powershell_script "Install IIS" do
  code "add-windowsfeature Web-Server"
  action :run
end
```

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

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
...
# All rights reserved - Do Not Redistribute
#
powershell_script "Install IIS" do
  action :run
  code "add-windowsfeature Web-Server"
end

service "w3svc" do
  action [ :enable, :start ]
end
```

**SAVE FILE!**

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

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
...
service "w3svc" do
  action [ :enable, :start ]
end

cookbook_file 'c:\inetpub\wwwroot\Default.htm' do
  source "Default.htm"
  rights :read, "Everyone"
end
```

**SAVE FILE !**

# Windows: Full contents of the IIS\_demo recipe

```
powershell_script "Install IIS" do
  action :run
  code "add-windowsfeature Web-Server"
end

service "w3svc" do
  action [ :enable, :start ]
end

cookbook_file 'c:\inetpub\wwwroot\Default.htm' do
  source "Default.htm"
  rights :read, "Everyone"
end
```

## Notice we didn't say how to restart the service

- 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 determines the platform the node is running to determine the correct provider for a resource

# 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"
end
```

## Windows Exercise: Add Default.htm to your cookbook's files/default directory

**OPEN IN EDITOR:** cookbooks\iis\_demo\files\default\Default.htm

```
<html>
  <body>
    <h1>Hello, Windows World!</h1>
  </body>
</html>
```

**SAVE FILE !**

# Windows Exercise: Upload the cookbook

```
ws> knife cookbook upload iis_demo
```

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

# The Run List

- The Run List is the ordered set of recipes and roles that the Chef Client will execute on a node
  - Recipes are specified by “`recipe[ name ]`”
  - Roles are specified by “`role[ name ]`”
  - If you leave off the recipe name, the default recipe is assumed (`default.rb`)

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

```
ws> knife node run_list add <nodename> "recipe[iis_demo]"
```

```
windowsnode:  
  run_list: recipe[iis_demo]
```

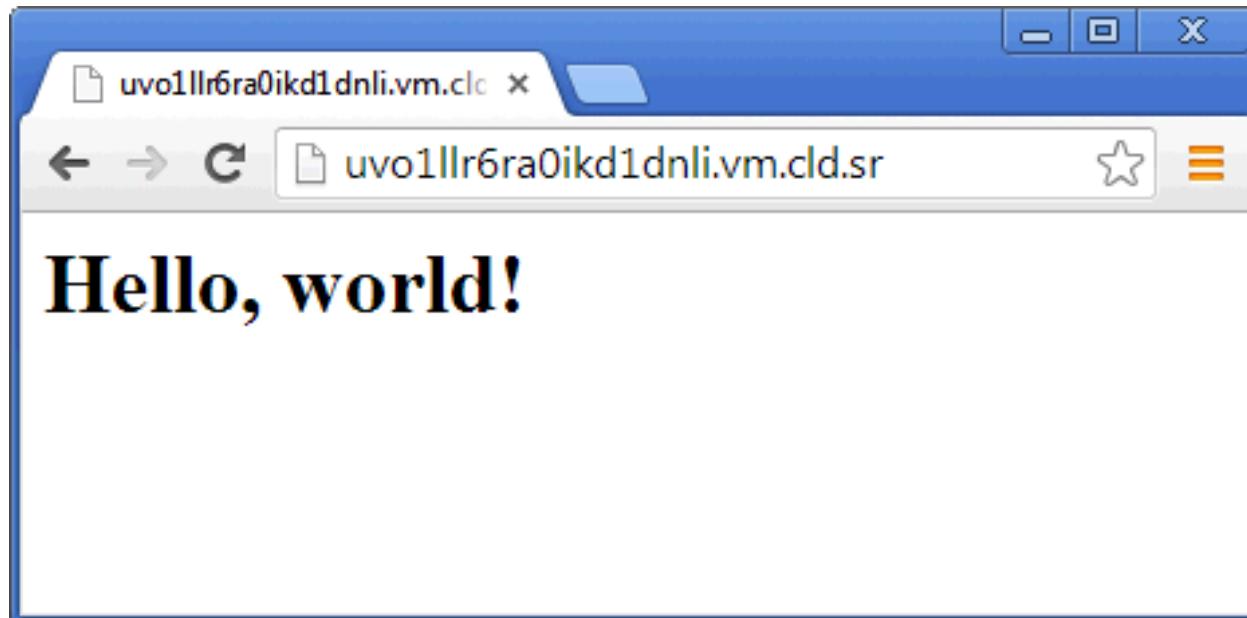
# Windows Exercise: Run the chef-client on your test Windows node

```
rn> chef-client
```

```
Starting Chef Client, version 11.10.4
[2014-02-24T05:37:54-08:00] INFO: *** Chef 11.10.4 ***
[2014-02-24T05:37:54-08:00] INFO: Chef-client pid: 260
[2014-02-24T05:38:09-08:00] INFO: Run List is [recipe[iis_demo]]
[2014-02-24T05:38:09-08:00] INFO: Run List expands to [iis_demo]
[2014-02-24T05:38:09-08:00] INFO: Starting Chef Run for windowsnode
[2014-02-24T05:38:09-08:00] INFO: Running start handlers
[2014-02-24T05:38:09-08:00] INFO: Start handlers complete.
resolving cookbooks for run list: ["iis_demo"]
[2014-02-24T05:38:10-08:00] INFO: Loading cookbooks [iis_demo]
Synchronizing Cookbooks:
[2014-02-24T05:38:10-08:00] INFO: Storing updated cookbooks/iis_demo/recipes/
default.rb in the cache.
...
```

## Exercise: Verify that the home page works

- Open a web browser
- Type in the URL for your test node



# Congratulate yourself!

- You have just written your first Chef cookbook!
- (clap!)

# Windows: Reading the output of a chef-client run

```
* cookbook_file[c:\inetpub\wwwroot\Default.htm] action create
[2014-02-24T06:25:33-08:00] INFO: Processing cookbook_file[c:\inetpub\wwwroot\Default.htm] action create (iis_demo::default line 18)
[2014-02-24T06:25:33-08:00] INFO: cookbook_file[c:\inetpub\wwwroot\Default.htm] created file c:\inetpub\wwwroot\Default.htm

  - create new file c:\inetpub\wwwroot\Default.htm[2014-02-24T06:25:33-08:00]
INFO: cookbook_file[c:\inetpub\wwwroot\Default.htm] updated file contents c:
\inetpub\wwwroot\Default.htm

  - update content in file c:\inetpub\wwwroot\Default.htm from none to 231d53
    --- c:\inetpub\wwwroot\Default.htm      2014-02-24 06:25:33.000000000 -0800
    +++ C:/Users/chef/AppData/Local/Temp/4/Default.htm20140224-1896-13q72ev
2014-02-24 06:25:33.000000000 -0800
     @@ -1 +1,7 @@
     +<html>
     +  <body>
     +    <h1>Hello, world!</h1>
     +  </body>
     +</html>
    +[2014-02-24T06:25:33-08:00] INFO: cookbook_file[c:\inetpub\wwwroot\Default.htm] permissions changed to [Everyone	flags:0/mask:80000000]

  - change dacl
```

- Check for an Default.htm file
- If there is already one in place, backup the file
- A diff of the written file is shown with the modified lines called out
- Set permissions on the file

# Windows: Reading the output of a chef-client run

```
Converging 3 resources
Recipe: iis_demo::default
  * powershell_script[Install IIS] action run[2014-02-24T06:08:29-08:00] INFO: Processing
powershell_script[Install IIS]
  action run (iis_demo::default line 9)

Success Restart Needed Exit Code      Feature Result
----- ----- ----- -----
True    No          Success        {Common HTTP Features, Default Document, D...
WARNING: Windows automatic updating is not enabled. To ensure that your newly-installed role
or feature is automatically updated, turn on Windows Update.
[2014-02-24T06:09:16-08:00] INFO: powershell_script[Install IIS] ran successfully
```

- `powershell_script` is not idempotent
- `Add-WindowsFeature` is!

# Reading the output of a chef-client run

```
[2014-02-24T06:25:34-08:00] INFO: Chef Run complete in 17.432572377 seconds
[2014-02-24T06:25:34-08:00] INFO: Running report handlers
[2014-02-24T06:25:34-08:00] INFO: Report handlers complete
Chef Client finished, 2 resources updated
```

- Time to complete the Chef run is displayed
- Report and exception handlers are now run

# Idempotence

- Actions on resources in Chef are designed to be idempotent
- In practical terms, this means they only change the state of the system if they have to
- If a resource in Chef is properly configured, we move on to the next resource

# Windows Exercise: Re-run the Chef Client

```
rn> chef-client
```

```
Converging 3 resources
Recipe: iis_demo::default
 * powershell_script[Install IIS] action run[2014-02-24T06:42:15-08:00] INFO: Processing powershell_script[Install IIS]
   action run (iis_demo::default line 9)

Success Restart Needed Exit Code      Feature Result
----- ----- ----- -----
True      No          NoChangeNeeded {}

[2014-02-24T06:42:16-08:00] INFO: powershell_script[Install IIS] ran successfully

 - execute "powershell.exe" -NoLogo -NonInteractive -NoProfile -ExecutionPolicy RemoteSigned -InputFormat None -File
"C:/Users/chef/AppData/Local/Temp/4/chef-script20140224-4608-1fwjxog.ps1"
 * service[w3svc] action enable[2014-02-24T06:42:16-08:00] INFO: Processing service[w3svc] action enable (iis_demo::default
line 14)
(up to date)
 * service[w3svc] action start[2014-02-24T06:42:16-08:00] INFO: Processing service[w3svc] action start (iis_demo::default
line 14)
(up to date)
 * cookbook_file[c:\inetpub\wwwroot\Default.htm] action create[2014-02-24T06:42:16-08:00] INFO: Processing cookbook_file[c:
\inetpub\wwwroot\Default.htm] action create (iis_demo::default line 18)
(up to date)
[2014-02-24T06:42:17-08:00] INFO: Chef Run complete in 3.416521 seconds
...
```

# Best Practice: Omit the default action

So this:

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

Becomes this:

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

Becomes this:

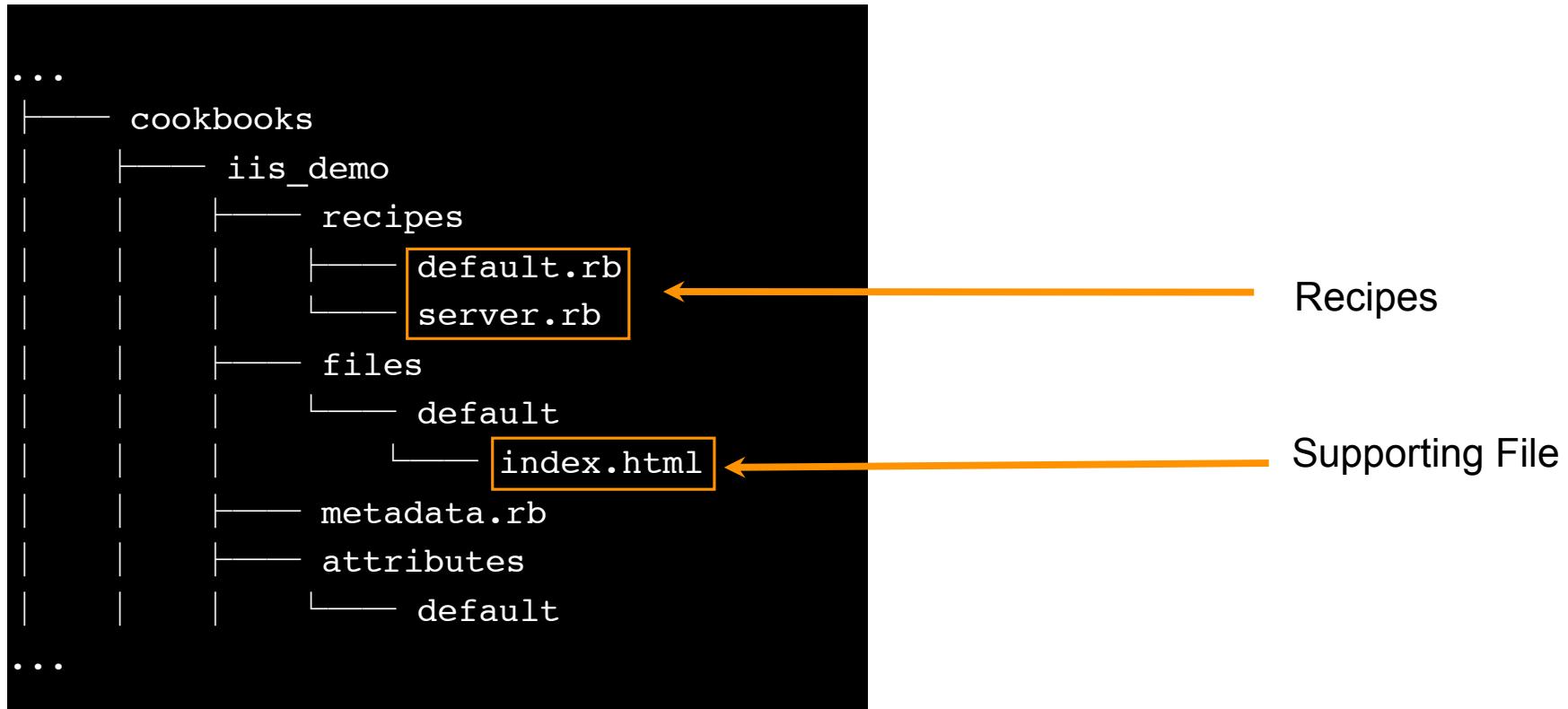
```
package "httpd"
```

# What's with the 'default' subdirectory?

- Chef allows you to select the most appropriate file (or template) within a cookbook depending on the node's platform
  - Host node name (foo.bar.com)
  - platform-version (centos-6.3)
  - platform-major (centos-6)
  - platform
  - default
- **99% of the time, you will just use default**

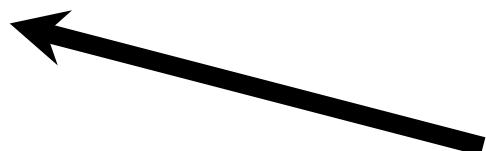
# Review:

## Cookbooks contain recipes and supporting files



## Review:

# Cookbooks are installed as artifacts on Chef Server

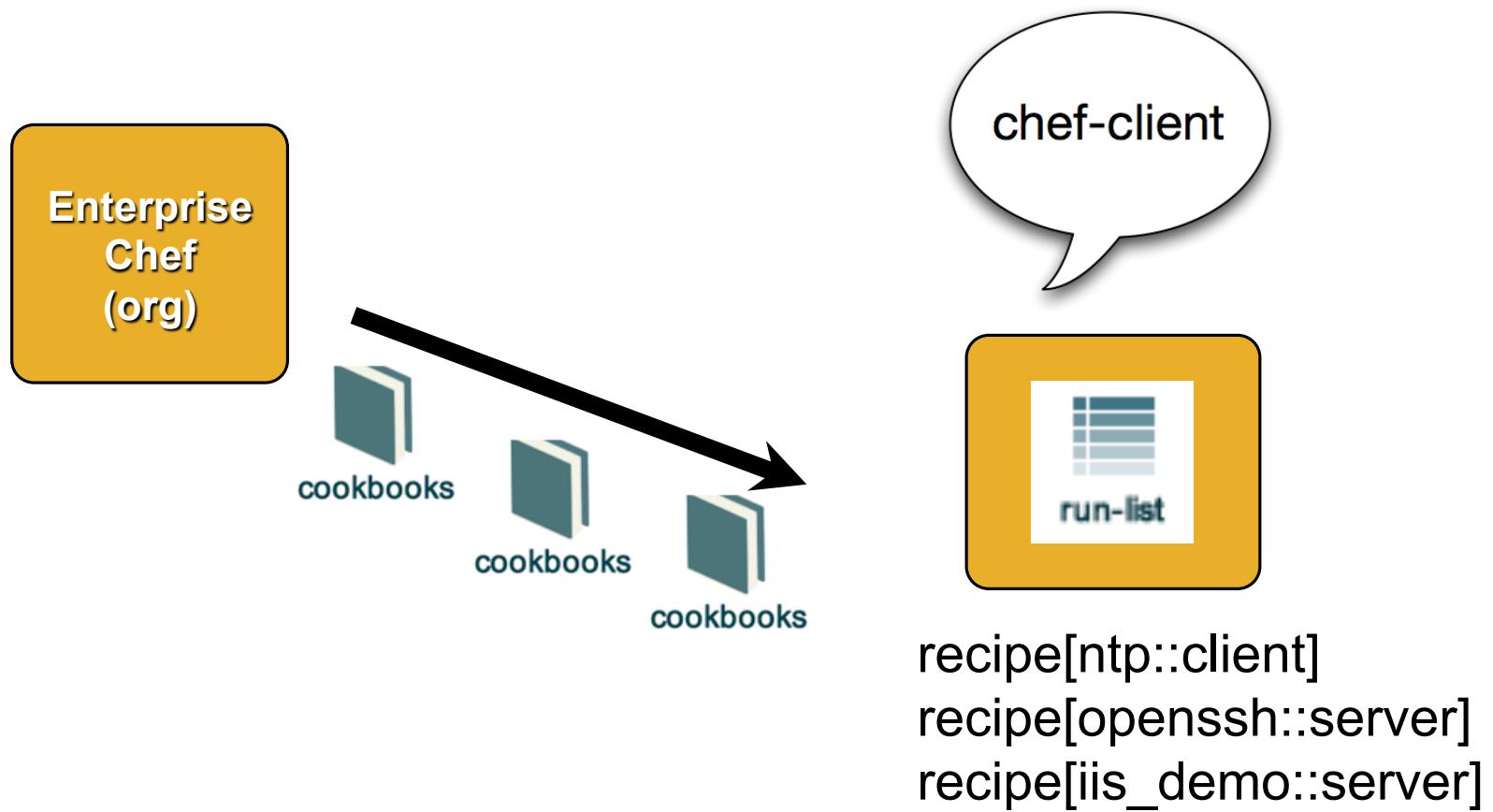


iis\_demo v0.1.0

ntp v0.2.0

openssh v2.4.7

# Review: Nodes have run\_lists made of recipes



# Questions

- How do you create a new cookbook?
- How do you upload a cookbook to the Chef Server?
- How do you add a recipe to the run list using knife?
  - How do you remove a recipe from the run list using knife?

# The Anatomy of a Chef run

v2.1.0\_DUAL

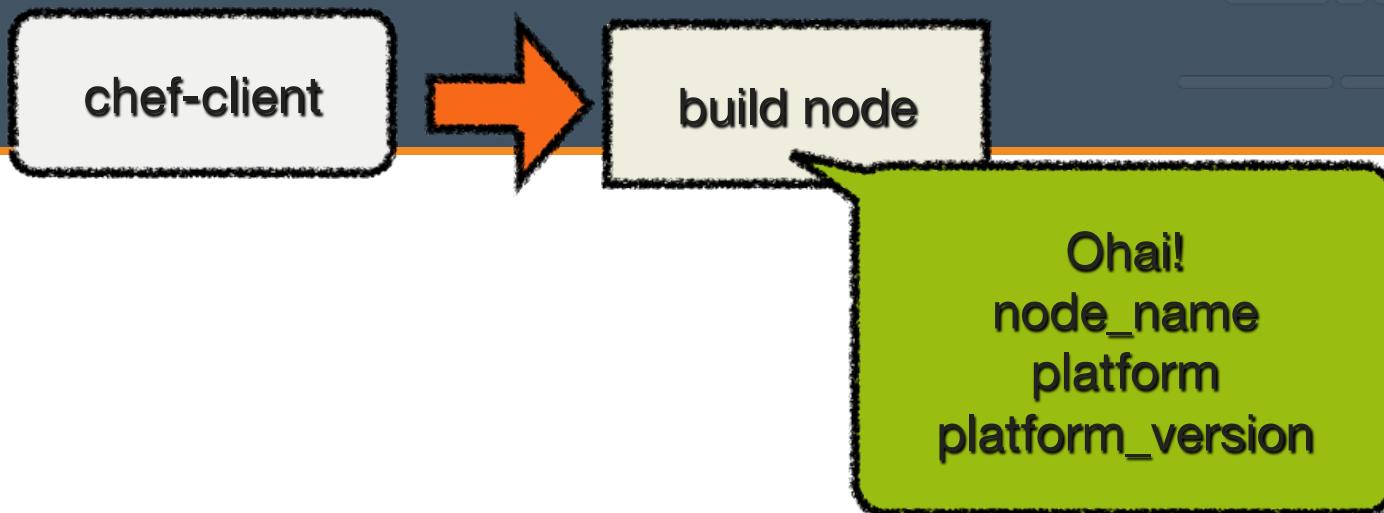


# Lesson Objectives

- After completing the lesson, you will be able to
  - List all the steps taken by **chef-client** during a run

**chef-client**

**Note: this happens AFTER the bootstrap is complete**

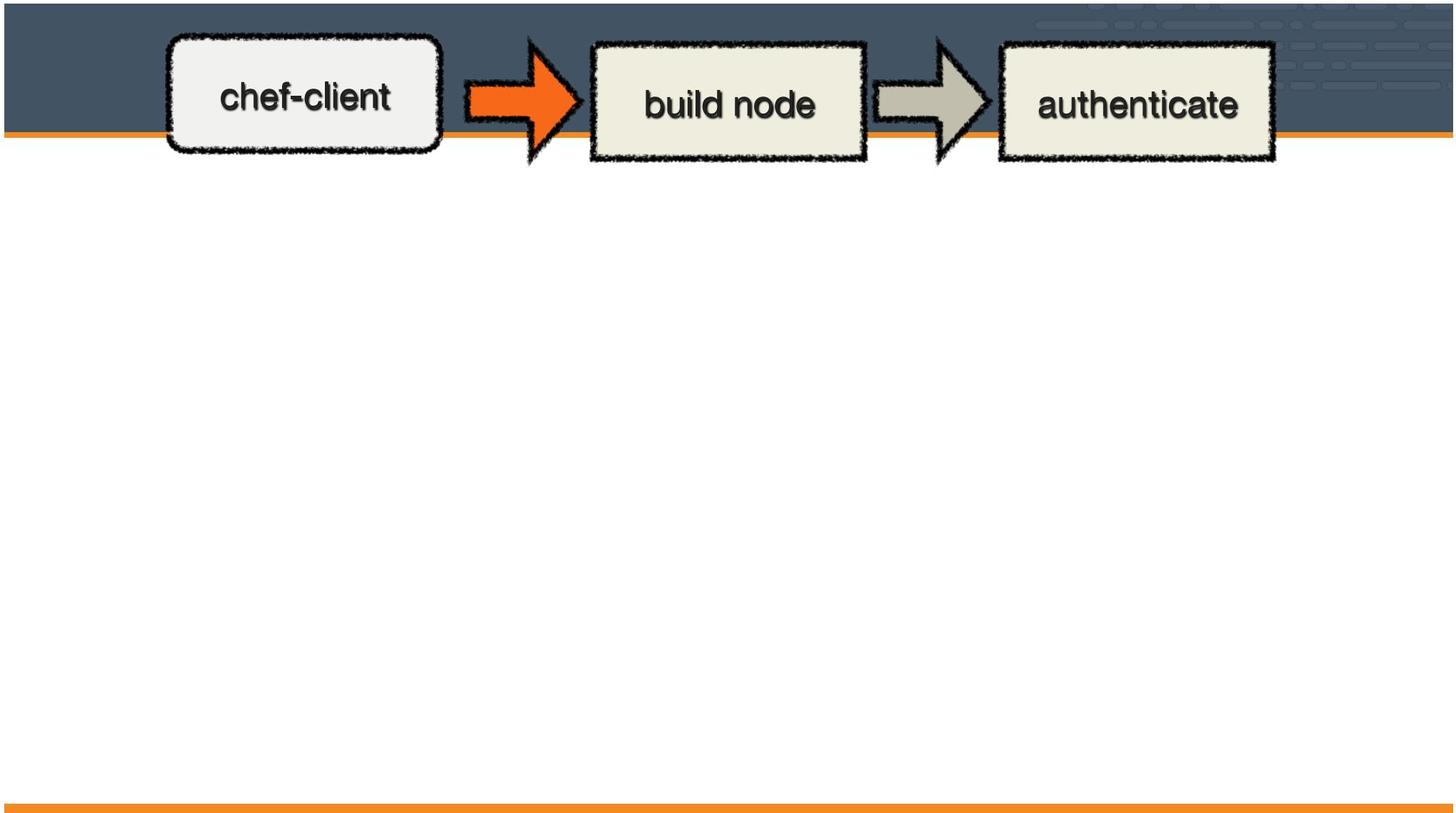


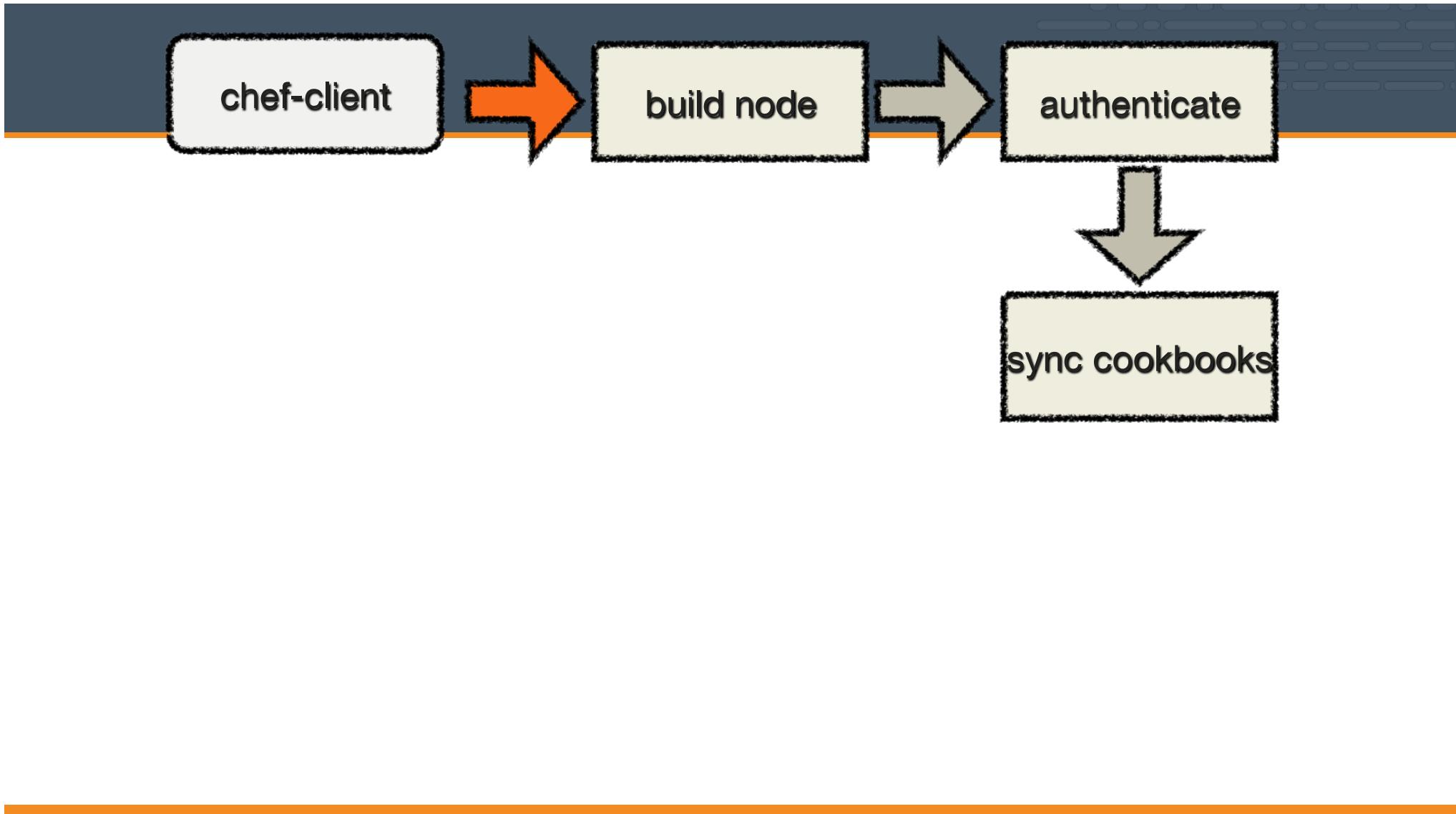
# Example of Ohai output from a Node

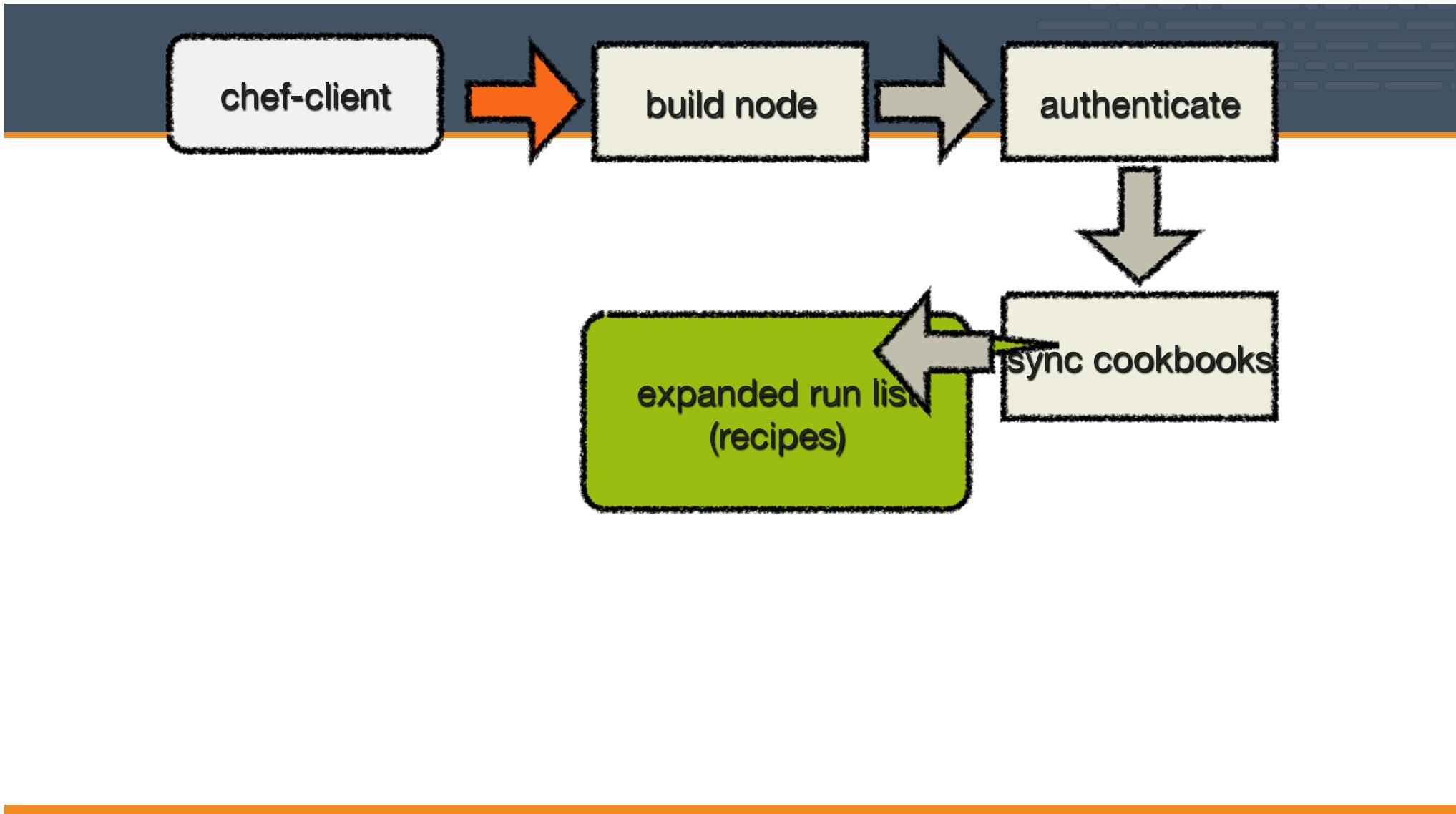
```
"languages": {  
  "ruby": {  
    "platform": "i386-  
mingw32",  
    "version": "1.9.3",  
    "target": "i386-pc-  
mingw32",  
    "target_cpu": "i386",  
    "target_vendor": "pc",  
    "target_os": "mingw32",  
    "host": "i686-pc-  
mingw32",  
    "host_cpu": "i686",  
    "host_os": "mingw32",  
    "host_vendor": "pc",  
  },  
  "perl": {  
    "version": "5.8.8",  
    "archname": "msys-64int"  
  },  
  ...  
}
```

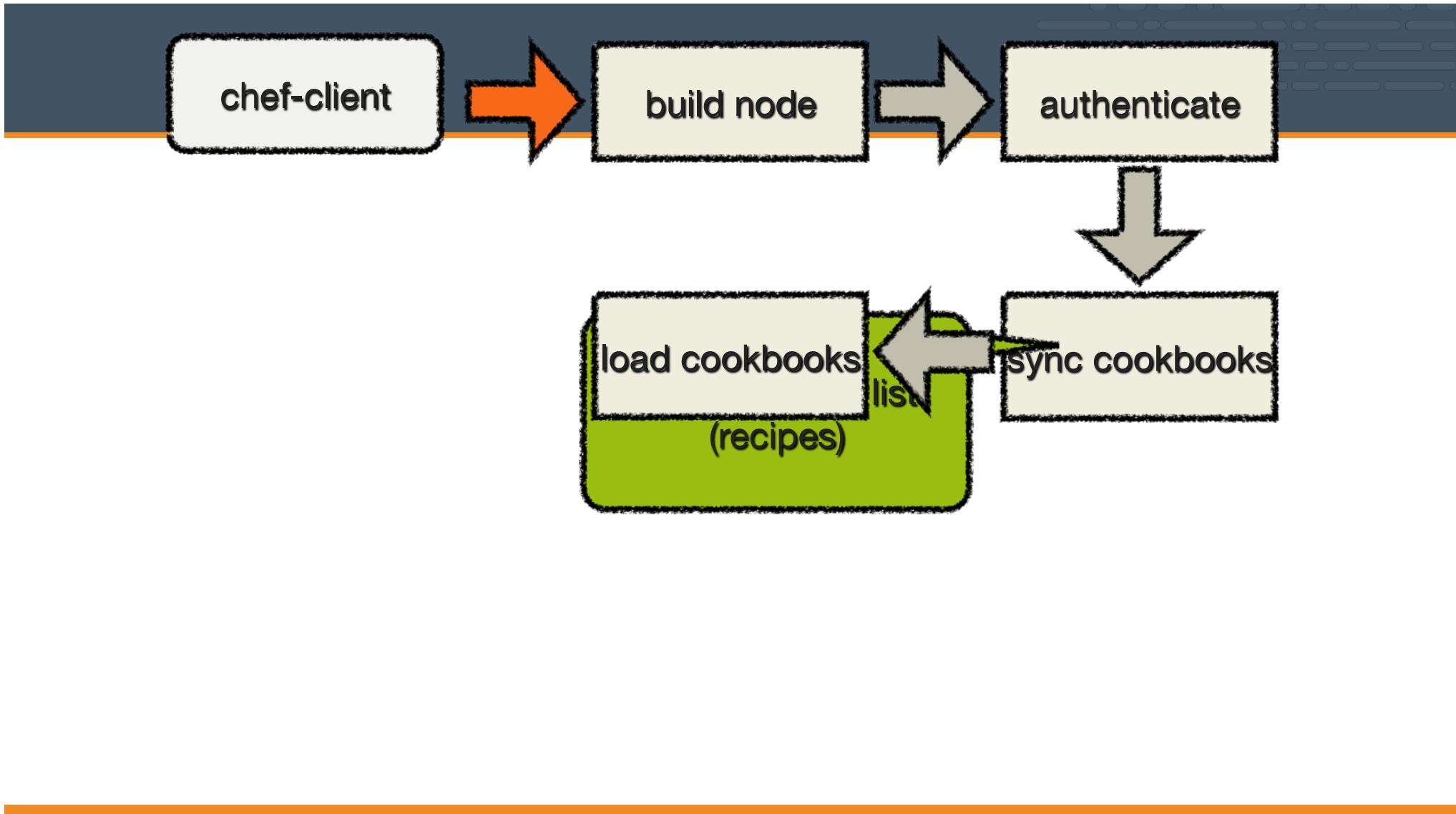
```
"kernel": {  
  "os_info": {  
    "boot_device": "\Device\  
\HddiskVolume1",  
    "build_number": "9200",  
    "build_type": "Multiprocessor  
Free",  
    "caption": "Microsoft Windows  
Server 2012 Standard",  
    "code_set": "1252",  
    "country_code": "1",  
    "creation_class_name":  
      "Win32_OperatingSystem",  
    "cs_creation_class_name":  
      "Win32_ComputerSystem",  
    "csd_version": null,  
    "cs_name": "C1263834251"},  
  "machine": "x86_64",  
  "pnp_drivers": {  
    "SWD\\PRINTENUM\\{B86F2C50-  
C174-459A-AE9E-0097FCE113EE}":  
    ...  
  }  
}
```

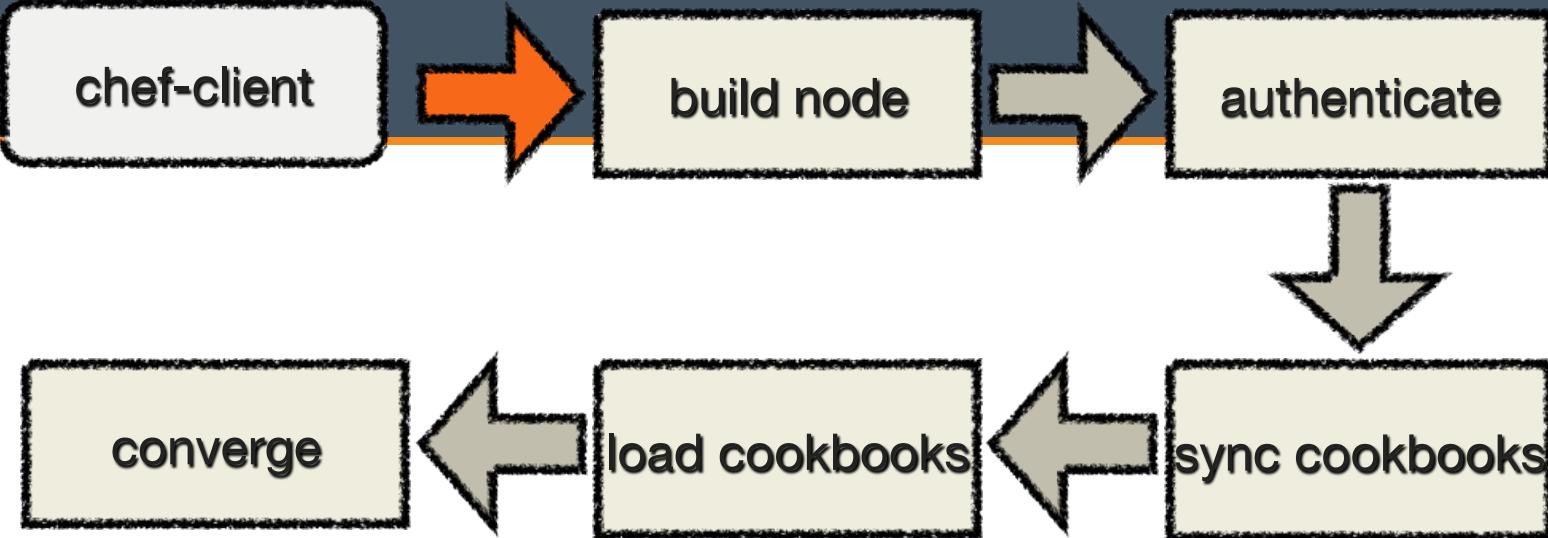
```
"network": {  
  "interfaces": {  
    "lo": {  
      "mtu": "16436",  
      "flags": [  
        "LOOPBACK", "UP", "LOWER_UP"  
      ],  
      "encapsulation": "Loopback",  
      "addresses": {  
        "127.0.0.1": {  
          "family": "inet",  
          "netmask": "255.0.0.0",  
          "scope": "Node"  
        },  
        "::1": {  
          "family": "inet6",  
          "scope": "Node"  
        }  
      },  
      "...  
    },  
    "eth0": {  
      "type": "eth",  
      "number": "0",  
      "mac": "00:0C:29:4D:4D:00",  
      "ip4_addresses": [  
        {  
          "ip": "192.168.1.10",  
          "netmask": "255.255.255.0",  
          "broadcast": "192.168.1.255",  
          "scope": "Global",  
          "link_layer": "00:0C:29:4D:4D:00",  
          "physical": true  
        }  
      ],  
      "ip6_addresses": [  
        {  
          "ip": "fe80::20c:29ff:fe4d:4d00",  
          "netmask": "ffff:ffff:ffff:ffff::ffff:ffff:ffff:ffff",  
          "scope": "LinkLocal",  
          "link_layer": "00:0C:29:4D:4D:00",  
          "physical": true  
        }  
      ]  
    }  
  }  
}
```

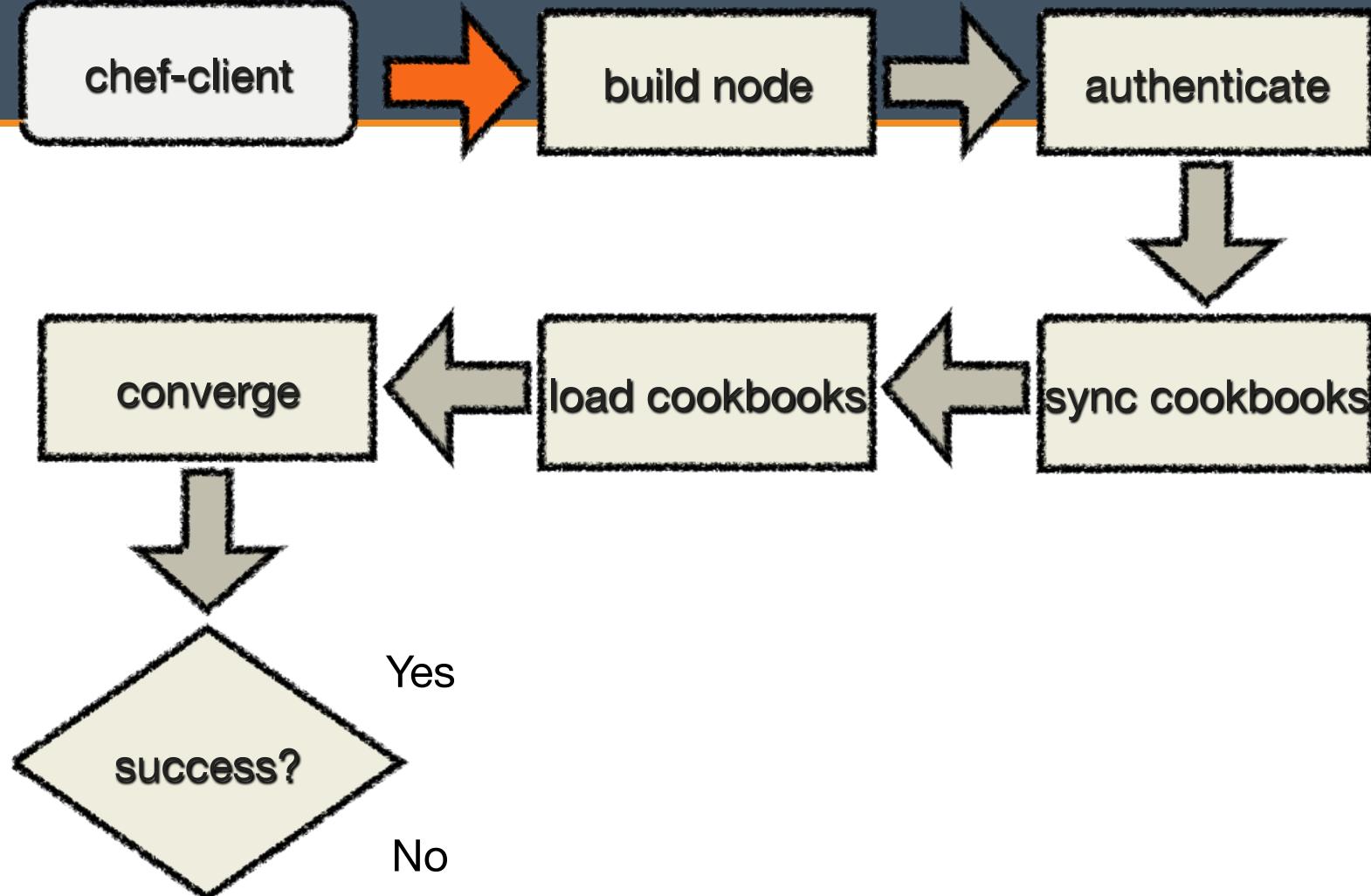


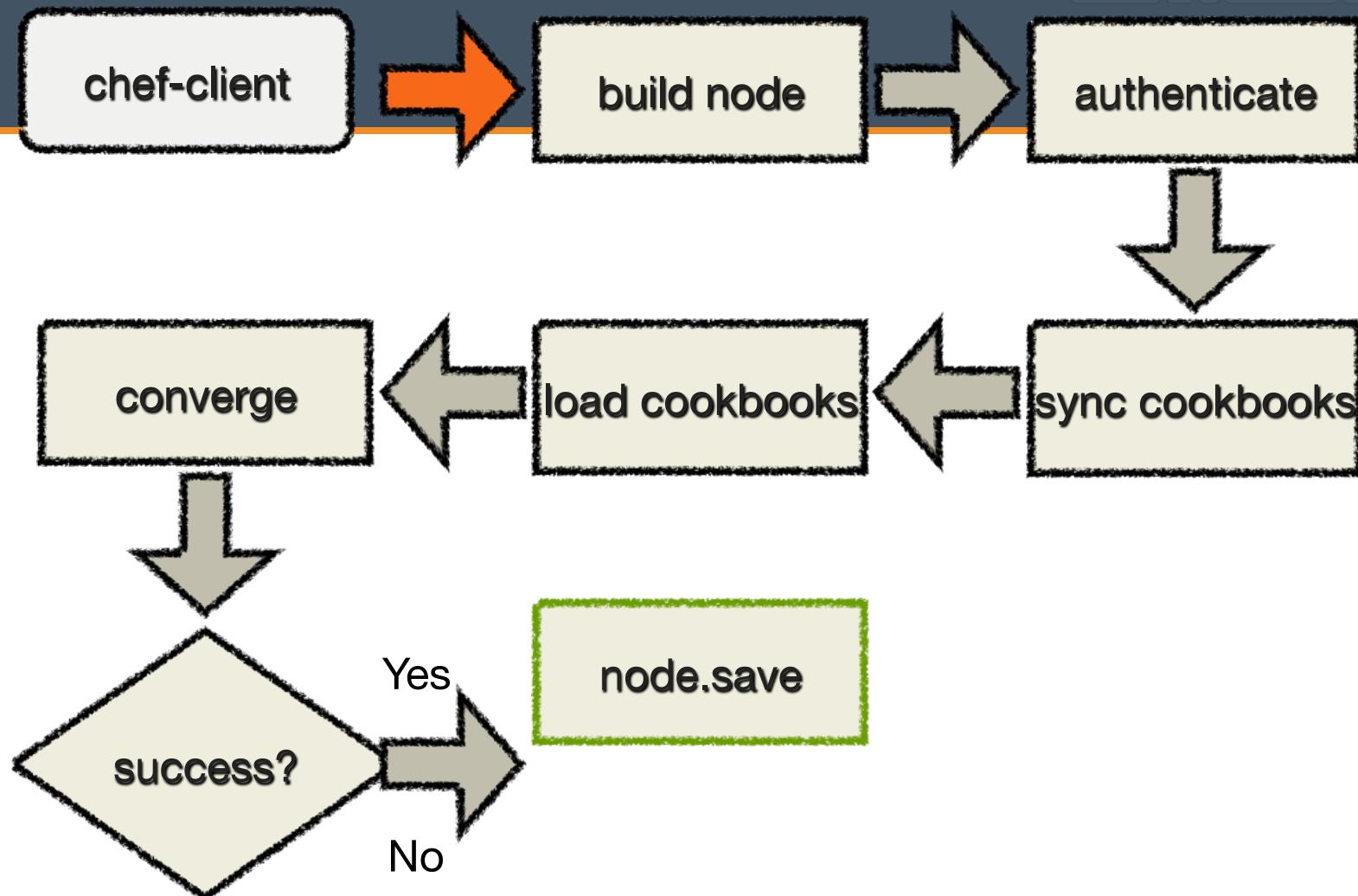


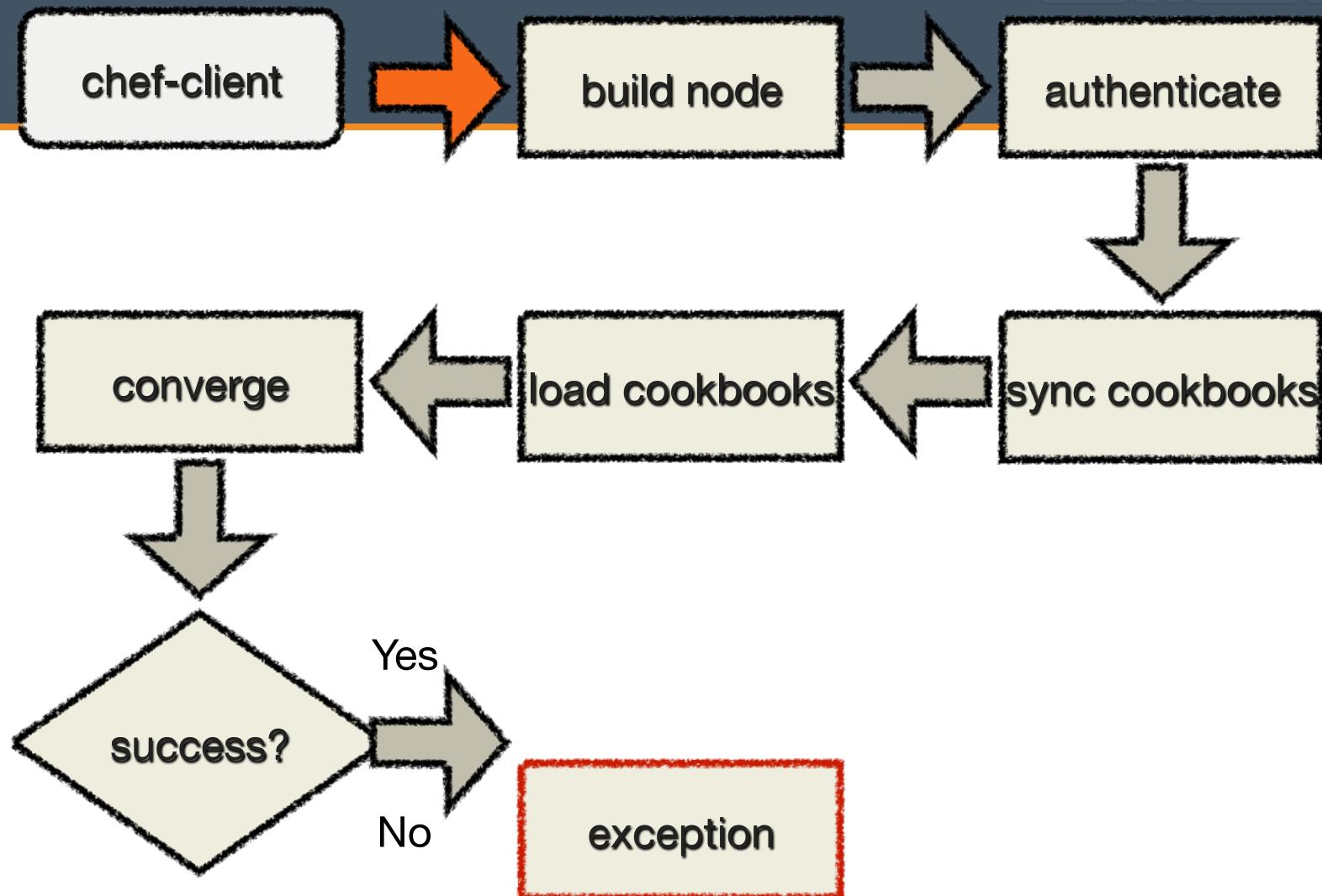


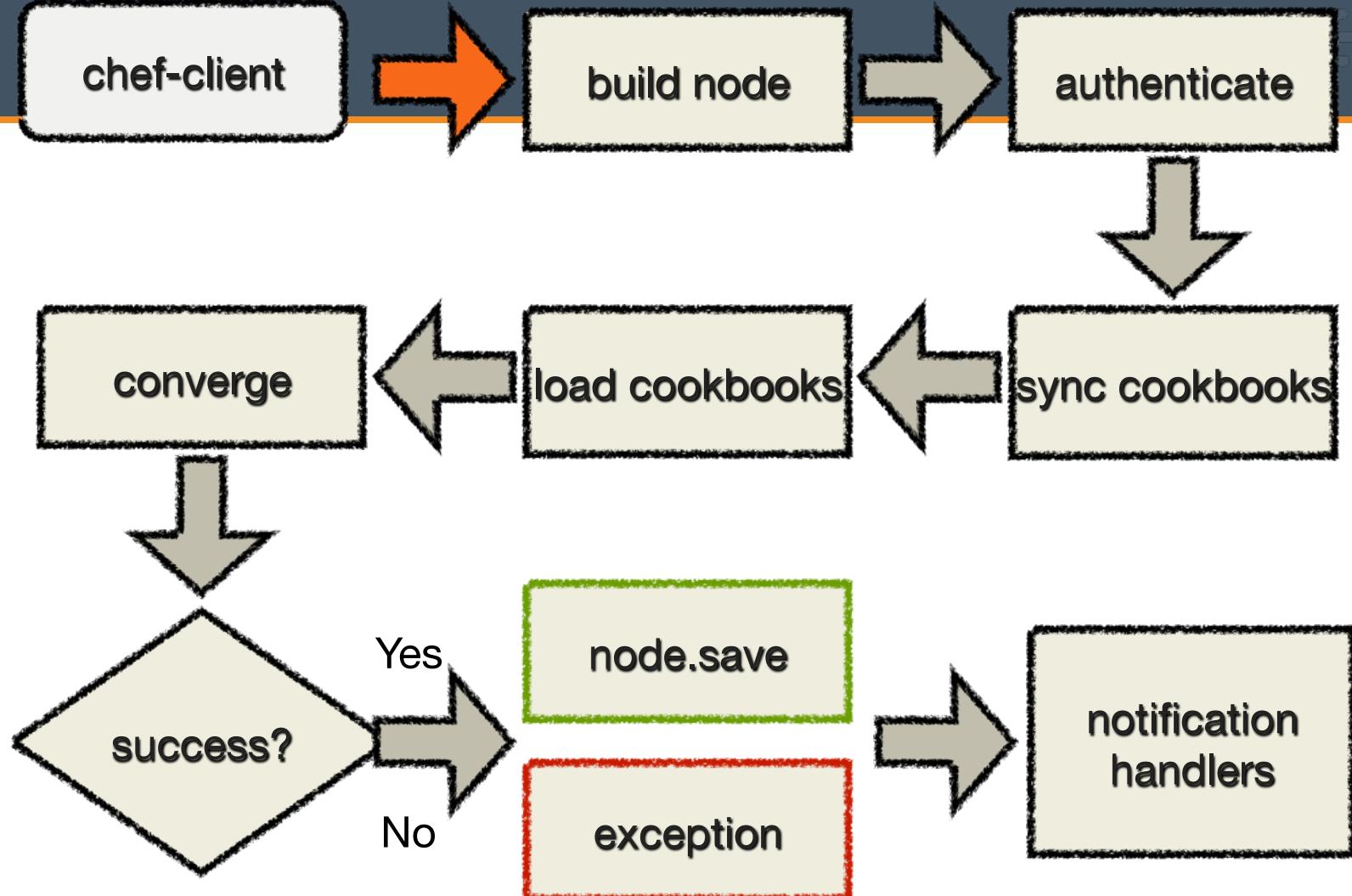












# Secure Communications between the Node and Chef Server

v2.1.0\_DUAL

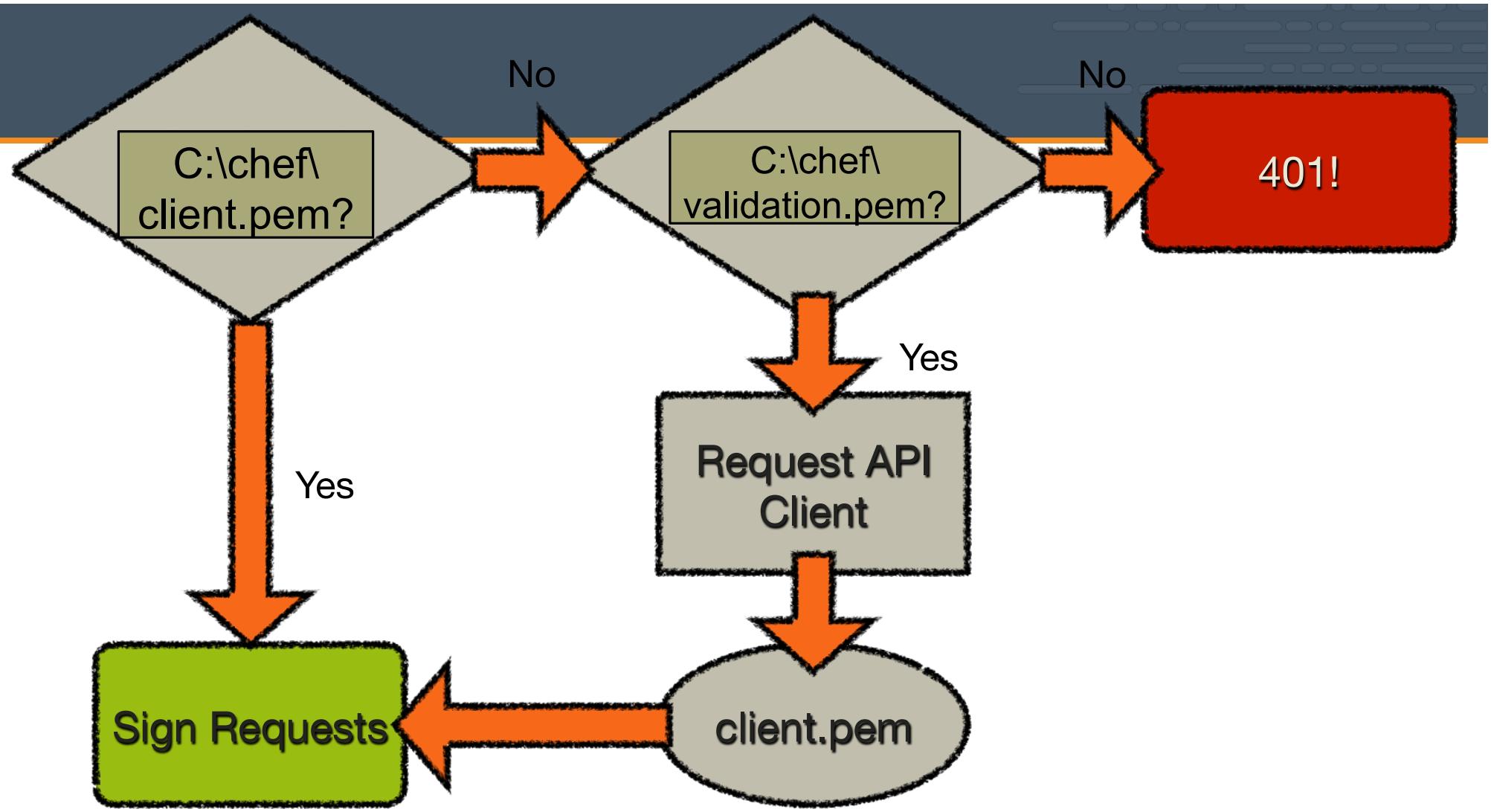


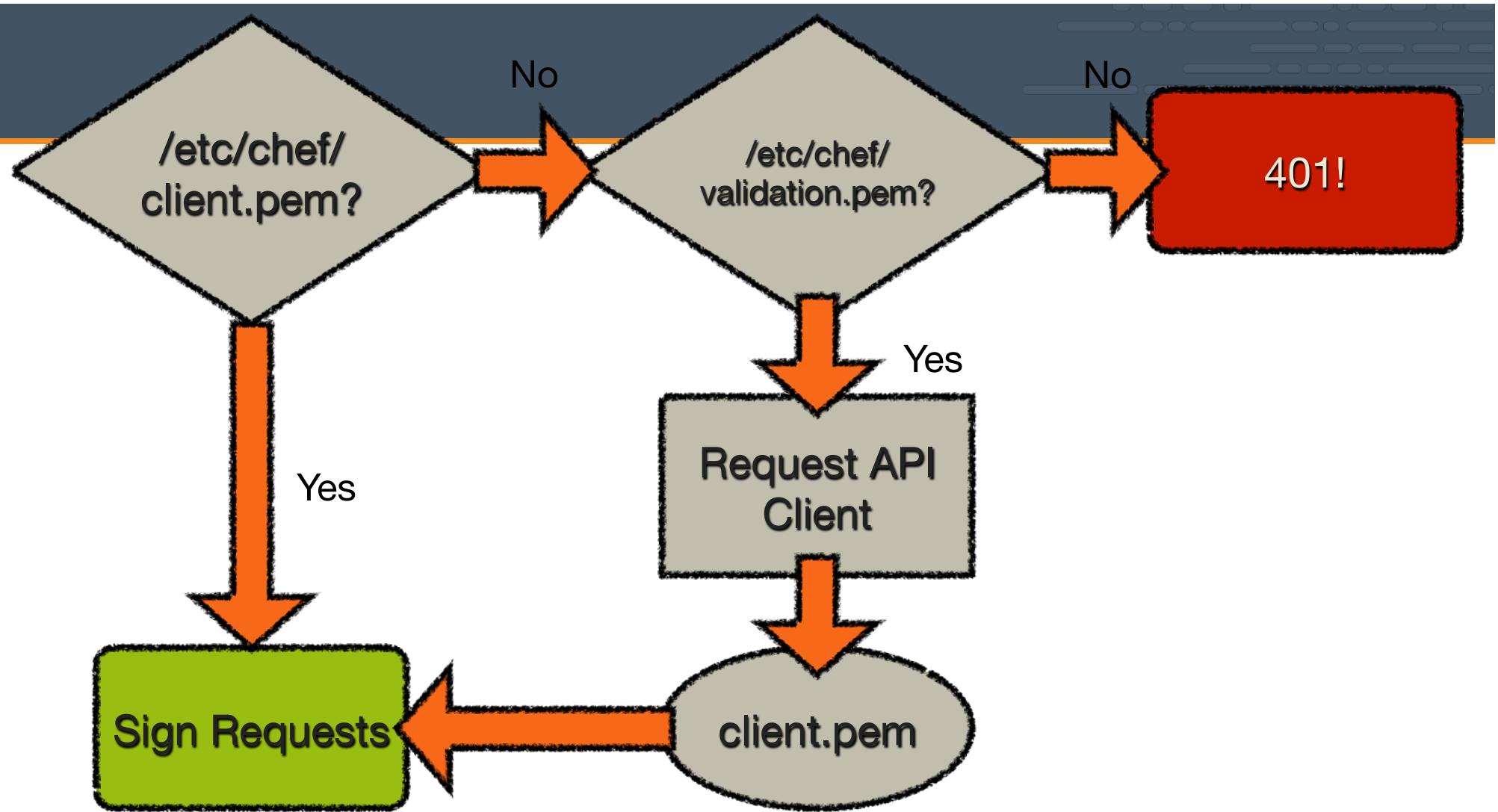
# Lesson Objectives

- After completing the lesson, you will be able to a run
  - Explain the basic security model of Chef

# Private Keys

- 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...



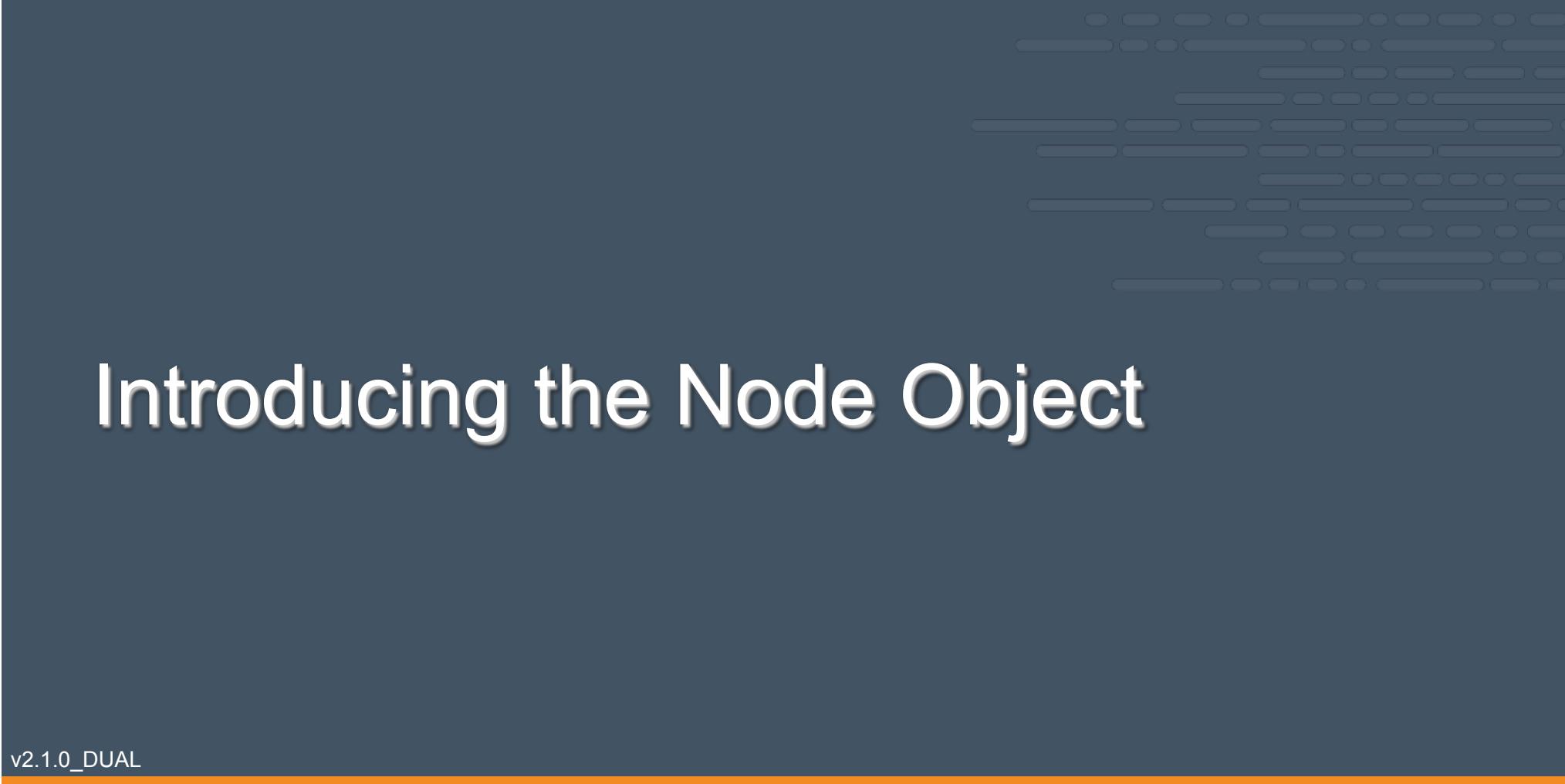


# Private Key Notes

- **validation.pem** on the node is an exact match of **ORGNAME-validator.pem** on your workstation
  - It was placed on the node at bootstrap to allow for the creation of client.pem
- validation.pem on the node poses a **security risk**, so delete it (using Chef!)
- The **chef-client** cookbook (see [supermarket.chef.io](https://supermarket.chef.io)) has a recipe to do this for you

# Exercise: Delete validation.pem

- 1. Create a cookbook, name it del\_val**
- 2. Write a recipe to delete the validation.pem file**
- 3. Read the docs ([docs.chef.io](https://docs.chef.io)) to find a file resource and to see its available actions and options**
- 4. Make sure you specify to not make backups!**
- 5. Test it: upload the cookbook, add the recipe to the run\_list, converge the node and verify that validation.pem has been deleted**



# Introducing the Node Object

v2.1.0\_DUAL



# Lesson Objectives

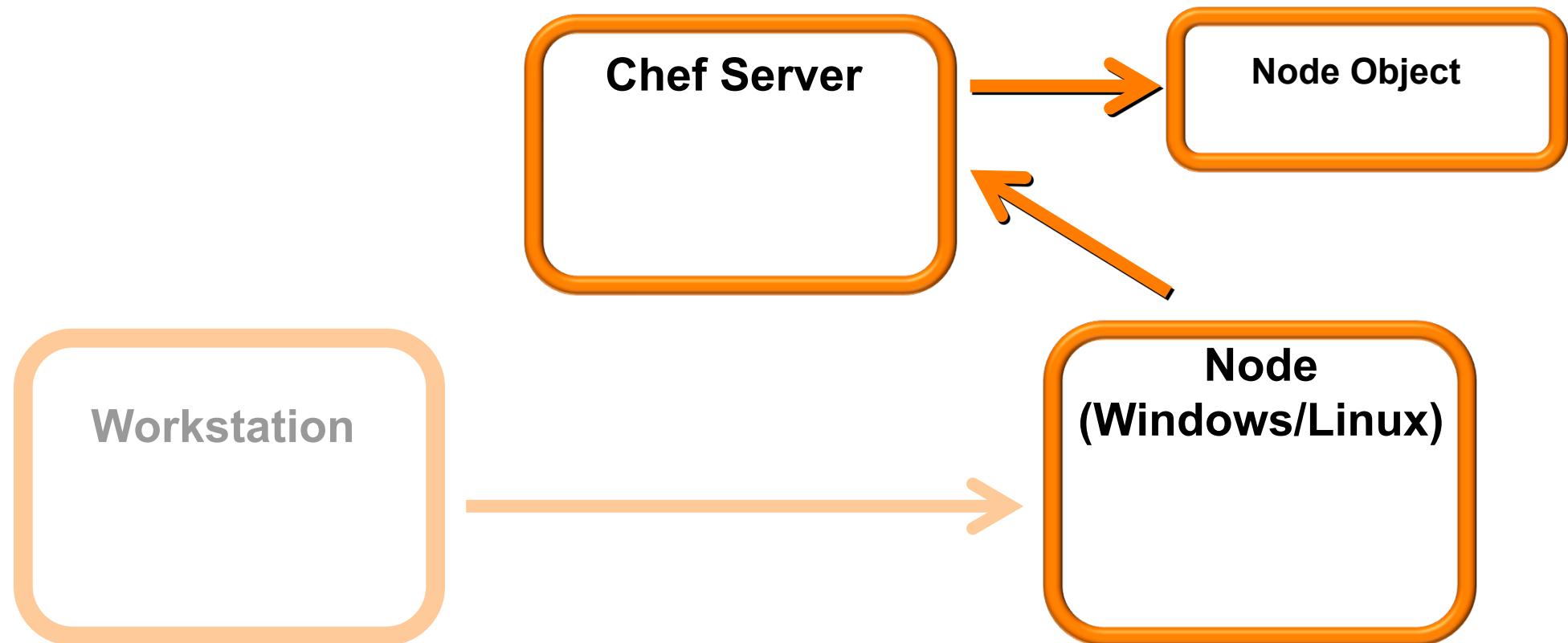
- We are going to:
  - Explain what the Node Object represents in Chef
  - Show the content of the Node Object
  - Describe what Node Attributes are
  - Show a Node's Attributes

# What is the Node Object

A **Node** is any physical, virtual, or cloud machine that is configured to be maintained by Chef

But, the '**Node Object**' is the representation of that node, stored as JSON, and cached on the Chef Server

# Node Object Representation



## Why is the Node Object so Awesome?

When you are writing Recipes, the Node object is always available to you

- This means **any node** in your Organization can discover information about **any other node** in your Organization

## Node Object Use Case

A Load Balancer (LB) wants to get the IP Address of every available Webserver:

1. LB searches all Node Objects within the Organization that have a **Role** of “**Webserver**”
2. LB builds its configuration file from the list of IP Addresses returned from the Node Object search
3. Now, the LB knows the IP address of every node to which it can direct webserver traffic

# How Is The Node Object Created?

1. During each convergence, Ohai runs on the node and gathers all the system information
2. When the node connects to the Chef Server, it uploads the information discovered by Ohai
3. Chef Server adds what it knows about the node (the run list, roles and additional attributes) to this Ohai information
4. All of this information is stored as the **Node Object**

# Example of Ohai output from a Node

```
"languages": {  
  "ruby": {  
    "platform": "i386-  
mingw32",  
    "version": "1.9.3",  
    "target": "i386-pc-  
mingw32",  
    "target_cpu": "i386",  
    "target_vendor": "pc",  
    "target_os": "mingw32",  
    "host": "i686-pc-  
mingw32",  
    "host_cpu": "i686",  
    "host_os": "mingw32",  
    "host_vendor": "pc",  
  },  
  "perl": {  
    "version": "5.8.8",  
    "archname": "msys-64int"  
  },  
  ...  
}
```

```
"kernel": {  
  "os_info": {  
    "boot_device": "\Device\  
\HddiskVolume1",  
    "build_number": "9200",  
    "build_type": "Multiprocessor  
Free",  
    "caption": "Microsoft Windows  
Server 2012 Standard",  
    "code_set": "1252",  
    "country_code": "1",  
    "creation_class_name":  
      "Win32_OperatingSystem",  
    "cs_creation_class_name":  
      "Win32_ComputerSystem",  
    "csd_version": null,  
    "cs_name": "C1263834251"},  
  "machine": "x86_64",  
  "pnp_drivers": {  
    "SWD\\PRINTENUM\\{B86F2C50-  
C174-459A-AE9E-0097FCE113EE}":  
    ...  
  }  
}
```

```
"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",  
      "ip4_addresses": [  
        {  
          "ip": "192.168.1.10",  
          "netmask": "255.255.255.0",  
          "broadcast": "192.168.1.255",  
          "scope": "Global",  
          "link_layer": "00:0C:29:4D:4A:00",  
          "physical": true  
        }  
      ],  
      "ip6_addresses": [  
        {  
          "ip": "fe80::20c:29ff:fe4d:4a00",  
          "netmask": "ffff:ffff:ffff:ffff::ffff:ffff:ffff:ffff",  
          "scope": "LinkLocal",  
          "link_layer": "00:0C:29:4D:4A:00",  
          "physical": true  
        }  
      ]  
    }  
  }  
}
```

# Contents of the Node object

Node objects are made up of **Attributes**

- Most attributed are discovered automatically by **Ohai** (eg: platform, ipaddress, number of CPUs)
- You can add additional attributes to the Node Object from Roles, Environments, Attribute Files and directly from within Recipes

# View Node Object on Chef Server

- Click the 'Attributes' tab
- This reads the Node Object
- Created by combining Ohai output (from the node) with Chef Server output (i.e. run\_list, roles and attributes)

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, a sidebar on the left has a 'Nodes' section 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 includes columns for Node Name, Platform, FQDN, IP Address, Uptime, and Last Checkin. Below the table, a specific node is selected ('Node: node1'). A tab bar at the bottom of this section includes 'Details', 'Attributes' (which is highlighted with a yellow oval), and 'Permissions'. Under the 'Attributes' tab, there's a 'Attributes' section with 'Expand All' and 'Collapse All' buttons. It lists several tags and their values: languages (languages), kernel (kernel), os (windows), os\_version (6.2.9200), chef\_packages (hostname: C1263834251, fqdn: C1263834251, domain:), network (network), and counters (counters). At the bottom of the page, there's a footer with copyright information, help links, and a feedback button.

Copyright © 2012 – 2014 Chef, Inc.

Need help? If you have questions or are stuck, we are here to help.

Feedback

# How Is The Node Object Updated?

1. The Node Object is updated at the beginning of each convergence with Chef Server
2. The Node Object is again updated **at the end of each convergence** to record any differences to the node caused by the convergence

# Exercise: Showing details about a node

```
ws> knife node show <nodename>
```

```
Node Name: windowsnode
Environment: _default
FQDN: C1263834251
IP: 10.160.33.236
Run List: recipe[iis_demo]
Roles:
Recipes: iis_demo, iis_demo::default
Platform: windows 6.2.9200
Tags:
```

## Exercise: Showing all the attributes of a node

```
ws> knife node show <nodename> -l
```

```
Node Name: windowsnode
Environment: _default
FQDN: C1263834251
IP: 10.160.33.236
Run List: recipe[iis_demo]
Roles:
Recipes: iis_demo, iis_demo::default
Platform: windows 6.2.9200
Tags:
Attributes:
...
```

## Exercise: Format Node Object output in JSON

```
ws> knife node show <nodename> -l -Fj
```

```
{  
  "name": "windowsnode",  
  "chef_environment": "_default",  
  "run_list": ["recipe[iis_demo]"],  
  "normal": {"tags": []}  
}
```

## Exercise: Traverse the Node Object Array

```
ws> knife node show <nodename>  
-a ipaddress
```

```
linuxnode:  
  ipaddress: 10.35.46.77
```

## Exercise: Traverse the Node Object Array

```
ws> knife node show <nodename>  
-a virtualization.system
```

```
linuxnode:  
  virtualization.system: vmware
```

## Windows Exercise: Traverse the Node Object Array

```
ws> knife node show <nodename>  
-a kernel.os_info.build_number
```

```
windowsnode:  
  kernel.os_info.build_number: 9200
```

# Questions

- What is the Node object?
- What is a Node Attribute?
- How do you display all the attributes of a Node?
- How would you return just the cpu attribute of your node?

# Setting and Retrieving Node Attributes

v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - Understand the value of attributes
  - List where you can set attributes
  - Describe how attributes are set and retrieved
  - Explain the attribute merge order and precedence rules

# Why Are Attributes Useful?

- Attributes keep the program code separate from the data
- Example:
  - You create a MySQL cookbook for your company
  - Users of the cookbook set the **username**, **password** and **port number** in a separate “attributes” file
    - Not much technical expertise required for this
  - Users never have to touch (or even understand) the MySQL recipes or configuration files

# Attribute Sources

- Attributes can be set at various levels
  - From the node itself (by Ohai)
  - In roles
  - In environments
  - In cookbook recipes (recommended only for debugging)
  - In cookbook attribute files (the most common place)

# Merge Order and Precedence

Precedence levels	Attribute locations (where it is set)			
	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		

# Setting Node Attributes

- Use this format to **SET** an attribute:
  - *precedence[“attribute\_name”] = “value”*
- Example:

precedence      attribute name      attribute value  
↓                ↓                ↓  
`default[ "indexfile" ] = "Default.htm"`

# Retrieving Node Attributes

- Use this format to **GET** an attribute:
  - **node[“attribute\_name”]**
- Example:

Get a node  
attribute      attribute name

```
graph TD; A[Get a node attribute] --> B[node["indexfile"]]; C[attribute name] --> D["indexfile"];
```

node[ "indexfile" ]

This will return the attribute value “**Default.htm**”

## Best Practice: Include cookbook in attribute name

- If the cookbook “`iis_demo`” contains an attribute named “`indexfile`”
  - Include the cookbook name to make it easier to find the attribute later
  - **Set with:** `default[“iis_demo”][“indexfile”] = “Default.htm”`
  - **Retrieve with:** `node[“iis_demo”][“indexfile”]`

# Retrieving Node Attributes

- Node attributes can be set in different locations, with different precedence levels, so which value should be returned?

**We want whichever one wins!**

- Which one wins here, set in the attribute file:

**default[“iis\_demo”][“indexfile”] = “file1.htm”**

**override[“iis\_demo”][“indexfile”] = “file2.htm”**

**normal[“iis\_demo”][“indexfile”] = “file3.htm”**

# Merge Order and Precedence

Precedence levels	Attribute locations (where it is set)			
	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		

# Retrieving Node Attributes

- Retrieve the attribute with:
  - `node["iis_demo"]["indexfile"]`

`default["iis_demo"]["indexfile"] = "file1.htm"`

`override["iis_demo"]["indexfile"] = "file2.htm"`

`normal["iis_demo"]["indexfile"] = "file3.htm"`

**Override** is the highest precedence in this example,  
so the result will be “**file2.htm**”

# Retrieving Node Attributes

- What will be returned in this case? The attribute is set in 3 different locations, with different precedence levels:

Set in Role: `override["filename"] = "homepage.html"`

Set in Environment: `default ["filename"] = "Default.htm"`

Set in Attribute file: `force_override["filename"] = "index.html"`

# Merge Order and Precedence

Precedence levels	Attribute locations (where it is set)			
	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		

# Retrieving Node Attributes

- What will be returned in this case? The attribute is set in 3 different locations, with different precedence levels:

Set in Role: `override["filename"] = "homepage.html"`

Set in Environment: `default ["filename"] = "Default.htm"`

Set in Attribute file: `force_override["filename"] = "index.html"`

**Winner: index.html**

# The Problem and the Success Criteria

- **The Problem:** We have hard coded our homepage in the recipe, but we may want to change that without altering the recipe
- **Success Criteria:** We can change the homepage by changing an attribute value

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

**OPEN IN EDITOR:** \cookbooks\iis\_demo\metadata.rb

```
name          'iis_demo'
maintainer    'YOUR_COMPANY_NAME'
maintainer_email 'YOUR_EMAIL'
license        'All rights reserved'
description    'Installs/Configures iis_demo'
long_description IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version        '0.1.1'
```

**SAVE FILE!**

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

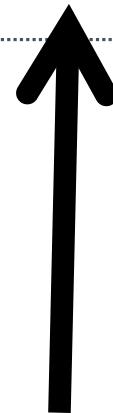
# Windows Exercise: Set an attribute in the IIS\_DEMO attributes file

**OPEN IN EDITOR:** cookbooks\iis\_demo\attributes\default.rb

```
default[ "iis_demo" ][ "indexfile" ] = "Default1.htm"
```

**SAVE FILE!**

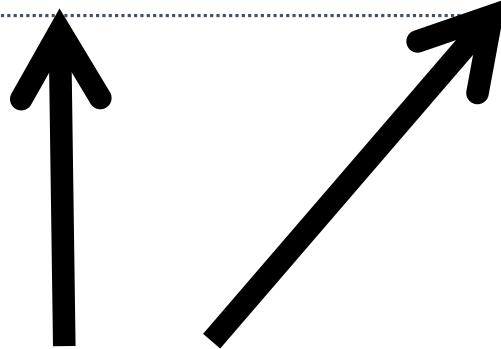
- Set an attribute to a specific value in the attributes file
- Note the addition of the number “1” in the attribute value



# Windows Exercise: Set an attribute in the IIS\_DEMO attributes file

**OPEN IN EDITOR:** cookbooks\iis\_demo\attributes\default.rb

```
default[ "iis_demo" ][ "indexfile" ] = "Default1.htm"
```



**No spaces here**

## Windows Exercise: Add Default1.htm to cookbook's files/default directory

**OPEN IN EDITOR:** cookbooks\iis\_demo\files\default\Default1.htm

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>This is Default1.htm</h2>
    <p>We configured this in the attributes file</p>
  </body>
</html>
```

**SAVE FILE!**

# Windows Exercise: Retrieve attribute in a recipe

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
service "w3svc" do
  action [:enable, :start]
end

cookbook_file 'c:\inetpub\wwwroot\Default.htm' do
  source "Default.htm" ← Replace this value
  rights :read, "Everyone"
end
```

**SAVE FILE!**

# Windows Exercise: Retrieve attribute in a recipe

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
service "w3svc" do
  action [:enable, :start]
end

cookbook_file 'c:\inetpub\wwwroot\Default.htm' do
  source node["iis_demo"]["indexfile"]
  rights :read, "Everyone"
end
```

**SAVE FILE!**

# Windows Exercise: Retrieve attribute in a recipe

**OPEN IN EDITOR:** cookbooks\iis\_demo\recipes\default.rb

```
service "w3svc" do
  action [:enable, :start]
end

cookbook_file 'c:\inetpub\wwwroot\Default.htm' do
  source node["iis_demo"]["indexfile"]
  rights :read, "Everyone"
end
```

**Put a space here  
(between “source” and “node”)**

**SAVE FILE!**

**No spaces  
in the orange box**

## Windows Exercise: Upload the iis\_demo cookbook

```
ws> knife cookbook upload iis_demo
```

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

# Windows Exercise: Re-run Chef Client

```
rn> chef-client
```

```
...
* cookbook_file[c:\inetpub\wwwroot\Default.htm] action create[2014-02-25T01:39:45-08:00] INFO: Processing cookbook_file[c:\inetpub\wwwroot\Default.htm] action create (iis_demo::default line 18)
[2014-02-25T01:39:45-08:00] INFO: cookbook_file[c:\inetpub\wwwroot\Default.htm] backed up to c:/chef/backup/inetpub/wwwroot\Default.htm.chef-20140225013945.548152
[2014-02-25T01:39:45-08:00] INFO: cookbook_file[c:\inetpub\wwwroot\Default.htm] updated file contents c:\inetpub\wwwroot\Default.htm

- update content in file c:\inetpub\wwwroot\Default.htm from 231d53 to 442a41
  --- c:\inetpub\wwwroot\Default.htm      2014-02-24 06:25:33.000000000 -0800
  +++ C:/Users/chef/AppData/Local/Temp/4/Default.htm20140225-1272-1evzs75 2014-02-25 01:39:45.000000000 -0800
@@ -1,7 +1,8 @@
<html>
- <body>
-   <h1>Hello, world!</h1>
- </body>
-</html>
-
+ <body>
+   <h1>Hello, world!</h1>
+   <h2>This is Default1.htm</h2>
+   <p>We configured this in the attributes file</p>
+ </body>
+</html>
[2014-02-25T01:39:46-08:00] INFO: Chef Run complete in 3.556801 seconds
```

# Exercise: Verify new homepage works

- Open a web browser
- The homepage takes the attribute file value



# Viewing Attributes via WebUI

The screenshot shows the CHEF Manage interface. The top navigation bar includes the CHEF logo, a search bar, and tabs for Nodes, Reports, Policy, and Administration. The main content area is titled "Showing All Nodes". A table lists one node: "node1" (Platform: windows, FQDN: C1263834251, IP Address: 10.160.33.236). Below the table, the "Node: node1" details are shown, with tabs for Details, Attributes, and Permissions. The Attributes tab is selected, displaying a hierarchical list of attributes. The "iis\_demo" attribute under "tags" is highlighted with an orange border. Other visible attributes include "languages", "kernel", "os", "os\_version", "chef\_packages", "hostname", "fqdn", "domain", "network", and "counters".

Node Name	Platform	FQDN	IP Address
node1	windows	C1263834251	10.160.33.236

**Node: node1**

Attributes

Expand All Collapse All

- iis\_demo
  - indexfile: Default2.htm
- + languages
- + kernel
- os: windows
- os\_version: 6.2.9200
- + chef\_packages
  - hostname: C1263834251
  - fqdn: C1263834251
  - domain:
- + network
- + counters

# Windows: Viewing specific attributes using knife

```
ws> knife node show <nodename> -a iis_demo.indexfile
```

Default Attributes:  
iis\_demo:  
indexfile: Default1.htm

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

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

```
name          'apache'
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        '0.1.1'
```

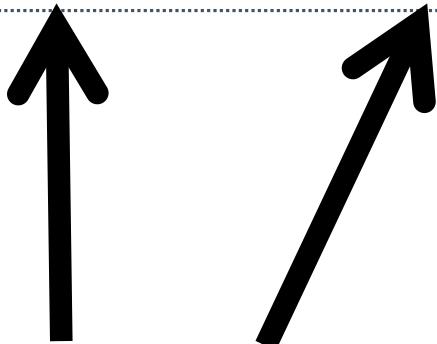
**SAVE FILE!**

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

# Linux Exercise: Set an attribute in the Apache attributes file

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

```
default["apache"]["indexfile"] = "index1.html"
```



**No spaces here**

**SAVE FILE!**

## Linux Exercise: Add index1.html to cookbook's files/default directory

**OPEN IN EDITOR:** cookbooks/apache/files/default/index1.html

```
<html>
  <body>
    <h1>Hello, world!</h1>
    <h2>This is index1.html</h2>
    <p>We configured this in the attributes file</p>
  </body>
</html>
```

**SAVE FILE!**

# Linux Exercise: Set attribute in a recipe

**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" ← Replace this value
  mode "0644"
end
```

**SAVE FILE!**

# Linux Exercise: Set attribute in a recipe

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

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

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

**SAVE FILE!**

# Linux Exercise: Set attribute in a recipe

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

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

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

**Put a space here  
(between “source” and “node”)**

**SAVE FILE!**

**No spaces  
in the orange box**

# Linux Exercise: Verify new homepage works

1. Upload the cookbook
2. Converge your node
3. Open a web browser
  - The homepage takes the attribute file value



# Exercise: Viewing Attributes via WebUI

The screenshot shows the CHEF MANAGE web interface. The top navigation bar includes links for Nodes, Reports, Policy, and Administration. The left sidebar under the 'Nodes' heading contains options: Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area displays 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 section for 'Node: node1' is shown with tabs for Details, Attributes, and Permissions. The 'Attributes' tab is selected, showing a tree view of node attributes. An attribute named 'iis\_demo' is expanded, revealing its value 'indexfile: Default2.htm'. This expanded node is highlighted with an orange border. Other collapsed attributes listed include tags, languages, kernel, os, os\_version, chef\_packages, hostname, fqdn, domain, network, and counters.

Node Name	Platform	FQDN	IP Address
node1	windows	C1263834251	10.160.33.236

**Node: node1**

Attributes

Expand All Collapse All

- iis\_demo  
indexfile: Default2.htm
- + languages
- + kernel
- os: windows
- os\_version: 6.2.9200
- + chef\_packages  
hostname: C1263834251  
fqdn: C1263834251  
domain:  
+ network  
+ counters

# Linux: Viewing specific attributes using knife

```
ws> knife node show <nodename> -a  
apache.indexfile
```

```
...  
Attributes:  
tags:  
  
Default Attributes:  
apache:  
  indexfile: index1.html  
  
Override Attributes:  
  
Automatic Attributes (Ohai Data):  
chef_packages:  
  chef:  
    chef_root: /opt/chef/embedded/lib/ruby/gems/1.9.1/gems/chef-11.10.4-x86-mingw32/lib  
    version: 11.10.4  
  ohai:  
    ohai_root: /opt/chef/embedded/lib/ruby/gems/1.9.1/gems/ohai-6.20.0/lib/ohai  
    version: 6.20.0  
command:  
counters:  
  network:  
...  
...
```

## Linux Exercise: Revert to the original homepage

1. Change the attribute file back to “index.html”
2. Upload the cookbook, converge your node
3. Open a web browser
  - You should see the original page displayed

# Questions

- Where can attributes be set?
- What knife commands can you use to view attributes?
- How many levels of attribute precedence are there?
- Which takes precedence, a default attribute set in the attribute file or in a recipe?

# Attributes, Templates, and Cookbook Dependencies

Writing a Linux MOTD Cookbook

v2.1.0\_DUAL



# Lesson Objectives

- In this lesson, we will:
  - Introduce the Template resource
  - Use ERB (Embedded Ruby)
  - Specify cookbook dependencies

# The /etc/motd file

- On Linux systems, the contents of /etc/motd is displayed when a user logs onto the console
- In the past, this was used by Administrators to send messages to users when they logged in, such as:
  - “Server going down at 2:30 pm for maintenance”
  - “You must change your login password before Monday”

# 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 (or when we run 'cat /etc/motd')

## Best Practice: One cookbook – One purpose

- Create **one** cookbook who's job it is to write the correct content to the /etc/motd file
- Create a **separate** cookbook to determine PCI Compliance
  - MOTD cookbook will write the /etc/motd file based on the result of the PCI cookbook

# Exercise: Create a cookbook named ‘pci’

Create a new cookbook named ‘pci’

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

# Note about PCI

- In the real world, the PCI cookbook might have a complicated procedure to determine if a particular server is truly in compliance or not
- For our purposes, we're simply setting an attribute to 'true' or 'false' as a way to determine compliance:
  - `pci.in_scope = true`
  - `pci.in_scope = false`

# 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 literal **true**
- Notice there are no quotes around **true**

# Linux Exercise: Create a cookbook named 'motd'

```
ws> knife cookbook create motd
```

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

# Linux 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

# What resource should we use?

- We could try and use a `cookbook_file` here, and rely on the file copy rules
  - This would create the exact same file on each server
- 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

# 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 -%>
```

- To print the resolution of the ruby evaluation:
  - Add an equal sign <%=

# 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 print the resolution of the ruby evaluation:
  - Add an equal sign <%=

# 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
  - Note there are no spaces in 'node[ "pci" ][ "in\_scope" ]'

# 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 %>
```

- Ending with -%> will not insert a line in the resulting 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!**

# Exercise: Upload both cookbooks

```
ws> knife cookbook upload motd pci
```

```
Uploading motd [ 0.1.0 ]
Uploading pci [ 0.1.0 ]
Uploaded 2 cookbooks.
```

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

```
ws> knife node run_list add <nodename> "recipe[motd]"
```

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

## Exercise: Show the motd recipe in your test node's run list

```
ws> knife node show <nodename>
```

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

# Exercise: Re-run the Chef Client

```
rn> chef-client
```

```
Starting Chef Client, version 11.8.2...
 - motdCompiling Cookbooks...Converging 4 resourcesRecipe: 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 10) (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[2014-01-06T09:14:36-05:00] INFO: Processing cookbook_file[/var/www/html/index.html]
action create (apache::default line 10) (up to date)  * recipe: motd: fault * template[/etc/
motd] action create[2014-01-06T09:14:36-05:00] INFO: Processing template[/etc/motd] action
create (motd: default line 10) (up to date)  * file /etc/motd -> /etc/motd
motd-----Error
motd=====Chef:
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)
```

# 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::TemplateEr<----->
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
```

# nil:NilClass error

- The bottom line is when you see this ruby error

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

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

## Linux 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"
```

## Linux 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

# Exercise: Upload the motd cookbook

Upload the MOTD cookbook, since you made changes to it

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

# Exercise: Re-run the Chef Client

```
rn> chef-client
```

```
Forking chef instance to converge...Starting Chef Client, version 11.8.2*** Chef 11.8.2 ***Chef-client pid: 15152Run List is [recipe[apache], recipe[motd]]Run List expands to [apache, motd]Starting Chef Client on fedora-18-x86_64 (node 64ebb9) - update content in file /etc/motd (from 3b03e06ebb9 to 6ebb9) (new content is binary) - restore selinux security context (chef-client completed in 0.3587 seconds)Removing cookbook /apache/files/default/index.html from the cache; it is no longer needed by chef-client.Running report handlersReport handlers completeChef Client finished, 1 resources updatedSending resource update report (run-id: f923449e-cc7b-45f7-a9fb-ac1da3653557)
```

# Exercise: Check your work

```
ws> knife cookbook upload motd
```

```
rn> chef-client
```

```
rn> 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 node show <nodename> -a pci
```

```
1 items found
linuxnode:
  pci:
    in_scope: true
```

## 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
- If you are including a recipe from another cookbook, you must add a ‘depends’ to metadata.rb
  - `include_recipe` is discussed later in the course.

## Exercise: Change attribute to false

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

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

1. Change the attribute to **false** and upload the cookbook
2. Make sure **log\_level is set to :info** (in /etc/chef/client.rb)
3. converge the node
4. See the changes in the **/etc/motd** file
5. Show the pci.in\_scope attribute from the command line

## Exercise: Find the /etc/motd backup

- Look at your chef-client run output
- In the log output, find the location where chef-client placed the **/etc/motd** backup
- View the backup file

# Congratulations!

- You now know the most important resources for configuration management!
  - Package (Linux)
  - Template
  - Service

# Attributes, Templates, and Cookbook Dependencies

Writing a Windows Hosts Cookbook

v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - Use ERB Templates in Chef
  - Specify cookbook dependencies
  - Perform the cookbook creation, upload, and test loop

# The Problem and the Success Criteria

- **The Problem:** To comply with local regulations, we need to block access from our Windows servers located in the country of Fitzmandia to the New York Times website.
- **How:** We will use the Windows hosts file to set the New York Times IP address to 0.0.0.0.
- **Success Criteria:** We see the hosts file updated on the Windows server.

# We have a small problem...

- But we don't have an attribute that reflects which region the server is in
  - We could add a node attribute for Location, but that will become very tiresome at scale

# Solution

- The best thing to do is create a Datacenter cookbook, and add our attribute there
- Well-factored cookbooks only contain the information relevant to their domain
- We know we will likely have other things related to Location (security settings, for example)

## Windows Exercise: Create a cookbook named ‘datacenter’

```
> knife cookbook create datacenter
```

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

## Windows Exercise: Create a default.rb attribute file in the datacenter cookbook

**OPEN IN EDITOR:** cookbooks\datacenter\attributes\default.rb

```
default[ "datacenter" ][ "location" ] = "Fitzmandia"
```

**SAVE FILE!**

- The 'datacenter' in the attribute is just convention so you know the cookbook the attribute was set in. This is not an enforced syntax

## Windows Exercise: Upload the Datacenter cookbook

```
> knife cookbook upload datacenter
```

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

## Windows Exercise: Create a cookbook named ‘hosts’

Create the new cookbook named ‘hosts’

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

# Windows Exercise:

## Open the default recipe in your editor

**OPEN IN EDITOR:** cookbooks\hosts\recipes\default.rb

```
#  
# Cookbook Name:: hosts  
# Recipe:: default  
#  
# Copyright 2014, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
#
```

# What resource should we use?

- We could try and use a `cookbook_file` resource here, and rely on the file copy rules
  - This would create the exact same file on each server
- Obviously, that's dramatically inefficient
- Instead, we will render a **template resource** - a file that is a mixture of the contents we want, and embedded Ruby code

# The Template Resource

**OPEN IN EDITOR:** cookbooks\hosts\recipes\default.rb

```
#  
# Cookbook Name:: hosts  
# Recipe:: default  
# Copyright 2013, YOUR_COMPANY_NAME  
#  
# All rights reserved - Do Not Redistribute  
  
template "#{ENV['windir']}\\System32\\drivers\\etc\\hosts" do  
  source "hosts.erb"  
end
```

**SAVE FILE!**

# Windows Exercise: Open hosts.erb in your Editor

**OPEN IN EDITOR:** cookbooks\hosts\templates\default\hosts.erb

```
#This file is managed by the server <%= node[ "fqdn" ] %>
<% case node[ "datacenter" ][ "location" ] -%>
  <% when "Fitzmandia" -%>
    0.0.0.0      www.nytimes.com
<% end -%>
```

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

# Windows Exercise: Open hosts.erb in your Editor

**OPEN IN EDITOR:** cookbooks\hosts\templates\default\hosts.erb

```
#This file is managed by the server <%= node[ "fqdn" ] %>
<% case node[ "datacenter" ][ "location" ] -%>
  <% when "Fitzmandia" -%>
    0.0.0.0      www.nytimes.com
<% end -%>
```

# Windows Exercise: Open hosts.erb in your Editor

**OPEN IN EDITOR:** cookbooks\hosts\templates\default\hosts.erb

```
#This file is managed by the server <%= node[ "fqdn" ] %>
<% case node[ "datacenter" ][ "location" ] -%>
  <% when "Fitzmandia" -%>
    0.0.0.0      www.nytimes.com
<% end -%>
```

# Windows Exercise: Open hosts.erb in your Editor

**OPEN IN EDITOR:** cookbooks\hosts\templates\default\hosts.erb

```
#This file is managed by the server <%= node[ "fqdn" ] %>
<% case node[ "datacenter" ][ "location" ] -%>
  <% when "Fitzmandia" -%>
    0.0.0.0      www.nytimes.com
<% end -%>
```

# Windows Exercise: Open hosts.erb in your Editor

**OPEN IN EDITOR:** cookbooks\hosts\templates\default\hosts.erb

```
#This file is managed by the server <%= node[ "fqdn" ] %>
<% case node[ "datacenter" ][ "location" ] -%>
  <% when "Fitzmandia" -%>
    0.0.0.0      www.nytimes.com
<% end -%>
```

# Windows Exercise: Open hosts.erb in your Editor

**OPEN IN EDITOR:** cookbooks\hosts\templates\default\hosts.erb

```
#This file is managed by the server <%= node[ "fqdn" ] %>
<% case node[ "datacenter" ][ "location" ] -%>
  <% when "Fitzmandia" -%>
    0.0.0.0      www.nytimes.com
<% end -%>
```

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

## Windows Exercise: Upload the hosts cookbook

Upload the hosts cookbook

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

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

```
> knife node run_list add <nodename> 'recipe[hosts]'
```

```
windowsnode:
  run_list:
    recipe[iis_demo]
    recipe[hosts]
```

## Windows Exercise: Re-run the Chef Client

```
rn> chef-client
```

# FAIL!

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

Template Context:

---

on line #2

```
1: #This file is managed by the server <%= node["fqdn"] %>
2: <% case node["datacenter"]["location"] -%>
3: <% when "Fitzmandia" -%>
4:   0.0.0.0  www.nytimes.com
5: <% 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)

# Scroll up

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

```
=====
=Error executing action `create` on resource 'template[c:\Windows
\System32\driver\etc\hosts]'
=====
=Chef::Mixin::Template::TemplateError-----
undefined method `[]' for nil:NilClassResource
Declaration:-----# In c:\opscode\cache\cookbooks\hosts\recipes
\default.rb 10: template "c:\Windows\System32\drivers\etc\hosts" do 11:
source "hosts.erb"
```

# nil:NilClass error

- The bottom line is when you see this ruby error

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

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

## Windows Exercise: Add a dependency on the datacenter cookbook to the hosts cookbook

**OPEN IN EDITOR:** \cookbooks\hosts\metadata.rb

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

depends        'datacenter'
```

## Windows Exercise: Upload the hosts cookbook

Upload the hosts cookbook

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

## Windows Exercise: Re-run the Chef Client

```
rn> chef-client
```

**WIN!**

## Windows Exercise: Check your work

```
rn> gc C:\Windows\System32\drivers\etc\hosts
```

```
# This file is managed by the server C1263834251
0.0.0.0      www.nytimes.com
```

# Windows Exercise: Check automated backups

```
rn> gci C:\chef\backup\Windows\System32\drivers\etc
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
-a---	7/25/2012 10:26 PM	824	hosts.chef-20140225031457.550034

## Windows Exercise: Show your test node's datacenter attribute

```
> knife node show <nodename> -a datacenter
```

```
1 items found
windowsnode:
  datacenter:
    location: Fitzmandia
```

## Exercise: Change your node's Location

**OPEN IN EDITOR:** cookbooks\datacenter\attributes\default.rb

```
default["datacenter"]["location"] = "Russia"
```

- 1.Modify the attributes file**
- 2.Upload the cookbook you modified**
- 3.Re-converge**
- 4.Look at the hosts file on the node**

```
rn> gc C:\Windows\System32\drivers\etc\hosts
```

```
# This file is managed by the server C1263834251
```

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

```
> knife node show windowsnode -a datacenter
```

```
windowsnode:  
  datacenter:  
    location: Russia
```

# Congratulations!

- You now know the most important resources for configuration management!
  - Powershell\_script (Windows)
  - Template
  - Service

# Questions

- What goes in a cookbook's attribute files?
- When do you need to specify a cookbook dependency?
- What does <%= mean, and where will you encounter it?
- Where are .erb files stored on the Workstation?
- What are the most important resources in Windows configuration management?
- Where is Fitzmandia?

# Template Variables, Notifications, and Controlling Idempotency

Refactoring the Windows iis\_demo cookbook for multiple websites

v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - Control idempotence manually with `not_if`
  - Describe the Directory resource
  - Implement resource notifications
  - Use Template Variables to dynamically generate web pages

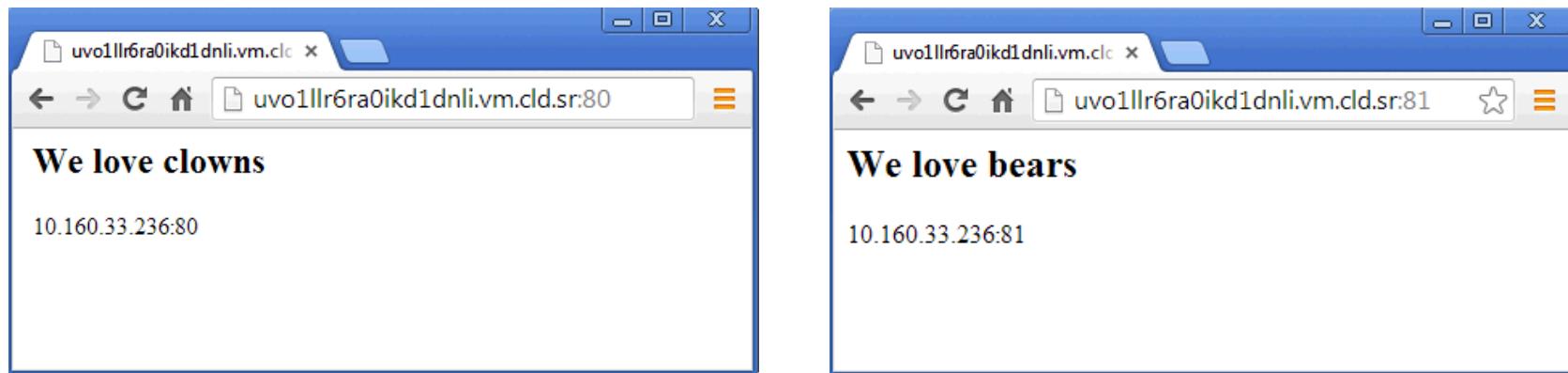
# The Problem and the Success Criteria

- **The Problem:** We need to deploy multiple custom home pages running on different ports
- **Success Criteria:** Be able to view our custom home pages
- First we'll write a Windows cookbook, then a Linux cookbook

# 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 pages

# Exercise: Here is what we want to see



**Note the different port numbers**

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

**OPEN IN EDITOR:** \cookbooks\iis\_demo\metadata.rb

```
name          'iis_demo'
maintainer    'YOUR_COMPANY_NAME'
maintainer_email 'YOUR_EMAIL'
license        'All rights reserved'
description    'Installs/Configures iis_demo'
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/>

## Windows Exercise: Create a default.rb attribute file in the `iis_demo` cookbook

OPEN IN EDITOR: \cookbooks\iis\_demo\attributes\default.rb

```
default["iis_demo"]["indexfile"] = "Default1.htm"
```

```
default["iis_demo"]["sites"]["clowns"] = { "port" => 80 }
default["iis_demo"]["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
  - You can ignore the existing **indexfile** attribute

## Windows Exercise: Comment out the existing cookbook\_file resource

**OPEN IN EDITOR:** \cookbooks\iis\_demo\recipes\default.rb

```
service "w3svc" do
  action [:enable, :start ]
end

#cookbook_file "c:\\inetpub\\wwwroot\\Default.htm" do
#  source node["iis_demo"]["indexfile"]
#  rights :read, "Everyone"
#end
```

**SAVE FILE!**

- Comment out (or delete) the cookbook\_file resource from recipe

## Windows Exercise: Stop the Default Web Site with Powershell resource

**OPEN IN EDITOR:** \cookbooks\iis\_demo\recipes\default.rb

```
service "w3svc" do
  action [:enable, :start ]
end

#cookbook_file "c:\\inetpub\\wwwroot\\Default.htm" do
#  source node["iis_demo"]["indexfile"]
#  rights :read, "Everyone"
#end

powershell_script "disable default site" do
  code 'get-website "Default Web Site*" | where {$_.state -ne "Stopped"} | Stop-Website'
end
```

**SAVE FILE!**

- Use powershell resource to stop Default Web Site if running.

# Idempotency Guarantees

```
powershell_script "disable default site" do
  code 'get-website "Default Web Site*" | where {$_.state -ne "Stopped"} | Stop-Website'
end
```

- Stop the existing default web site
- With `powershell_script` resource it's up to you to make resources idempotent
- Either do it in Powershell, or Chef idempotency guards (later)

# Windows Exercise: Iterate over each IIS site

**OPEN IN EDITOR:** \cookbooks\iis\_demo\recipes\default.rb

```
powershell_script "disable default site" do
  code 'get-website "Default Web Site*" | where {$_.state -ne "Stopped"} | Stop-Website'
end

node["iis_demo"]["sites"].each do |site_name, site_data|
```

**SAVE FILE!**

- `node["iis_demo"]["sites"]` is the Ruby hash, with keys and values we defined in our default attributes

# Windows Exercise: Iterate over each IIS site

```
node["iis_demo"]["sites"].each do |site_name, site_data|
```

- Calling `.each` loops over each site

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

- **First pass**

- `site_name = "clowns"`
- `site_data = { "port" => 80 }`

- **Second pass**

- `site_name = "bears"`
- `site_data = { "port" => 81 }`

# Windows Exercise: Iterate over each IIS site

**OPEN IN EDITOR:** \cookbooks\iis\_demo\recipes\default.rb

```
powershell_script "disable default site" do
  code 'get-website "Default Web Site*" | where {$_.state -ne "Stopped"} | Stop-Website'
end

node["iis_demo"]["sites"].each do |site_name, site_data|
  site_dir = "#{ENV['SYSTEMDRIVE']}\\inetpub\\wwwroot\\#{site_name}"
```

**SAVE FILE!**

- Create a ruby variable (for use only inside this loop) for the site directory

# Windows Exercise: Add the Directory resource

```
node["iis_demo"]["sites"].each do |site_name, site_data|
  site_dir = "#{ENV['SYSTEMDRIVE']}\\inetpub\\wwwroot\\#{site_name}"
```

```
directory site_dir
```

What do you think is the default action for the ‘directory’ resource?

# Windows: Variables and interpolation

- Given the following variable for 'document\_root'

```
site_dir = "c:\foo\foobar"
```

- What directory will be created in each of the following?

```
directory site_dir do ...
```

c:\foo\foobar

```
directory "site_dir" do ...
```

c:\site\_dir

```
directory "#{site_dir}" do ...
```

c:\foo\foobar

## Powershell\_script resources are generally not idempotent

- Chef will stop your run if a resource fails
- Most command line utilities are not idempotent - they assume a human being is interacting with, and understands, the state of the system
- The result is - it's up to you to make powershell\_script resources idempotent

## Windows:

### Enter the `not_if` and `only_if` metaparameters

```
not_if "C:\\Windows\\System32\\inetsrv\\appcmd.exe  
list apppool #{site_name}"
```

- 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

# Windows Exercise: Add an iis\_site resource to install the new sites

```
node["iis_demo"]["sites"].each do |site_name, site_data|
  site_dir = "#{ENV['SYSTEMDRIVE']}\\inetpub\\wwwroot\\#{site_name}"

  directory site_dir

  powershell_script "create app pool for #{site_name}" do
    code "New-WebAppPool #{site_name}"
    not_if "C:\\Windows\\System32\\inetsrv\\appcmd.exe list appPool #{site_name}"
  end
end
```

**SAVE FILE!**

- Create App Pool, but only if one doesn't already exist.

# Windows Exercise: Add a Web Site with Powershell\_script resource

```
powershell_script "create app pool for #{site_name}" do
  code "New-WebAppPool #{site_name}"
  not_if "C:\\Windows\\System32\\inetsrv\\appcmd.exe list apppool #{site_name}"
end
```

```
powershell_script "new website for #{site_name}" do
  code <<-EOH
    Import-Module WebAdministration
    if(-not(test-path IIS:\\Sites\\#{site_name})){
      New-WebSite -name #{site_name} -Port #{site_data["port"]} -PhysicalPath
      #{site_dir} -ApplicationPool #{site_name}
    }
  EOH
end
```

**SAVE FILE!**

- Create Web Site, but only if one doesn't already exist.

# 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

## Windows Exercise: Use a template to create Default.htm for each site

```
.... #{site_dir} -ApplicationPool #{site_name}
}
EOH
end

template "#{site_dir}\\Default.htm" do
  source "Default.htm.erb"
  rights :read, "Everyone"
  variables(
    :site_name => site_name,
    :port => site_data["port"]
  )
  notifies :restart, "service[w3svc]"
end
end
```

# Notifications

```
  notifies :restart, "service[w3svc]"
```

- 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 iis”
- 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: Add Default.htm.erb to your templates directory

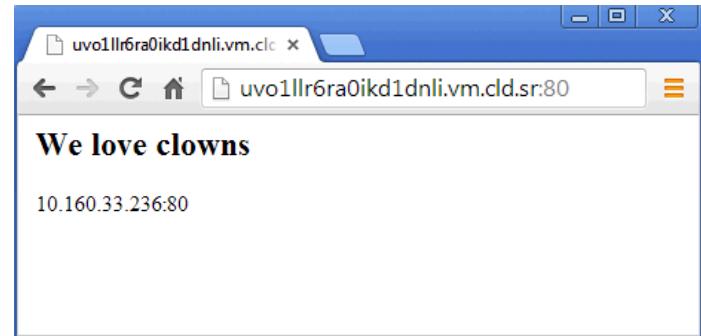
**OPEN IN EDITOR:** cookbooks\iis\_demo\templates\default\Default.htm.erb

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

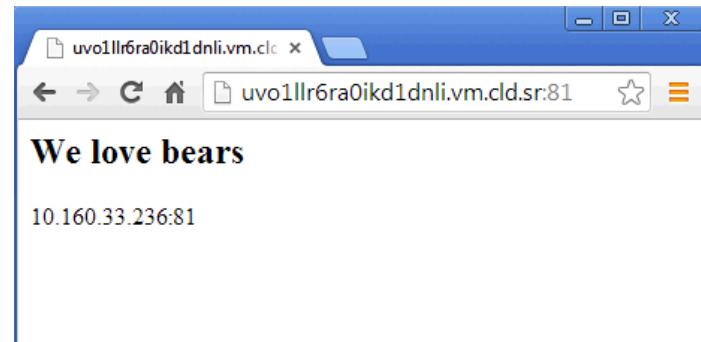
- **Note:** reference variables passed into the template by prefixing the variable with an @ symbol

# Exercise: Converge and test your code

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



- 1.Upload your cookbook**
- 2.Converge your node**
- 3.Verify the new websites are working on their respective port numbers**



# The entire recipe

Windows Powershell Example:

<https://gist.github.com/johnfitzpatrick/9207666>

# Exercise: Add zebras!

- Modify your attributes file
- Add “zebras” on port “82”
- Upload, converge, and see if port 82 works

# Chef resources on Windows

- Resources in Chef that work on Windows:
  - `file`, `remote_file`, `cookbook_file`, `template`
  - `registry`
  - `directory`, `remote_directory`
  - `user`, `group`
  - `mount`, `env`
  - `service`
  - `execute`
    - `ruby_block`, `powershell_script`, `batch`
- Check the Docs...

# Questions

- How do you control the idempotence of a **powershell\_script** 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?

# Template Variables, Notifications, and Controlling Idempotency

Refactoring the Linux apache cookbook for multiple websites

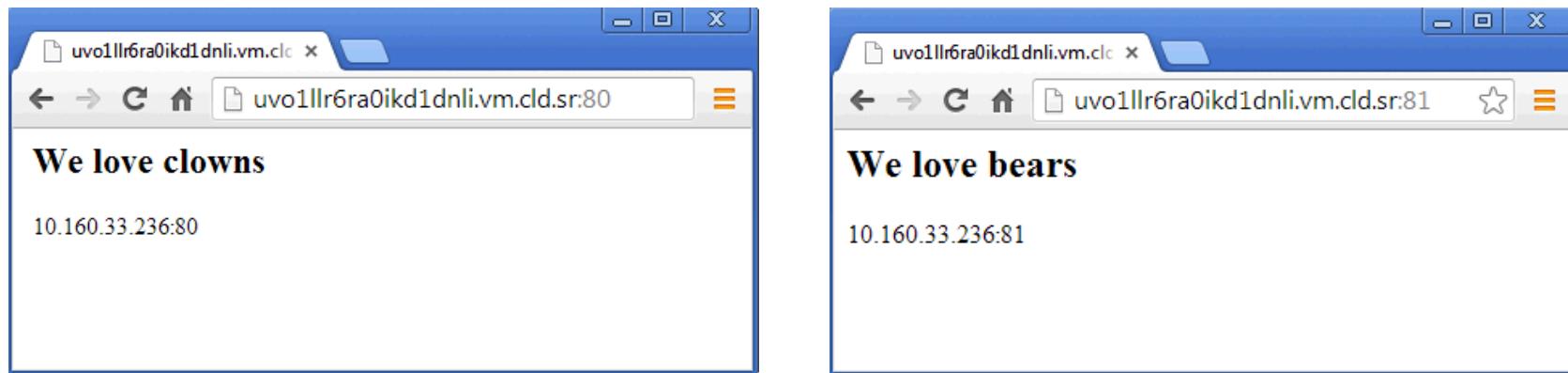
v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - Use the `execute` resource
  - Control idempotence manually with `not_if` and `only_if`
  - Describe the `Directory` resource
  - Implement resource notifications
  - Use Template Variables to dynamically generate web pages

# What we want to accomplish



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

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

```
name          'apache'
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/>

## Linux Exercise:

### Create a default.rb attribute file in the apache cookbook

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

```
default["apache"]["indexfile"] = "index.html"

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

- We add information about the sites we need to deploy
  - One about Clowns, running on port 80
  - One about Bears, running on port 81
  - You can ignore the **indexfile** attribute

## Linux Exercise: Comment out the 'cookbook\_file' resource

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

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

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

**SAVE FILE!**

- Comment out the cookbook\_file resource from recipe

## Linux Exercise: Stop the Default Web Site with 'execute' resource

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

```
service "httpd" do
  action [:enable, :start]
end
#cookbook_file "/var/www/html/index.html" do
#  source node["apache"]["indexfile"]
#  mode "0644"
#end
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
```

**SAVE FILE!**

- Comment out the cookbook\_file resource from recipe
- Use execute resource to stop Default Web Site if running.

# Linux 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[ "apache" ][ "sites" ].each do |site_name, site_data|
```

**SAVE FILE!**

- `node[ "apache" ][ "sites" ]` is the Ruby hash, with keys and values we defined in our default attributes

# Linux Exercise: Iterate over each IIS site

```
node["apache"]["sites"].each do |site_name, site_data|
```

- 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 }`

# Windows: Variables and interpolation

- Given the following variable for 'document\_root'

```
document_root = "/foo/foobar"
```

- What directory will be created in each of the following?

```
directory document_root do ...
```

/foo/foobar

```
directory "document_root" do ...
```

document\_root

```
directory "#{document_root}" do ...
```

/foo/foobar

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

**SAVE FILE!**

- `node["apache"]["sites"]` is the Ruby hash, with keys and values we defined in our default attributes

# Exercise: Add the Directory resource

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

```
directory document_root do
  mode "0755"
  recursive true
end
```

# Linux Exercise: Add an apache resource to configure the new sites

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

  directory document_root do
    mode "0755"
    recursive true
  end

  # 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
end
```

# Linux Exercise:

## Use a template to create index.html for each site

```
:document_root => document_root,
:port => site_data["port"]
)
notifies :restart, "service[httpd]"
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
```

**SAVE FILE!**

# Linux Exercise: Move Service block to the END of the 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
```

**SAVE FILE!**

# The entire recipe

Linux Example:

<https://gist.github.com/johnfitzpatrick/8954699>

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

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

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

- **Note:** reference variables passed into the template by prefixing the variable with an @ symbol

## Exercise: Add custom.erb to your templates directory

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

- Copy the following file:  
**<http://tinyurl.com/n4ggmh8>**  
Follow the instructions to create your custom.erb file.
- Note the two **template variables** are prefixed with an @ symbol
- We are using a conditional if here

## Custom.erb solution

- Solution to the custom.erb re-write
  - <http://tinyurl.com/mfw9xln>
- What apache sees
  - <http://tinyurl.com/loakgdz>

# Linux Exercise: Upload the apache cookbook

## Upload the cookbook

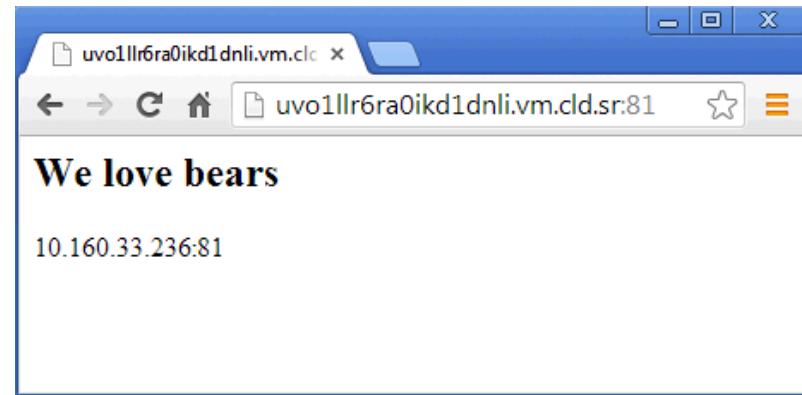
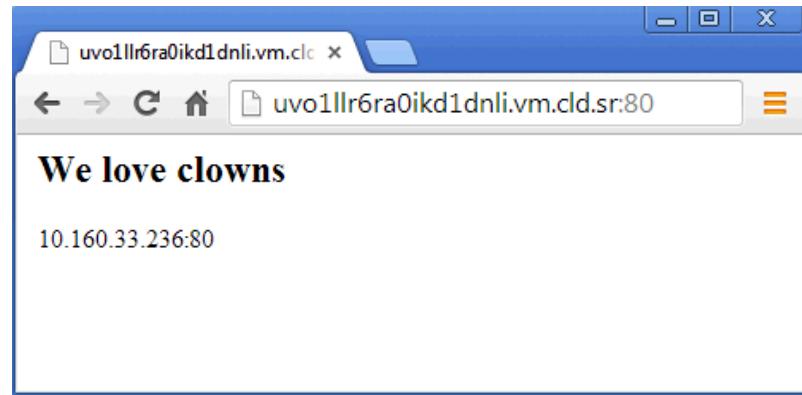
```
Uploading apache [0.2.0]
Uploaded 1 cookbook.
```

# Linux Exercise: Re-run the Chef Client

## Run chef-client on the node

```
"/opt/chef/AppData/Local/Temp/2/chef-script20130918-1312-ebgd1i.ps1"
  * template[/etc/httpd/conf.d/bears/index.html] action create
INFO: Processing template[/etc/httpd/conf.d/bears/index.html] action create
(apache::default line 45)
  (up to date)
Recipe: hosts::default
  * template[/etc/hosts] action create
INFO: Processing template[/etc/hosts] action create (hosts::default line 11)
  (up to date)
[2013-09-18T21:02:52+00:00] INFO: Chef Run complete in 13.1508 seconds
[2013-09-18T21:02:52+00:00] INFO: Running report handlers
[2013-09-18T21:02:52+00:00] INFO: Report handlers complete
Chef Client finished, 6 resources updated
```

# Exercise: Verify our two sites are working!



# Exercise: Add zebras!

- Modify your attributes file
- Add “zebras” on port “82”
- Converge, and see if port 82 works

# 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?

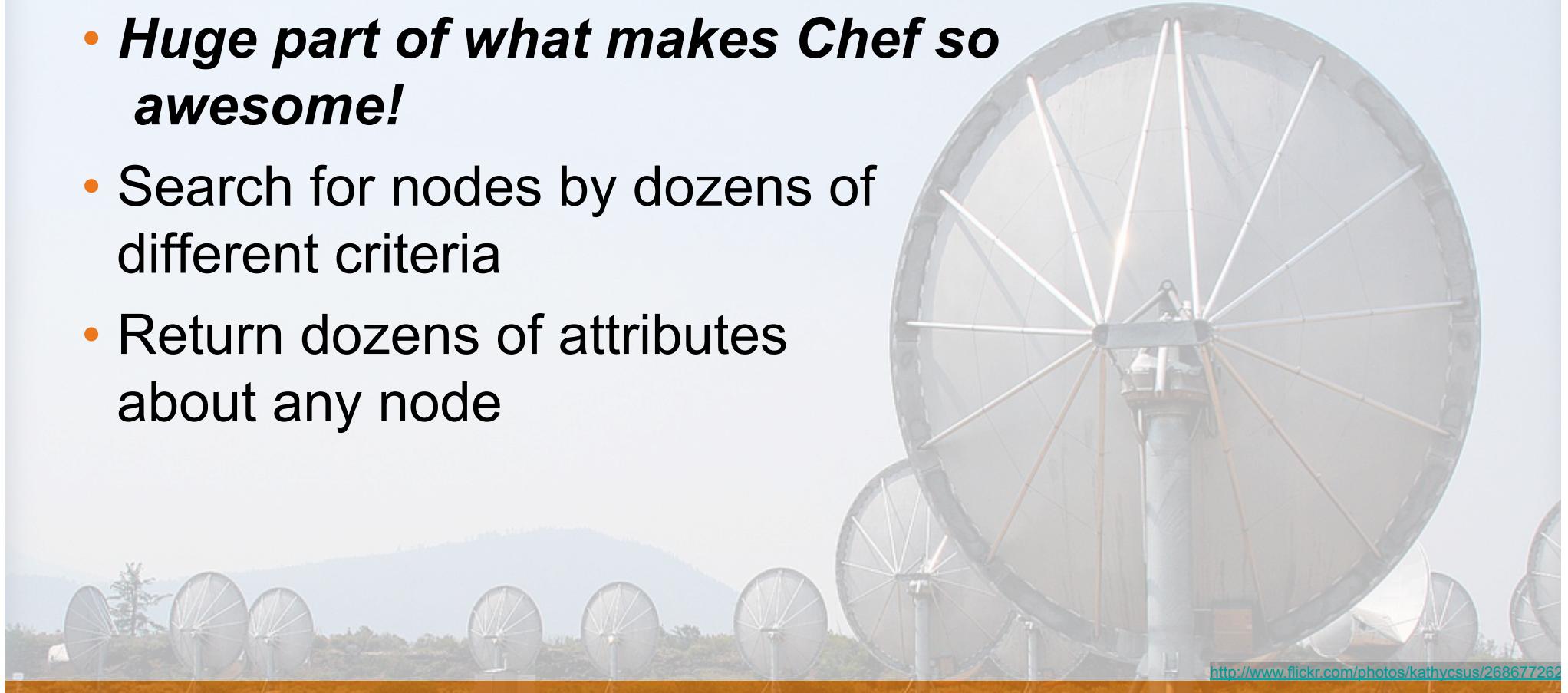
# Search

v2.1.0\_DUAL



# Search

- ***Huge part of what makes Chef so awesome!***
- Search for nodes by dozens of different criteria
- Return dozens of attributes about any node

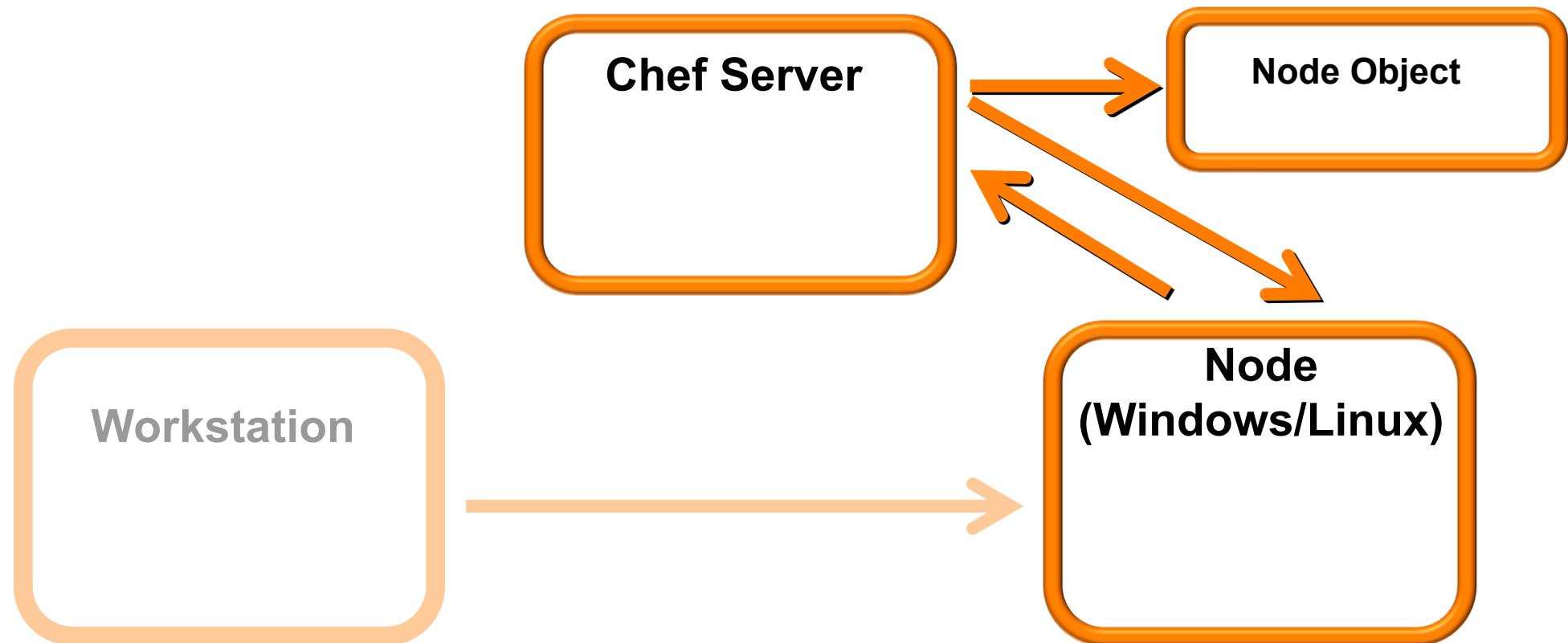


<http://www.flickr.com/photos/kathycsus/268677262>

# Lesson Objectives

- After completing the lesson, you will be able to
  - Describe the query syntax used in Search
  - Invoke a search from command line
  - Build a search into your recipe code

# Node Object Review



# What is search?

- Use search to query data indexed on the chef server
  - Eg: from the Node Object
- Syntax is

```
knife search INDEX SEARCH_QUERY
```
- Where **index** can be client, environment, node, role, (or the name of a data bag)
- Search query runs on the Chef Server and is invoked from within a recipe, using knife or via WebUI

# Query Syntax

- Chef search uses the Solr 'key:search\_pattern' syntax

```
knife search node "ipaddress:10.20.30.40"
```

- Use an asterisk ("\*") for >=0 chars in a wildcard search

```
knife search node "ipaddress:10.*"
```

```
knife search node "platfo*:windows"
```

- Use a question mark ("?") to replace a single character

```
knife search node "platform_version:6.2.920?"
```

# Exercise: Use knife search

```
ws> knife search node "*:*"
```

```
Node Name: windowdnode
Environment: _default
FQDN: C1263834251
IP: 10.160.33.236
Run List: recipe[iis_demo], recipe[hosts]
Roles:
Recipes: iis_demo, hosts, iis_demo::default, hosts::default
Platform: windows 6.2.9200
Tags:
```

## Exercise: Find a node with the specific attribute value

```
ws> knife search node "ipaddress:10.*"
```

```
2 items found
```

```
Node Name: windowsnode
Environment: _default
FQDN: C1263834251
IP: 10.160.33.236
Run List: recipe[iis_demo], recipe[hosts]
Roles:
Recipes: iis_demo, hosts, iis_demo::default, hosts::default
Platform: windows 6.2.9200
Tags:
```

## Exercise: Using Knife to find an attribute value

```
ws> knife search node "*:*" -a ipaddress
```

```
2 items found
```

```
windowsnode:
```

```
  ipaddress: 10.160.33.236
```

```
linuxnode:
```

```
  ipaddress: 10.20.134.17
```

## Exercise: Return just the attribute value

```
ws> knife search node "ipaddress:10.*" -a ipaddress
```

```
2 items found
```

```
windowsnode:  
  ipaddress: 10.160.33.236  
linuxnode:  
  ipaddress: 10.20.134.17
```

# Exercise: Boolean matching

```
> knife search node "ipaddress:10* AND platform:windows"
```

```
> knife search node "ipaddress:10* AND platform:centos"
```

```
Node Name:      node1
Environment:    _default
FQDN:          C1263834251
IP:            10.160.33.236
Run List:       recipe[iis_demo], recipe[hosts]
Roles:
Recipes:        iis_demo, hosts, iis_demo::default, hosts::default
```

# Exercise: Boolean matching

```
> knife search node "ipaddress:[10.0.* TO 10.2.*]"
```

```
1 item found
```

```
Node Name: windowsnode
Environment: _default
FQDN: C1263834251
IP: 10.160.33.236
Run List: recipe[iis_demo], recipe[hosts]
Roles:
Recipes: iis_demo, hosts, iis_demo::default, hosts::default
Platform: windows 6.2.9200
Tags:
```

# Search Use Case

- 5,500 total nodes in our Organization
- 4,750 of them are **webservers**
- Our **monitoring server** needs to know the **IP Address** and **Hostname** of every available **webserver**
- The monitoring server needs to **automatically** update this list every **30 minutes**
- **Method:** Search for every **webserver**, then add its **IP Address** and **Hostname** to the correct config file

# Search for nodes – Windows Example

This is inside a recipe:

```
my_webservers = search("node","role:webserver")

template "C:\Monitoring App\cfg\servers.list" do
  source "servers.list.erb"
  variables :my_webservers =>my_webservers.uniq
  notifies :restart, "service[monitoring_svc]"
end
```

# Search for nodes – Windows Example

```
my_webservers = search("node","role:webserver")

template "C:\Monitoring App\cfg\servers.list" do
  source "servers.list.erb"
  variables :my_webservers => my_webservers.uniq
  notifies :restart, "service[monitoring_svc]"
end
```

# Search for nodes – Windows Example

```
my_webservers = search("node","role:webserver")

template "C:\Monitoring App\cfg\servers.list" do
  source "servers.list.erb"
  variables :my_webservers => my_webservers.uniq
  notifies :restart, "service[monitoring_svc]"
end
```

# Search for nodes – Windows Example

```
my_webservers = search("node","role:webserver")

template "C:\Monitoring App\cfg\servers.list" do
  source "servers.list.erb"
  variables :my_webservers =>my_webservers.uniq
  notifies :restart, "service[monitoring_svc]"
end
```

# Search for nodes – Windows Example

```
my_webservers = search("node","role:webserver")

template "C:\Monitoring App\cfg\servers.list" do
  source "servers.list.erb"
  variables :my_webservers => my_webservers.uniq
  notifies :restart, "service[monitoring_svc]"
end
```

# Pass the results into a Templates

Content of templates/default/servers.list.erb

```
# List of web servers to monitor
<% @my_webservers.each do |member| -%>
server <%= member[:hostname] %> <%= member[:ipaddress] %>:>
<% end -%>
```

**Loop through once for each webserver found, adding that webserver's info to the Monitoring Server's config file**

# Pass the results into a Templates

```
# List of servers to monitor
<% @my_webservers.each do |member| -%>
  server <%= member[:hostname] %> <%= member[:ipaddress] %>:>
<% end -%>
```

# Pass the results into a Templates

```
# List of servers to monitor
<% @my_webservers.each do |member| -%>
  server <%= member[:hostname] %> <%= member[:ipaddress] %>
<% end -%>
```

# Pass the results into a Templates

```
# List of servers to monitor
<% @my_webservers.each do |member| -%>
  server <%= member[:hostname] %> <%= member[:ipaddress] %>
<% end -%>
```

# Linux Search Use Case

- 5,500 total nodes in our Organization
- 4,750 of them are **webservers**
- Our **load balancer** needs to know the **IP Address** and **Hostname** of every available **webserver**
- The load balancer needs to **automatically** update this list every **30 minutes**
- **Method:** Search for every **webserver**, then add its **IP Address** and **Hostname** to the correct config file

# Search for Nodes – Linux Example

```
pool_members = search("node","role:webserver")
```

```
template "/etc/haproxy/haproxy.cfg" do
  source "haproxy-app_lb.cfg.erb"
  owner "root"
  group "root"
  mode 0644
  variables :pool_members => pool_members.uniq
  notifies :restart, "service[haproxy]"
end
```

# Search for Nodes – Linux Example

```
pool_members = search("node","role:webserver")
```

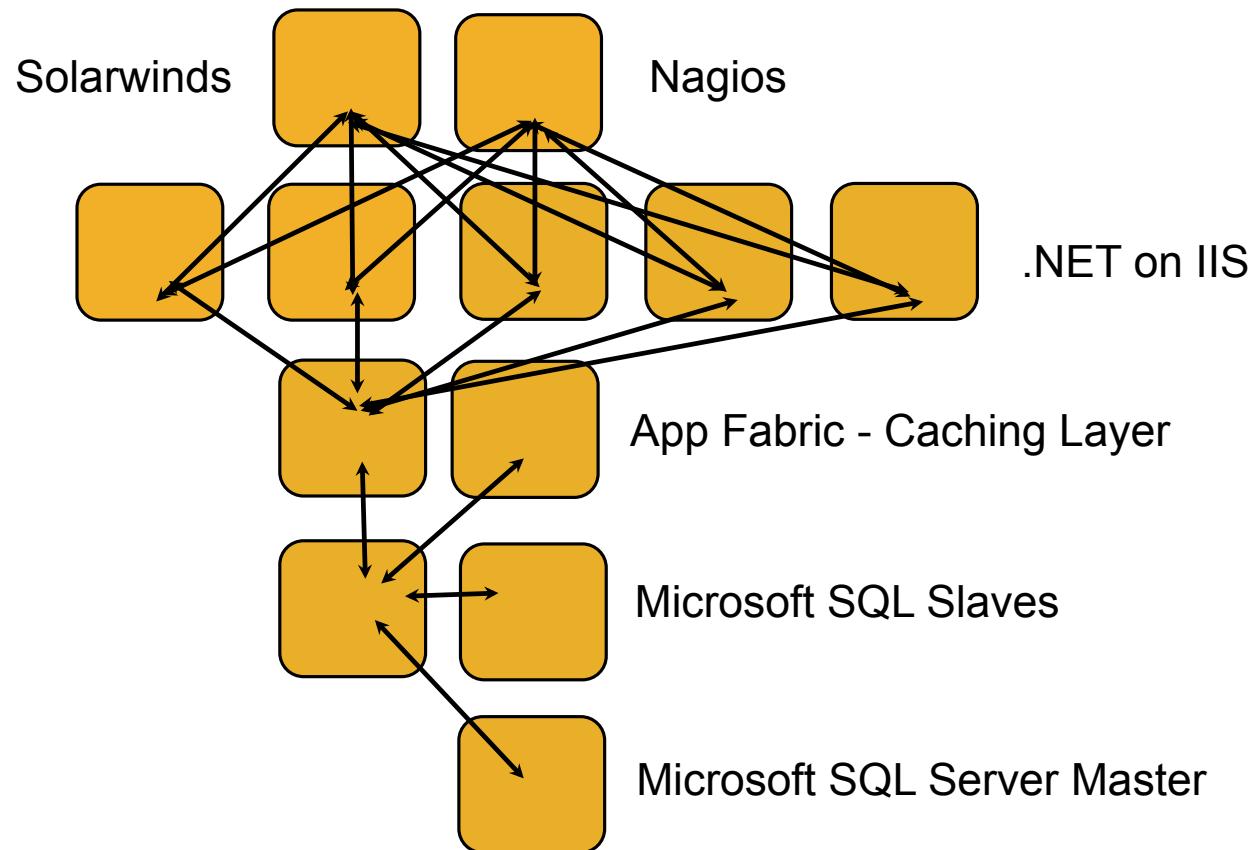
```
template "/etc/haproxy/haproxy.cfg" do
  source "haproxy-app_lb.cfg.erb"
  owner "root"
  group "root"
  mode 0644
  variables :pool_members => pool_members.uniq
  notifies :restart, "service[haproxy]"
end
```

# Pass results into Templates

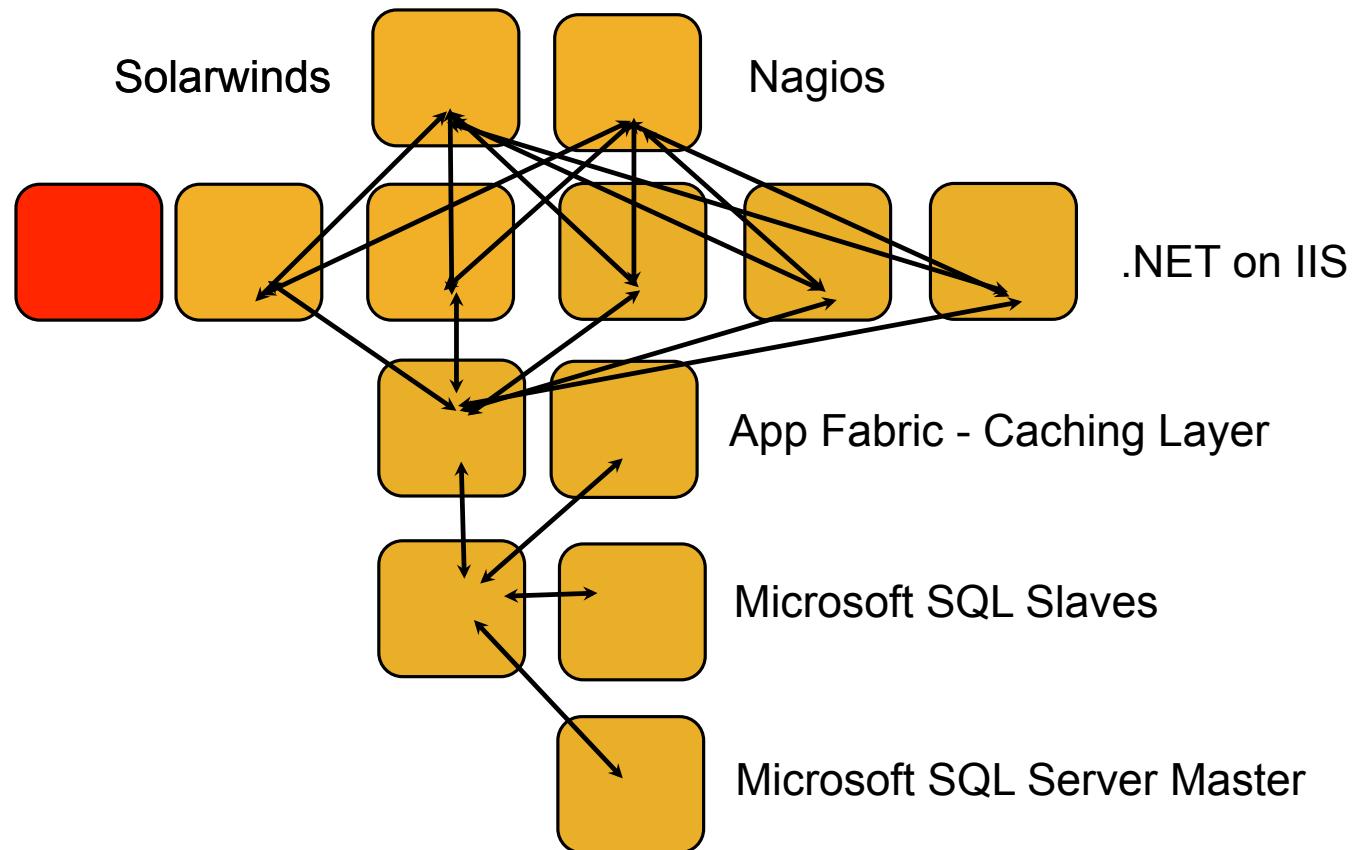
```
# Set up application listeners here.  
listen application 0.0.0.0:80  
    balance roundrobin  
    <% @pool_members.each do |member| -%>  
        server <%= member[:hostname] %> <%= member[:ipaddress] %>  
        weight 1 maxconn 1 check  
<% end -%>
```

**Loop through once for each webserver found, adding that webserver's info to the load balancer's config file**

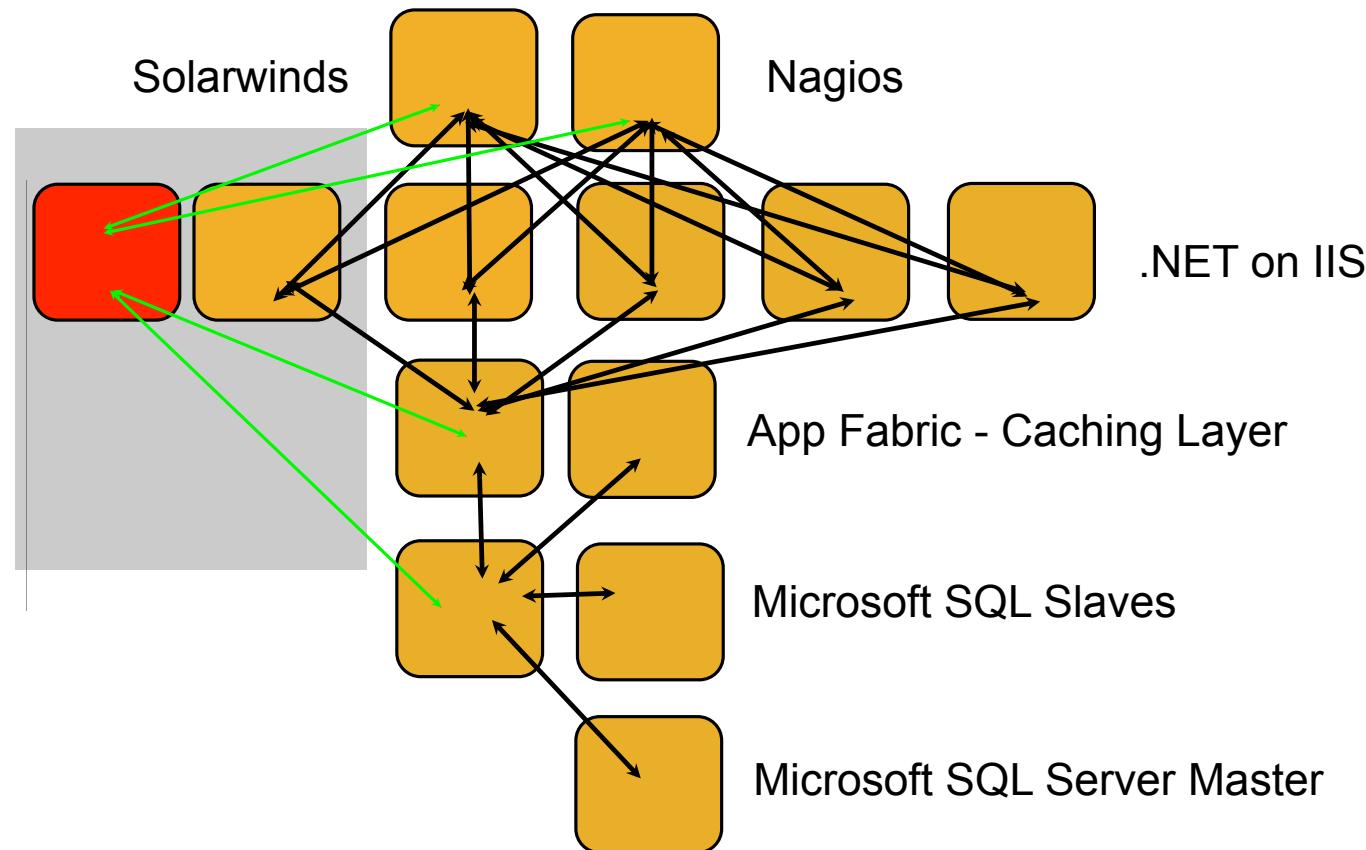
# So when this...



# ...becomes this



# ...this can happen automatically



# Search from within a recipe

- You can invoke a search within a recipe, and use the results to apply further Chef primitives

```
search(INDEX, SEARCH_QUERY).each do |result|
  foo
end
```

- For example

```
search("node", "role:webserver").each do |webserver|
  [register webserver with load balancer]
  or
  [configure firewall so webserver can access database]
end
```

# Exercise: Search for an attribute in a recipe

**OPEN IN EDITOR:** cookbooks/ip-logger/recipes/default.rb

```
search("node","ipaddress:10.*").each do |server|
  log "The servers in your organization have the following FQDN/IP
Addresses:- #{server['fqdn']}/#{server['ipaddress']}"
end
```

1. Create a new cookbook, called ip-logger
2. Write the default recipe (see content above)
3. Upload the cookbook
4. Add the recipe to the run\_list on the node(s)
5. Converge the node(s) and see the results in your chef-client output

# Questions

- What search engine is chef search built upon?
- What is searchable in chef?
- What characters can you use for a wildcard search in SOLR?
- Why should you not set an attribute and then immediately search for it?

# **Recipe Inclusion, Data Bags and Search in Cross-Platform recipes**

Writing a Users cookbook

v2.1.0\_DUAL



# 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
  - Describe the role Search plays in recipes
  - Understand how to write a cross-platform recipe
  - Use `include_recipe`

# The Problem and the Success Criteria

- **The Problem:** Employees should have local user accounts created on all 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 would duplicate 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
  - Passwords (in an encrypted data bag)

## Best Practice: Data Bags are commonly managed in **two ways**

1. By creating files in the `data_bags` directory of your `chef-repo`
  - We would use this to manage data bags by hand
  - This is the model we will be using in this section
2. By creating and managing data bag items directly through the Chef API via a custom script
  - We would use this for LDAP or AD integration

## Exercise: Make a data\_bags directory in chef-repo

```
ws> cd chef-repo  
ws> mkdir data_bags
```

```
Directory: C:\windows\temp\chef-repo
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	25-Feb-14 3:23 PM		data_bags

## Exercise: Create a data bag named users

```
ws> mkdir data_bags\users  
ws> knife data_bag create users
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	25-Feb-14 3:24 PM		users

```
Created data_bag[users]
```

## Exercise: Create a data bag named users

```
ws> mkdir data_bags\users  
ws> knife data_bag create users
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	25-Feb-14 3:24 PM		users

```
Created data_bag[users]
```

## Exercise:

### Create a data bag named groups

```
> mkdir data_bags\groups  
> knife data_bag create groups
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d---	9/18/2013 3:07 PM		groups

Created data\_bag [groups]

# Windows Exercise:

## Create a user item in the users data bag

data\_bags\users\bobo.json

```
{  
  "id": "bobo",  
  "comment": "Bobo T. Clown",  
  "password": "iAm@Cl0wN!!!",  
  "platform": "windows"  
}
```

data\_bags\users\frank.json

```
{  
  "id": "frank",  
  "comment": "Frank Belson",  
  "password": "h@ppyCl0wns!!",  
  "platform": "windows"  
}
```

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

data\_bags/users/daisy.json

```
{  
  "id": "daisy",  
  "comment": "Daisy Duke",  
  "uid": 2003,  
  "gid": 0,  
  "home": "/home/daisy",  
  "shell": "/bin/bash",  
  "platform": "centos"  
}
```

data\_bags/users/rosco.json

```
{  
  "id": "rosco",  
  "comment": "Rosco P.  
Coltrane",  
  "uid": 2004,  
  "gid": 0,  
  "home": "/home/roscos",  
  "shell": "/bin/bash",  
  "platform": "centos"  
}
```

## Exercise:

### Upload the data bag item to Chef Server

```
> knife data_bag from file users filename.json
```

- Upload all additional users to the USERS data\_bag

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

## Exercise:

### Show all the users in the 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
password: sP3nC3r4Hire!

chef_type: data_bag_item
comment: Bobo T. Clown
data_bag: users
gid: 0
home: /home/bobo
id: bobo
```

## Exercise: Find a user's id from the data bag

```
ws> knife search users 'comment:Bobo*' -a id
```

```
1 items found
```

```
data_bag_item_users_bobo:
```

```
  id: bobo
```

## Exercise: Find a user's id from the data bag

```
ws> knife search users 'comment:Daisy*' -a id
```

```
1 items found
```

```
data_bag_item_users_daisy:
```

```
  id: daisy
```

# Windows Exercise:

## Create a group item in the group data bag

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

```
{  
  "id": "win-clowns",  
  "members": [ "bobo", "frank" ],  
  "platform": "windows"  
}
```

**SAVE FILE!**

# Linux Exercise:

## Create a group item in the group data bag

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

```
{  
  "id": "lin-clowns",  
  "gid": 3000,  
  "members": [ "daisy", "rosco" ],  
  "platform": "centos"  
}
```

**SAVE FILE!**

## Exercise:

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

data\_bags/groups/lin-clowns.json

```
{  
  "id": "lin-clowns",  
  "gid": 3000,  
  "members": [ "daisy",  
  "rosco" ],  
  "platform": "centos"  
}
```

data\_bags/groups/win-clowns.json

```
{  
  "id": "win-clowns",  
  "members": [ "bobo",  
  "frank" ],  
  "platform": "windows"  
}
```

# Exercise: Upload the data bag items

**Upload all items in the GROUPS data bag**

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

# Exercise: Create a cookbook named ‘users’

## Create A Users Cookbook

- \*\* Creating cookbook users
- \*\* Creating README for cookbook: users
- \*\* Creating CHANGELOG for cookbook: users
- \*\* Creating metadata for cookbook: users

# Using Variables In Search (don't write this recipe yet)

```
THIS ->
user "bobo" do
  comment "Bobo T. Clown"
  password "iAm@C10wN!!!"
end
```

```
BECOMES THIS ->
search(:users, "platform:windows").each do |user_data|
  user user_data["id"] do
    comment user_data["comment"]
    password user_data["password"]
  end
end
```

# Multi-Platform Exercise: Create a new recipe: create-users.rb

**OPEN IN EDITOR:** cookbooks/users/recipes/create-users.rb

```
case node["platform"]
when "windows"
  search("users", "platform:windows").each do |user_data|
    user user_data["id"] do
      comment user_data["comment"]
      password user_data["password"]
    end
  end
end
```

//File continues on next slide

# Multi-Platform Exercise: Create a new recipe: create-users.rb

**OPEN IN EDITOR:** cookbooks/users/recipes/create-users.rb

```
//This is a continuation of the previous slide
when "centos"
  search("users", "platform:centos").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
end
```

# Full create-users.rb recipe

```
case node["platform"]
when "windows"
  search("users", "platform:windows").each do |user_data|
    user user_data["id"] do
      comment user_data["comment"]
      password user_data["password"]
    end
  end
when "centos"
  search("users", "platform:centos").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
end
```

# Multi-Platform Exercise: Create a new recipe: `create-groups.rb`

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

```
case node["platform"]
when "windows"
  search("groups", "platform:windows").each do |group_data|
    group group_data["id"] do
      members group_data["members"]
    end
  end
end
```

**//File continues on next slide**

# Multi-Platform Exercise: Create a new recipe: `create-groups.rb`

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

**//Continuation of the previous slide**

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

# Full create-groups.rb recipe

```
case node["platform"]

when "windows"
  search(:groups, "platform:windows").each do |group_data|
    group group_data["id"] do
      members group_data["members"]
    end
  end

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

# `include_recipe`

- Allows one recipe to launch another recipe
- Syntax to be used in the calling recipe:  
**`include_recipe "cookbook::recipe"`**
- Can reference a recipe in the same cookbook or in a different cookbook
- If the included recipe is located in a different cookbook, add to the metadata.rb file of **calling** cookbook:  
**`depends "<cookbook_name>"`**
- Chef will only run a named recipe one time in a run list

# Multi-Platform Exercise: Add an Include at the end of the create-users recipe

**OPEN IN EDITOR:** `cookbooks/users/recipes/create-users.rb`

```
when "centos"
  search("users", "platform:centos").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
end

include_recipe "users::create-groups"
```

# Windows Exercise: Open the default recipe in your editor

**OPEN IN EDITOR:** cookbooks/users/recipes/windows-users.rb

```
search(:users, "platform:windows").each do |user_data|
  user user_data["id"] do
    comment user_data["comment"]
    password user_data["password"]
  end
end
```

- 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

## Windows Exercise: Open the users::group recipe in your editor

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

```
search(:groups, "platform:windows").each do |group_data|
  group group_data["id"] do
    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

# `include_recipe`

- Allows one recipe to launch another recipe
- Syntax to be used in the calling recipe:  
**`include_recipe "cookbook::recipe"`**
- Can reference a recipe in the same cookbook or in a different cookbook
- If the included recipe is located in a different cookbook, add to the metadata.rb file of **calling** cookbook:  
**`depends "<cookbook_name>"`**
- Chef will only run a named recipe one time in a run list

# Windows Exercise:

## Open the windows-users.rb recipe in your editor

**OPEN IN EDITOR:** [cookbooks/users/recipes/windows-users.rb](#)

```
search(:users, "platform:windows").each do |user_data|
  user user_data["id"] do
    comment user_data["comment"]
    password user_data["password"]
  end
end

include_recipe "users::win-groups"
```

# Linux Exercise: Open the default recipe in your editor

**OPEN IN EDITOR:** cookbooks/users/recipes/linux-users.rb

```
search(:users, "platform:centos").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
```

- 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

## Linux Exercise: Open the users::group recipe in your editor

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

```
search(:groups, "platform:centos").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

# `include_recipe`

- Allows one recipe to launch another recipe
- Syntax to be used in the calling recipe:  
**`include_recipe "cookbook::recipe"`**
- Can reference a recipe in the same cookbook or in a different cookbook
- If the included recipe is located in a different cookbook, add to the metadata.rb file of **calling** cookbook:  
**`depends "<cookbook_name>"`**
- Chef will only run a named recipe one time in a run list

# Linux Exercise: Open the linux-users.rb recipe in your editor

**OPEN IN EDITOR:** [cookbooks/users/recipes/linux-users.rb](#)

```
search(:users, "platform:centos").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::lin-groups"
```

# Exercise: Upload the users cookbook

```
> knife cookbook upload users
```

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

## Exercise:

Add the correct users recipe to your node's run list

```
> knife node run_list add <nodename>
'recipe[cookbook::recipe]'
```

```
windowsnode:
  run_list:
    recipe[iis_demo]
    recipe[hosts]
    recipe[users::recipe-name]
```

# Exercise: Reconverge Windows nodes

```
> chef-client
```

```
...
Recipe: users::default
  * user[bobo] action create[2014-02-25T07:56:24-08:00] INFO: Processing user[bobo] action
create (users::default line 10)
[2014-02-25T07:56:24-08:00] INFO: user[bobo] created
  - create user user[bobo]
  * user[frank] action create[2014-02-25T07:56:24-08:00] INFO: Processing user[frank] action
create (users::default line 10)
[2014-02-25T07:56:24-08:00] INFO: user[frank] created
  - create user user[frank]
Recipe: users::groups
  * group[win-clowns] action create[2014-02-25T07:56:24-08:00] INFO: Processing group[win-
clowns] action create (users::groups line 2)
[2014-02-25T07:56:24-08:00] INFO: group[win-clowns] created
  - create group[win-clowns]
[2014-02-25T07:56:25-08:00] INFO: Chef Run complete in 6.203709 seconds
```

# Exercise:

## Verify the users and groups exist on the node

### Windows

```
rn> net users
```

```
User accounts for \\C1263834251
-----
Administrator          bobo
frank                  Guest
The command completed successfully.
```

```
rn> net localgroup
```

```
Aliases for \\WIN-BBNT36Q0RB2
*Administrators
...
*win-clowns
*Performance Log Users
```

### Linux

```
rn> cat /etc/passwd
```

```
daisy:x:2003:0:Daisy Duke:/home/daisy:/bin/bash
rosco:x:2004:0:Rosco P. Coltrane:/home/roscos:/bin/bash
```

```
rn> cat /etc/group
```

```
clowns:x:3000:daisy,rosco
```

# Windows Exercise: Verify the users and groups exist on the node

```
rn> net users
```

```
User accounts for \\C1263834251
-----
Administrator          bobo
frank                  Guest
The command completed successfully.
```

```
rn> net localgroup
```

```
Aliases for \\WIN-BBNT36Q0RB2
*Administrators
...
*win-clowns
*Performance Log Users
```

## Linux Exercise: Verify the users and groups exist on the node

```
user@hostname:~$ cat /etc/passwd
```

```
sheldon:x:2003:0:Daisy Duke:/home/daisy:/bin/bash  
rosco:x:2004:0:Rosco Noseeeum:/home/rosco:/bin/bash
```

```
user@hostname:~$ cat /etc/group
```

```
clowns:x:3000:sheldon,rosco
```

# Let's review..

- 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 fancier)
- Between Data Bags and Node Attribute precedence, Chef provides a plethora of ways to inform the patterns you use to configure your infrastructure

# Questions

- What are Data Bags?
- 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 webserver recipe?
- Where would you go to find out more?

# Environments

Cookbook Version Constraints,  
Override Attributes

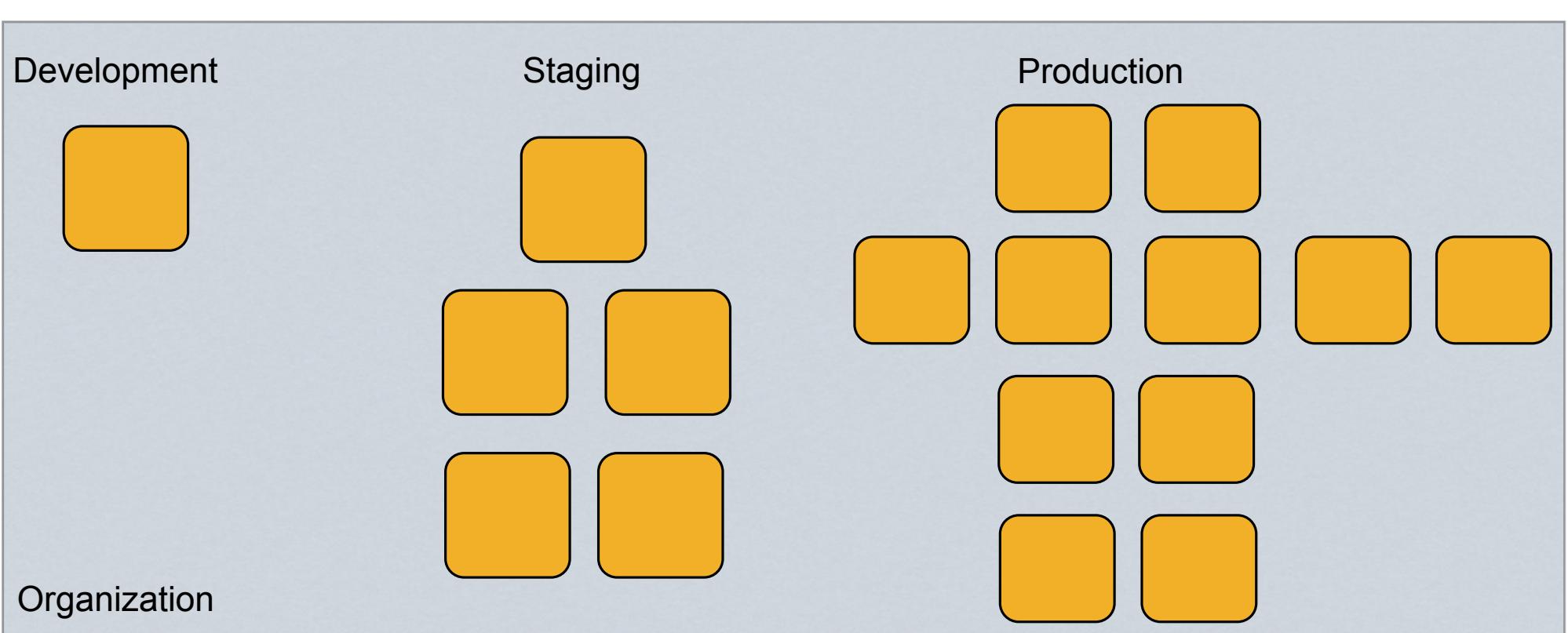
v2.1.0\_DUAL



# 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 how to set attributes in an environment

# Environments



# Environments

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

# Environment Best Practice

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

# 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

## Exercise: Make an environments directory on your workstation (in the chef-repo directory)

```
ws> cd chef-repo  
ws> mkdir environments
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	26-Feb-14 9:36 AM		environments

## Exercise: List the current environments

```
> knife environment list
```

```
_default
```

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

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

```
> knife cookbook show iis_demo
```

iis_demo	0.2.0	0.1.1	0.1.0
----------	-------	-------	-------

# Windows Exercise: Create a dev environment

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

```
name "windows-dev"
description "For developers!"
cookbook "iis_demo", "= 0.2.0"
```

**SAVE FILE!**

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

# Windows Exercise: Create a dev environment

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

```
name "windows-dev"
description "For developers!"
cookbook "iis_demo", "= 0.2.0"
```

**SAVE FILE!**

You need a space here

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

## Windows Exercise: Create the dev environment

```
> knife environment from file windows-dev.rb
```

Updated Environment windows-dev

## Windows Exercise: Show your Chef dev environment

```
> knife environment show windows-dev
```

```
chef_type:           environment
cookbook_versions:
  iis_demo: = 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 ‘linuxnode’
- Select dev from the ‘Environments’ drop-down list
- Click ‘Save’

The screenshot shows the Chef Manage web interface. The top navigation bar has tabs for Nodes, Reports, Policy, and Administration. The Nodes tab is active. Below the navigation is a search bar labeled 'Search Nodes...' with a magnifying glass icon. The main area displays a table titled 'Showing All Nodes' with columns: Node Name, Platform, FQDN, IP Address, Uptime, Last Check-In, Environment, and Actions. One row is highlighted for 'node1' (Windows, FQDN C1263834251, IP 10.160.33.236, 5 hours uptime, last checked in 16 hours ago, environment \_default). To the right of the table is a gear icon. Below the table, a modal dialog is open for 'Node: node1'. It has tabs for Details, Attributes, and Permissions. Under Details, it shows 'Last Check In: 16 Hours Ago' (2014-02-25 18:05:26) and 'Uptime: 5 Hours' (Since 2014-02-26 04:50:46 U). On the right side of the modal, there is a dropdown menu for 'Environment' currently set to 'dev'. A yellow callout box points to this dropdown with the text 'Node environment changed from \_default to dev.'. At the bottom of the modal are 'Cancel' and 'Save' buttons. The footer of the page includes copyright information (Copyright © 2012–2014 Chef, Inc.), help links ('Need help? If you have questions or are stuck, we are here to help.'), and navigation links ('Feedback', 'What's New?', 'About Chef Manage').

## Exercise: Re-converge the nodes

### Converge the Windows nodes

```
* template[/etc/hosts] action create[2014-02-26T01:57:44-08:00] INFO: Processing template[/etc/hosts] action create (hosts::default line 9)
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] backed up to /var/chef/backup/etc/hosts.chef-20140226015744.972901
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] updated file /etc/hosts

- update content in file /etc/hosts from 1f8bdf to 94a261
  --- /etc/hosts      2014-02-25 03:26:02.000000000 -0800
  +++ /home/chef/AppData/Local/Temp/2/chef-rendered-template20140226-3540-w4vntg
2014-02-26 01:57:44.000000000 -0800
  @@ -1,3 +1,4 @@
    #This file is managed by server at centos63.example.com
  +  0.0.0.0    nytimes.com
```

# Windows Exercise: Create a production environment

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

- Make sure the apache cookbook is set to version 0.1.0
- Set an override attribute for being in\_scope for PCI

```
name "win-production"
description "For Prods!"
cookbook "iis_demo", "= 0.1.0"
override_attributes({
  "datacenter" => {
    "location" => "Chile"
  }
})
```

**SAVE FILE!**

# Windows Exercise: Create a production environment

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

- Make sure the apache cookbook is set to version 0.1.0
- Set an override attribute for being in\_scope for PCI

```
name "win-production"
description "For Prods!"
cookbook "iis demo", "= 0.1.0"
override_attributes({
  "datacenter" => {
    "location" => "Chile"
  }
})
```

**SAVE FILE!**

# 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	

## Exercise: Create the production environment

### Upload the Production Environment

Updated Environment win-production

## Exercise: Move your node to new environment

```
ws> knife node environment set  
<nodename> <new environment>
```

```
ws> knife node environment set  
windowsnode win-production
```

```
1 items found

Node Name: windowsnode
Environment: win-production
FQDN:
IP:          10.160.201.90
Run List:
Roles:
Recipes:
Platform:    windows
```

## Exercise: Re-converge the nodes

### Converge the Windows nodes

```
* template[/etc/hosts] action create[2014-02-26T01:57:44-08:00] INFO: Processing template[/etc/hosts] action create (hosts::default line 9)
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] backed up to /var/chef/backup/etc/hosts.chef-20140226015744.972901
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] updated file /etc/hosts

- update content in file /etc/hosts from 1f8bdf to 94a261
  --- /etc/hosts      2014-02-25 03:26:02.000000000 -0800
  +++ /home/chef/AppData/Local/Temp/2/chef-rendered-template20140226-3540-w4vntg
2014-02-26 01:57:44.000000000 -0800
  @@ -1,3 +1,4 @@
    #This file is managed by server at centos63.example.com
  +  0.0.0.0    nytimes.com
```

## Linux Exercise:

### Use knife to show the available cookbook versions

```
> knife cookbook show apache
```

```
apache      0.2.0      0.1.1      0.1.0
```

# Linux Exercise: Create a dev environment

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

```
name "linux-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

# Linux Exercise: Create a dev environment

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

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

**SAVE FILE!**



You need a space here

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

## Linux Exercise: Create the dev environment

```
> knife environment from file linux-dev.rb
```

Updated Environment linux-dev

# Linux Exercise: Show your Chef dev environment

```
> knife environment show linux-dev
```

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

## Exercise: Move your node to new environment

```
ws> knife node environment set  
<nodename> <new environment>
```

```
Looking for an fqdn of node1 or name of node1  
Setting environment to linux-dev  
1 items found
```

```
Node Name:    linuxnode  
Environment:  linux-dev  
FQDN:  
IP:          10.160.201.90  
Run List:  
Roles:  
Recipes:  
Platform:    centos  
Tags:
```

# Exercise:

## Re-converge the nodes

```
* template[/etc/hosts] action create[2014-02-26T01:57:44-08:00] INFO: Processing template[/etc/hosts] action create (hosts::default line 9)
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] backed up to /var/chef/backup/etc/hosts.chef-20140226015744.972901
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] updated file /etc/hosts

- update content in file /etc/hosts from 1f8bdf to 94a261
  --- /etc/hosts      2014-02-25 03:26:02.000000000 -0800
  +++ /home/chef/AppData/Local/Temp/2/chef-rendered-template20140226-3540-w4vntg
2014-02-26 01:57:44.000000000 -0800
  @@ -1,3 +1,4 @@
    #This file is managed by server at centos63.example.com
  +  0.0.0.0    nytimes.com
```

# Linux Exercise: Create a production environment

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

- Make sure the apache cookbook is set to version 0.1.0
- Set an override attribute for being `in_scope` for PCI
- Use **knife node show linuxnode –a pci** to see how `pci` is currently set

```
Name "lin-production"
Description "For Prods!"
cookbook "apache", "= 0.1.0"
override_attributes(
  {"pci" => {"in_scope" => true}}  
)
```

## Exercise: Create the production environment

```
> knife environment from file lin-production.rb
```

Updated Environment lin-production

## Exercise: Move your node to new environment

```
ws> knife node environment set  
<nodename> <new environment>
```

```
ws> knife node environment set  
linuxnode lin-production
```

```
1 items found

Node Name:    linuxnode
Environment:  lin-production
FQDN:
IP:          10.160.201.90
Run List:
Roles:
Recipes:
Platform:    centos
```

## Exercise: Re-converge the nodes

### Converge the Linux nodes

```
* template[/etc/hosts] action create[2014-02-26T01:57:44-08:00] INFO: Processing template[/etc/hosts] action create (hosts::default line 9)
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] backed up to /var/chef/backup/etc/hosts.chef-20140226015744.972901
[2014-02-26T01:57:44-08:00] INFO: template[/etc/hosts] updated file /etc/hosts

- update content in file /etc/hosts from 1f8bdf to 94a261
  --- /etc/hosts      2014-02-25 03:26:02.000000000 -0800
  +++ /home/chef/AppData/Local/Temp/2/chef-rendered-template20140226-3540-w4vntg
2014-02-26 01:57:44.000000000 -0800
  @@ -1,3 +1,4 @@
    #This file is managed by server at centos63.example.com
  +  0.0.0.0    nytimes.com
```

# 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 webserver cookbook
- While the recipe applied fine, investigating the system will reveal the webserver 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.

# Example of contra resources

```
service "httpd" do
  action :stop
end

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

```
directory "/var/www" do
  action :delete
  recursive true
end

directory "/etc/httpd" do
  action :delete
  recursive true
end
```

# Questions?

- What is an Environment?
- How is an Environment different from an Organization?
- What is a cookbook version constraint?
- What is the best way to move nodes between environments

# Roles

Simplifying The Run List

v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - Explain what Roles are, and how they are used to provide clarity
  - Show a Role with Knife
  - Describe nested Roles
  - Bootstrap a node using Roles and Environments

# 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”

# Why do we use Roles?

- 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

## Pre-lab Exercise: Request additional node(s) from CloudShare (for use later)

- Provision new instance(s) now, for use at the end of this module (we'll use our existing nodes for the majority of this module)
- Return to the CloudShare URL from the Welcome email to get new instances, but use different email addresses
- The email address doesn't need to be a real address, you only need it to log into CloudShare (Eg. bogusWindowsNode@gmail.com)

# Windows Bug Regarding Role Names

1. If you see “recipe[roles]” in your run\_list, there is a bug in Microsoft Visual C runtime  
See: <http://sourceforge.net/p/mingw/bugs/1621/>
2. This causes a failure in Windows if there is a letter ‘s’ in the role name (i.e. role[webserver])
3. Workaround 1: use back-ticks to escape the single-quotes  
```role[webserver]```
4. Workaround 2: don’t use the letter ‘s’ in role name

# Windows Exercise: Create the webserver role

**OPEN IN EDITOR:** chef-repo\roles\win-webhead.rb

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

```
name "win-webhead"  
description "IIS Web Server"  
run_list "recipe[iis_demo]"
```

**SAVE FILE!**

## Windows Exercise: Upload the role to Chef Server

```
> knife role from file win-webhead.rb
```

Updated Role win-webhead!

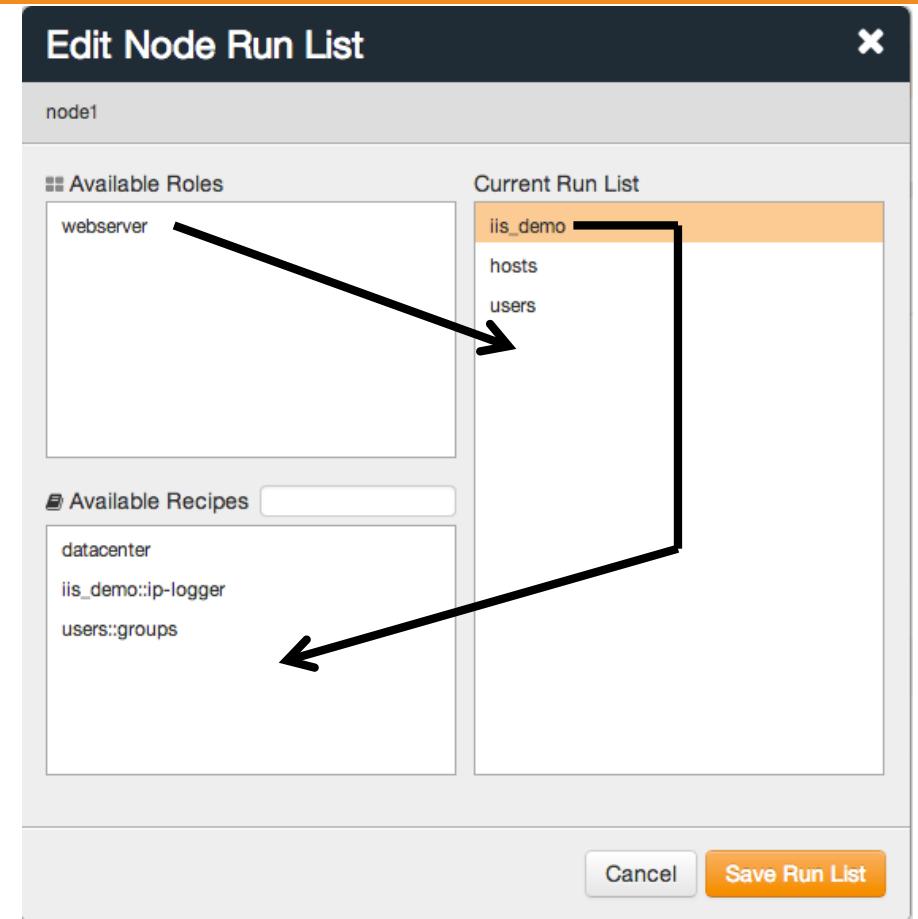
## Windows Exercise: Show the role with knife

```
> knife role show win-webhead
```

```
chef_type:          role
description:        IIS Web Server
env_run_lists:
json_class:        Chef::Role
name:              win-webhead
override_attributes:
run_list:          recipe[iis_demo]
```

# Windows Exercise: Replace `recipe[iis_demo]` with `role[win-webhead]` in the run list, using the GUI

- Click the ‘Nodes’ tab then select node ‘windowsnode’
- Click ‘Edit Run List’ from left navigation bar
- Drag ‘iis\_demo’ over from ‘Current Run List’ to ‘Available Recipes’
- Drag ‘win-webhead’ over from ‘Available Roles’ to the top of ‘Current Run List’
- Click ‘Save Run List’



# Windows Exercise: Re-run the Chef Client

```
rn> chef-client
```

```
[2014-02-25T09:44:14-08:00] INFO: Run List is [role[win-webhead],  
recipe[hosts], recipe[users]]  
[2014-02-25T09:44:14-08:00] INFO: Run List expands to [iis_demo, hosts,  
users]  
[2014-02-25T09:44:14-08:00] INFO: Starting Chef Run for node1  
[2014-02-25T09:44:14-08:00] INFO: Running start handlers  
[2014-02-25T09:44:14-08:00] INFO: Start handlers complete.  
resolving cookbooks for run list: ["iis_demo", "hosts", "users"]  
[2014-02-25T09:44:15-08:00] INFO: Loading cookbooks [datacenter, hosts,  
iis_demo, users]  
Synchronizing Cookbooks:  
  - iis_demo  
  - hosts
```

# Windows Exercise: Add an attribute to the role

**OPEN IN EDITOR:** roles\win-webhead.rb

- You can set default node attributes within a role.

```
name "win-webhead"
description "IIS Web Server"
run_list "recipe[iis_demo]"
default_attributes({
  "iis_demo" => {
    "sites" => {
      "admins" => {
        "port" => 8000
      }
    }
  }
})
```

**SAVE FILE!**

## Node Attributes that are hashes are merged

- The `iis_demo` cookbooks attribute file

```
Default["iis_demo"]["sites"]["clowns"] = { "port" => 80 }
Default["iis_demo"]["sites"]["bears"] = { "port" => 81 }
```

```
default_attributes({
  "iis_demo" => {
    "sites" => {
      "admins" => {
        "port" => 8000
      }
    }
  }
})
```

- While our role has...

## Windows Exercise: Verify the new homepage works

1. Upload the Role you just modified
2. Converge your Windows node **(on your existing node, not the node you just launched)**
3. Open a web browser  
Go to `http://<windows-hostname>:8000`  
You should see the “We love Admins” home page

## Windows Exercise: Search for the IIS sites attribute on all nodes with the role iis-webserver

```
> knife search node 'role:iis-webhead' -a iis_demo.sites
```

```
1 items found
```

```
windowsnode:
```

```
  iis_demo.sites:
```

```
    admins:
```

```
      port: 8000
```

```
    bears:
```

```
      port: 81
```

```
    clowns:
```

```
      port: 80
```

# Linux Exercise: Create the webserver role

**OPEN IN EDITOR:** chef-repo\roles\lin-webhead.rb

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

```
name "lin-webhead"  
description "Apache Web Server"  
run_list "recipe[apache]"
```

**SAVE FILE!**

## Linux Exercise: Upload the role to Chef Server

```
> knife role from file lin-webhead.rb
```

Updated Role lin-webhead!

## Linux Exercise: Show the role with knife

```
> knife role show lin-webhead
```

```
chef_type:          role
description:       Apache Web Server
env_run_lists:
json_class:        Chef::Role
name:              lin-webhead
override_attributes:
run_list:
```

## Lindows Exercise: Use knife to replace **recipe[apache]** with **role[lin-webhead]** in run list

First, remove the existing apache recipe from the run list

```
> knife node run_list remove linuxnode  
'recipe[apache]'
```

Next, add the lin-webhead role to the run list, placing it after 'recipe[users]'

```
> knife node run_list add linuxnode  
'role[lin-webhead]'  
-a 'recipe[users]'
```

# Linux Exercise: Re-run the Chef Client

```
rn> chef-client
```

```
[2014-02-25T09:44:14-08:00] INFO: Run List is [role[lin-webhead],  
recipe[motd], recipe[users]]  
[2014-02-25T09:44:14-08:00] INFO: Run List expands to [apache, motd, users]  
[2014-02-25T09:44:14-08:00] INFO: Starting Chef Run for node1  
[2014-02-25T09:44:14-08:00] INFO: Running start handlers  
[2014-02-25T09:44:14-08:00] INFO: Start handlers complete.  
resolving cookbooks for run list: ["apache", "motd", "users"]  
[2014-02-25T09:44:15-08:00] INFO: Loading cookbooks [pci, motd, apache,  
users]  
Synchronizing Cookbooks:  
  - apache  
  - motd
```

# Linux Exercise: Add an attribute to the role

**OPEN IN EDITOR:** roles/lin-webhead.rb

1. Modify role file
2. Upload Role
3. Converge Node
4. Browse to  
<hostname>:8000
5. See “We Love  
Admins” in  
homepage

```
name "lin-webhead"
description "Apache Web Server"
run_list "recipe[apache]"
default_attributes({
  "apache" => {
    "sites" => {
      "admins" => {
        "port" => 8000
      }
    }
  }
})
```

**SAVE FILE!**

## Node Attributes that are hashes are merged

- The `iis_demo` cookbooks attribute file

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

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

- While our role has...

## Linux Exercise: Search for the Apache sites attribute on all nodes with the role apache-webserver

```
> knife search node "role:lin-webhead" -a apache.sites
```

```
1 items found

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

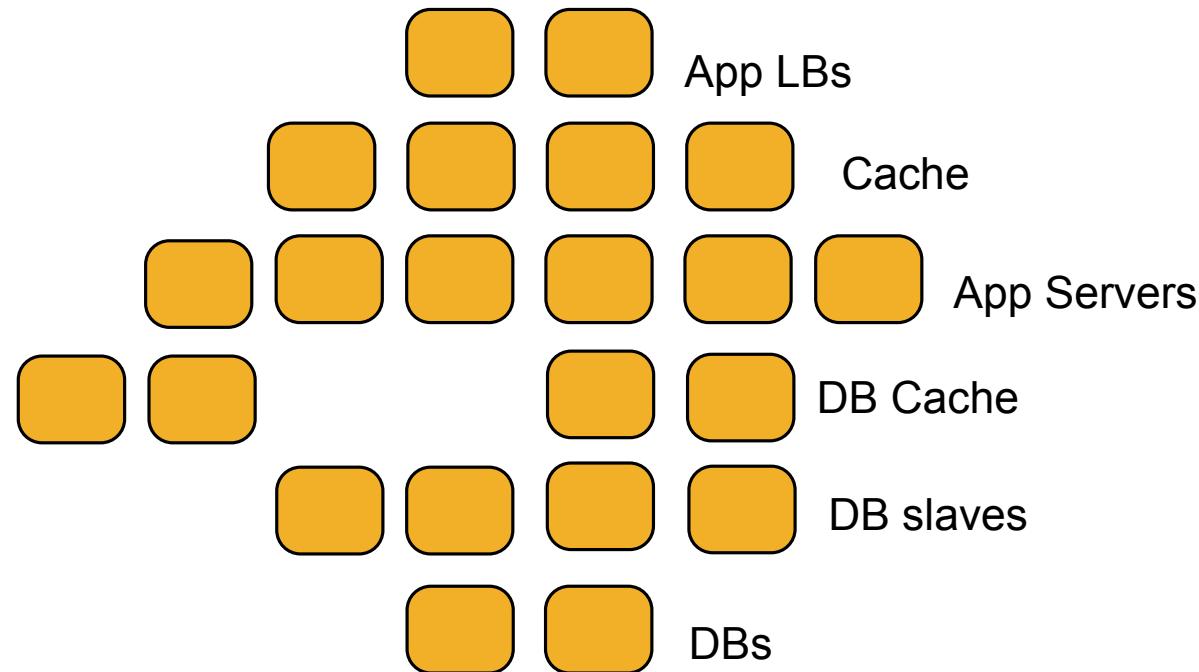
## Best Practice: Roles get default attributes

- 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

# When to use a Base Role



## Windows Exercise: Create the base role

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

```
name "base"
description "Base Windows Role"
run_list "recipe[del_val]", "recipe[hosts]",
"recipe[users]", "recipe[ip-logger"]"
```

**SAVE FILE!**

# Windows Exercise: Add the base role to the win-webhead role's run list

**OPEN IN EDITOR:** roles\win-webhead.rb

```
name "win-webhead"
description "Windows Web Server"
run_list ["role[base]", "recipe[iis_demo]"]
```

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

**SAVE FILE!**

## Exercise:

Set the run list to just the webhead role

- Remove all the entries in the node's **run list** other than the **webserver** role, then save and close.

## Exercise: Upload the roles and converge

- Upload the base role
- Upload the webserver role
- Converge the Node, look for the following:

```
[2014-02-25T10:05:36-08:00] INFO: Run List is [role/win-webhead]
[2014-02-25T10:05:36-08:00] INFO: Run List expands to [del-val, hosts, users,
ip-logger, iis_demo]
resolving cookbooks for run list: ["del-val", "hosts", "users", "ip-logger",
"iis_demo"]

[2014-02-25T10:05:37-08:00] INFO: Loading cookbooks [del_val, hosts, users,
iis_demo]
Synchronizing Cookbooks:
 - del_val
 - hosts
 - users
 - iis_demo
Compiling Cookbooks...
```

# Linux Exercise: Create the base role

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

```
name "base"
description "Base Linux Role"
run_list "recipe[del_val]", "recipe[motd]",
"recipe[users]", "recipe[ip-logger]"
```

**SAVE FILE!**

# Linux Exercise:

## Add the base role to the lin-webhead role's run list

**OPEN IN EDITOR:** roles/lin-webhead.rb

```
name "lin-webhead"
description "Linux Web Server"
run_list ["role[base]", "recipe[apache]"]
```

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

**SAVE FILE!**

## Exercise:

Set the run list to just the webhead role

- Remove all the entries in the node's **run list** other than the **webserver** role, then save and close.

## Exercise: Upload the roles and converge

- Upload the base role
- Upload the lin-webhead role
- Converge the Node, look for the following:

```
[2014-02-25T10:05:36-08:00] INFO: Run List is [role[lin-webhead]]
[2014-02-25T10:05:36-08:00] INFO: Run List expands to [del-val, motd, users,
ip-logger, apache]
resolving cookbooks for run list: ["del-val", "motd", "users", "ip-logger",
"apache"]
```

```
[2014-02-25T10:05:37-08:00] INFO: Loading cookbooks [del_val, motd, users, ip-
logger, apache]
Synchronizing Cookbooks:
 - del-val
 - motd
 - users
 - ip-logger
```

## 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

## Bootstrapping Nodes with Roles and Environments

- You can assign BOTH a Role and an Environment to a node when bootstrapping it
- This means the node goes from not being Chef-enabled, to existing as a complete Chef-enabled webserver (or database server, load balancer, etc.) in the Environment you specify
- Use `-r "role[rolename]"` and `-E "environmentname"`
- Note the capitalization of the `-r` and `-E` options

# "Bootstrap" with Role and Environment

To Bootstrap a **Windows** node

```
ws> knife bootstrap windows winrm <EXTERNAL_ADDRESS>
-x username -P password -N "windowsnode2"
-r `role[win-webhead]` -E "dev"
```

To Bootstrap a **Linux** node

```
ws> knife bootstrap <EXTERNAL_ADDRESS> -x username
-P password -N "linuxnode2"
-r `role[lin-webhead]` -E "dev"
```

**NOTE: If you are using more then one node for this course,  
name them differently or Chef Server will give you an error!**

# 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?
- How can you bootstrap using Roles and Environments?

# Using Community Cookbooks

Open Source: Saving you time!

v2.1.0\_DUAL



# Lesson Objectives

- After completing the lesson, you will be able to
  - Find, preview and download cookbooks from the Chef Community site
  - Use knife to work with the Community Site API

# The easy way...

- We've been writing cookbooks so far
- Hundreds already exist for a large number of use cases and purposes.
- Many (but only a fraction) are maintained by Chef.
- Think of it like RubyGems.org, CPAN.org, or other focused plugin-style distribution sites.
- **<https://supermarket.chef.io>**

# Exercise: Find and preview cookbooks on the site

**<https://supermarket.chef.io>**

The screenshot shows the homepage of the Chef Supermarket. At the top, there is a navigation bar with links for COOKBOOKS, CONTRIBUTORS, and TOOLS & PLUGINS. On the right side of the bar are links for CREATE ACCOUNT and SIGN IN. Below the navigation bar, a blue header banner displays the text "Supermarket is Now Open. Here is what you should know.". Below the banner is a search bar with the placeholder "Search Cookbooks" and a "GO" button. The main content area features a welcome message: "Welcome to Supermarket. Find, explore and view Chef cookbooks for all of your ops needs." Below this message are two buttons: "Sign Up For a Chef Account" and "Sign In With Your Chef Account".

## Explore

- [Browse Cookbooks](#)
- [Read the Chef Blog](#)

## Learn

- [Learn Chef](#)
- [Read the Chef Docs](#)

## Share

- [Share your cookbooks](#)
- [Chat on IRC at #chef on irc.freenode.net](#)

## Exercise: Search for a chef-client cookbook

The screenshot shows the Chef Supermarket search interface. At the top, there is a search bar with the placeholder text "Search for: chef-client". To the left of the search bar is a search input field containing "chef-client". To the right of the search bar is a blue "GO" button. Below the search bar, the results are displayed under the heading "25 Cookbooks". The first result is "chef-client", version 3.6.0, last updated July 30, 2014. The description for this cookbook is "Manages client.rb configuration and chef-client service". A call-to-action button at the bottom of this card says "cookbook 'chef-client', '~> 3.6.0'". The page also includes sections for "SUPPORTED PLATFORMS" (with icons for various operating systems) and metrics like "5560790 DOWNLOADS" and "125 FOLLOWERS".

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

- You can download cookbooks directly from the community site, but:
  - 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 includes a "cookbook site" plugin with some sub-commands:
  - search
  - show
  - download
  - ... and more!



# Download and use chef-client cookbook

v2.1.0\_DUAL



# Windows Versus Linux

- The following exercises will work the same on Windows or Linux
- You can work on either platform using the same commands
- We are using Windows for our Community Cookbook examples

## Exercise: Download the chef-client cookbook

```
ws> cd chef-repo  
ws> knife cookbook site download chef-client
```

Downloading chef-client from the cookbooks site at  
version 3.3.2 to /Users/YOU/chef-repo/chef-  
client-3.3.2.tar.gz

Cookbook saved: /Users/YOU/chef-repo/chef-  
client-3.3.2.tar.gz

## Exercise: Extract chef-client cookbook tarball

```
ws> tar -zxvf chef-client-3.9.0.tar.gz -C cookbooks
```

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

# 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 Administrator!

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

# 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

```
class ::Chef::Recipe
  include ::Opscode::ChefClient::Helpers
end

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::del_val]", "recipe[chef-client]", "recipe[hosts]",
  "recipe[users]", "recipe[ip-logger]"
```

**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 register 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',
  'winsw'
]
init_style = node["chef_client"]["init_style"]

# Services moved to recipes
if supported_init_styles.include? init_style
  include_recipe "chef-client::#{init_style}_service"
else
  log "Could not determine service init style, manual intervention required
  to start up the chef-client service."
end
```

## 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 files]
ERROR: Cookbook 'chef-client' depends on cookbook 'curl' version '>= 1.2.0',
ERROR: which current version uploaded and cannot be found on the
server.
```

**FAIL!**

# Exercise: Download the cron cookbook

```
> knife cookbook site download cron
```

```
Downloading cron from the cookbooks site at version  
1.2.8 to /Users/YOU/SOMEFOLDER/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-1.3.0.tar.gz -C cookbooks\
```

```
x cron/
x cron/CHANGELOG.md
x cron/CONTRIBUTING
x cron/Gemfile
x cron/LICENSE
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/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 files]
ERROR: Cookbook 'chef-client' depends on cookbook 'curl' version '>= 1.2.0',
ERROR: which current version uploaded and cannot be found on the
server.
```

**FAIL!**

# Exercise: Download the logrotate cookbook

```
> knife cookbook site download logrotate
```

```
Downloading logrotate from the cookbooks site at
version 1.5.0 to C:/Users/somefolder/
logrotate-1.5.0.tar.gz
Cookbook saved: C:/Users/somefolder/
logrotate-1.5.0.tar.gz
```

# Exercise: Extract logrotate cookbook tarball

```
> tar -zxvf logrotate-1.5.0.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 [ 1.3.0 ]
Uploaded 1 cookbook.
```

# Exercise: Upload the chef-client cookbook

```
> knife cookbook upload chef-client
```

```
Uploading chef-client [5.0.0]
ERROR: Cookbook 'chef-client' depends on cookbook 'node' version '>= 1.2.0',
ERROR: which is not currently being
server.
```

WIN!

# 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[hosts]", "recipe[users]"
```

**SAVE FILE!**

# Exercise: upload the base role

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

Updated Role base!

# Exercise: Re-run the Chef Client

```
rn> chef-client
```

```
Starting Chef Client, version 11.10.4
[2014-02-26T04:08:21-08:00] INFO: *** Chef 11.10.4 ***
[2014-02-26T04:08:21-08:00] INFO: Chef-client pid: 3412
[2014-02-26T04:08:36-08:00] INFO: Run List is [role[webserver], recipe[hosts], recipe[users]]
[2014-02-26T04:08:36-08:00] INFO: Run List expands to [chef-client::delete_validation, chef-client, hosts, users, iis_de
mo]
[2014-02-26T04:08:36-08:00] INFO: Starting Chef Run for node1
[2014-02-26T04:08:36-08:00] INFO: Running start handlers
[2014-02-26T04:08:36-08:00] INFO: Start handlers complete.
resolving cookbooks for run list: ["chef-client::delete_validation", "chef-client", "hosts", "users", "iis_demo"]
[2014-02-26T04:08:37-08:00] INFO: Loading cookbooks [chef-client, cron, datacenter, hosts, iis_demo, logrotate, users]
Synchronizing Cookbooks:
[2014-02-26T04:08:38-08:00] INFO: Storing updated cookbooks/chef-client/recipes/arch_service.rb in the cache. ipes/config.rb in the cache.
...
```

# Examine chef-client output

```
...
Recipe: chef-client::delete_validation
 * file[c:/chef/validation.pem] action delete[2014-02-26T04:08:55-08:00] INFO:
Processing file[c:/chef/validation.pem]
action delete (chef-client::delete_validation line 25)
[2014-02-26T04:08:55-08:00] INFO: file[c:/chef/validation.pem] deleted file at c:/
chef/validation.pem

 - delete file c:/chef/validation.pem
...
...
```

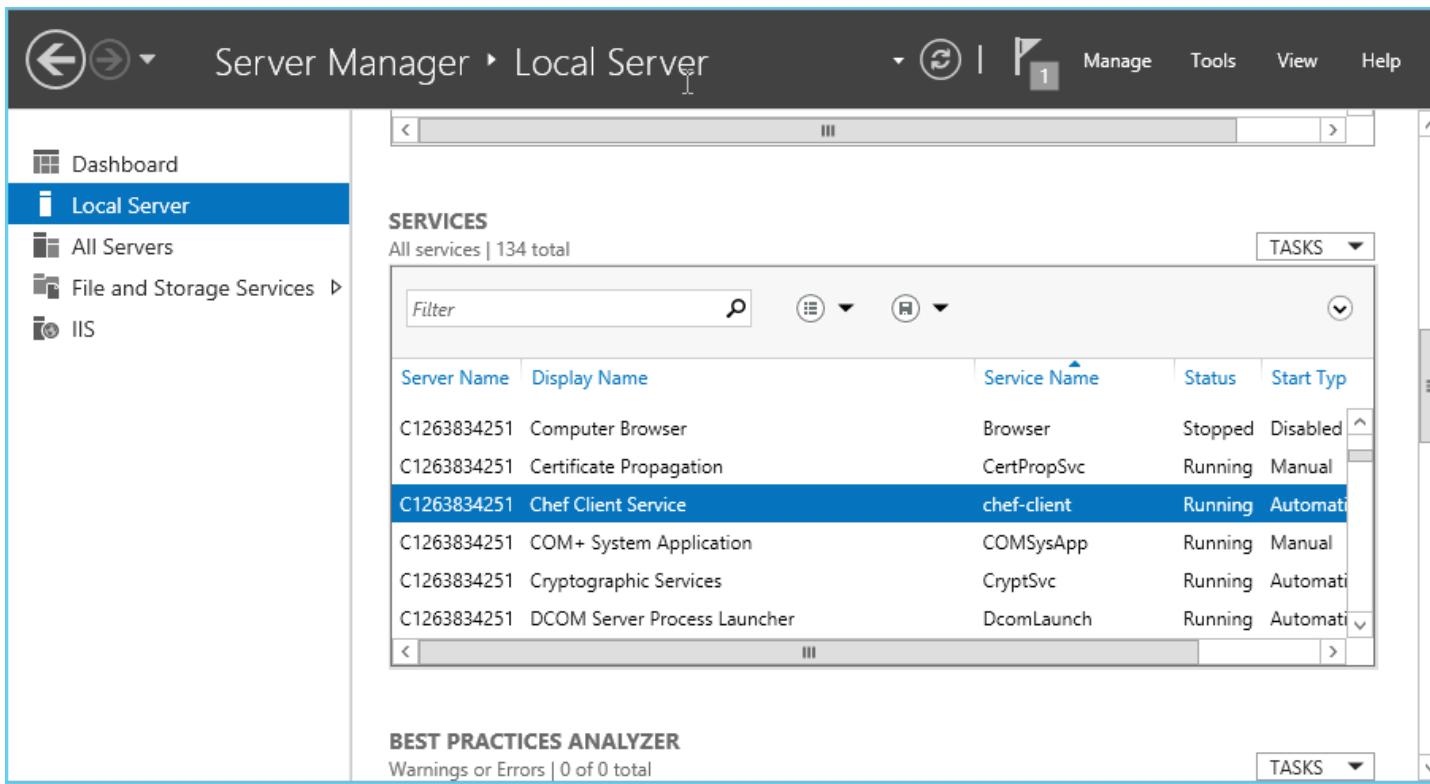
# Examine chef-client output

```
...
Recipe: chef-client::windows_service
...
  * execute[register-chef-service] action run[2014-02-26T04:08:55-08:00] INFO:
Processing execute[register-chef-service]
  action run (chef-client::windows_service line 34)
Service 'chef-client' has successfully been installed.
[2014-02-26T04:08:59-08:00] INFO: execute[register-chef-service] ran successfully

  - execute chef-service-manager -a install
  * service[chef-client] action enable[2014-02-26T04:08:59-08:00] INFO: Processing
service[chef-client] action enable (chef-client::windows_service line 39)
(up to date)
  * service[chef-client] action start[2014-02-26T04:08:59-08:00] INFO: Processing
service[chef-client] action start (chef-client::windows_service line 39)
[2014-02-26T04:09:07-08:00] INFO: service[chef-client] started

  - start service service[chef-client]
...
```

# Windows Exercise: Verify chef-client is running



The screenshot shows the Windows Server Manager interface for the Local Server. The left navigation pane is visible with options like Dashboard, Local Server (which is selected and highlighted in blue), All Servers, File and Storage Services, and IIS. The main content area is titled "SERVICES" and displays a list of all services on the server, indicating there are 134 total services.

The service list table has columns for Server Name, Display Name, Service Name, Status, and Start Type. One specific service, "Chef Client Service" (Display Name: "chef-client"), is highlighted in blue, indicating it is currently running. Other services listed include Computer Browser, Certificate Propagation, COM+ System Application, Cryptographic Services, and DCOM Server Process Launcher.

Server Name	Display Name	Service Name	Status	Start Type
C1263834251	Computer Browser	Browser	Stopped	Disabled
C1263834251	Certificate Propagation	CertPropSvc	Running	Manual
C1263834251	Chef Client Service	chef-client	Running	Automatic
C1263834251	COM+ System Application	COMSysApp	Running	Manual
C1263834251	Cryptographic Services	CryptSvc	Running	Automatic
C1263834251	DCOM Server Process Launcher	DcomLaunch	Running	Automatic

Below the service list, there is a section titled "BEST PRACTICES ANALYZER" which states "Warnings or Errors | 0 of 0 total".

## Linux Exercise: Verify chef-client is running

```
rn> 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
```

# Exercise: Verify chef-client is running

Windows

```
rn> get-service chef*
```

Status	Name
Display Name	
-----	-----
Running	chef-client
Chef Client Service	

Linux

```
rn> ps awux | grep chef-client
```

```
root 8933 0.3 2.2 130400 37816 ?
Sl 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 service, 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 on each run.

# Questions

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

# Further Resources

v2.1.0\_DUAL



# WORKING WITH CHEF, THE COMPANY

v2.1.0\_DUAL



# Chef Status

- Status for Chef related systems can be found by browsing to
  - <http://status.chef.io>, or
  - [https://twitter.com/opscode\\_status](https://twitter.com/opscode_status)

# How do I interact with Chef support?

- There are two ways to raise a ticket with Chef Support
  - Via the Web UI at <https://help2.chef.io>
  - By sending an email to [support@chef.io](mailto:support@chef.io)
- The Web UI allows you to submit any severity level ticket, however emailed tickets default to 'severity 3'
- The Web UI allows you to view previously submitted tickets for an org

# Dealing with Support

- When an org signs up for a support contract, the email domain of the primary user is used as a key, and anyone with an email in that domain can submit tickets
- E.g. [user1@domain.com](mailto:user1@domain.com) starts a support contract for an org, then [user2@domain.com](mailto:user2@domain.com) and [user3@domain.com](mailto:user3@domain.com) can also submit tickets on behalf of "domain.com" for that org
- Any user email address can be added explicitly to the Support Tool for your org by emailing [sales@chef.io](mailto:sales@chef.io) or submitting a ticket

# New releases

- Release Notes can be found here
  - <http://docs.chef.io>

# Further Resources

v2.1.0\_DUAL



# Further Resources

- <http://chef.io/>
- <http://supermarket.chef.io/>
- <http://docs.chef.io/>
- <http://learnchef.com>
- <http://lists.chef.io>
- <http://youtube.com/user/Opscode>
- [irc.freenode.net #chef, #chef-hacking, #learnchef](#)
- Twitter @chef.io #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

- Find your local DevOps or Chef meetup group!

Thank you for attending  
Chef Fundamentals  
for Windows and Linux!

v2.1.0\_DUAL



# Using Github

Using an external repo allows for  
team collaboration  
version control  
Chef code backups

Go to <http://github.com> to create a new Github account

v2.1.0\_DUAL

```
> git init
```

```
Initialized empty Git repository in  
/Users/your_name/chef-repo/.git/
```

v2.1.0

```
> git status
```

v2.1.0

```
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .berkshelf/
#       .chef/
#       .gitignore
#       Berksfile
#       README.md
#       Vagrantfile
#       chefignore
#       cookbooks/
#       roles/
nothing added to commit but untracked files present (use "git add" to track)
```

GETCHEF.COM



```
> git add .
```

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

v2.1.0

```
> git commit -m "add the starter kit from Chef"
```

```
[master (root-commit) 2af68fb] add the starter kit from Chef
13 files changed, 360 insertions(+)
create mode 100644 .berkshelf/config.json
create mode 100644 .chef/knife.rb
create mode 100644 .gitignore
create mode 100644 Berksfile
create mode 100644 README.md
create mode 100644 Vagrantfile
create mode 100644 chefignore
create mode 100644 cookbooks/starter/attributes/default.rb
create mode 100644 cookbooks/starter/files/default/sample.txt
create mode 100644 cookbooks/starter/metadata.rb
create mode 100644 cookbooks/starter/recipes/default.rb
create mode 100644 cookbooks/starter/templates/default/sample.erb
create mode 100644 roles/starter.rb
```

v2.1.0

```
> git push
```

```
Counting objects: 9, done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (5/5), done/  
Writing objects: 100% (5/5), 411 bytes | 0 bytes/s, done.  
Total 5 (delta 4), reused 0 (delta 0)  
To https://github.com/your_git_account/your_repo.git  
  0f555d4..3b7630e  master -> master
```

```
$
```

v2.1.0\_

```
> git pull
```

```
remote: Counting objects: 147, done.  
remote: Compressing objects: 100% (89/89), done.  
Receiving objects: 100% (145/145), 109.51 KiB | 0 bytes/s, done.  
Resolving deltas: 100% (31/31), done.  
From https://github.com/your_git_account/your_repo.git  
  2cf221..1585e9a master      -> origin/master  
Updating 2cf221..1585e9a  
Fast-forward  
  chef-client-3.6.0.tar.gz          | Bin 0 -> 33109 bytes  
  cookbooks/apache/CHANGELOG.md    | 13 +  
  cookbooks/apache/README.md       | 68 ++++  
v2.1.0_
```