

Chef Fundamentals - Windows

training@chef.io

Copyright (C) 2014 Chef Software, Inc.

v2.1.1_WIN

Introductions

v2.1.1_WIN

Instructor Introduction

- **Name:** Larry Eichenbaum
- **Current job role:** Automation Engineer
- **Previous job roles/background:** Automation Engineer for ‘other’ automation tools, Change/Release Mgr., QA Mgr., Technical Project Mgr., SysAdmin, farmer...
- **Experience with Chef/Config Management:** Well versed with variety of CM tools, process definition, etc.
- **Favorite Text Editor:** Sublime Text 2 and vi

Introduce Yourselves

- Name
- Current job role
- Previous job roles/background
- Experience with Chef and/or config management
- Favorite Text Editor

Course Objectives and Style

v2.1.1_WIN

Course Objectives

- Upon completion of this course you will be able to
 - Automate common infrastructure tasks with Chef
 - Describe Chef's architecture
 - Describe Chef's various tools
 - Apply Chef's primitives to solve your problems

How to learn Chef

- You bring the domain expertise about your business and problems
- Chef provides a framework for solving those problems
- Our job is to work together to teach you how to express solutions to your problems with Chef

Chef is a Language

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

Training is really a discussion

- We will be doing things the **hard way**
- We're going to do a **lot** of typing
- You can't be:
 - Absent
 - Late
 - Left Behind
- We will troubleshoot and fix bugs on the spot
- The result is you reaching fluency fast

Training is really a discussion

- I'll post objectives at the beginning of a section
- Ask questions when they come to you
- Ask for help when you need it
- You'll get the slides **after** class

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
 - We'll return to these items as time permits

Stages of Learning - Shuhari

- Shuhari - first learn, then detach, and finally transcend
 - shu - "obey" - traditional wisdom
 - ha - "detach" - break with tradition
 - ri - "separate" - transcend

守破離

Agenda

v2.1.1_WIN

Topics

- Overview of Chef
- Workstation Setup
- Organization Setup
- Test Node Setup
- Writing your first cookbook
- Dissecting your first Chef run
- Introducing the Node object
- Attributes, Templates, and Cookbook Dependencies

Topics

- Template Variables, Notifications, and Controlling Idempotency
- Data bags, search
- Roles
- Environments
- Using community cookbooks
- Further Resources

Breaks!

- We'll take a break between each section, or every hour, whichever comes first
- We'll obviously break for lunch :)

Overview of Chef

v2.1.1_WIN

Lesson Objectives

- After completing the lesson, you will be able to
 - Describe how Chef thinks about Infrastructure Automation
 - Define the following terms:
 - Node
 - Resource
 - Recipe
 - Cookbook
 - Run List
 - Roles
 - Search

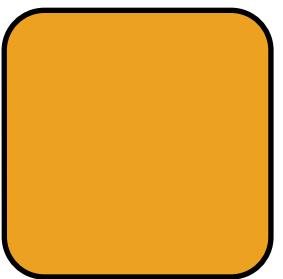
Complexity



Items of Manipulation (Resources)

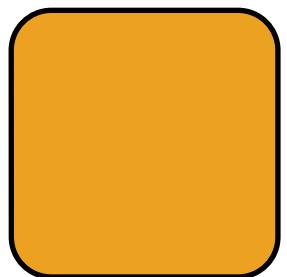
- Networking
- Files
- Directories
- Symlinks
- Mounts
- Registry Key
- Powershell Script
- Users
- Groups
- Packages
- Services
- Filesystems

A tale of growth...

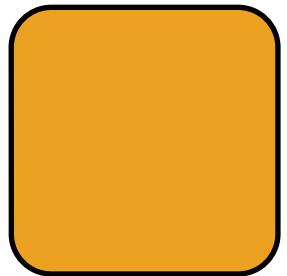


Application

Add a database

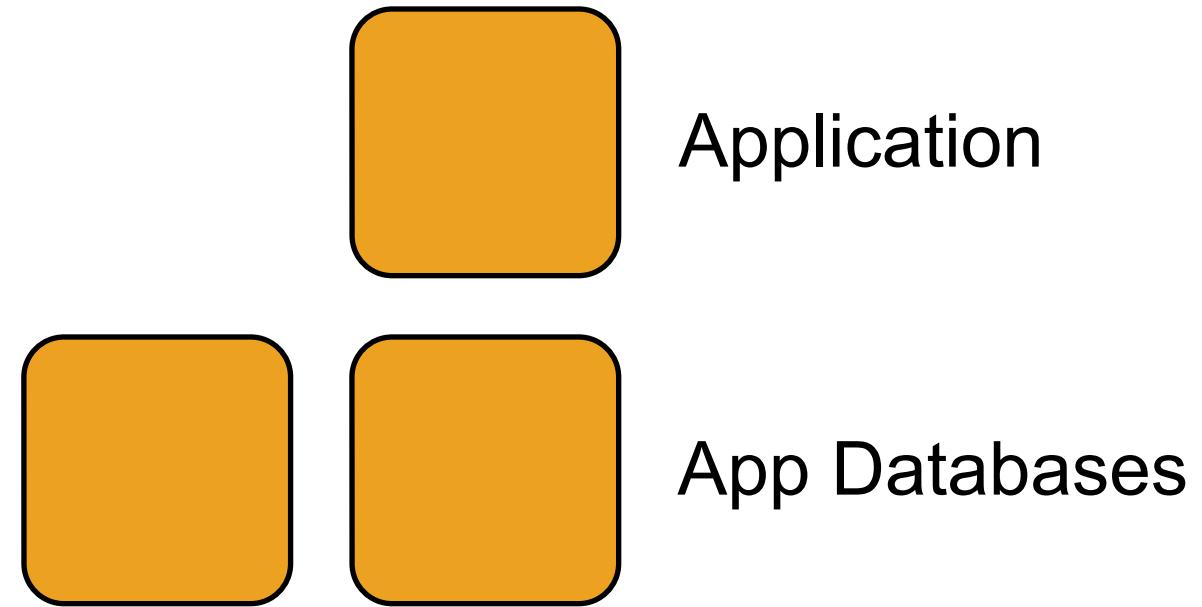


Application

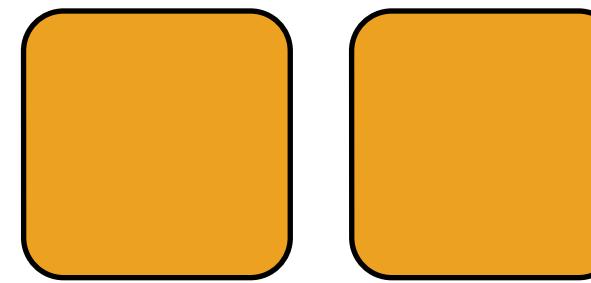


Application Database

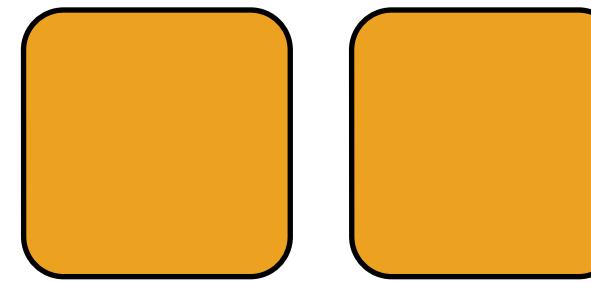
Make database redundant



Application server redundancy

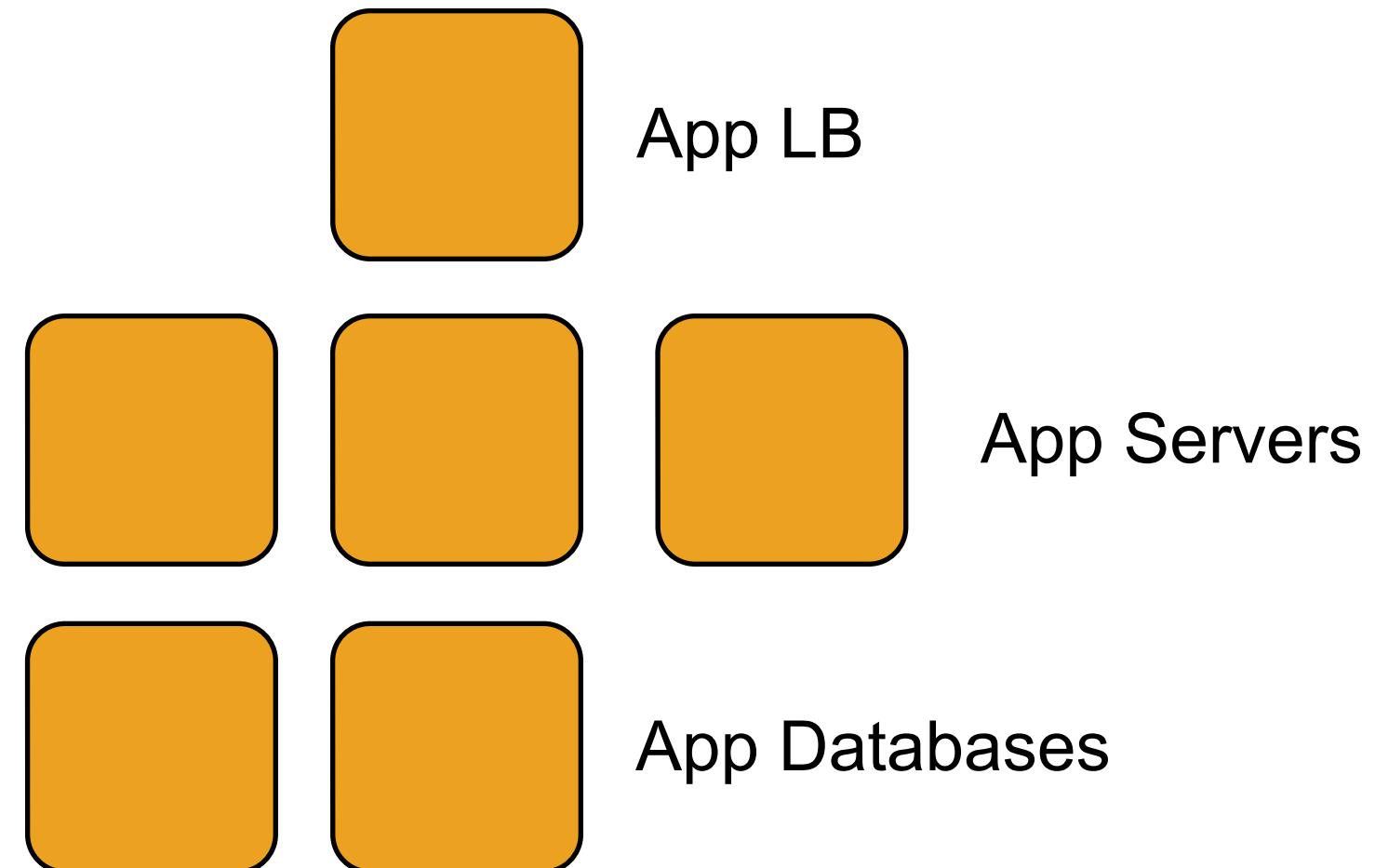


App Servers

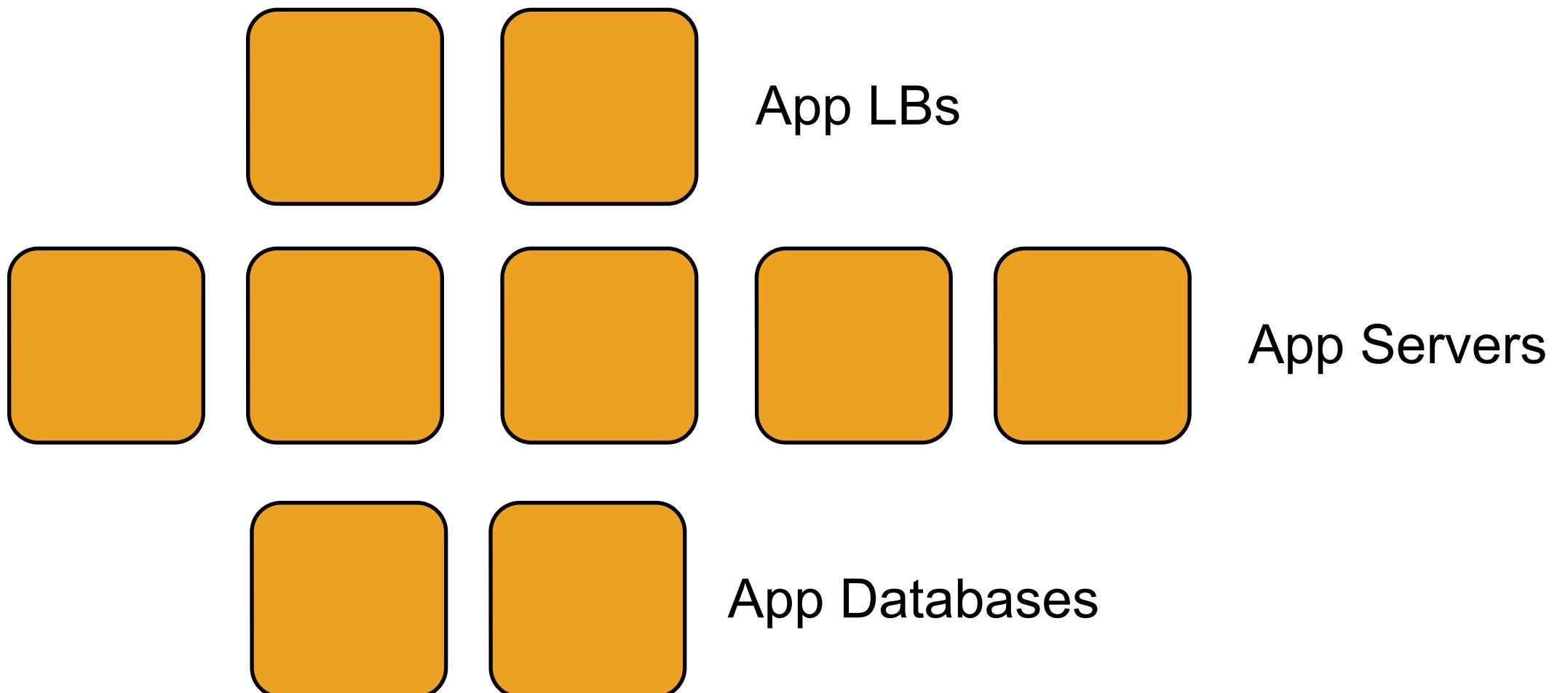


App Databases

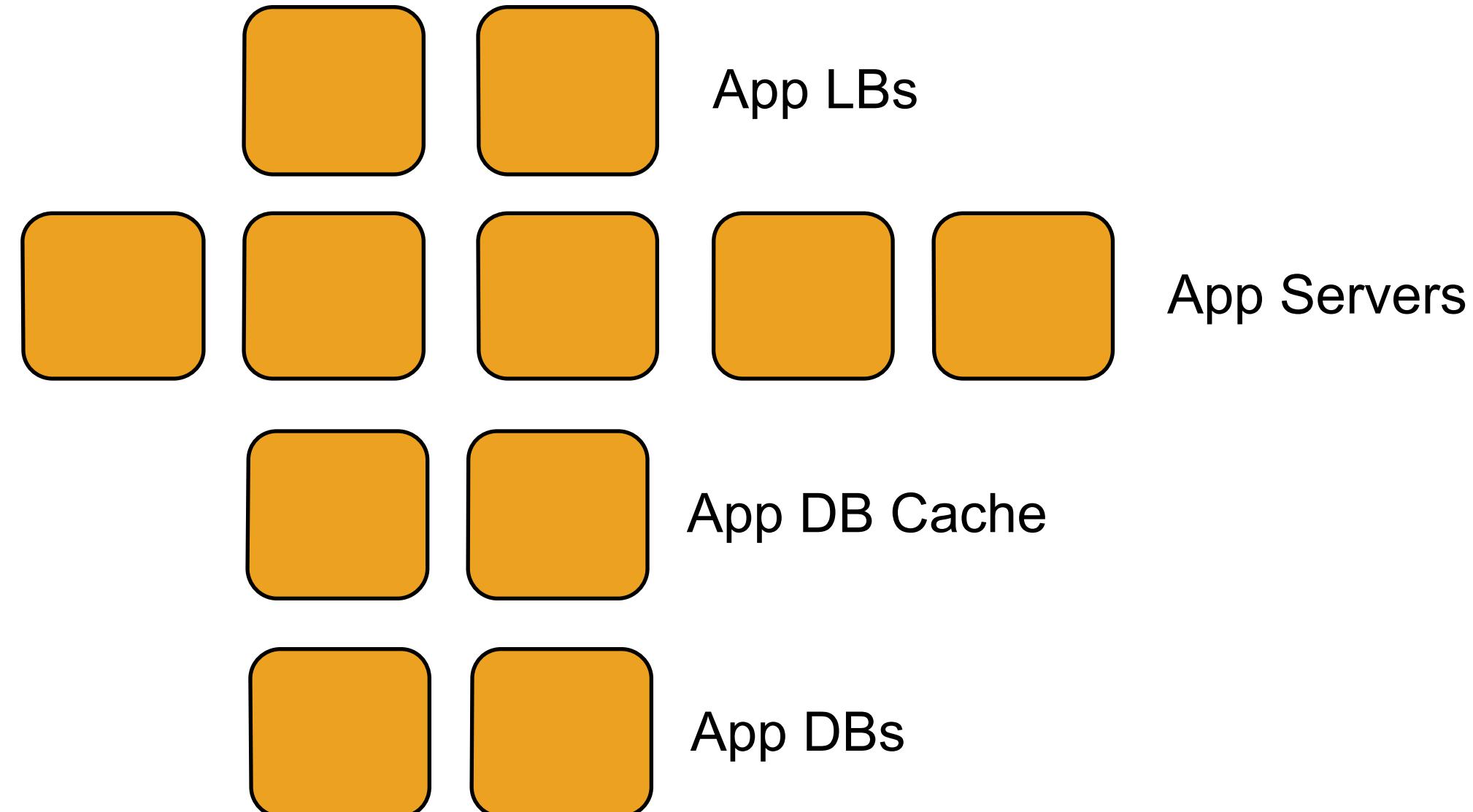
Add a load balancer



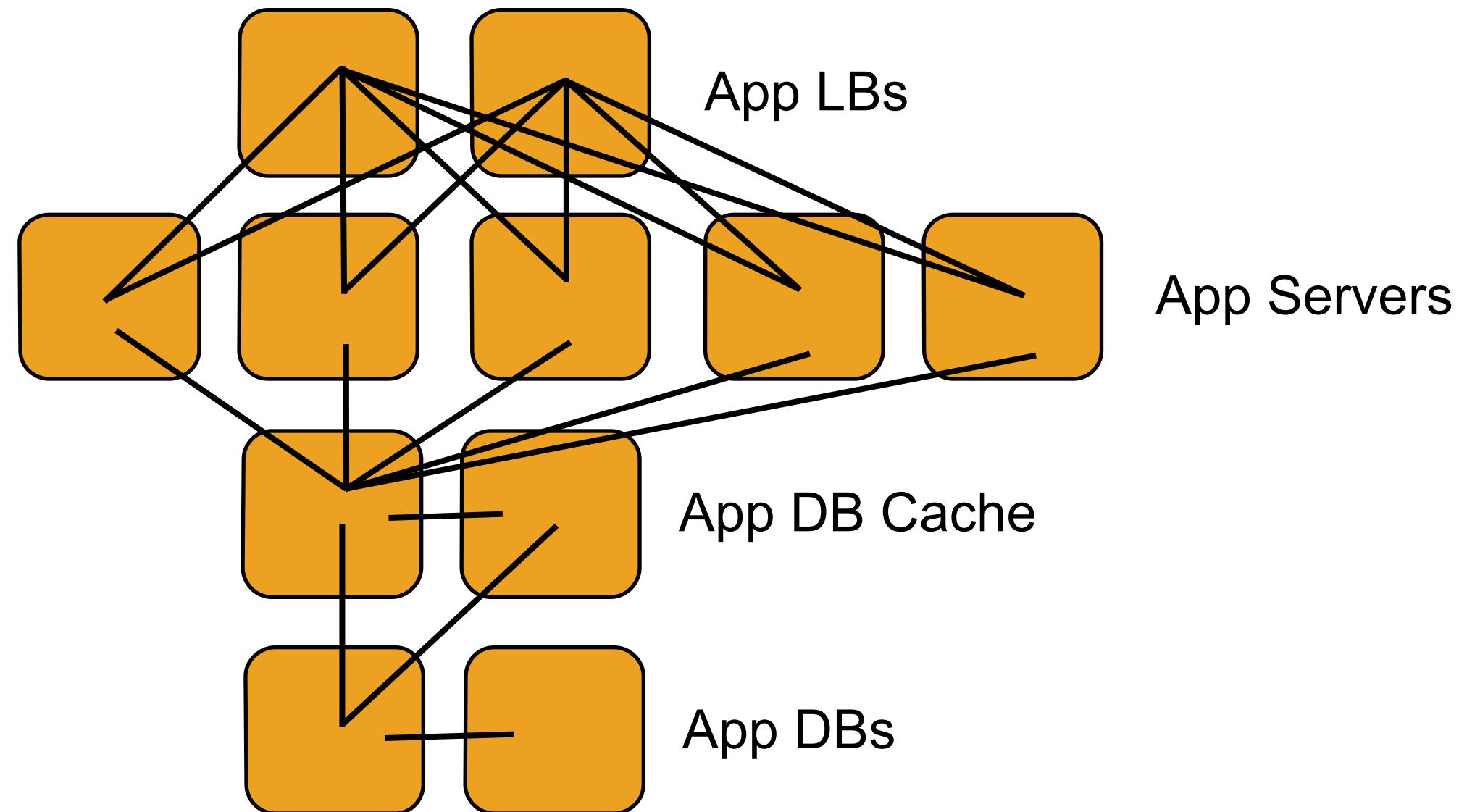
Webscale!



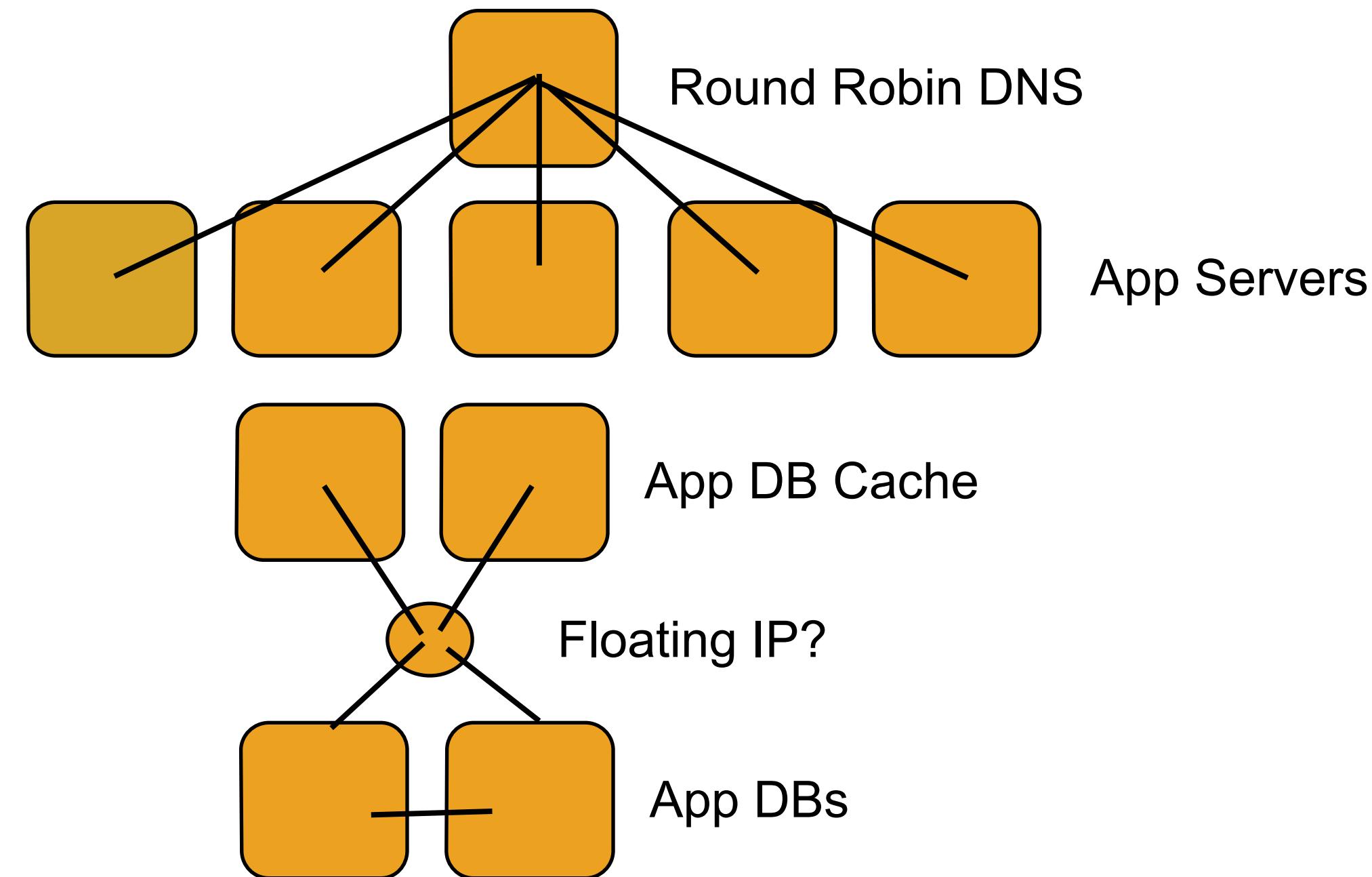
Now we need a caching layer



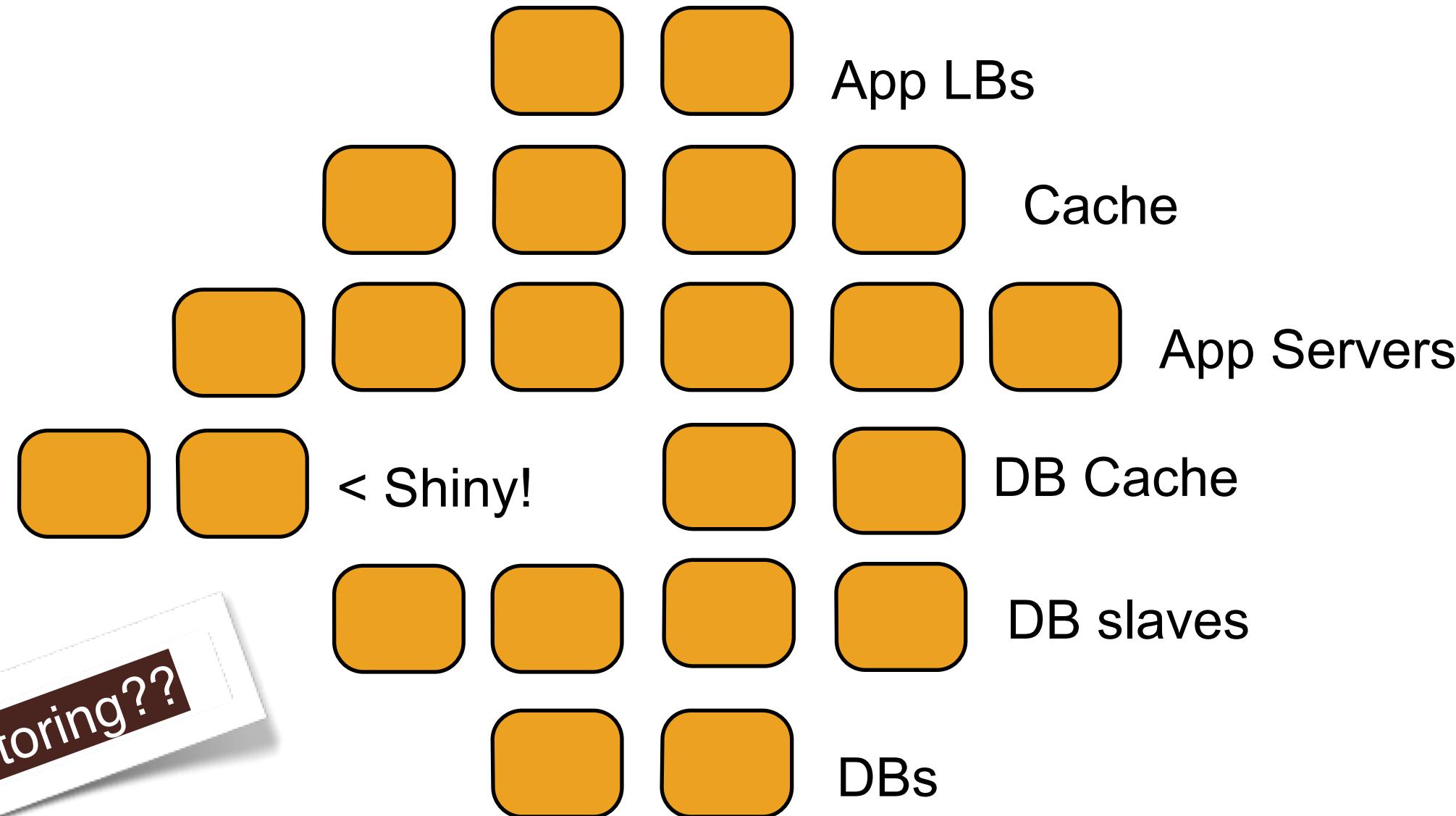
Infrastructure has a Topology



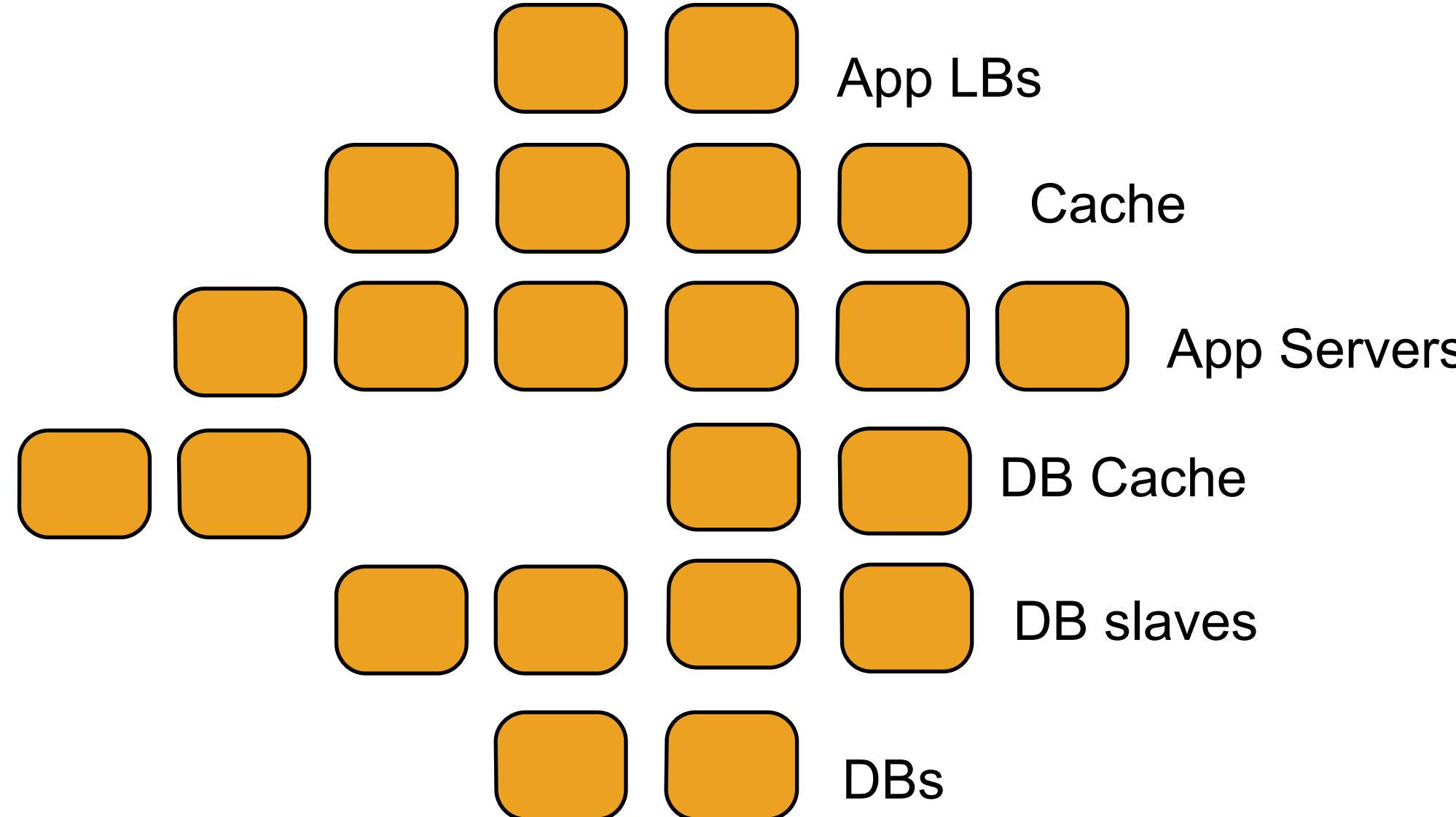
Your Infrastructure is a Snowflake



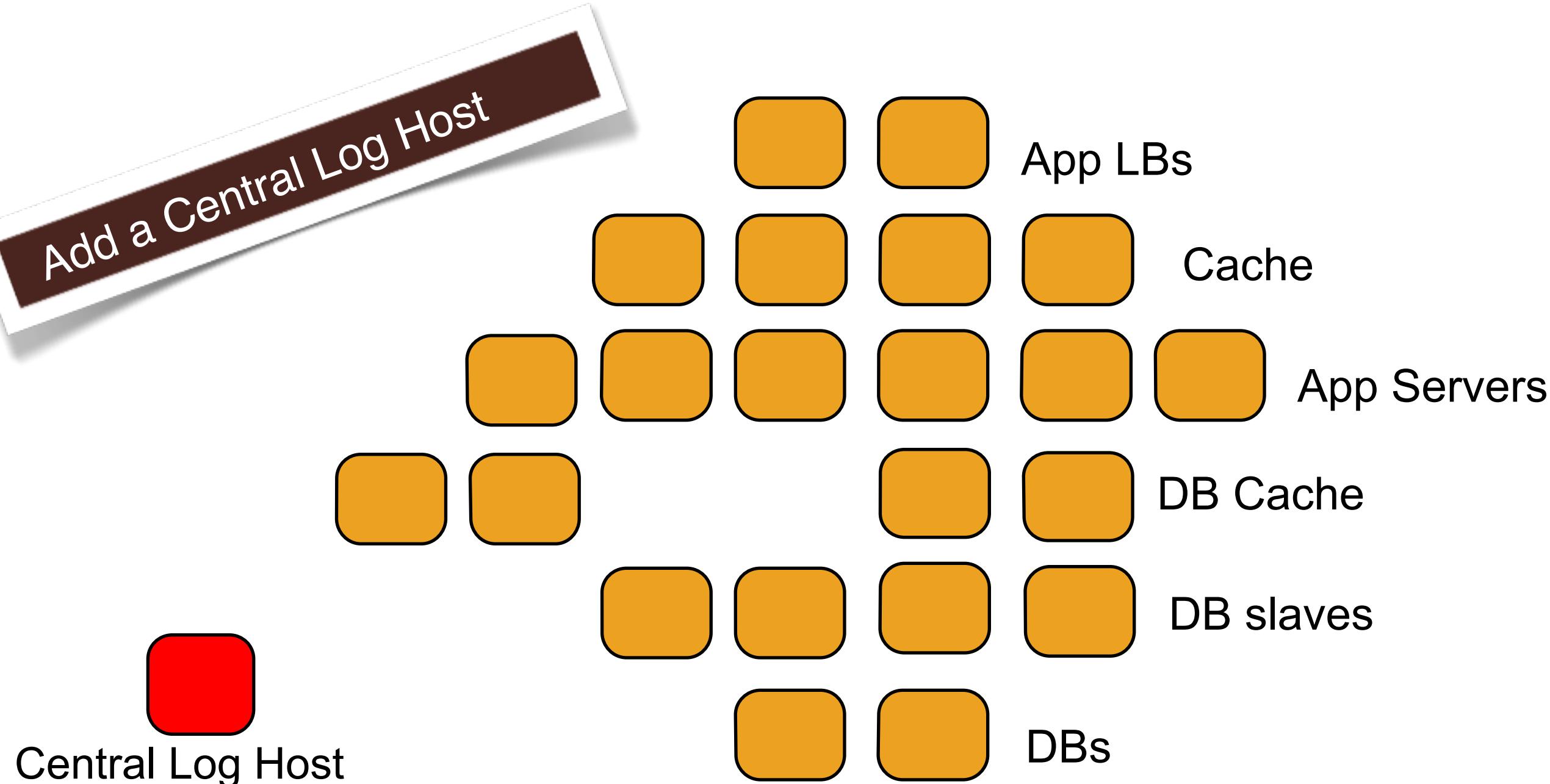
Complexity Increases Quickly



...and change happens!



...and change happens!



...and change happens!



Chef Solves This Problem

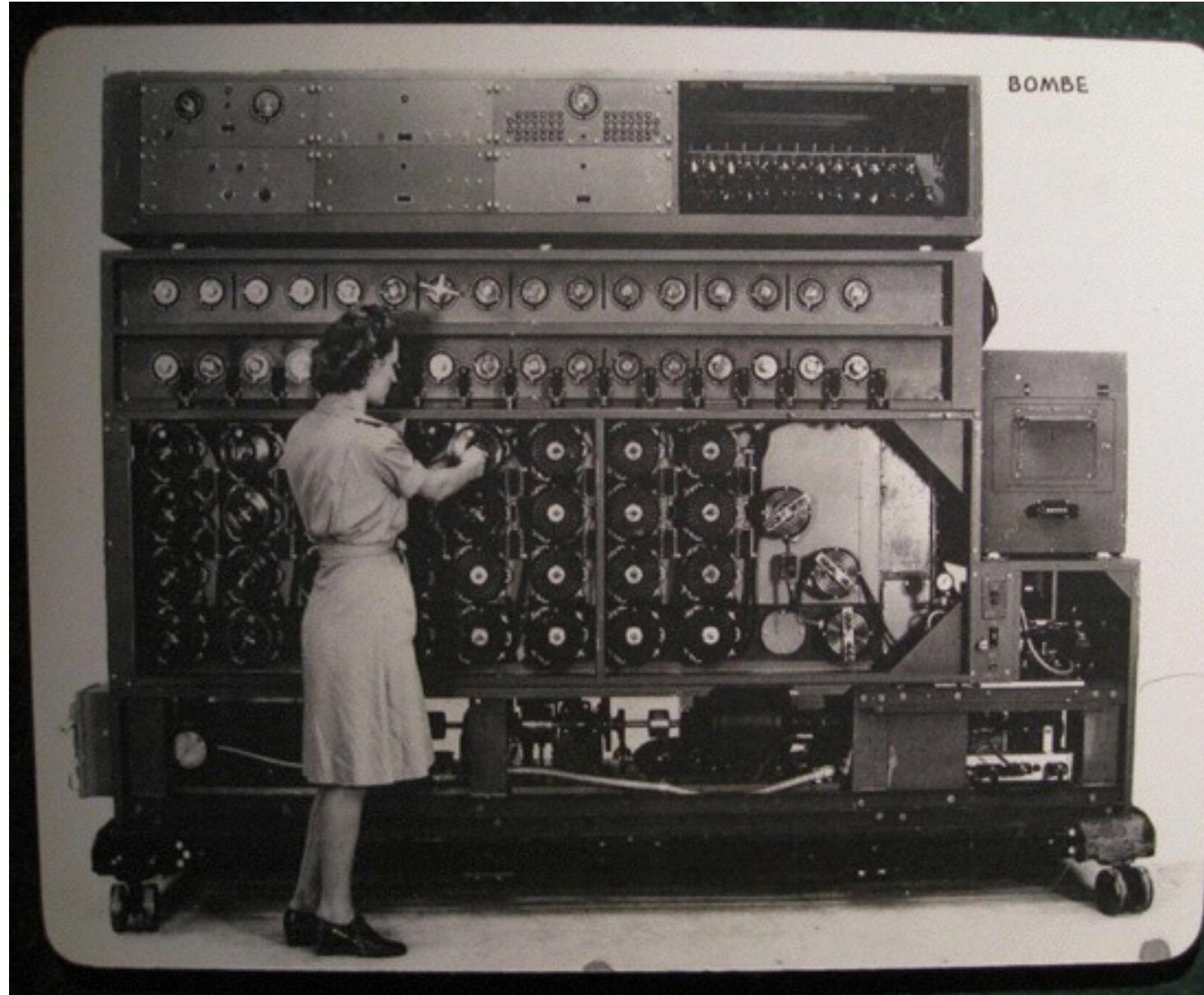


- But you already guessed that, didn't you?

CHEFTM
GETCHEF.COM

Chef is Infrastructure as Code

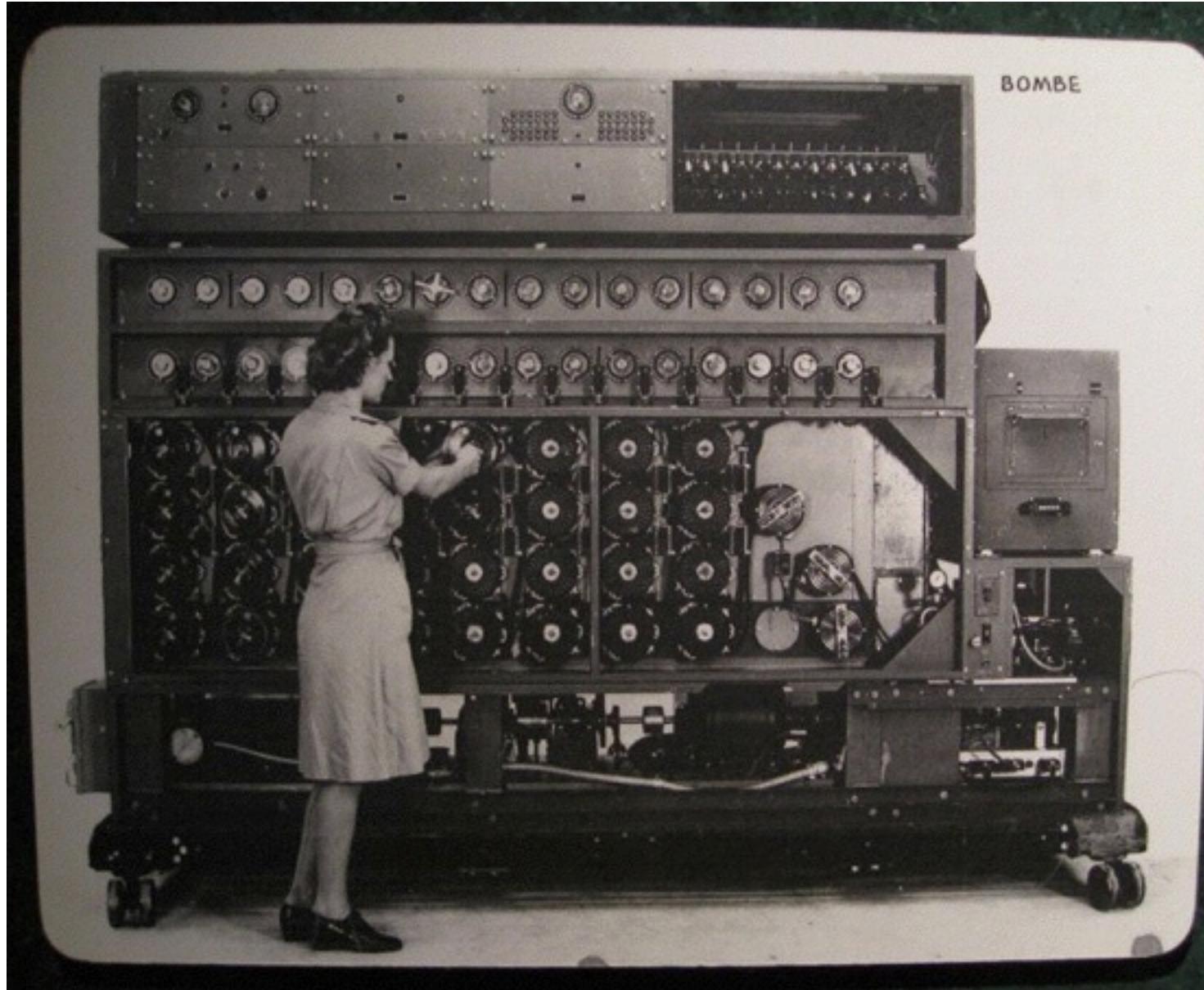
- Programmatically provision and configure components



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

Chef is Infrastructure as Code

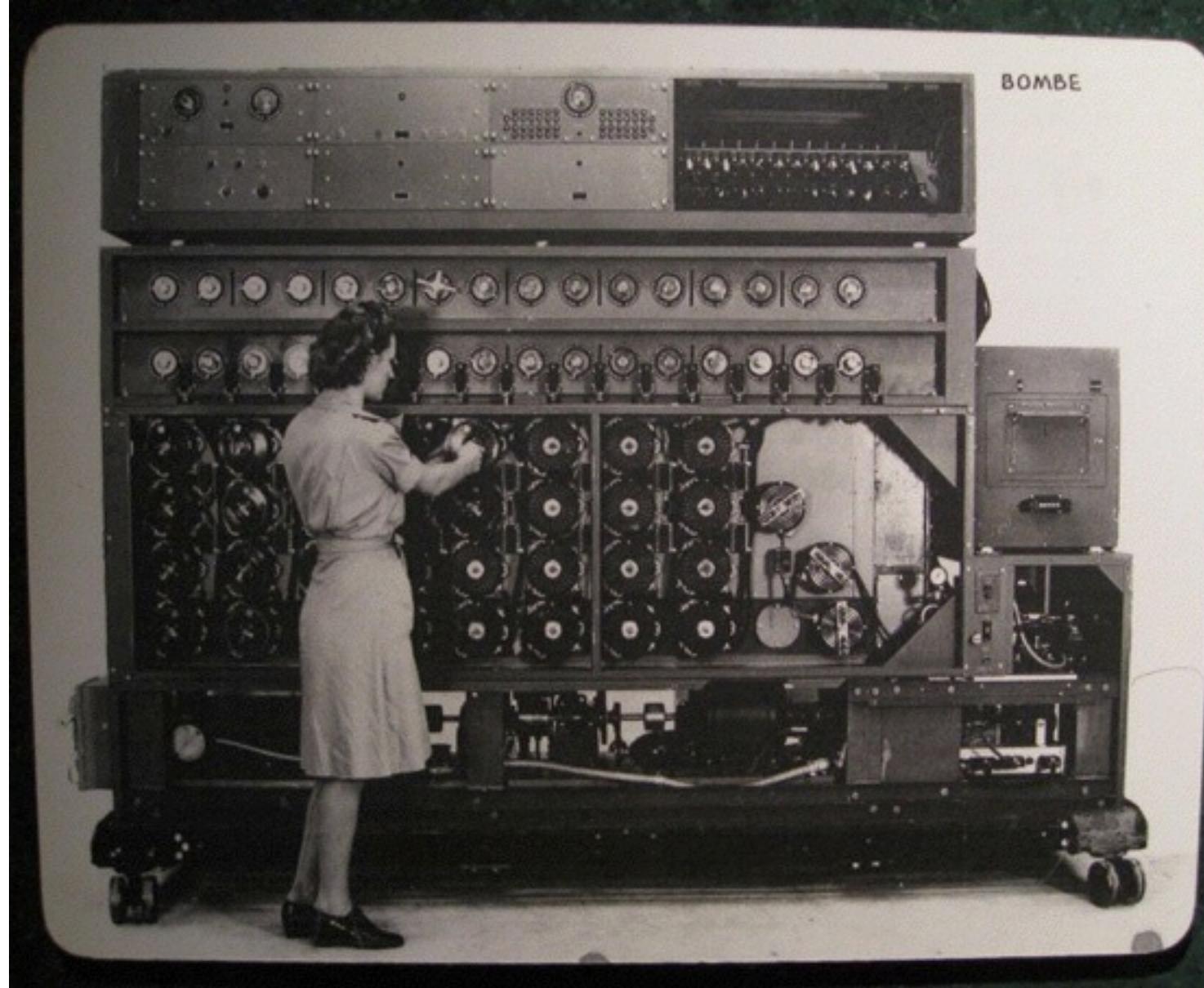
- Treat like any other code base



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

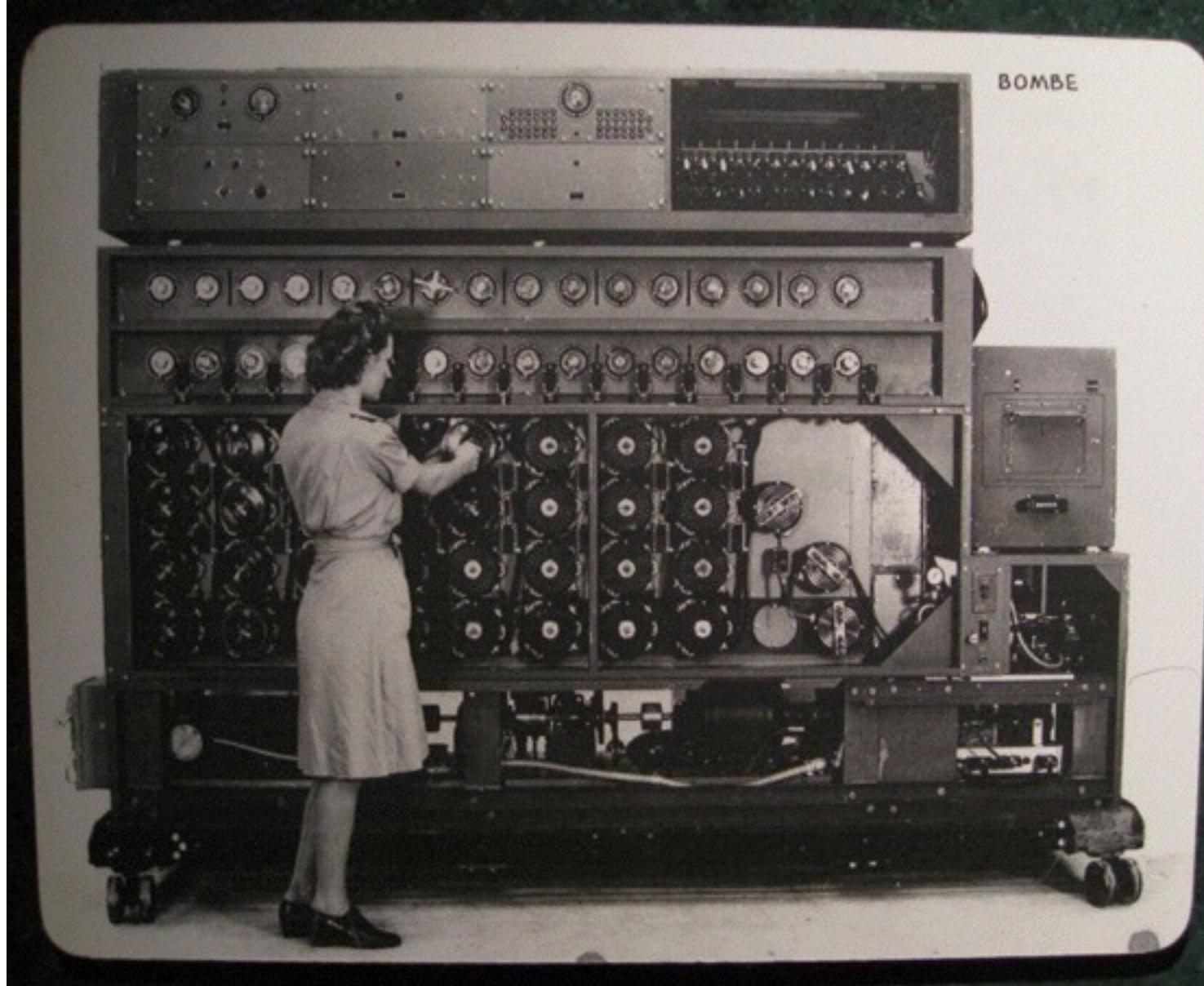
Chef is Infrastructure as Code

- Reconstruct business from **code repository, data backup, and compute resources**



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

Chef is Infrastructure as Code



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

Configuration Code

- Chef ensures each Node complies with the policy
- Policy is determined by the configurations in each Node's run list
- Reduce management complexity through abstraction
- Store the configuration of your infrastructure in version control

Declarative Interface to Resources

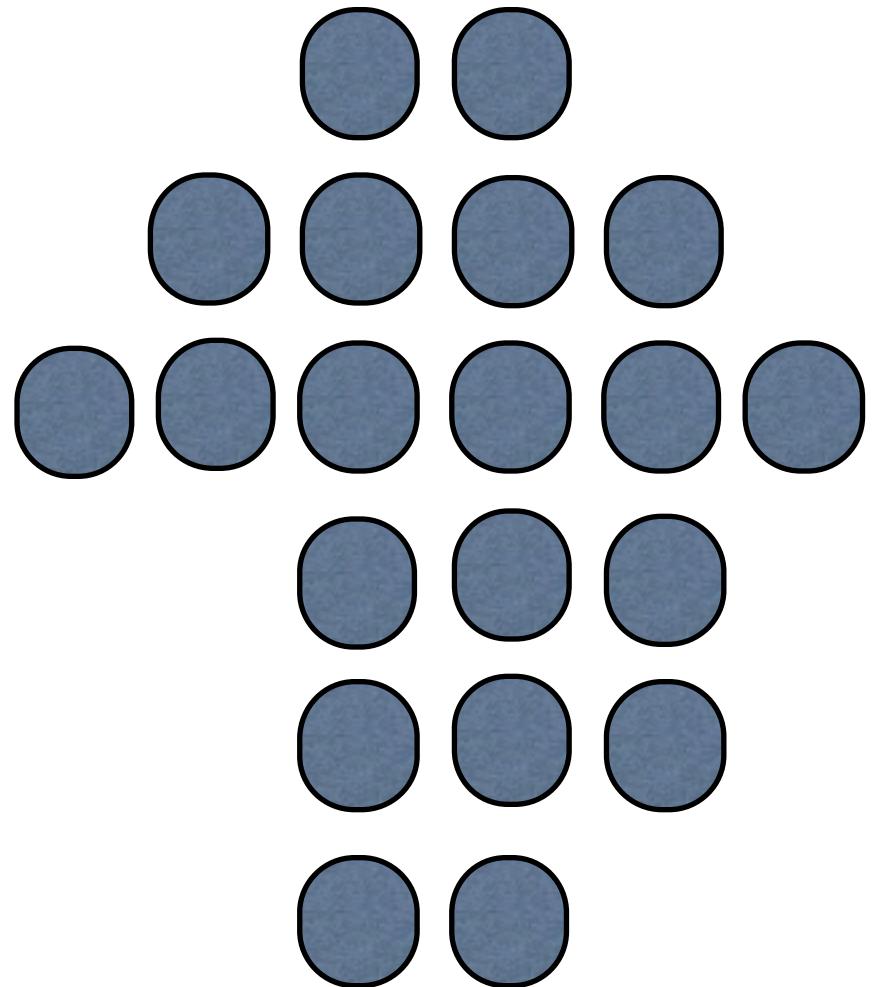
- You define the policy in your Chef configuration
- Your policy states what state each resource should be in, but not how to get there
- Chef-client will pull the policy from the Chef Server and enforce the policy on the Node

Managing Complexity

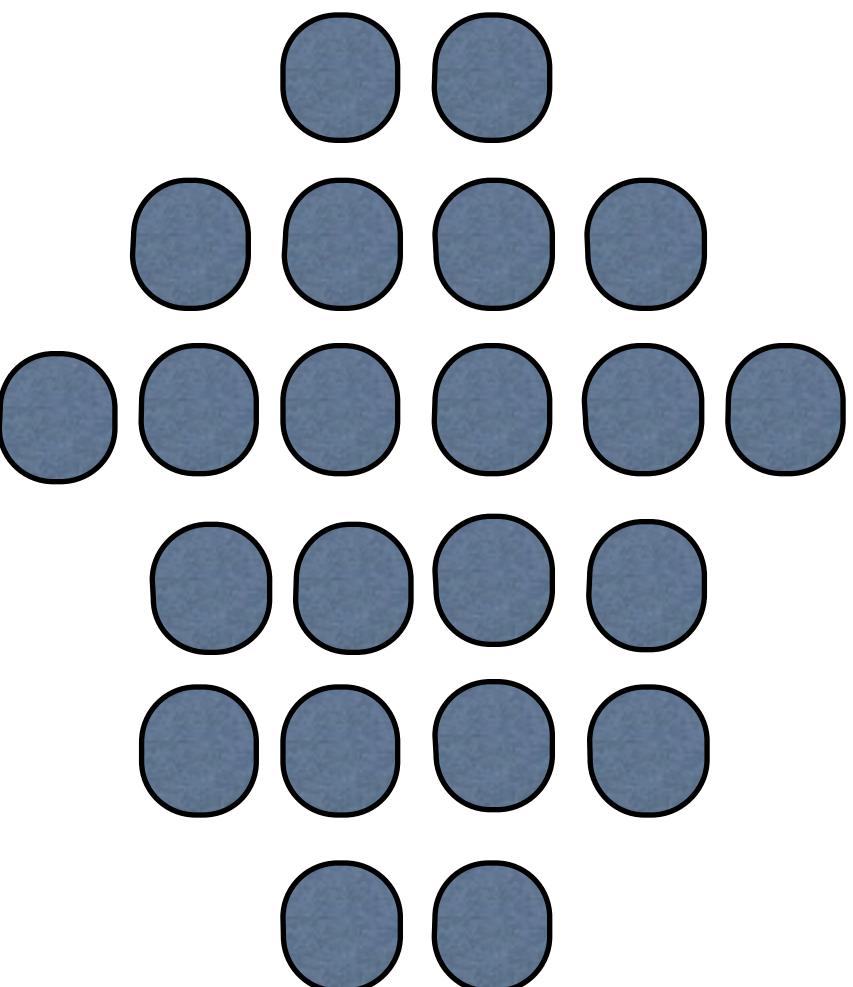
- Organizations
- Environments
- Roles
- Nodes
- Recipes
- Cookbooks
- Search

Organizations

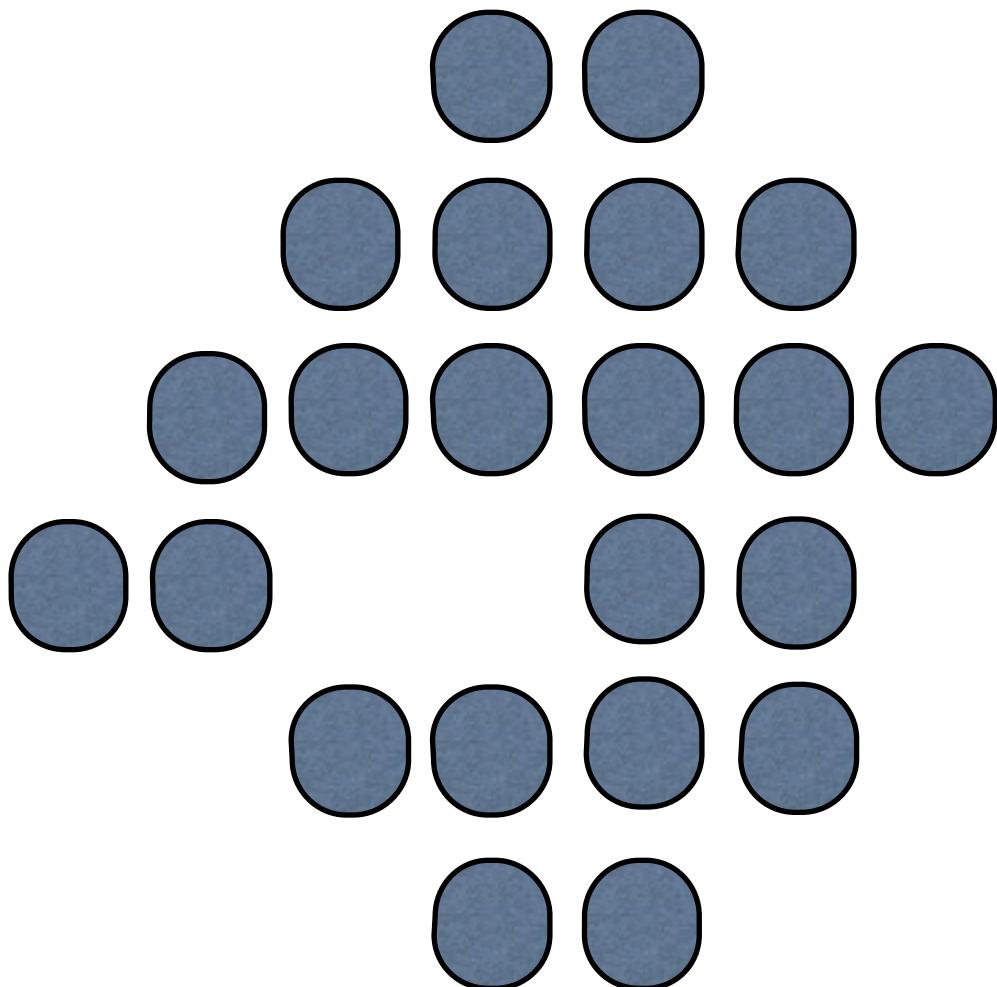
My Infrastructure



Your Infrastructure



Their Infrastructure

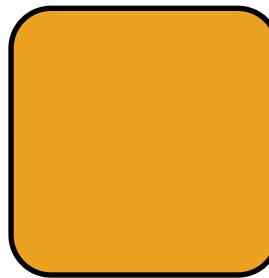


Organizations

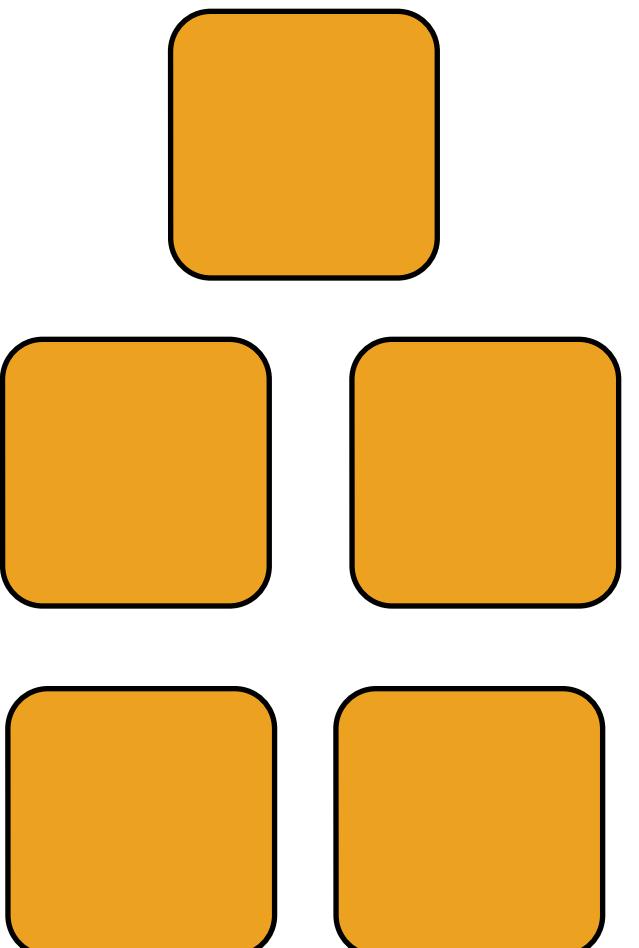
- Completely independent tenants of Enterprise Chef
- Share nothing with other organizations
- May represent different
 - Companies
 - Business Units
 - Departments

Environments

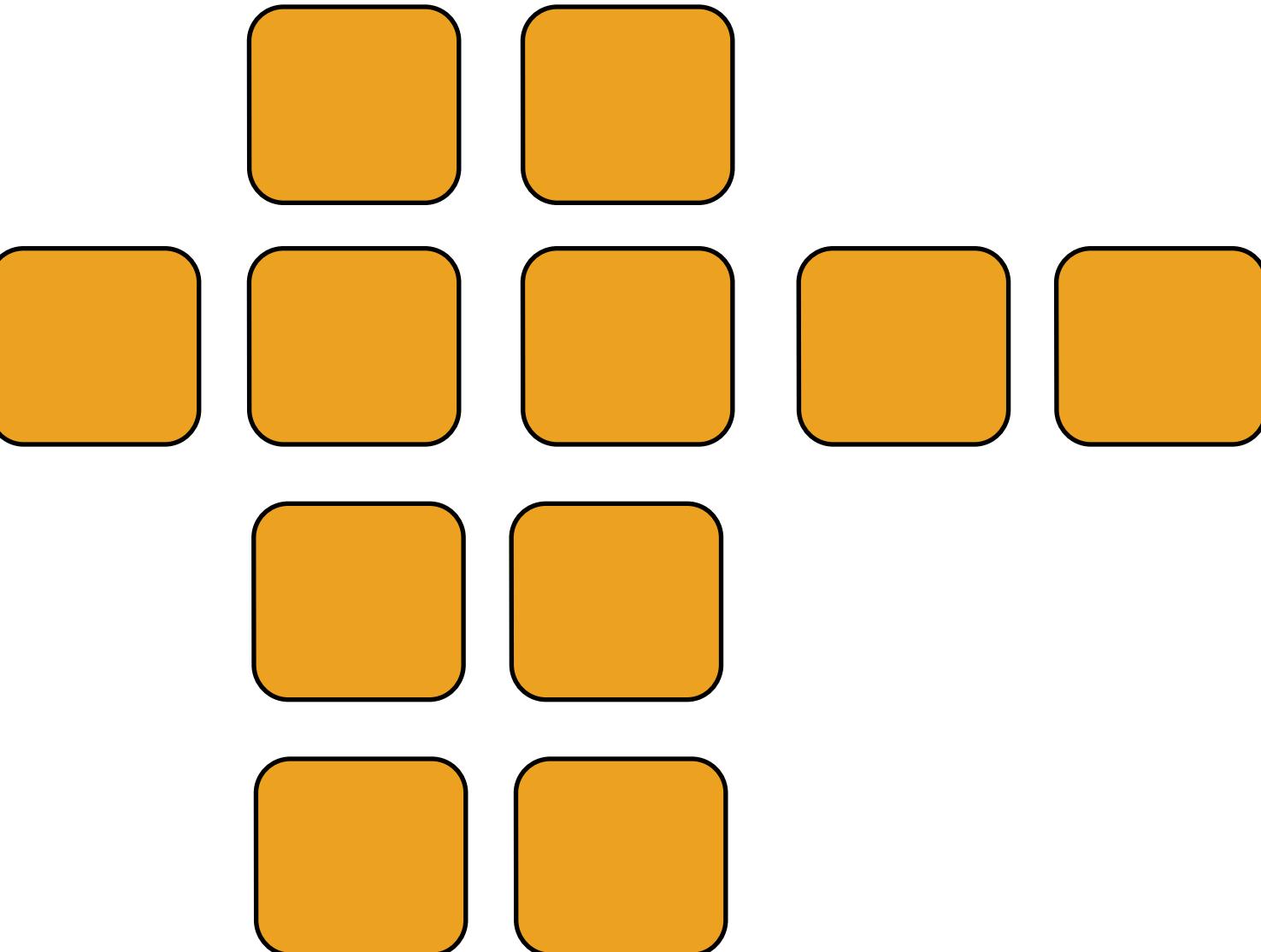
Development



Staging



Production



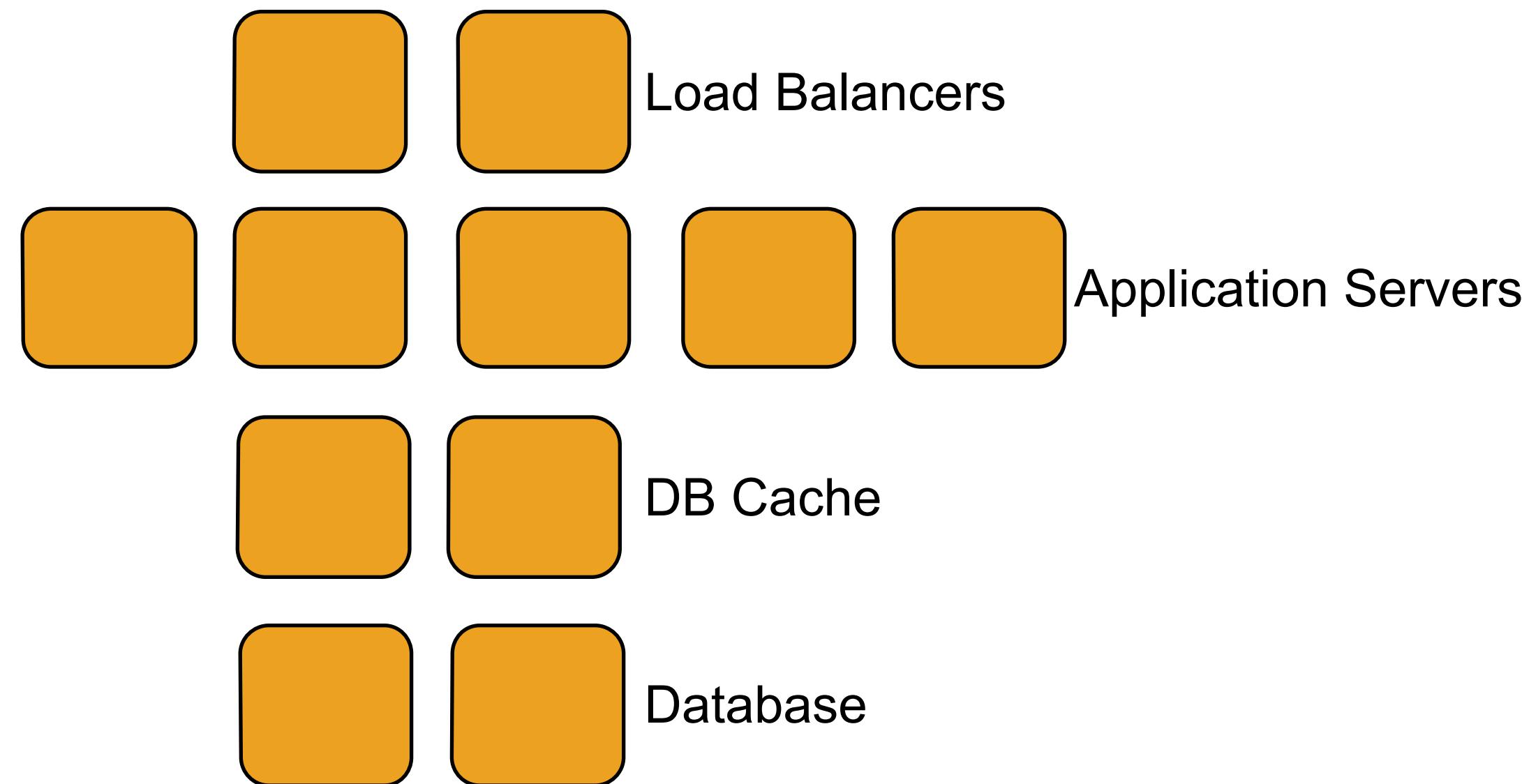
Environments

- Environments reflect your patterns and workflow, and can be used to model the life-stages of your applications
 - Development
 - Test
 - Staging
 - Production
 - etc.
- Every Organization starts with a single environment!

Environments Define Policy

- Environments may include data attributes necessary for configuring your infrastructure, e.g.
 - The URL of your payment service's API
 - The location of your package repository
 - The version of the Chef configuration files that should be used

Roles



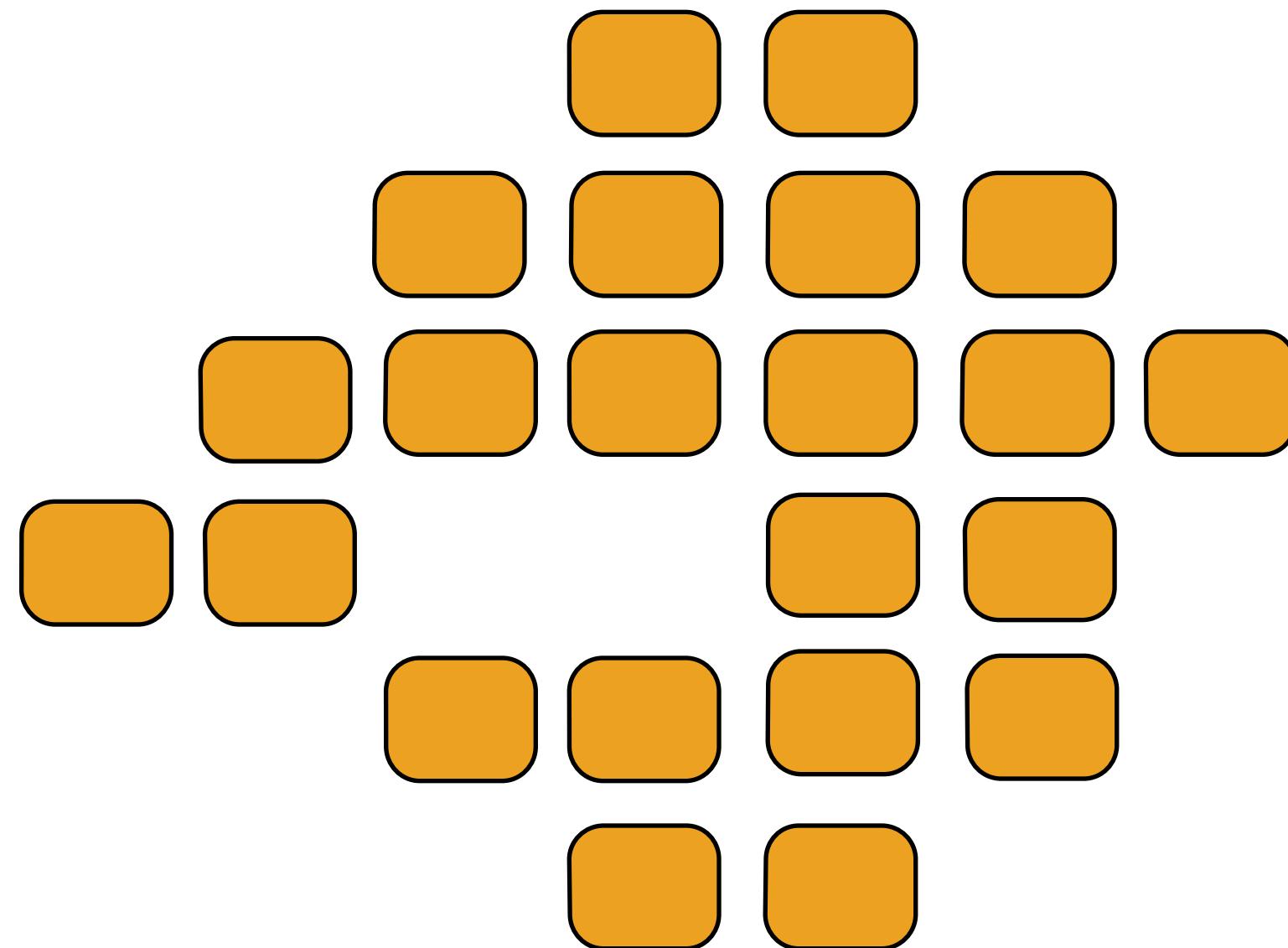
Roles

- Roles represent the types of servers in your infrastructure
 - Load Balancer
 - Application Server
 - Database Cache
 - Database
 - Monitoring

Roles Define Policy

- Roles may include an ordered list of Chef configuration files that should be applied
 - This list is called a Run List
 - Order is always important in the Run List
- Roles may include data attributes necessary for configuring your infrastructure, for example:
 - The port that the application server listens on
 - A list of applications that should be deployed

Nodes



Nodes

- Nodes represent anything that needs configuration
- Typically 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...

Node

- Each Node will
 - Belong to one Organization
 - Belong to one Environment
 - Have zero or more Roles

Nodes Adhere to Policy

- The chef-client application runs on each node, which
 - Gathers the current system configuration of the node
 - Downloads the desired system configuration policies from the Chef server for that node
 - Configures the node such that it adheres to those policies

Resources

- A Resource represents a piece of the system and its desired state
 - A package that should be installed
 - A service that should be running
 - A file that should be generated
 - A scheduled job that should be configured
 - A user that should be managed
 - and more

Resources in Recipes

- Resources are the fundamental building blocks of Chef configuration
- Resources are gathered into Recipes
- Recipes ensure the system is in the desired state

Recipes

- Configuration files that describe resources and their desired state
- Recipes can:
 - Install and configure software components
 - Manage files
 - Deploy applications
 - Execute other recipes
 - and more

Example 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
```

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

Or this...

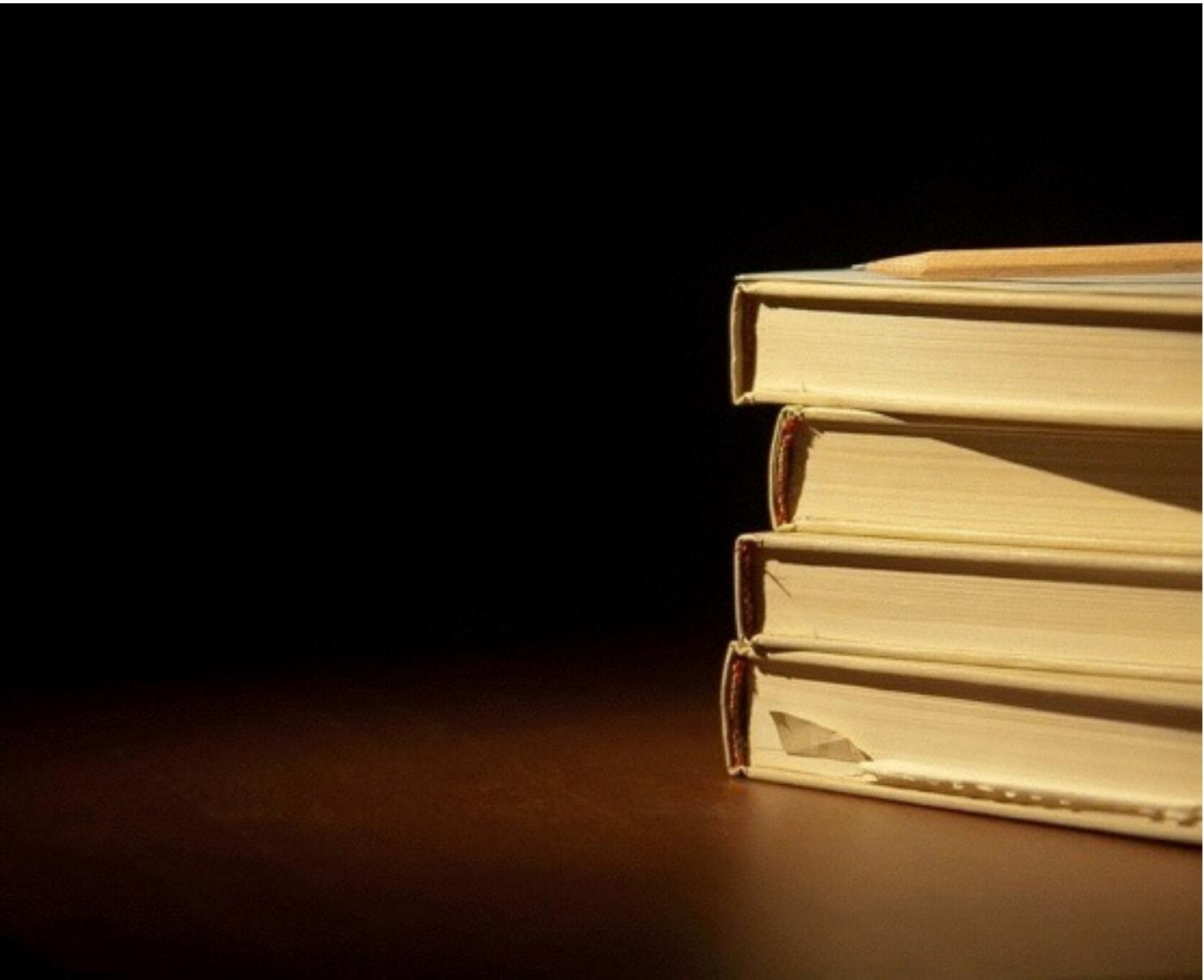
```
package "apache2"
```

```
template "/etc/apache2/apache2.conf" do
  source "apache2.conf.erb"
  owner "root"
  group "root"
  mode "0644"
  variables( :allow_override => "All" )
  notifies :reload, "service[apache2]"
end
```

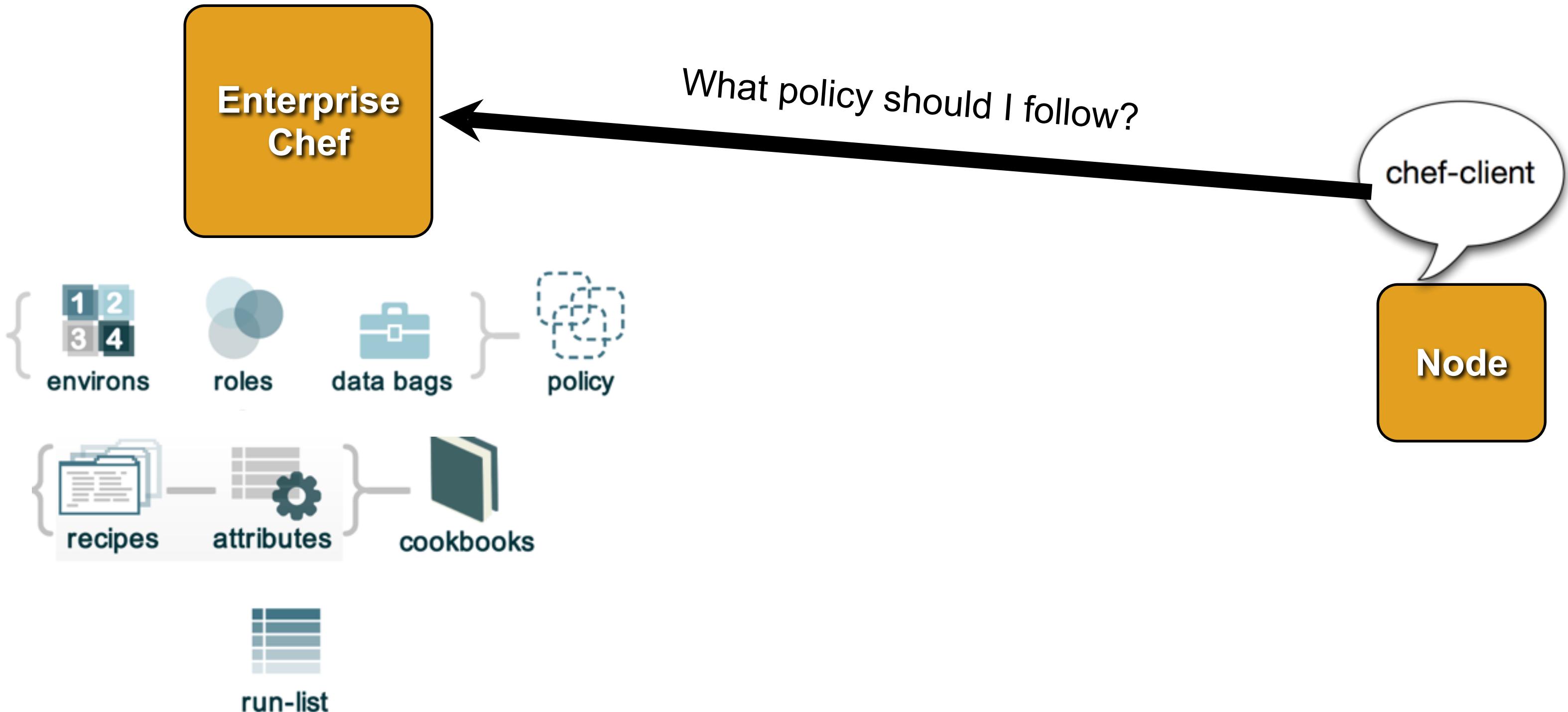
```
service "apache2" do
  action [ :enable, :start ]
  supports :reload => true
end
```

Cookbooks

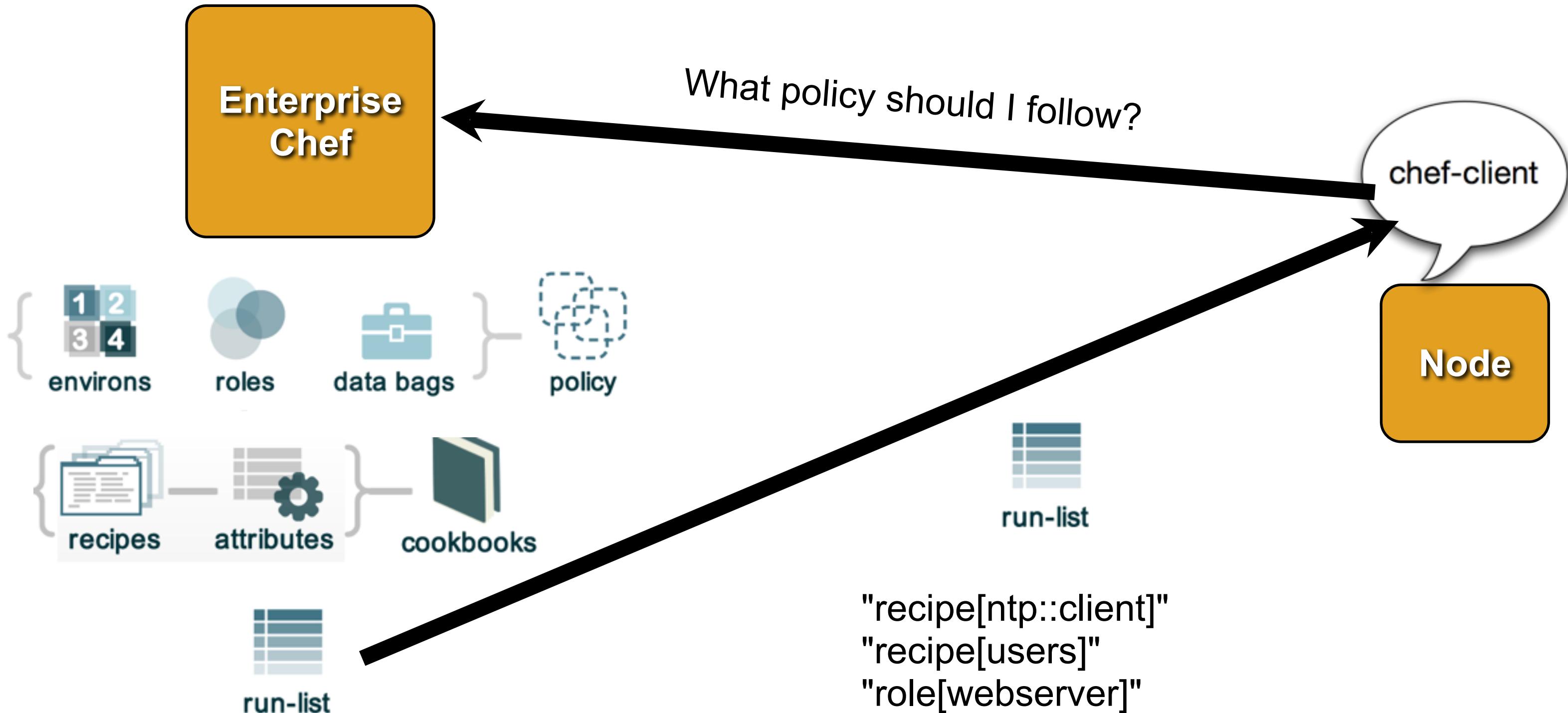
- Recipes are stored in Cookbooks
- Cookbooks contain recipes, templates, files, custom resources, etc
- Code re-use and modularity



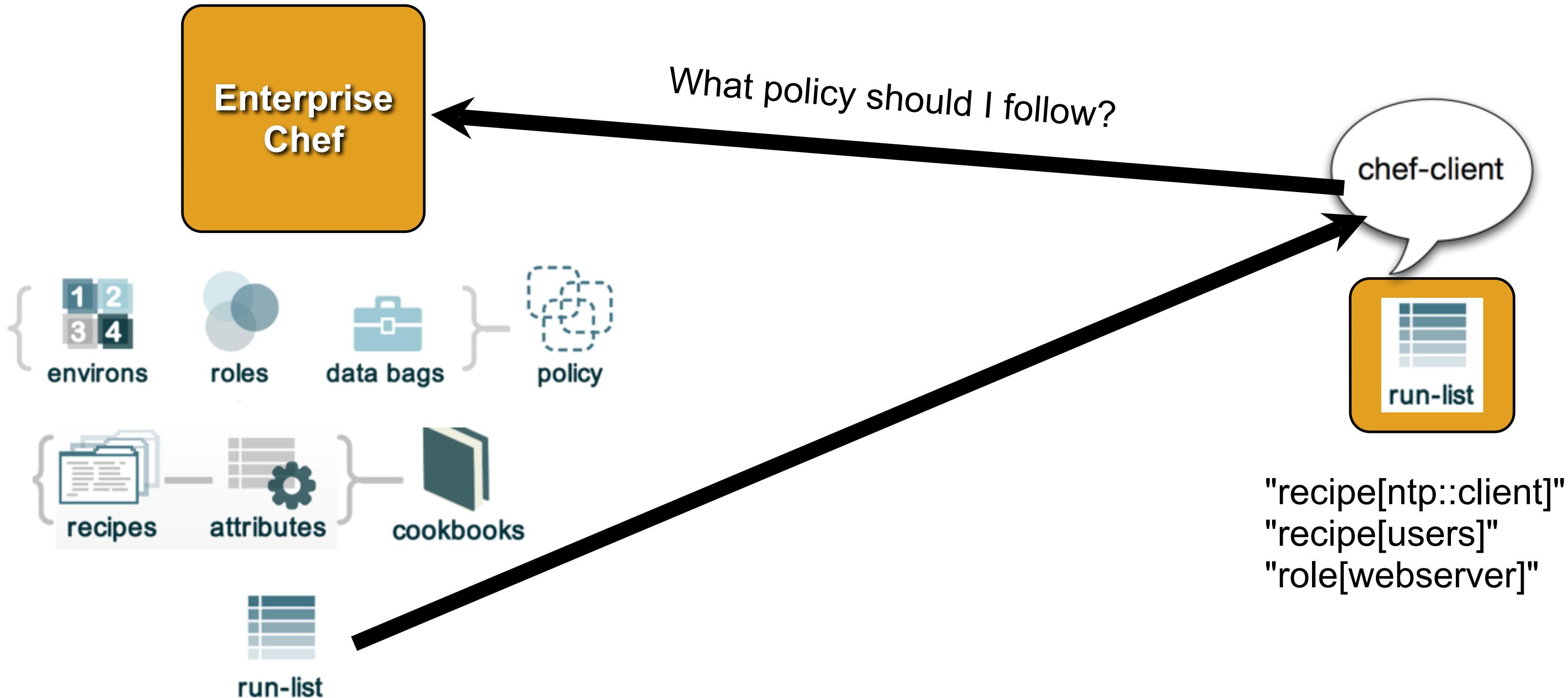
Run List



Run List



Run List



Run List Specifies Policy

- The Run List is an ordered collection of policies that the Node should follow
- Chef-client obtains the Run List from the Chef Server
- Chef-client ensures the Node complies with the policy in the Run List

Search

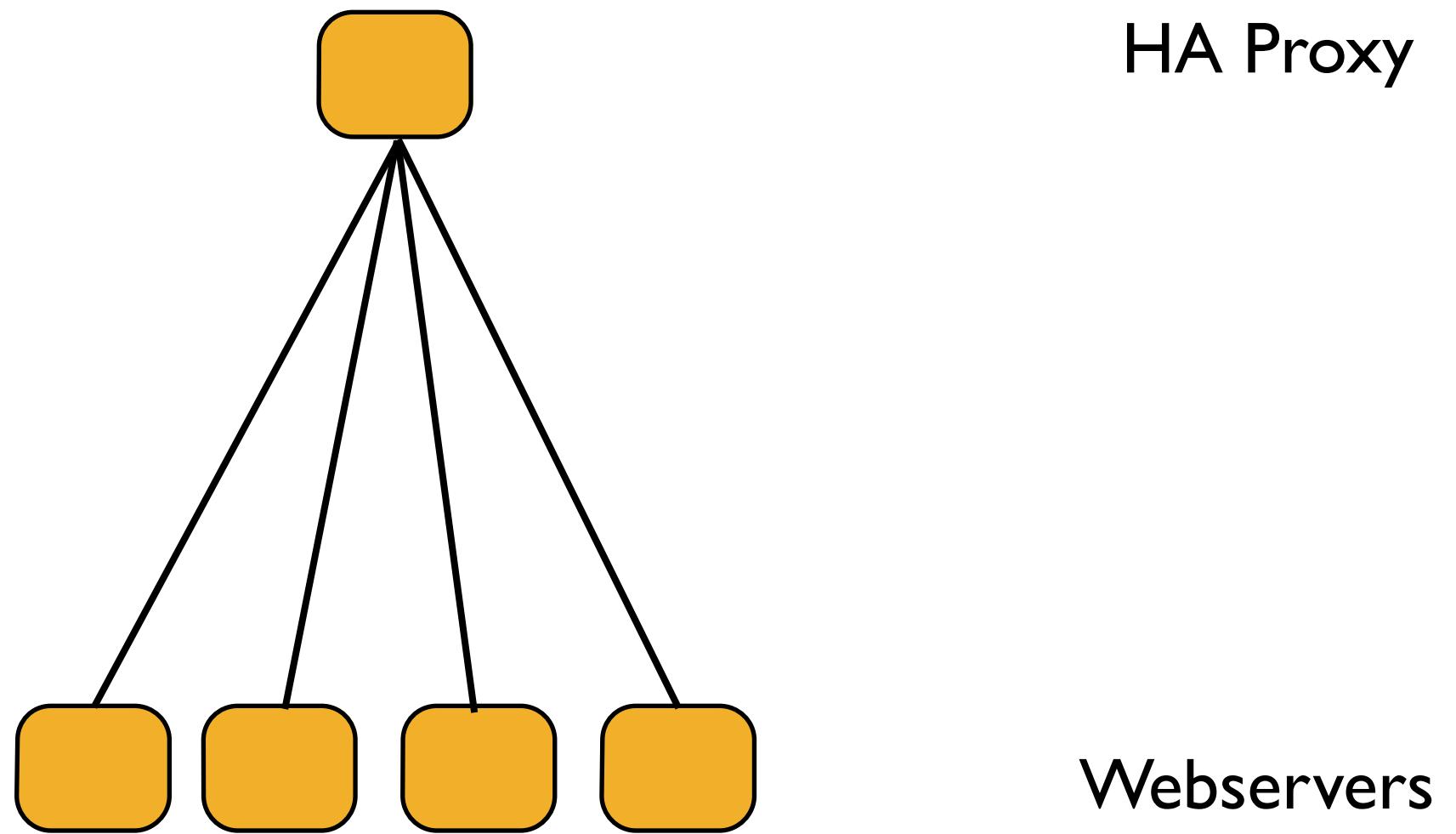
- Search for nodes with Roles
- Find Topology Data
- IP addresses
- Hostnames
- FQDNs
- Much more..

Search for Nodes - Recipe

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

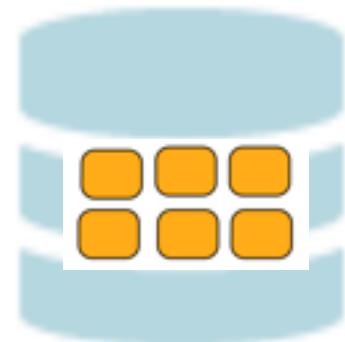
template "/etc/haproxy/haproxy.cfg" do
  source "haproxy-app_lb.cfg.erb"
  owner "root"
  group "root"
  mode 0644
  variables :pool_members => pool_members.uniq
  notifies :restart, "service[haproxy]"
end
```

HAProxy Configuration

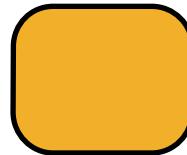


HAProxy Load Balancer

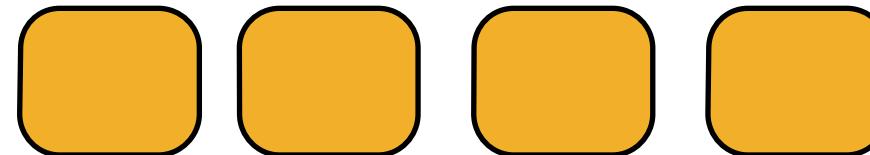
```
pool_members = search("node", "role:webserver")
```



Search Index



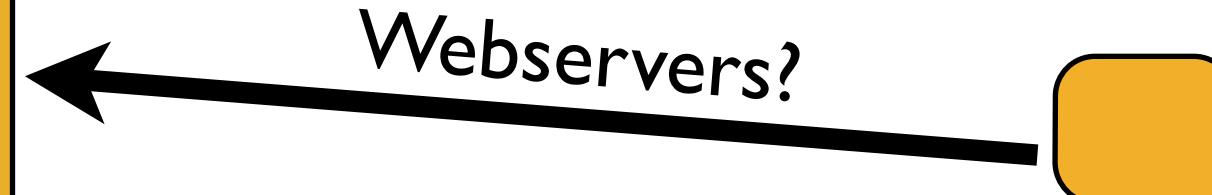
HA Proxy



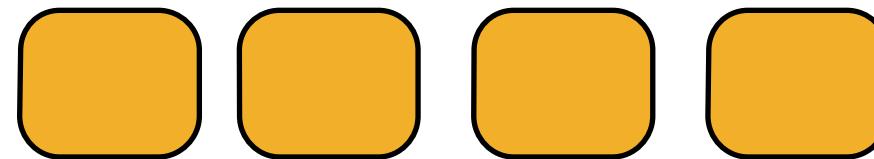
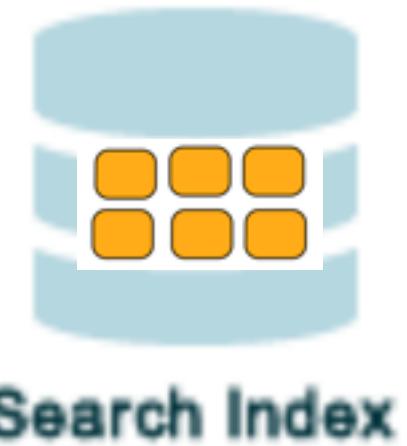
Webservers

HAProxy Load Balancer

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



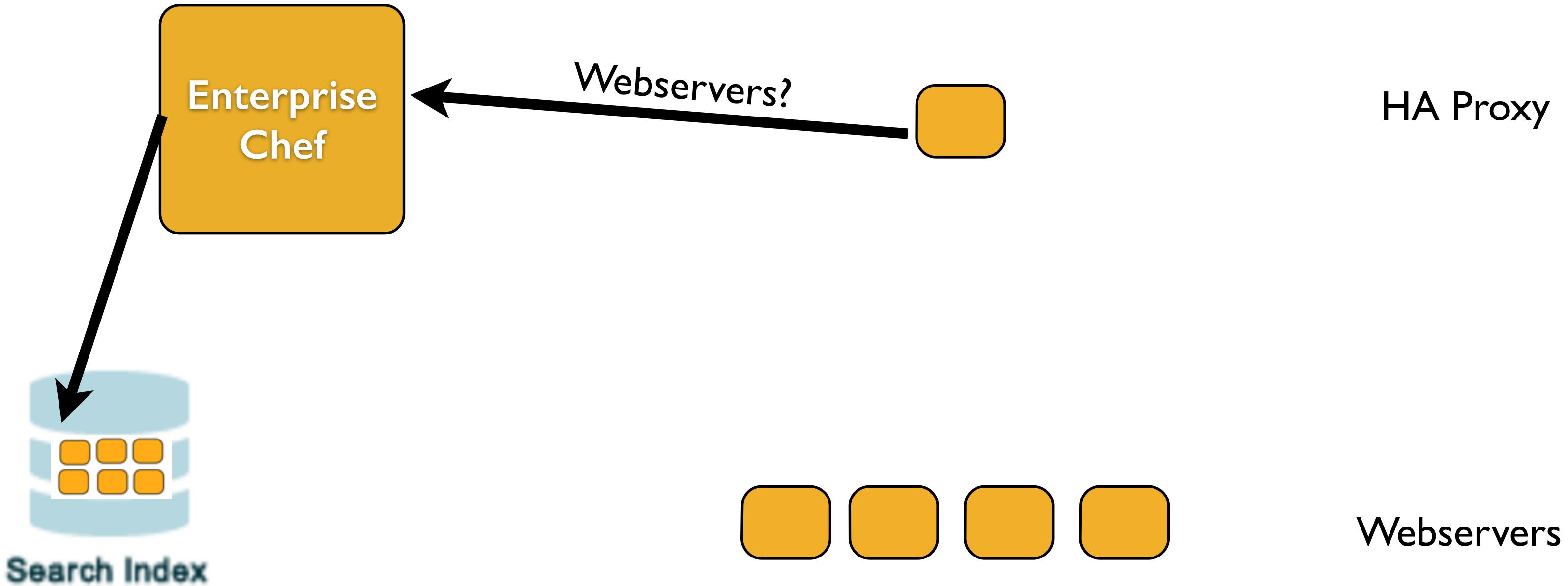
HA Proxy



Webservers

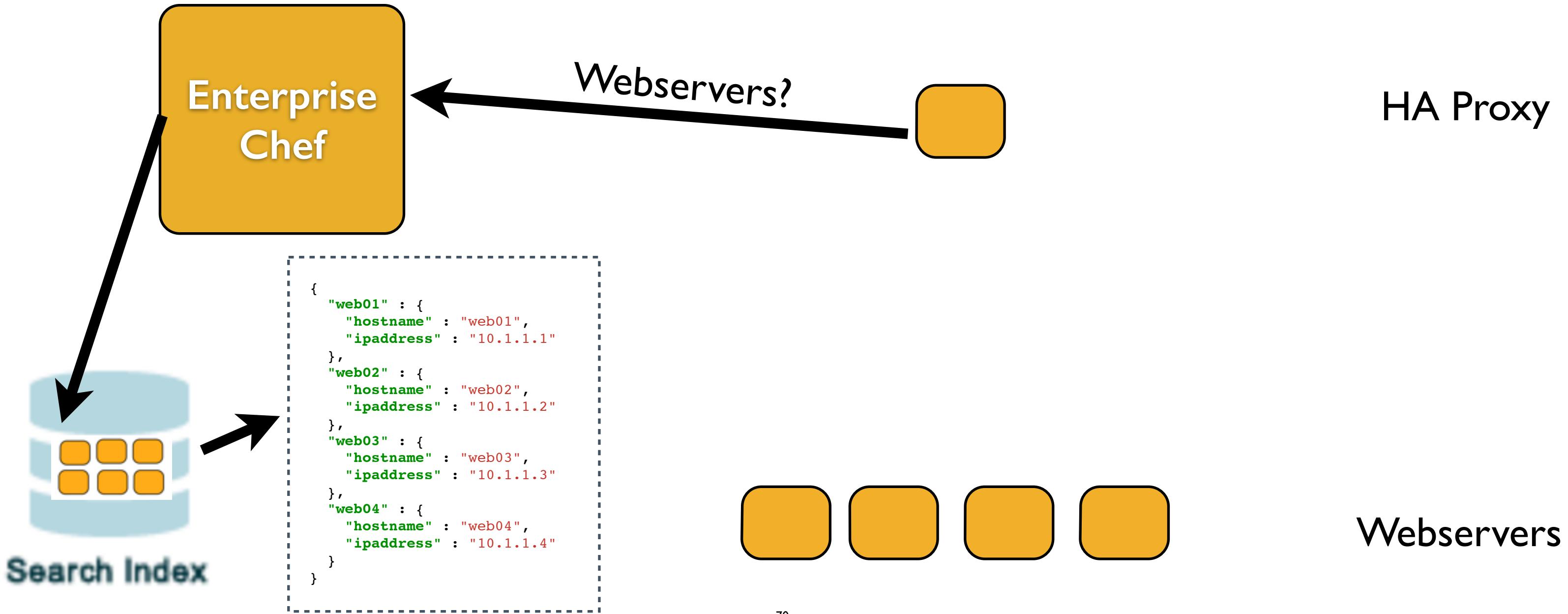
HAProxy Load Balancer

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



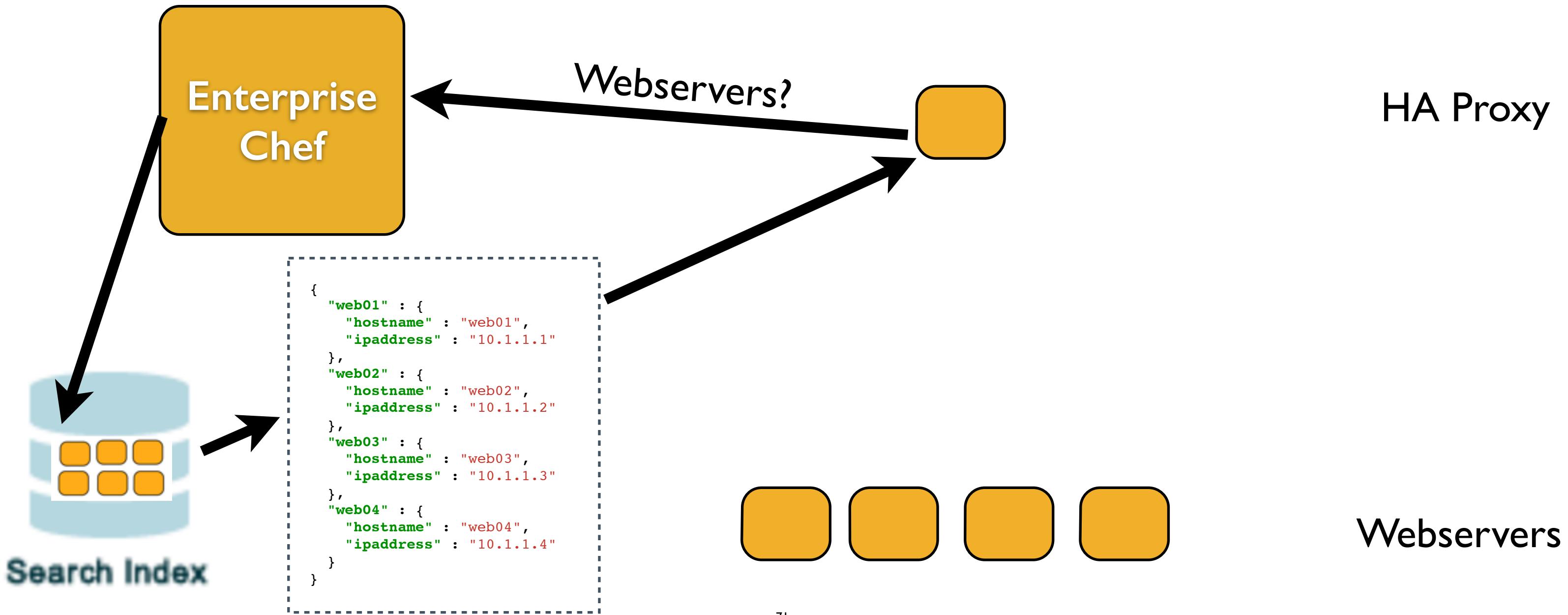
HAProxy Load Balancer

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



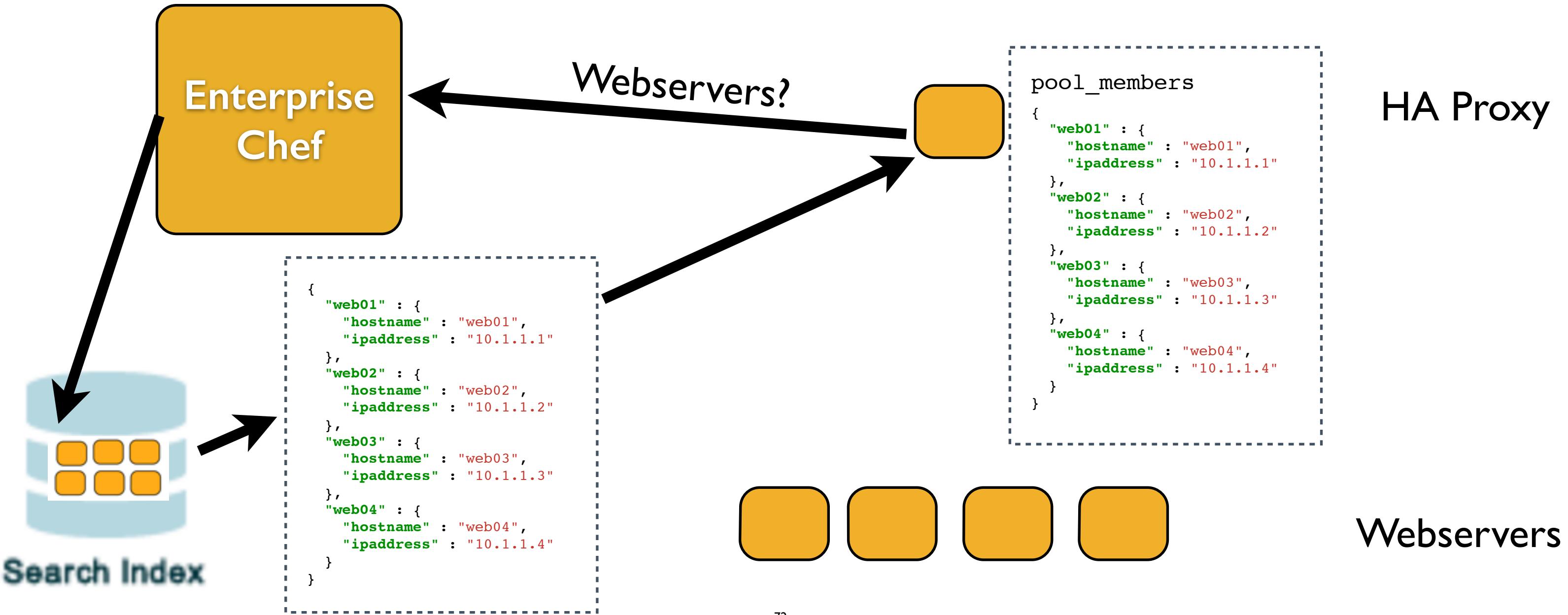
HAProxy Load Balancer

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



HAProxy Load Balancer

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



Search for Nodes

```
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 -%>

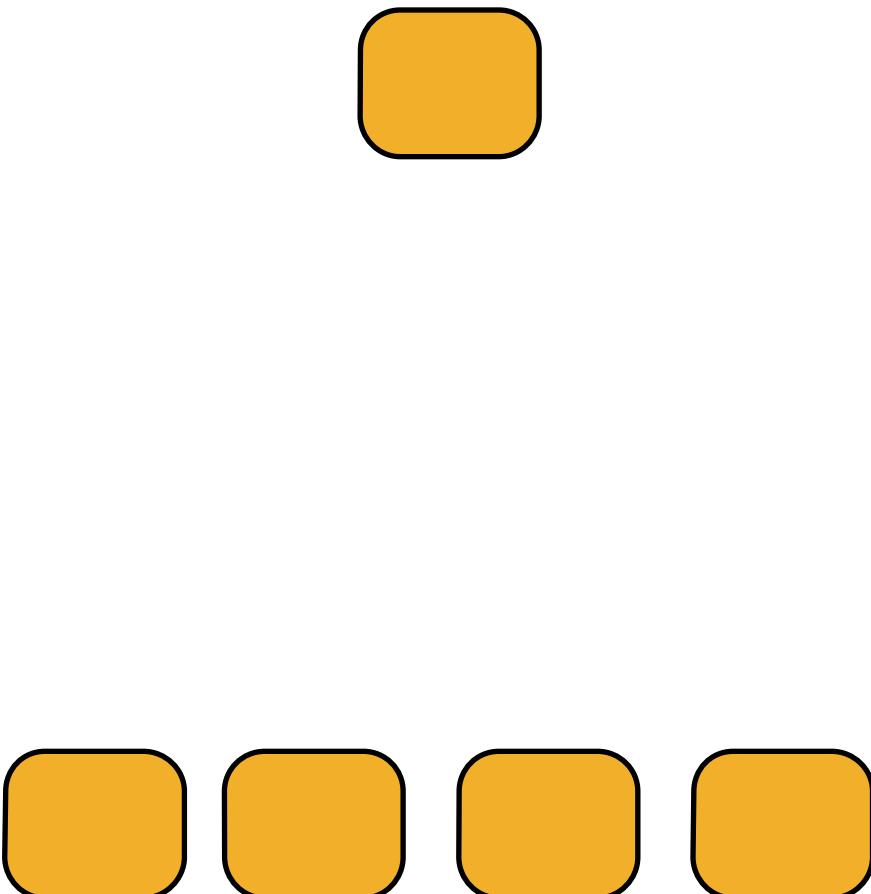
<% if node["haproxy"]["enable_admin"] -%>
listen admin 0.0.0.0:22002
  mode http
  stats uri /
<% end -%>
```

HAProxy Configuration

```
<% @pool_members.each do |member| -%>
server <%= member[:hostname] %> <%= member[:ipaddress] %>:> weight 1 maxconn 1 check
<% end -%>
```

```
pool_members
{
  "web01" : {
    "hostname" : "web01",
    "ipaddress" : "10.1.1.1"
  },
  "web02" : {
    "hostname" : "web02",
    "ipaddress" : "10.1.1.2"
  },
  "web03" : {
    "hostname" : "web03",
    "ipaddress" : "10.1.1.3"
  },
  "web04" : {
    "hostname" : "web04",
    "ipaddress" : "10.1.1.4"
  }
}
```

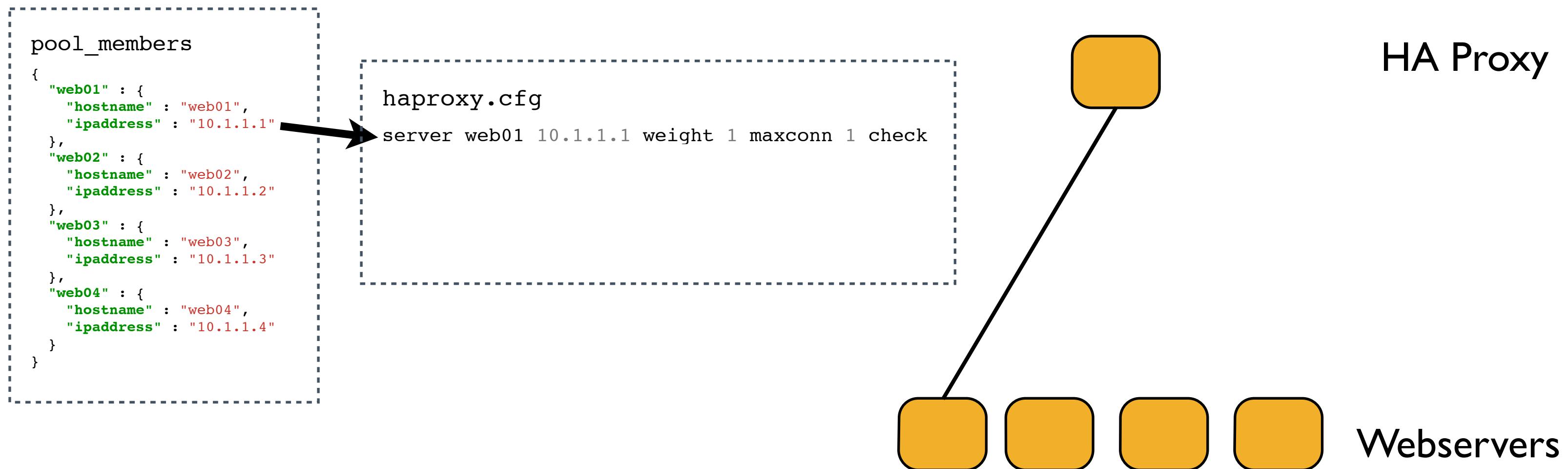
HA Proxy



Webservers

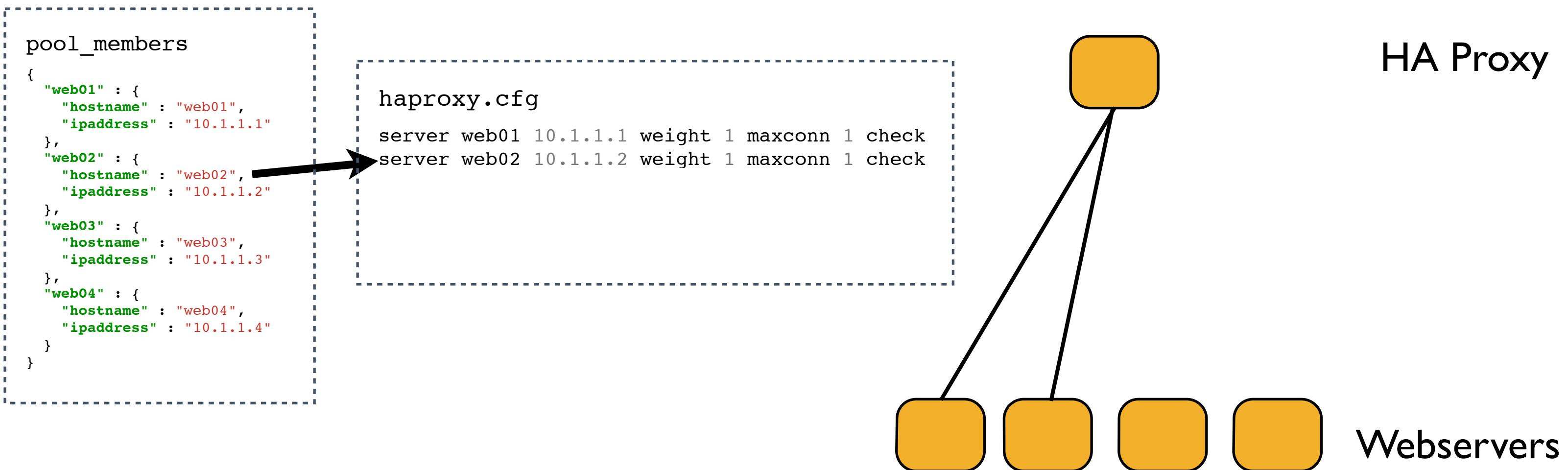
HAProxy Configuration

```
<% @pool_members.each do |member| -%>
server <%= member[:hostname] %> <%= member[:ipaddress] %>:> weight 1 maxconn 1 check
<% end -%>
```



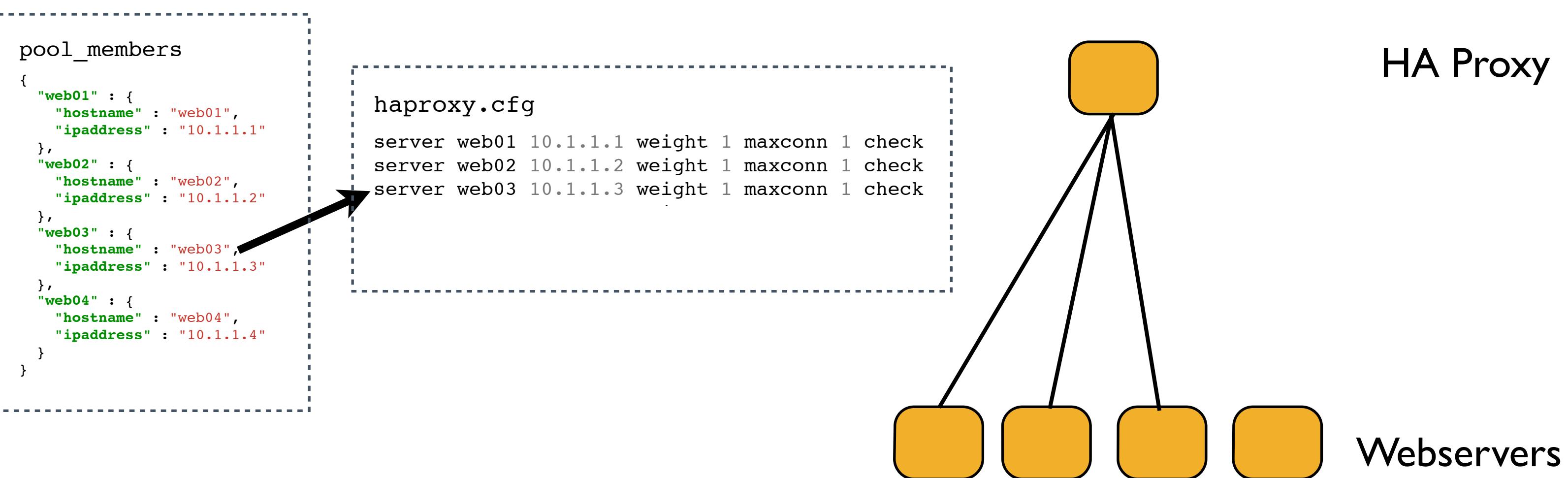
HAProxy Configuration

```
<% @pool_members.each do |member| -%>
server <%= member[:hostname] %> <%= member[:ipaddress] %>:> weight 1 maxconn 1 check
<% end -%>
```



HAProxy Configuration

```
<% @pool_members.each do |member| -%>
server <%= member[:hostname] %> <%= member[:ipaddress] %>:> weight 1 maxconn 1 check
<% end -%>
```

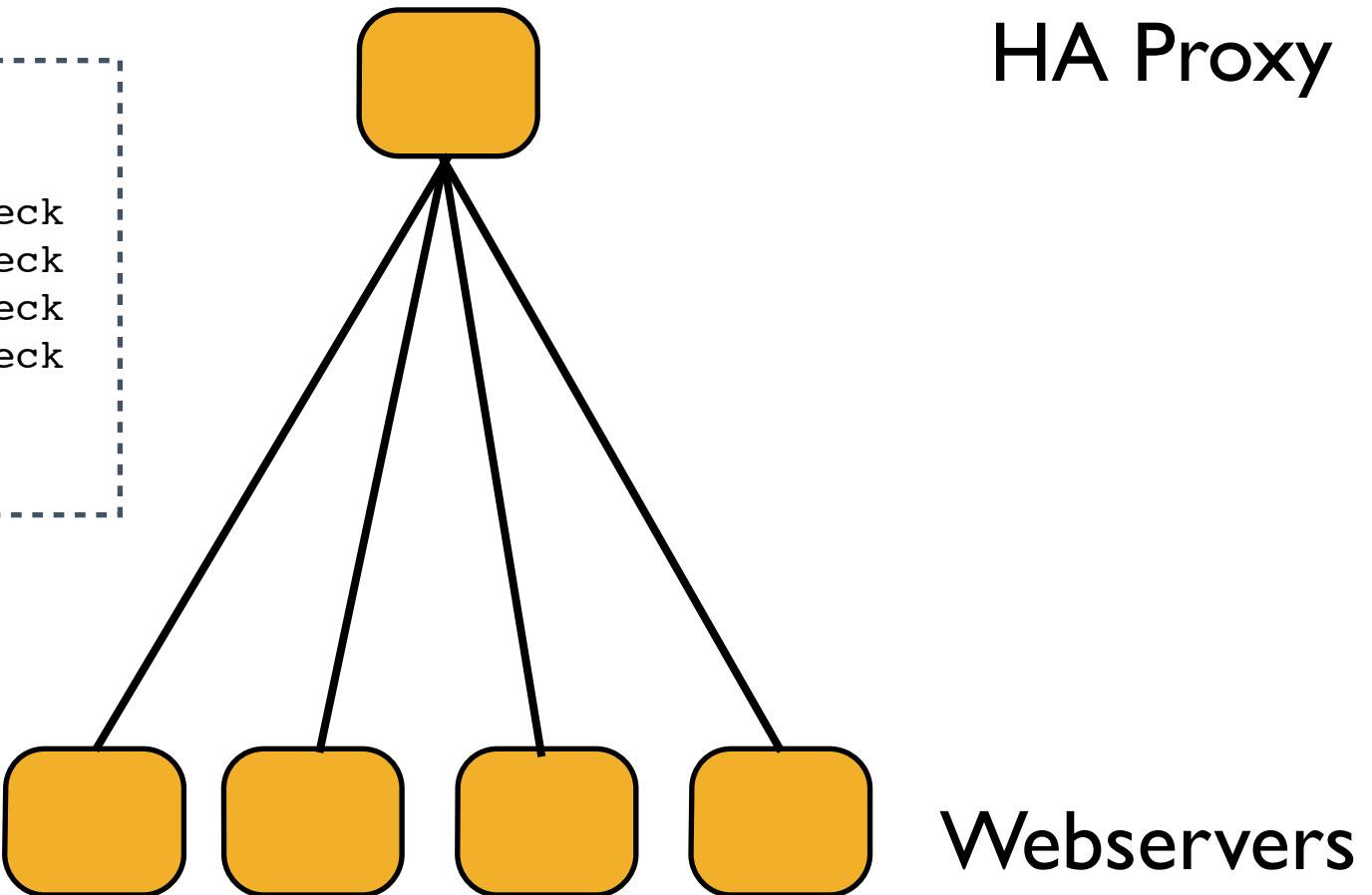


HAProxy Configuration

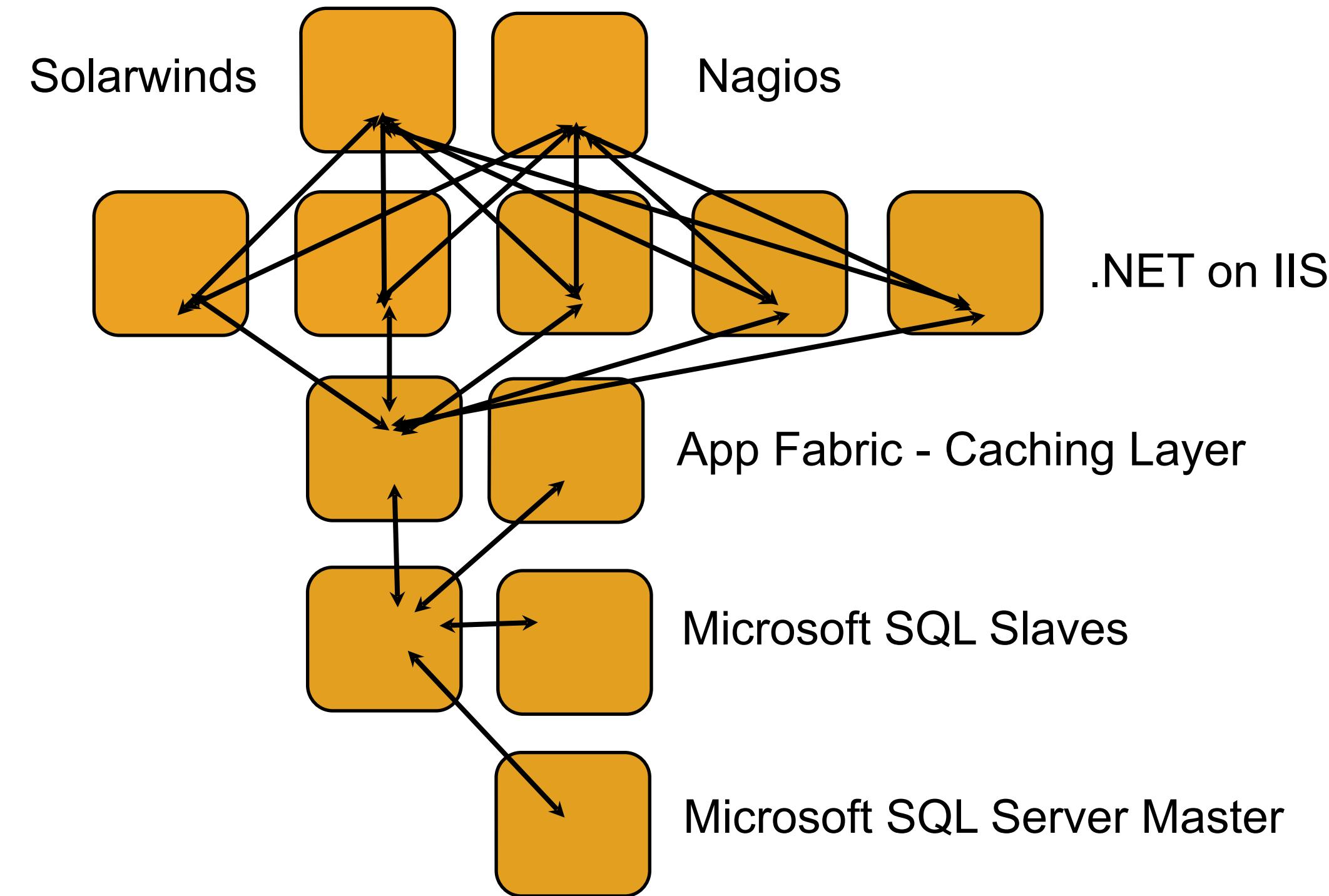
```
<% @pool_members.each do |member| -%>
server <%= member[:hostname] %> <%= member[:ipaddress] %>:> weight 1 maxconn 1 check
<% end -%>
```

```
pool_members
{
  "web01" : {
    "hostname" : "web01",
    "ipaddress" : "10.1.1.1"
  },
  "web02" : {
    "hostname" : "web02",
    "ipaddress" : "10.1.1.2"
  },
  "web03" : {
    "hostname" : "web03",
    "ipaddress" : "10.1.1.3"
  },
  "web04" : {
    "hostname" : "web04",
    "ipaddress" : "10.1.1.4"
  }
}
```

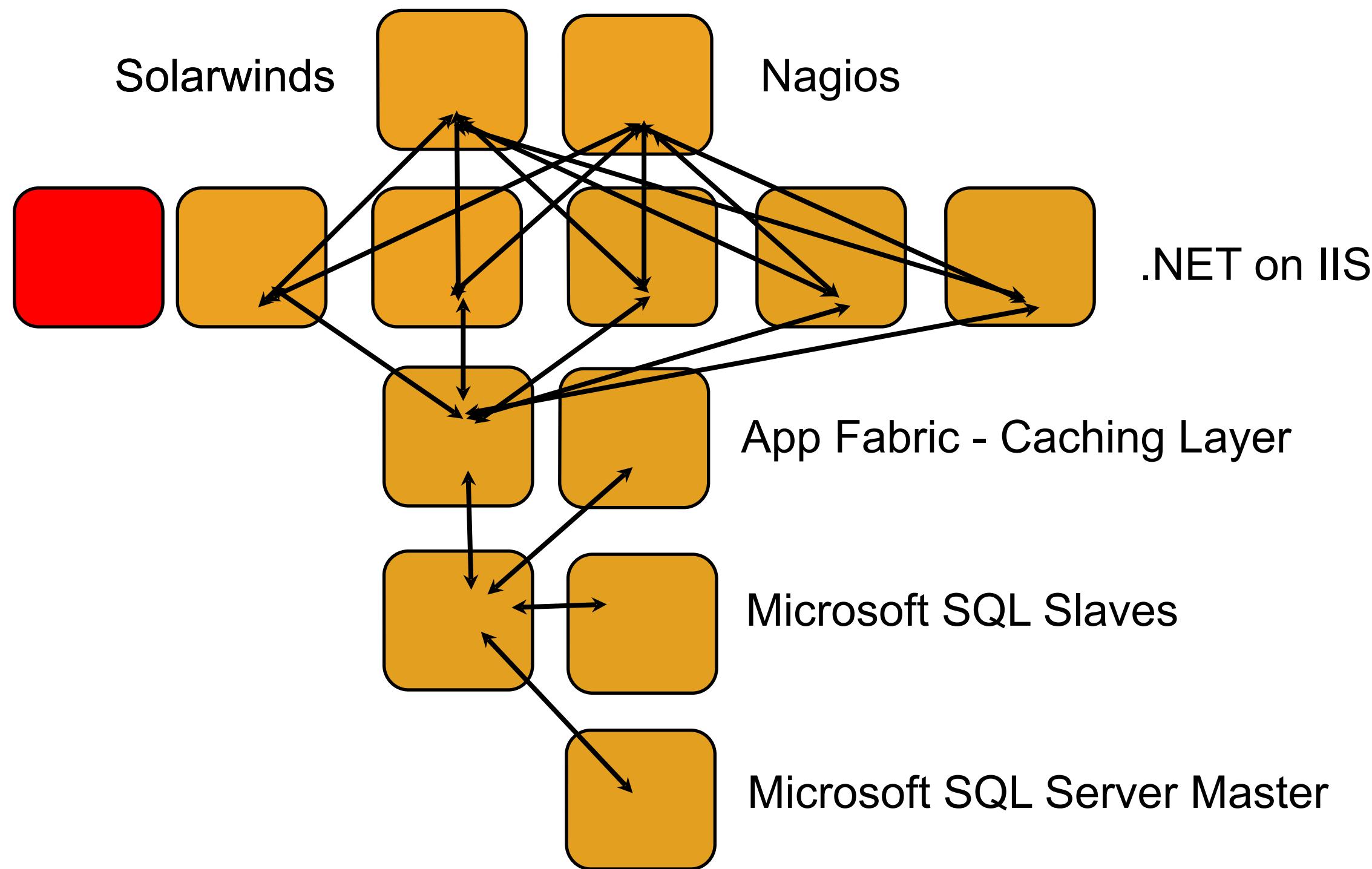
```
haproxy.cfg
server web01 10.1.1.1 weight 1 maxconn 1 check
server web02 10.1.1.2 weight 1 maxconn 1 check
server web03 10.1.1.3 weight 1 maxconn 1 check
server web04 10.1.1.4 weight 1 maxconn 1 check
```



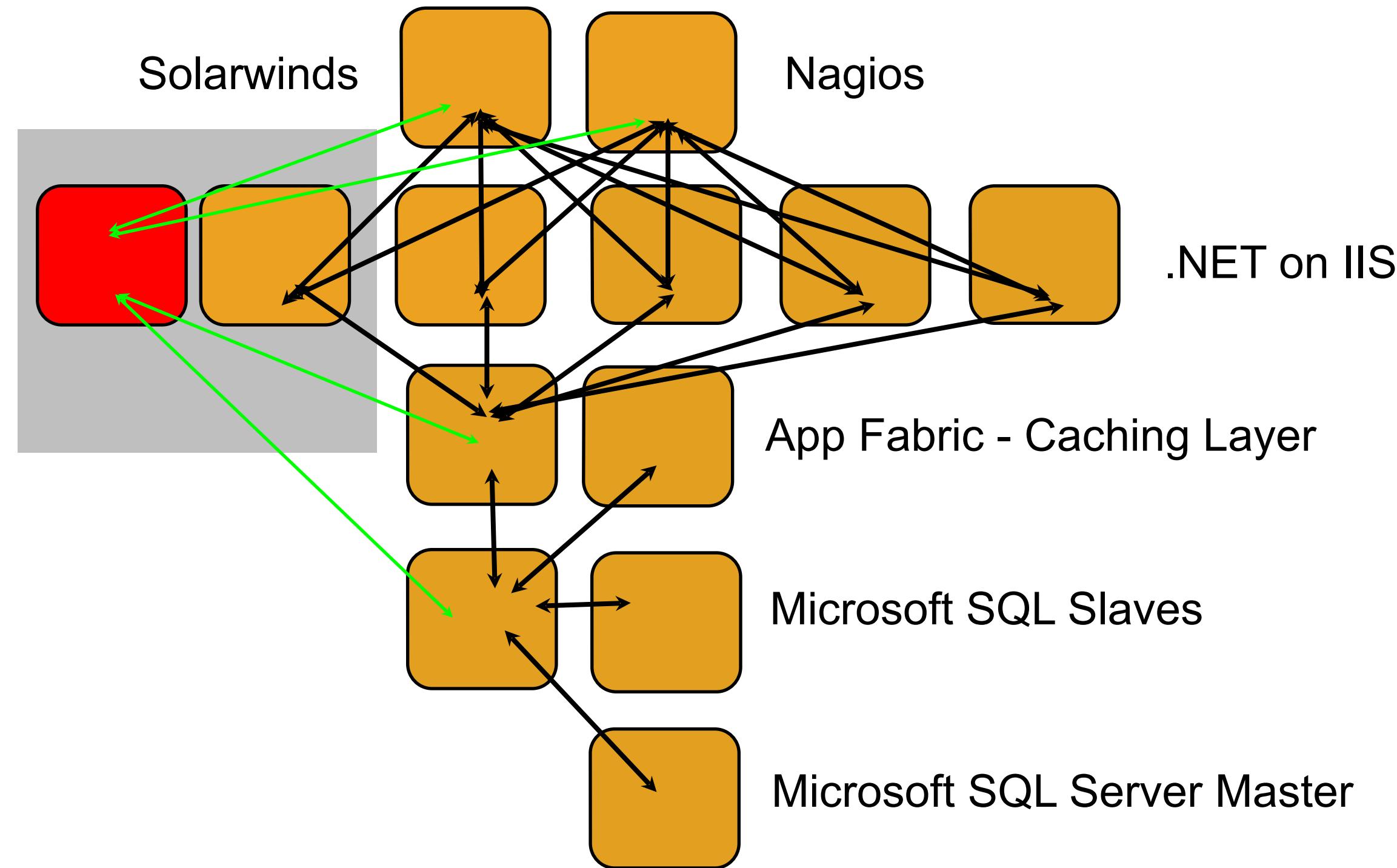
So when this...



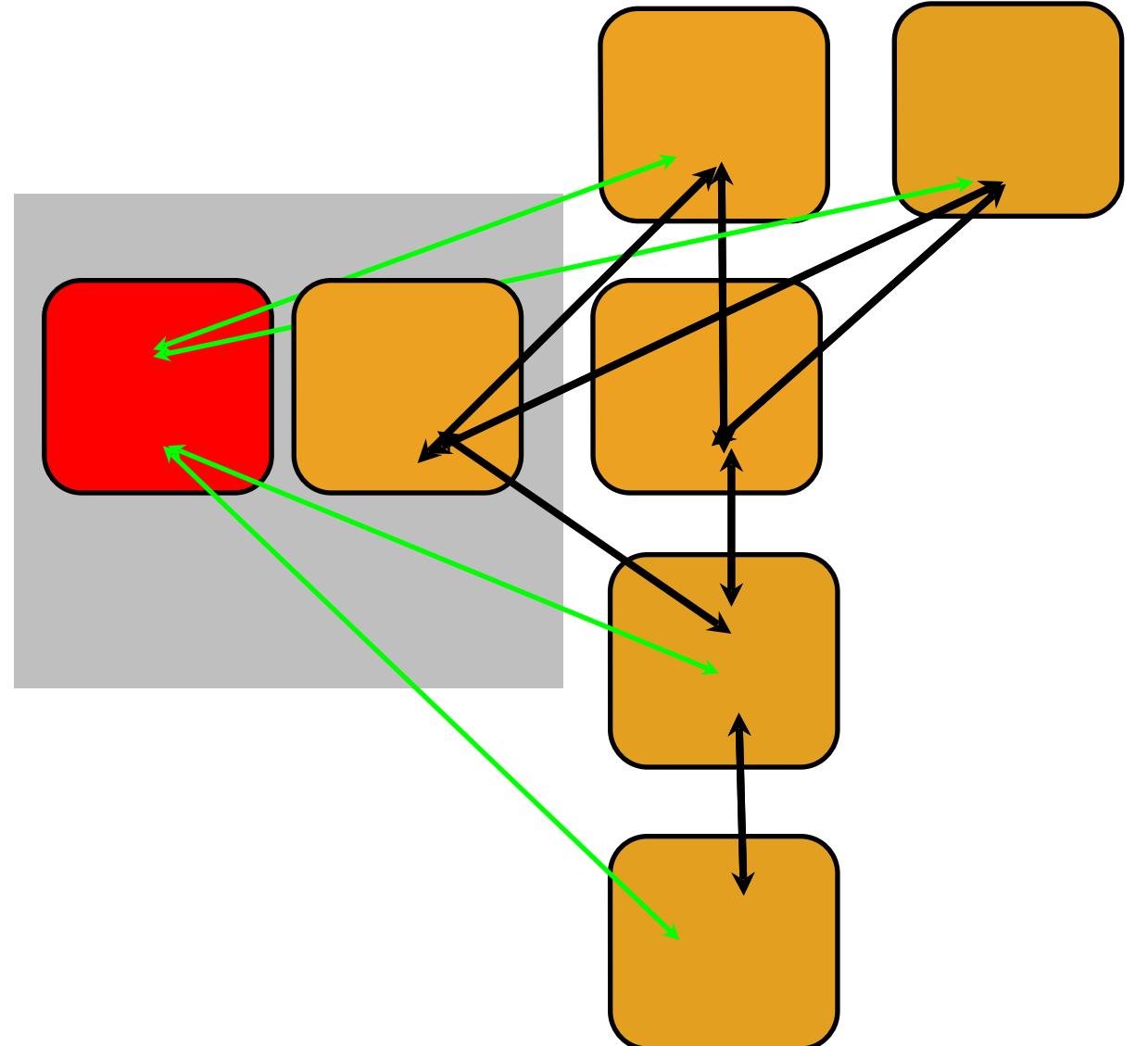
...becomes this



...this can happen automatically



Count the Resources



- 12+ resource changes for 1 node addition

- Load balancer config
- Nagios host ping
- Nagios host ssh
- Nagios host HTTP
- Nagios host app health
- Solarwinds CPU
- Solarwinds Memory
- Solarwinds Disk
- Solarwinds SNMP
- DB Cache firewall
- SQL Server firewall
- SQL Server ACL

Manage Complexity

- Determine the desired state of your infrastructure
- Identify the Resources required to meet that state
- Gather the Resources into Recipes
- Compose a Run List from Recipes and Roles
- Apply a Run List to each Node in your Environment
- Your infrastructure adheres to the policy modeled in Chef

Configuration Drift

- Configuration Drift happens when:
 - Your infrastructure requirements change
 - The configuration of a server falls out of policy
- Chef makes it easy to manage
 - Model the new requirements in your Chef configuration files
 - Run the chef-client to enforce your policies

Design Tenets of Chef

- Whiuptitude - whipping things up quickly
- Reasonability
- Sane defaults
- Flexibility
- Manipulexity - ability to manipulate complex things

Design Tenets of Chef

#ChefConf 2012

Blame Larry Wall - I do :)

- **Manipulexity:** the manipulation of complex things
- **Whipuptitude:** the aptitude for whipping things up

A diagram showing a 2D coordinate system with a vertical axis labeled "Manipulexity" and a horizontal axis labeled "Whipuptitude". A point labeled "C" is plotted in the first quadrant, representing a balance between the two tenets.

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 a Role?

Workstation Setup

Getting started

v2.1.1_WIN

Lesson Objectives

- After completing the lesson, you will be able to
 - Login to Enterprise Chef
 - View your Organization in Enterprise Chef
 - Describe Knife, the Chef command line utility
 - Use Knife on your Workstation

Legend

v2.1.1_WIN

Legend: Do I run that command on my workstation?

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

```
PS \> whoami  
i-am-a-workstation
```

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

```
Remote@PS\> whoami  
i-am-a-chef-node
```

Legend: Example Terminal Command and Output

```
PS \> 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

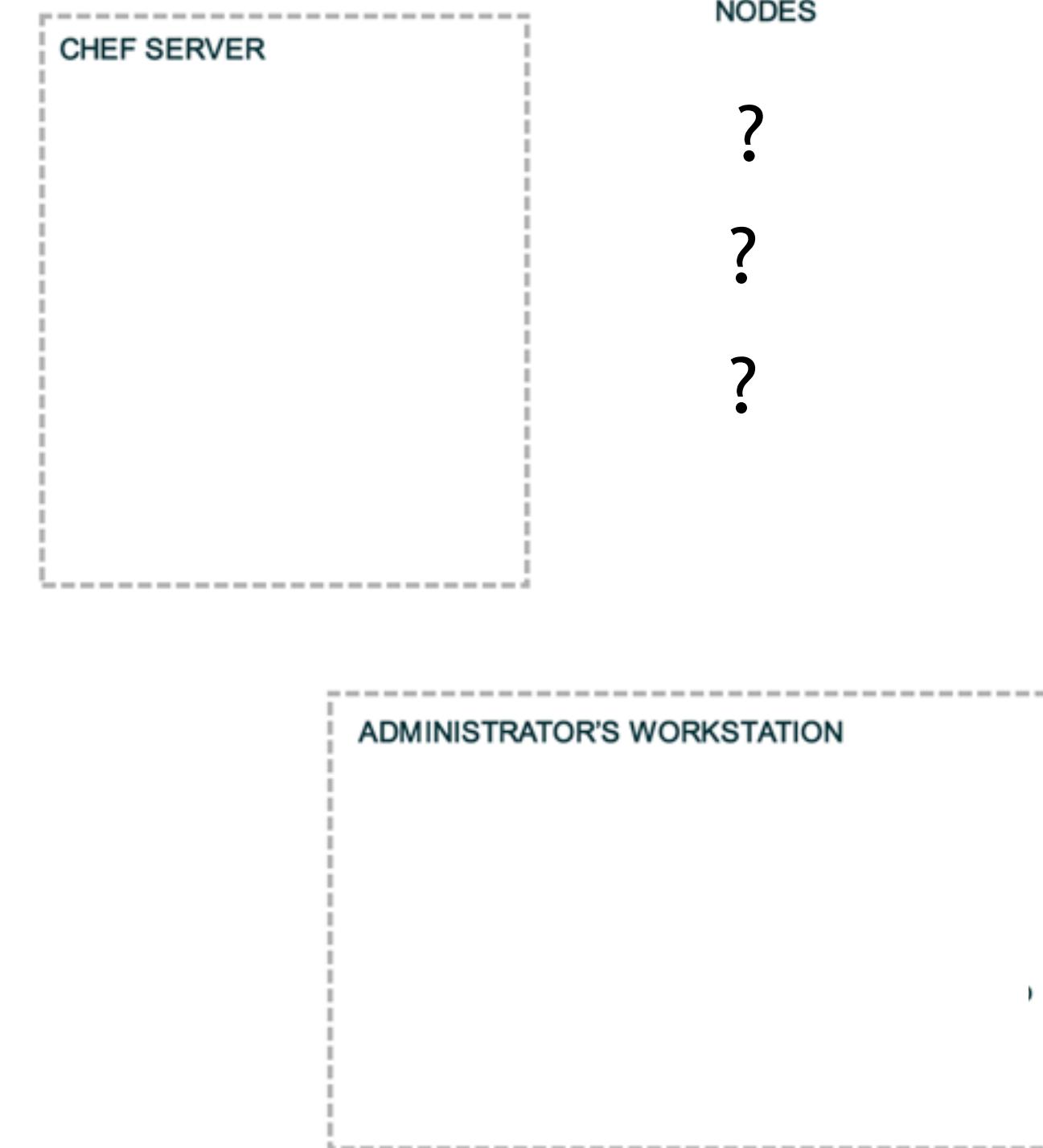
OPEN IN EDITOR: hello_world.txt

Hi!

I am a friendly file.

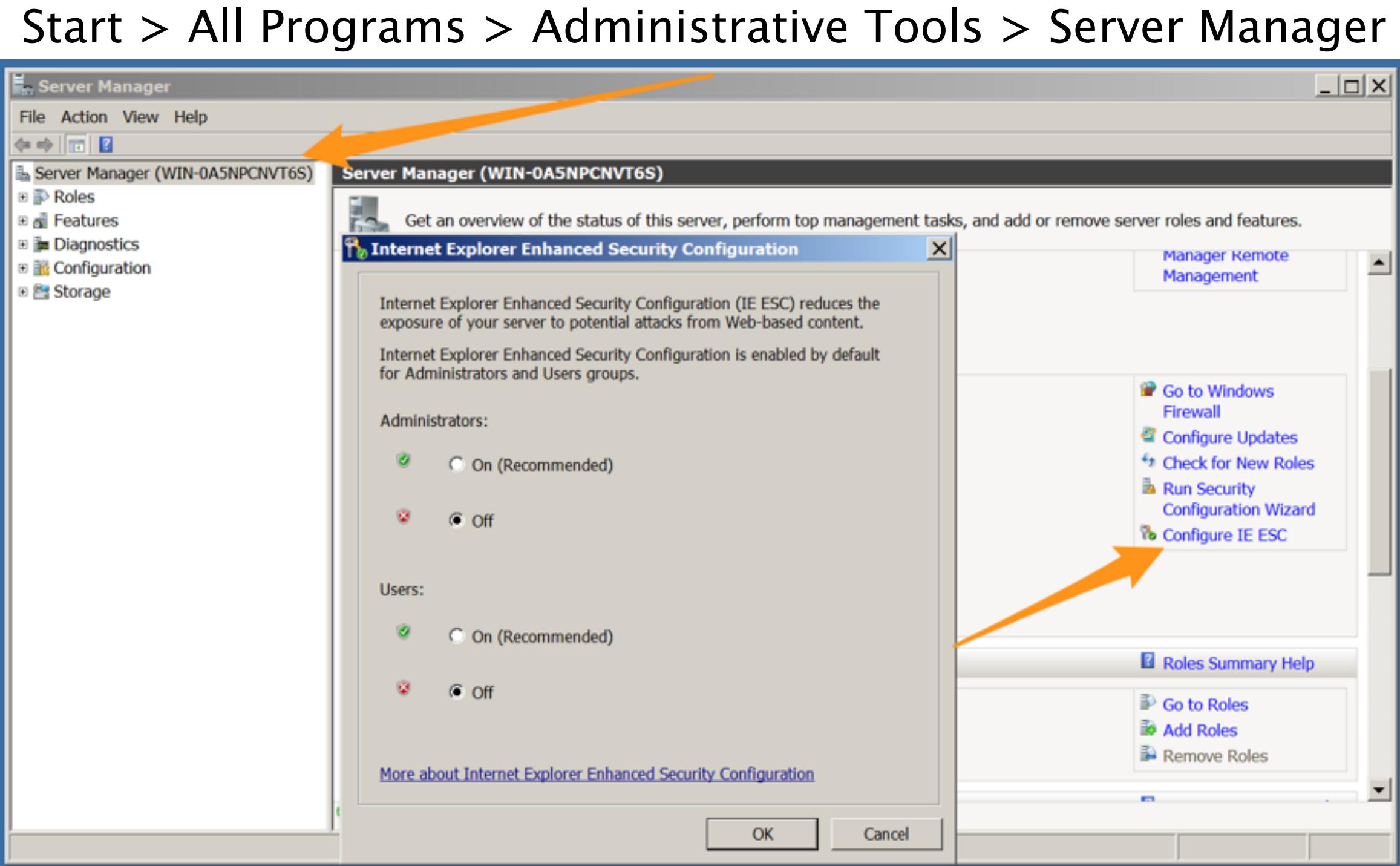
SAVE FILE!

Landscape of a Chef-managed Infrastructure



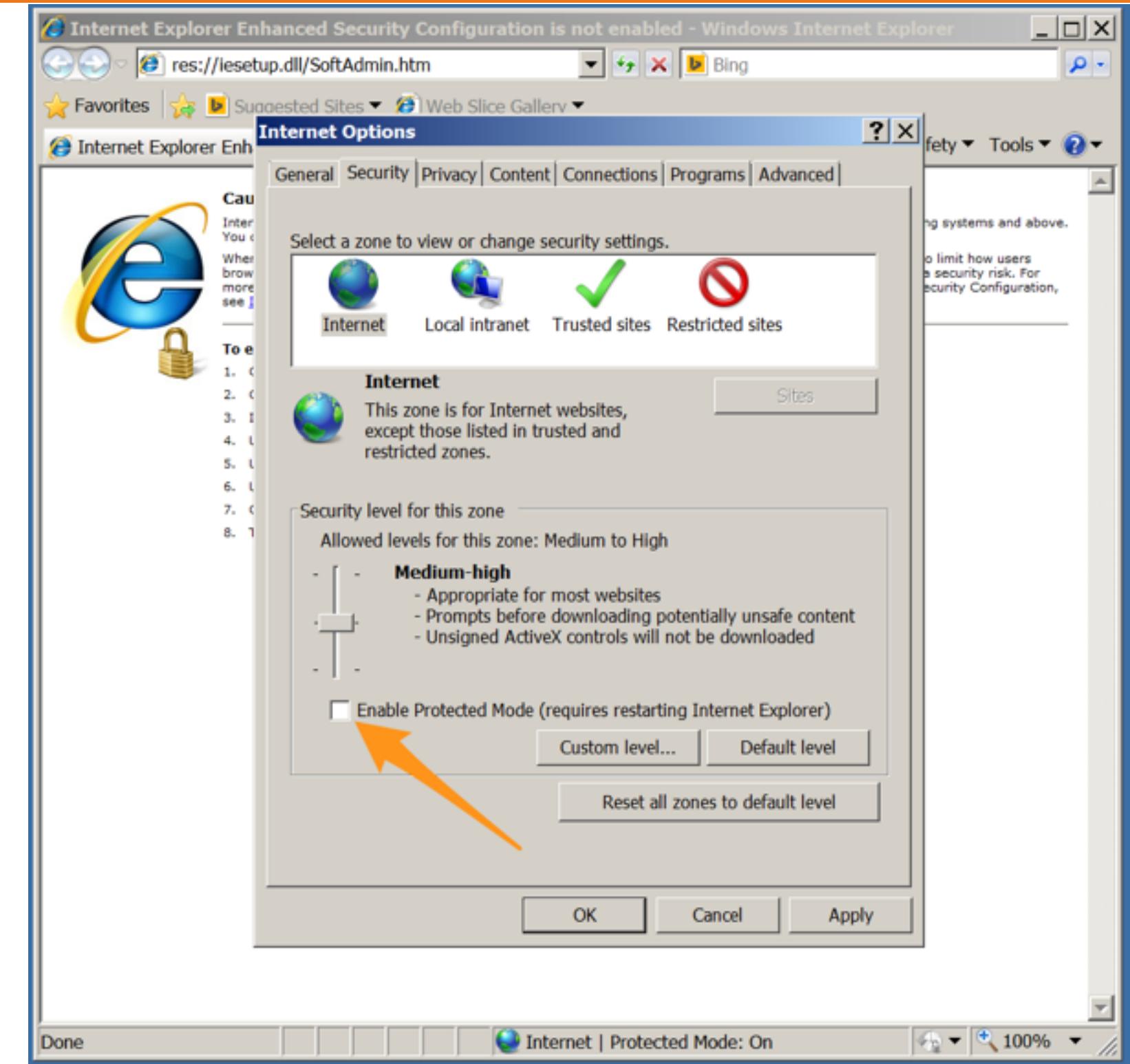
Exercise: Configure Management Station

- We're going to need to download a few items, so let's turn off IE Enhanced Security Configuration in Server Manager to make our downloads easier:



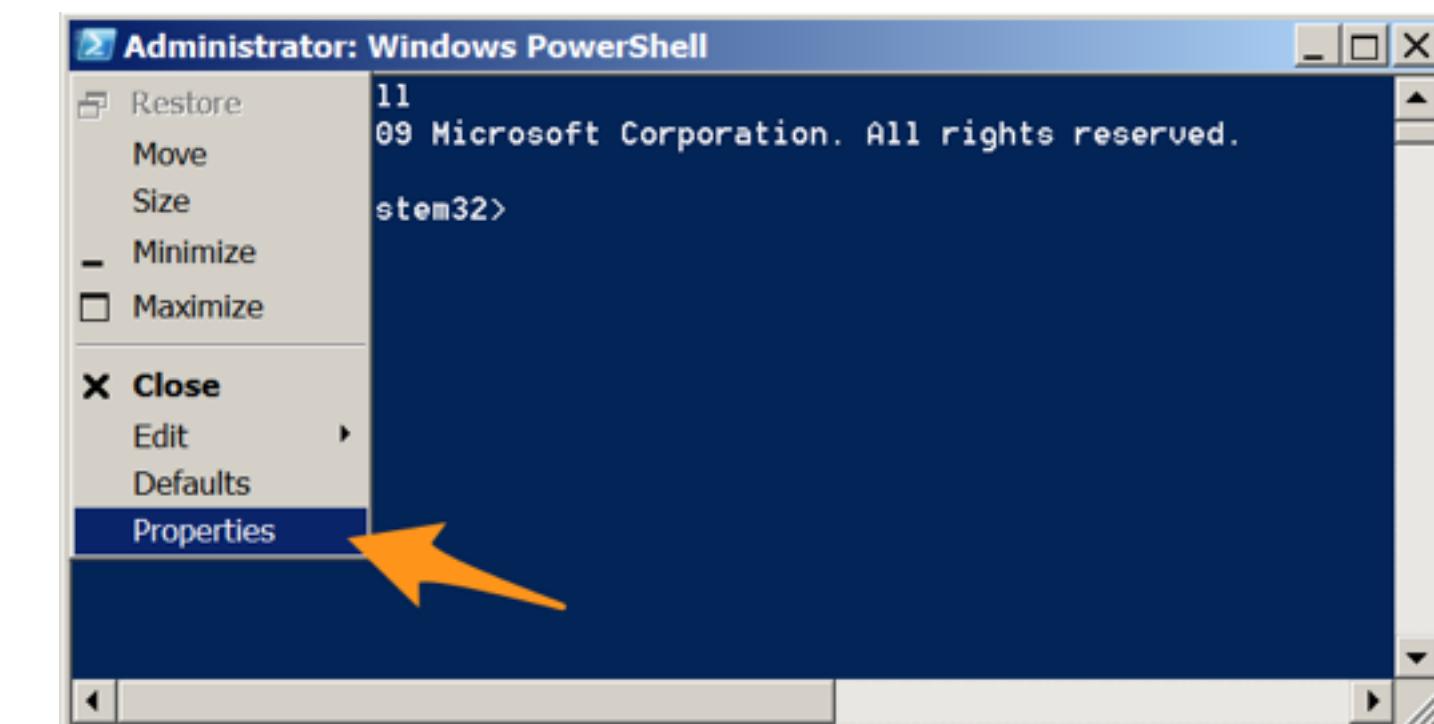
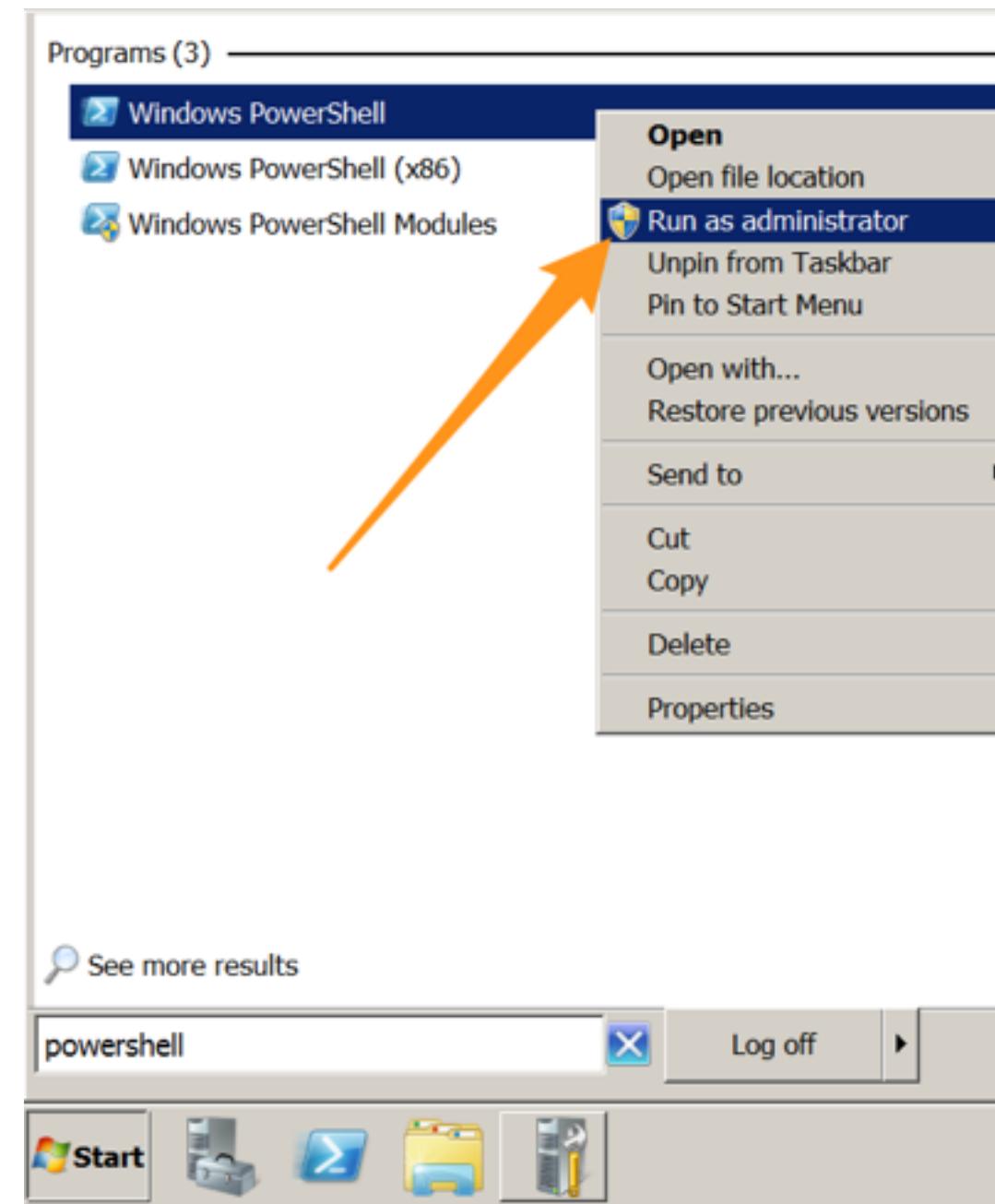
Exercise: Configure Management Station

- Turn off Enabled Protected mode in IE (IE Options --> Tools --> Uncheck “Enable Protected mode”)



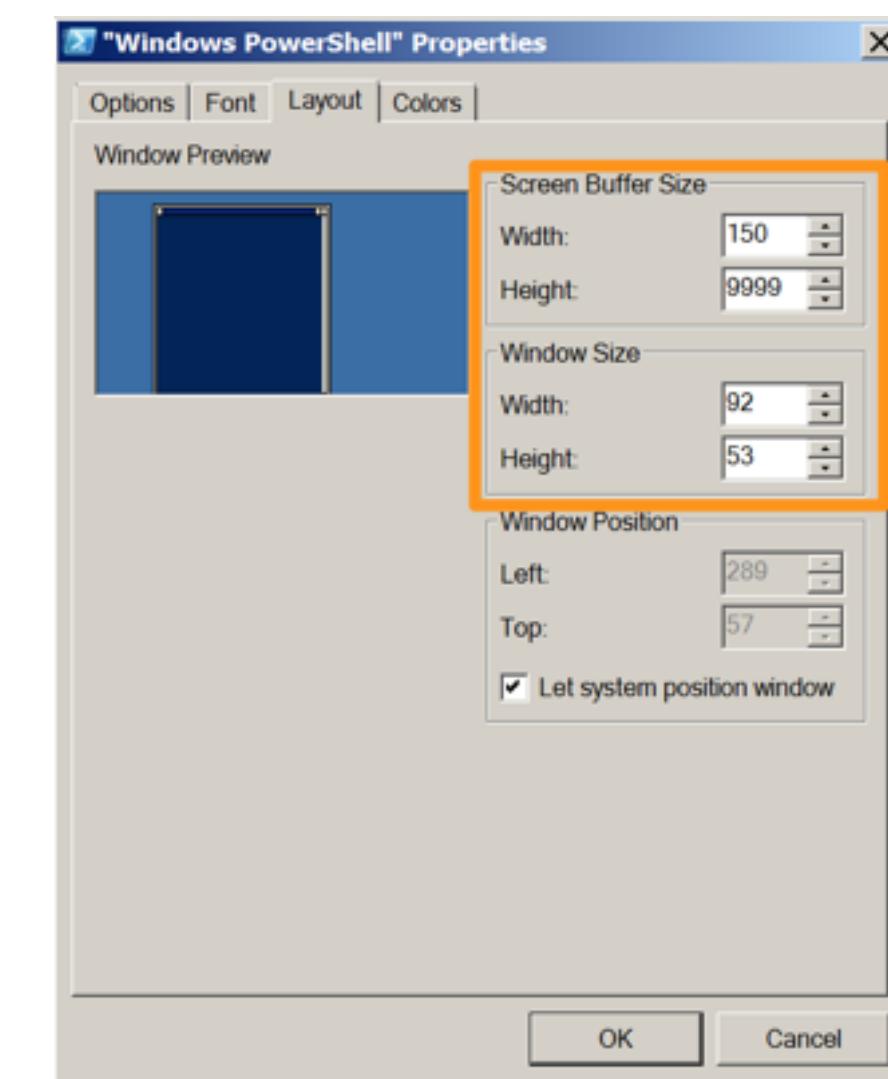
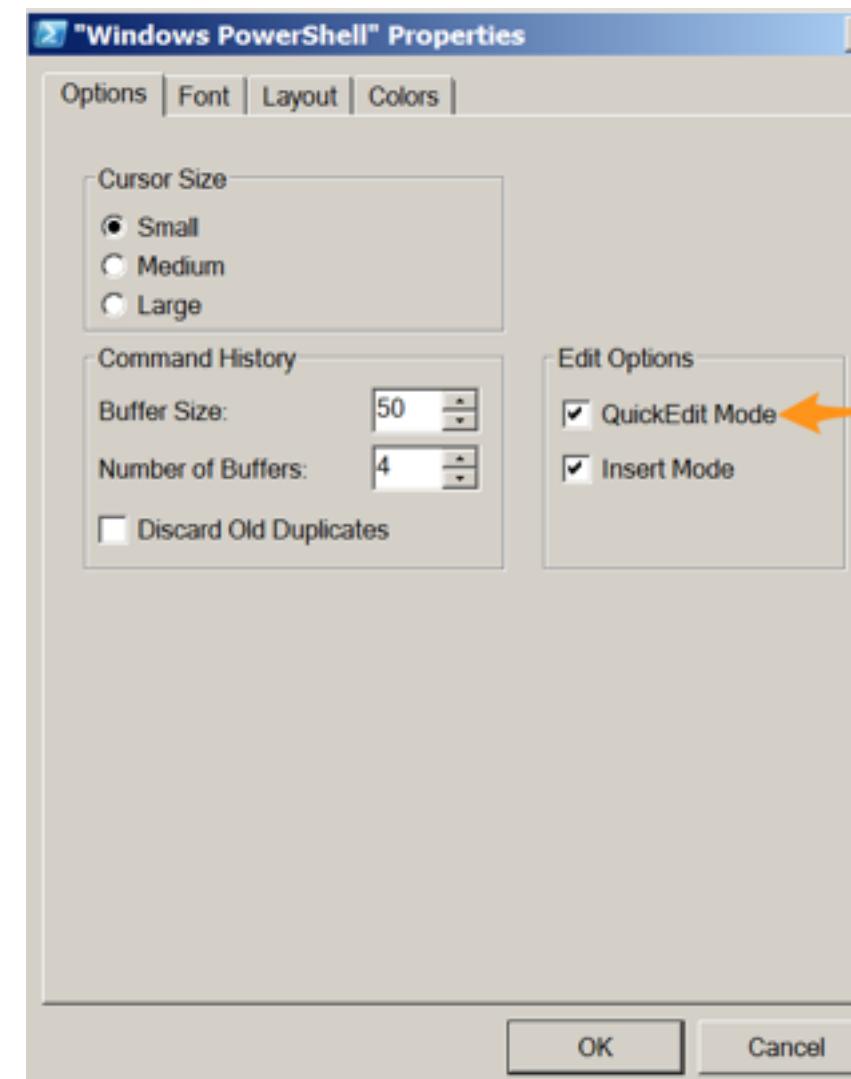
Configure PowerShell

- Run PowerShell as Administrator and open Properties



Configure PowerShell

- Enable QuickEdit Mode
- Set Height buffer to 9999



Workstation Setup

- <http://downloads.chef.io/>
- Select Chef-Client
- 2008 (Windows 7) or 2012 (Windows 8)
- i686 (32-bit) or x86_64 (64-bit)

have been tested
and are supported.

Windows

7

x86_64

11.16.4-1

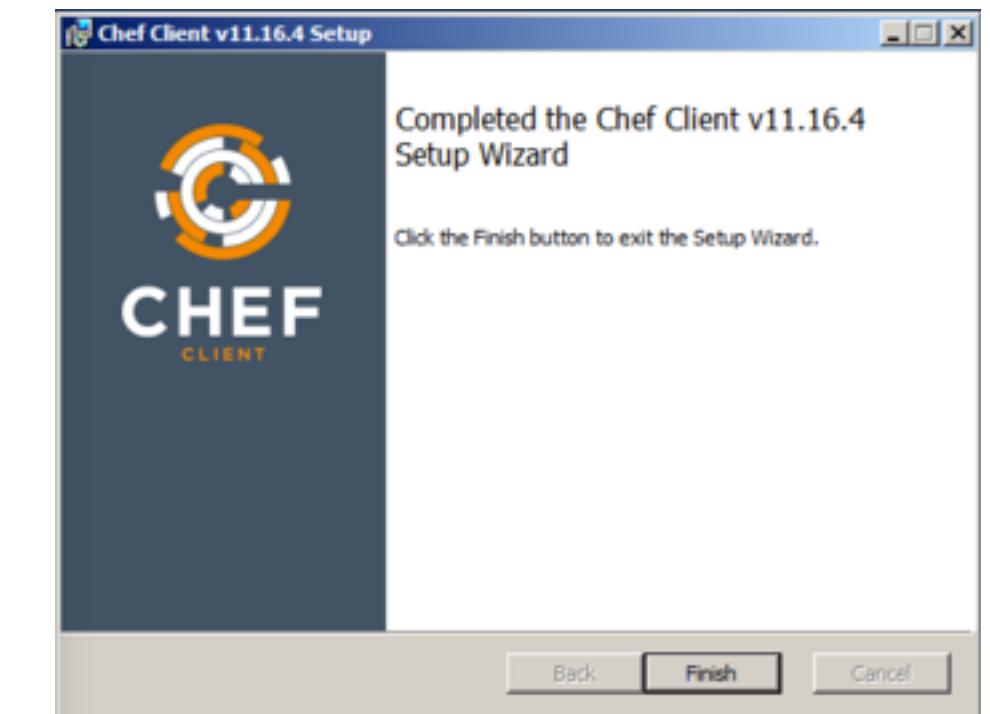
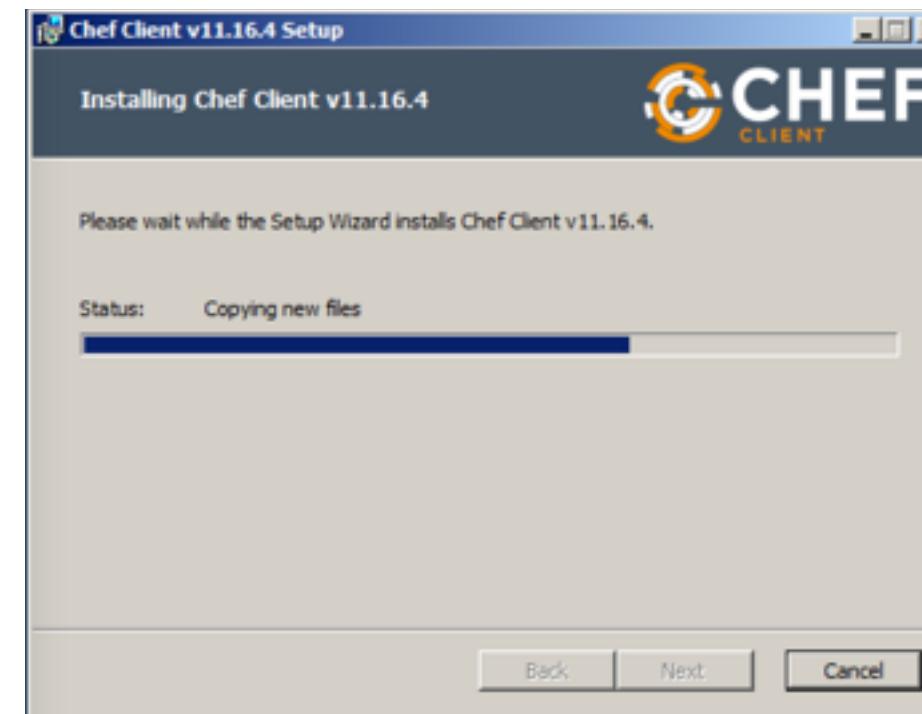
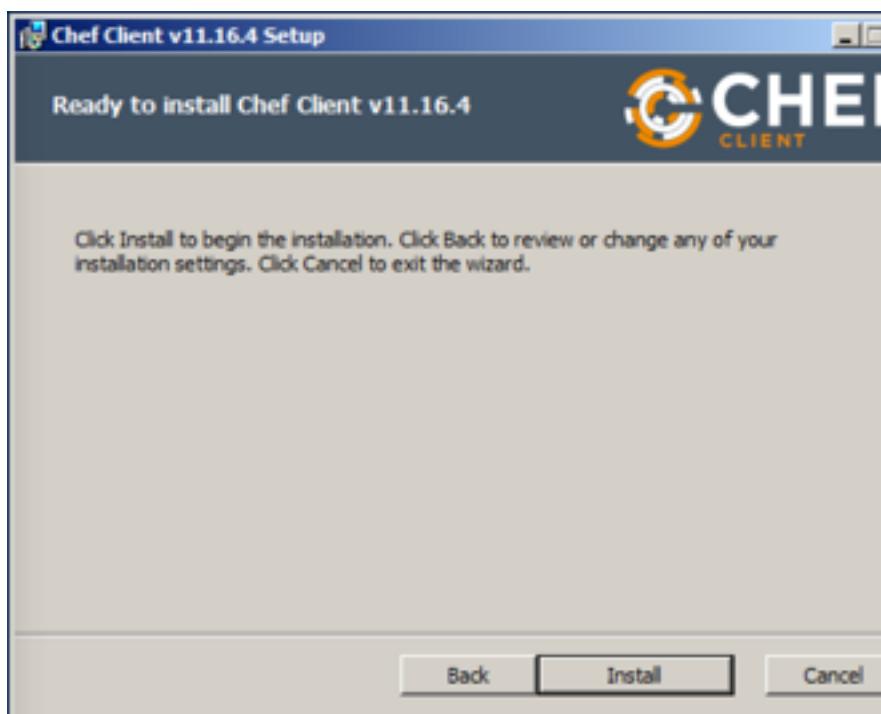
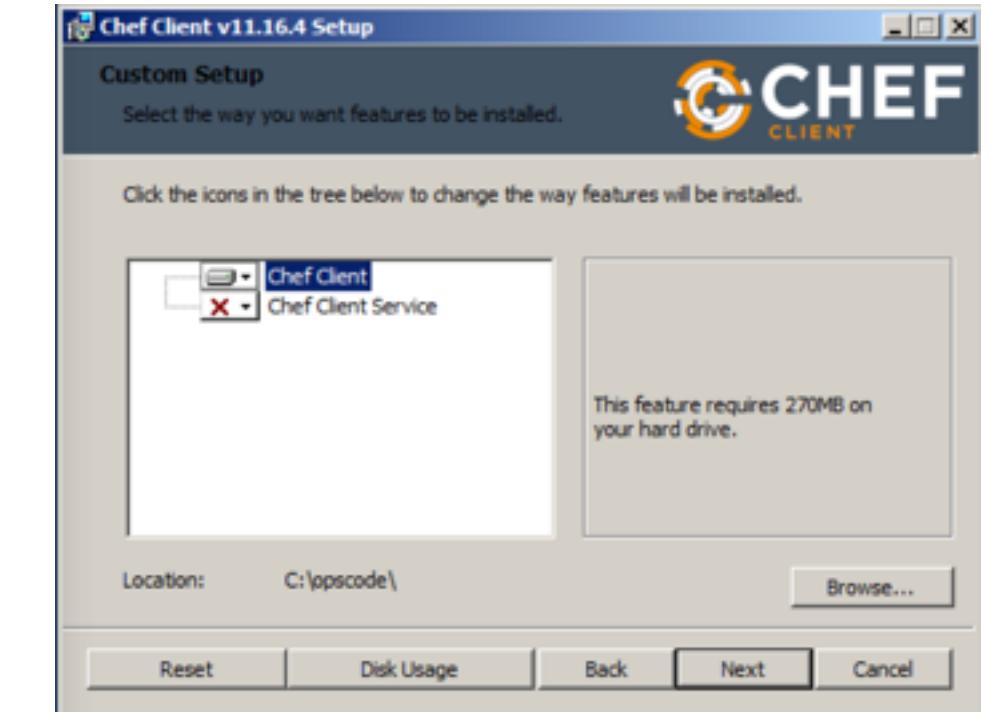
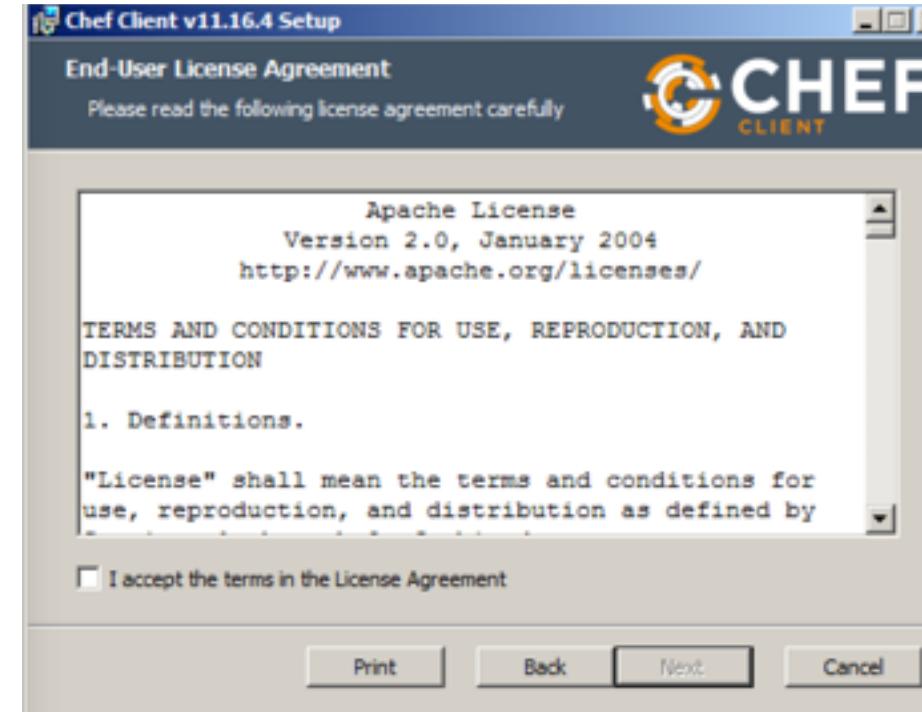
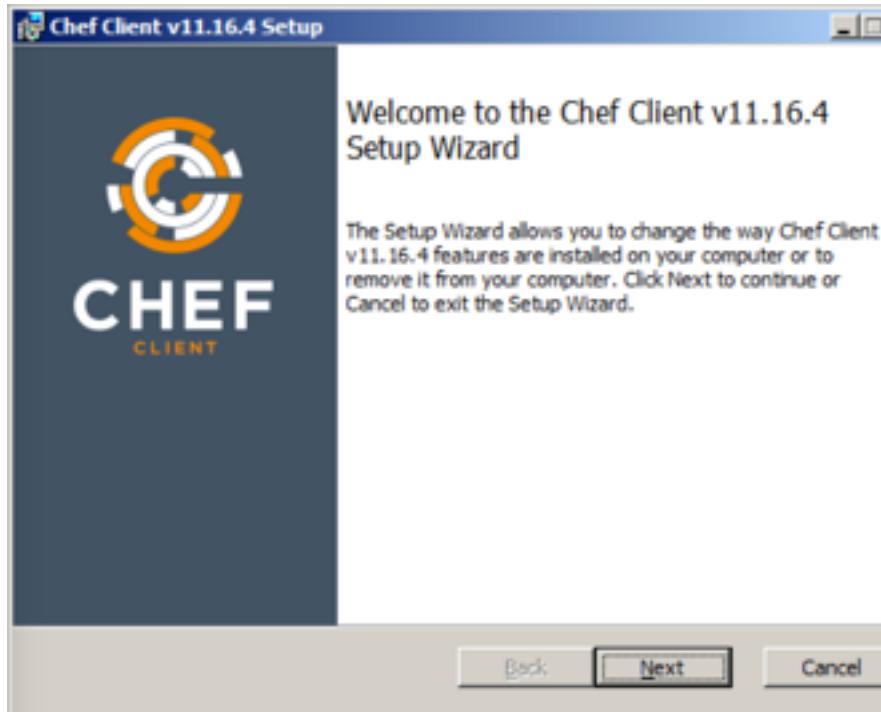
chef-windows-11.16.4-1.window



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

Download and install this file

Install on Windows



Workstation Setup - Mac OS X / Linux

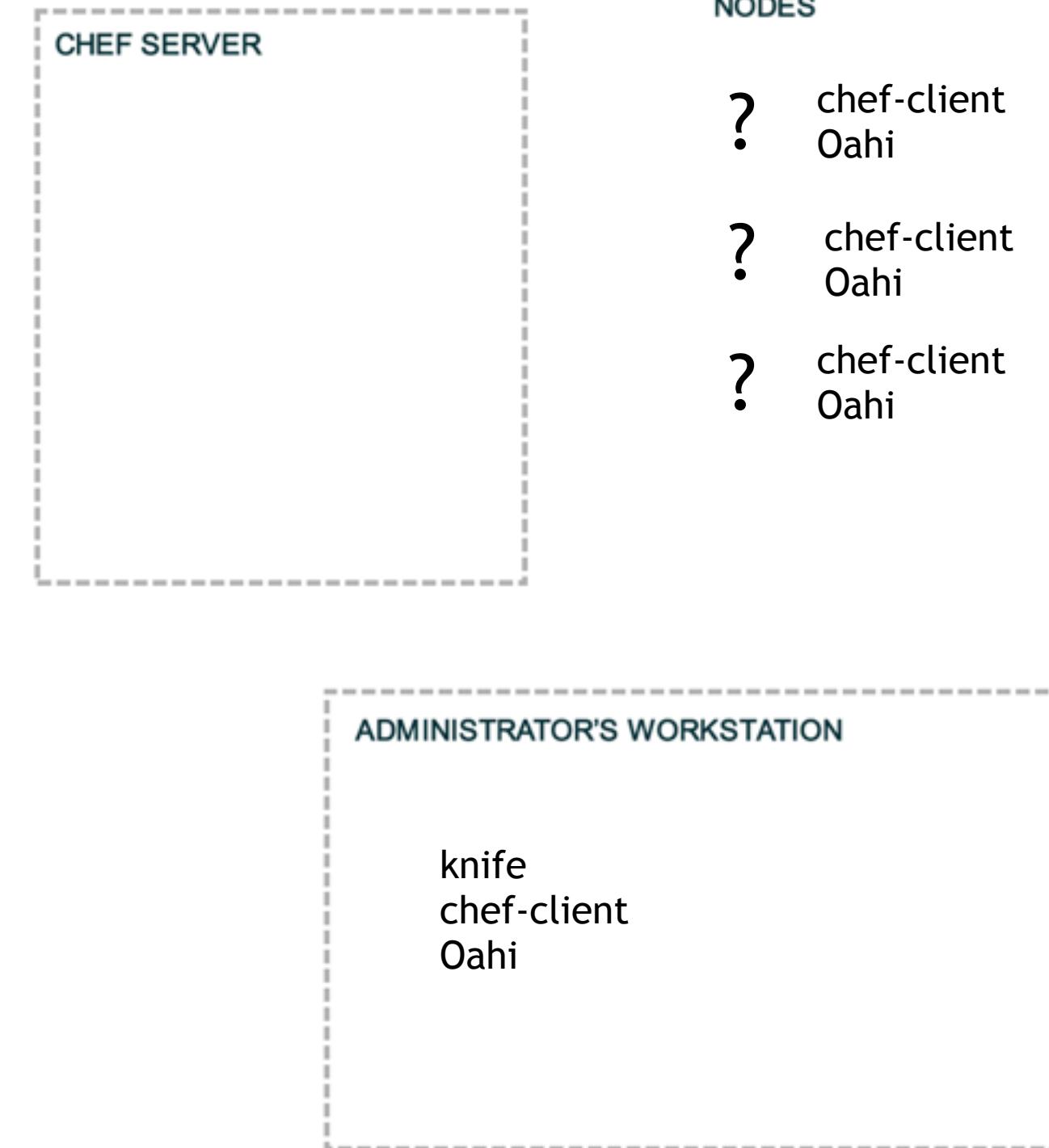
```
$ curl -L http://www.chef.io/chef/install.sh | sudo bash
```

```
% 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
```

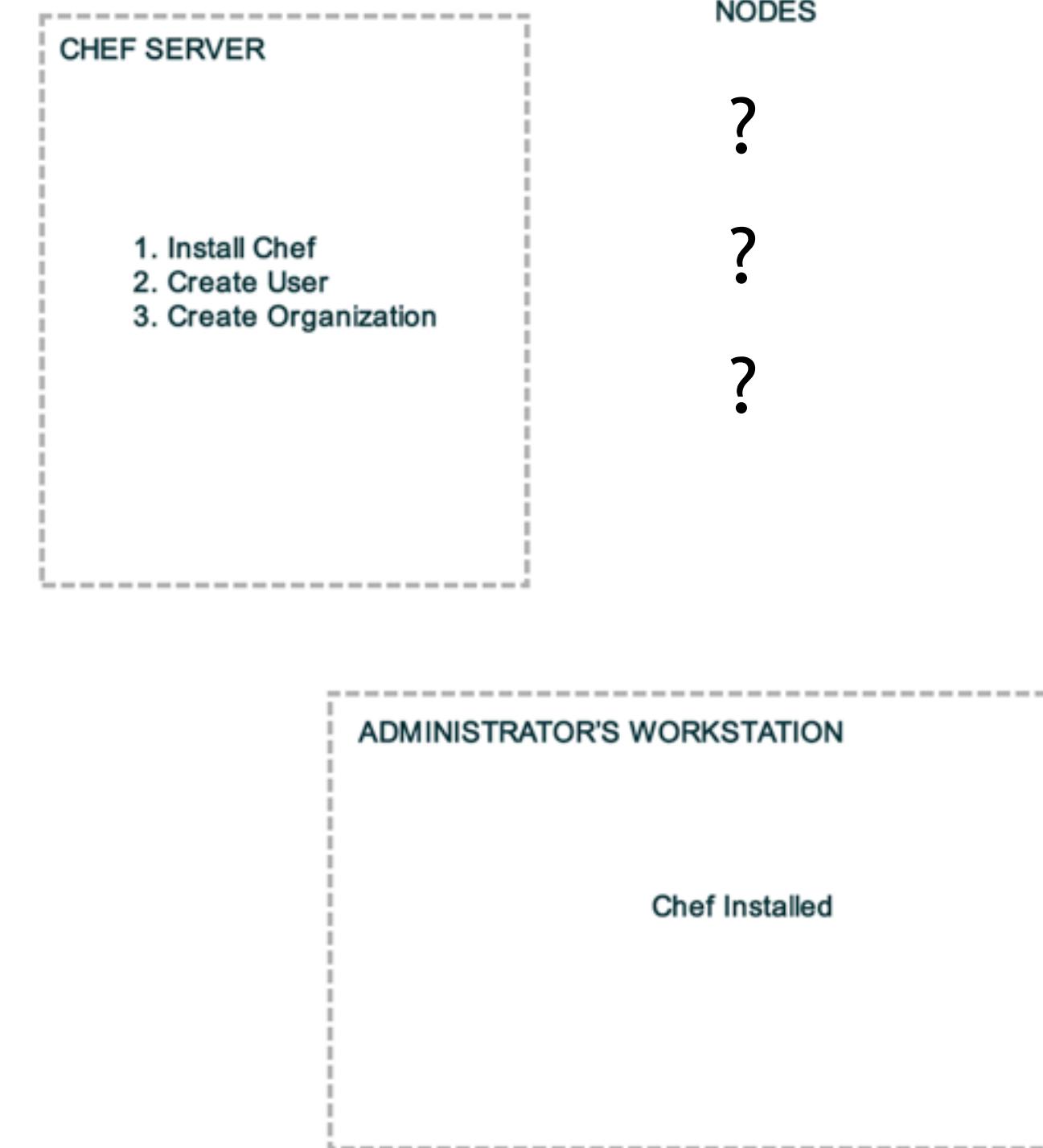
What just happened?

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

Workstation or Node?



Landscape of a Chef-managed Infrastructure



Your Chef Server for this class...

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

The image shows a screenshot of the Chef website. At the top, there is a navigation bar with links: 'WHAT IS CHEF?', 'LEARN CHEF', 'RESOURCES', and a prominent orange 'GET CHEF' button. Below this, a large banner features the text 'Choose your installation' and a subtext: 'Looking for the Chef Development Kit, Chef Client or Open Source Chef Server? [Start here.](#)' On the left, there is a section titled 'Simplification for Web-Scale IT' with a subtext: 'Delivers fast, scalable, flexible IT automation.' A large blue arrow points from the word 'HOSTED' in the 'HOSTED CHEF' section towards the 'GET HOSTED CHEF' button. The 'HOSTED CHEF' section contains the text 'HOSTED CHEF' and 'Let us manage Chef Server for you'. To the right, there is another section titled 'ON PREM CHEF' with the subtext 'Install & manage your own Chef Server' and a 'GET ON PREMISES CHEF' button. In the bottom left corner, there is a play button icon.

WHAT IS CHEF? LEARN CHEF RESOURCES GET CHEF

Choose your installation

Looking for the Chef Development Kit, Chef Client or Open Source Chef Server? [Start here.](#)

Simplification for Web-Scale IT

Delivers fast, scalable, flexible IT automation.

HOSTED CHEF

Let us manage Chef Server for you

GET HOSTED CHEF >

ON PREM CHEF

Install & manage your own Chef Server

GET ON PREMISES CHEF

Create new account

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

Start your free trial of Enterprise Chef

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

Full Name

Username

Email

Password

Company (Optional)

Chef Organization

Organization is the name of your instance of Enterprise Chef.

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

[Get Started](#)

Create new Organization

Welcome to
 **CHEF**
MANAGE

Thank you for using Chef!

You are not yet a member of any organizations, so please either create a new organization or accept a pending invitation.

If you are trying to join a specific organization and don't have any invitations, get someone in the organization to send you an invitation, then hit the refresh button.

[Sign Out](#)

[Create New Organization](#)

 [Accept Invite \(0 pending\)](#)

 **Create Organization** ×

Full Name (example: Chef, Inc.)
Any name. Doesn't have to be related to where you work

Short Name (example: chef)
i_watch_dora_the_exlorer

[Cancel](#)  [Create Organization](#)

Download Starter Kit

Download "Starter Kit" for your Org

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

Thank you for choosing Enterprise

Follow these three steps to be on your way:

[Download Starter Kit](#)

[Set up your Chef Infra](#)

What's next?

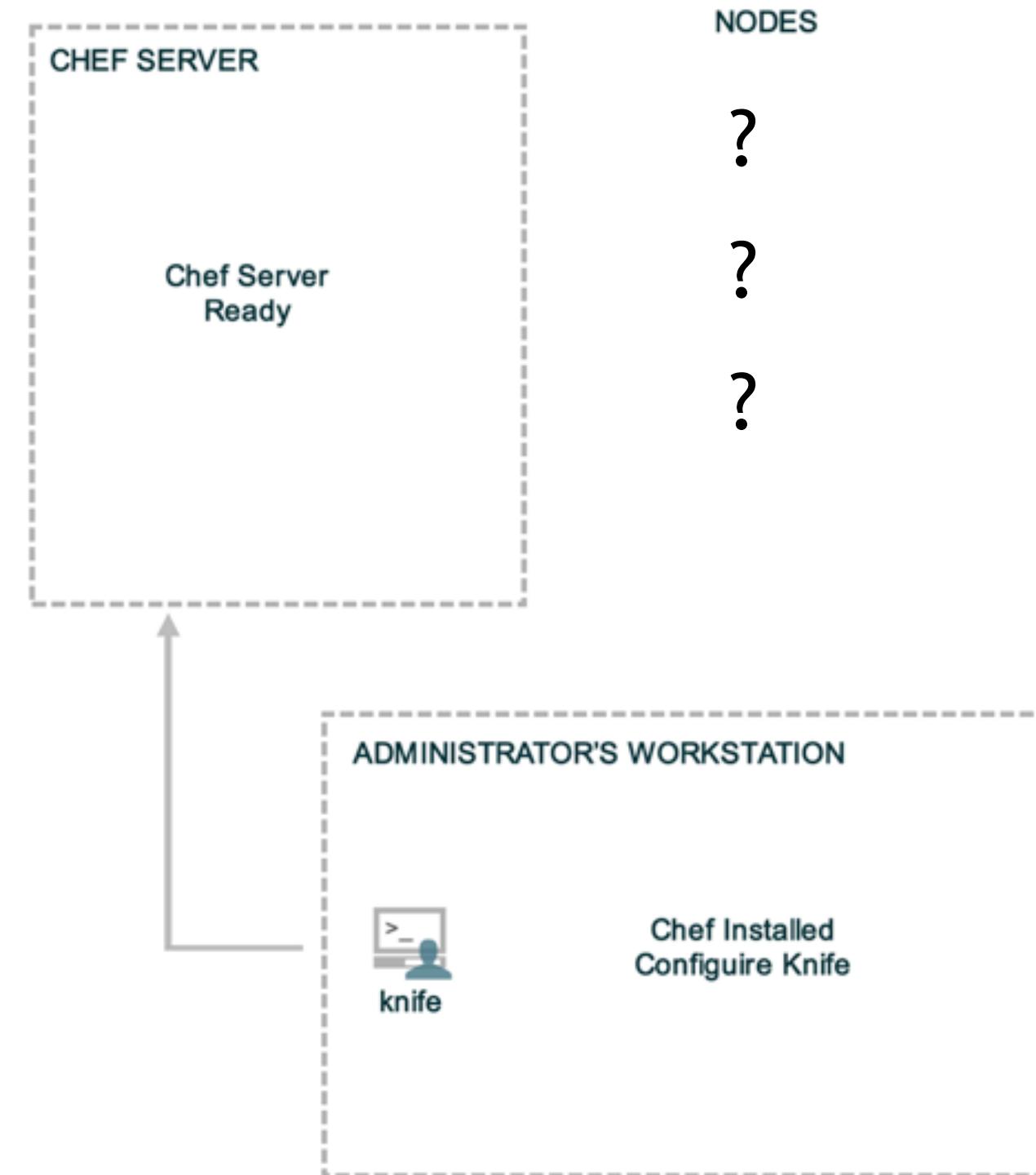
[Chef Documentation](#)

The best place to start learning about Chef in general.

[Browse Community Cookbooks](#)

Hundreds of members of the Chef community have contributed cookbooks you can use or draw inspiration from.

Landscape of a Chef-managed Infrastructure



A quick tour of the chef-repo

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

Verify that knife is working

```
PS\> cd SOMEDIR\chef-repo
```

```
PS \chef-repo>
```

A quick tour of the chef-repo

```
PS\> dir
```

Mode	LastWriteTime	Length	Name
----	-----	-----	-----
d----	9/24/2013 11:09 PM		.berkshelf
d----	9/24/2013 11:09 PM		.chef
d----	9/24/2013 11:09 PM		cookbooks
d----	9/24/2013 11:09 PM		roles
-a---	9/24/2013 11:09 PM	495	.gitignore
-a---	9/24/2013 11:09 PM	1433	Berksfile
-a---	9/24/2013 11:09 PM	588	chefignore
-a---	9/24/2013 11:09 PM	2292	README.md
-a---	9/24/2013 11:09 PM	3572	Vagrantfile

What's inside the .chef directory?

```
PS\> dir .chef
```

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

What's inside the .chef directory?

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

Knife is the command-line tool for Chef

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

knife.rb

- Default location
 - `~/.chef/knife.rb` (OSX)
 - `c:\Users\You\.chef\knife.rb` (Windows)
- Use a project specific configuration
 - `<project>/.chef/knife.rb` of the current directory
 - `chef-repo/.chef/knife.rb`
- http://docs.chef.io/config_rb_knife.html

knife.rb

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

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

Verify Knife

```
PS \> knife --version  
Chef: 11.16.4
```

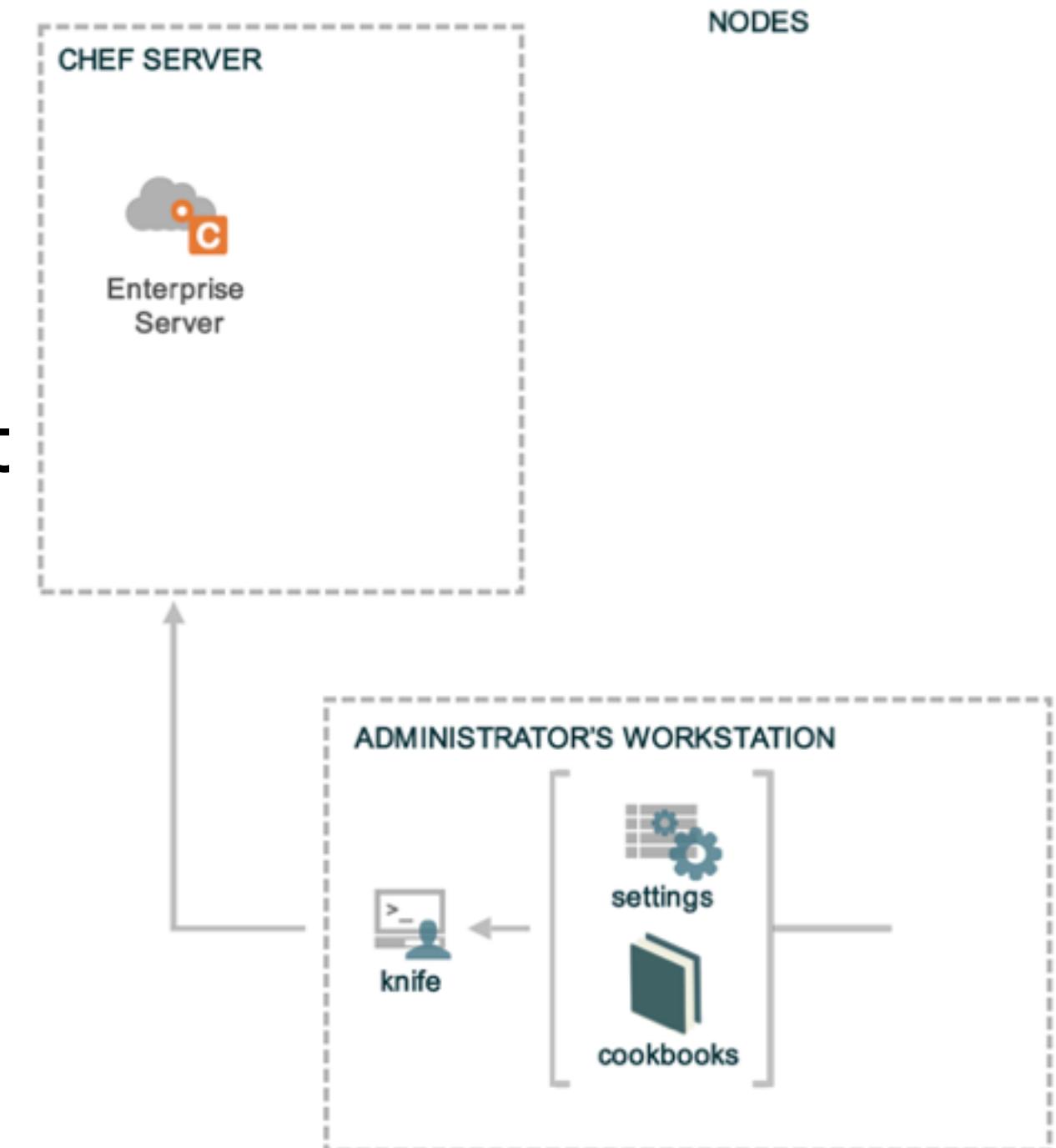
```
PS \> knife client list  
ORGNAME-validator
```

- Reopen PS for new path to take effect
- Your version may be different, that's ok!



knife client list

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



Exercise: Run ‘knife help list’

```
PS \> knife help list
```

Available help topics are:

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

Exercise: Run ‘knife client list’

```
PS \> knife client list
```

ORGNAME-validator

A few knife tips

- Commands are always structured as follows
 - `knife`
 - `NOUN` (`client`)
 - `VERB` (`list`)
- You can get more help with
 - `knife NOUN help`
 - `knife --help` just shows options

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

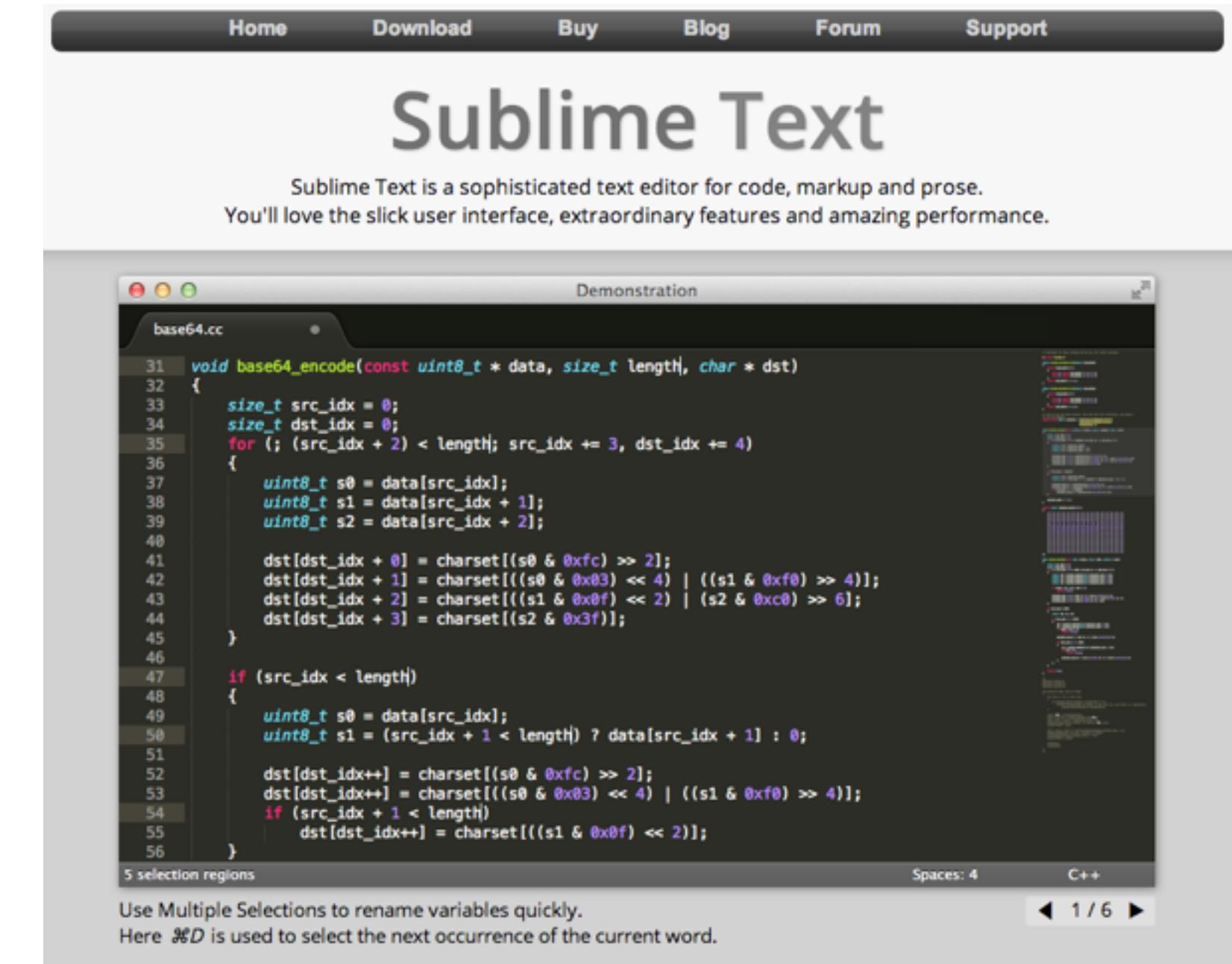
- Chef is about Infrastructure as Code
- People who code for a living use text editors that are designed for the task
 - Vim, Emacs, Sublime Text, Notepad++, etc.

Benefits of a good text editor

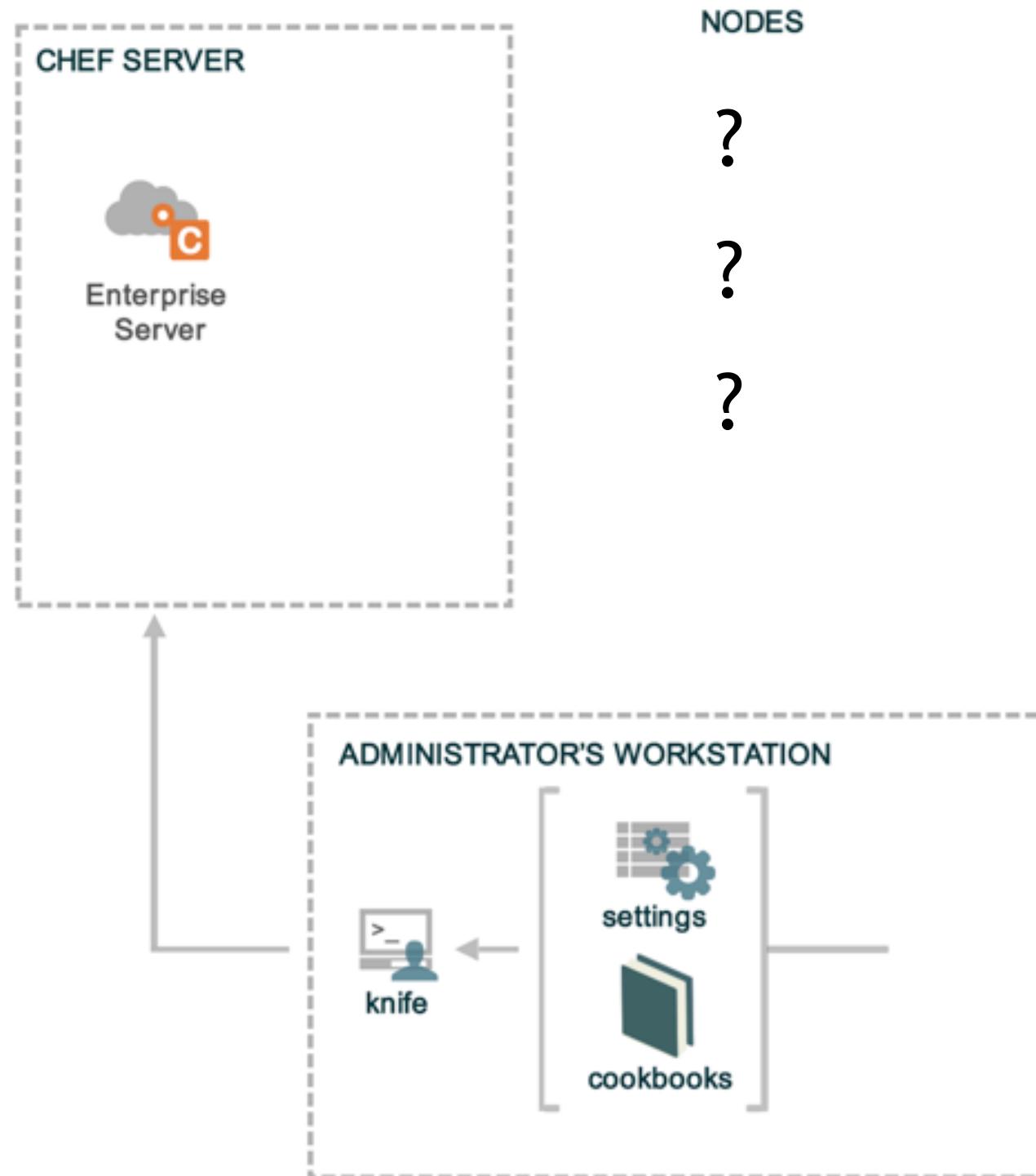
- A good editor will
 - Show line numbers
 - Highlight syntax
 - Autocomplete commands
 - Allow you to open multiple files

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

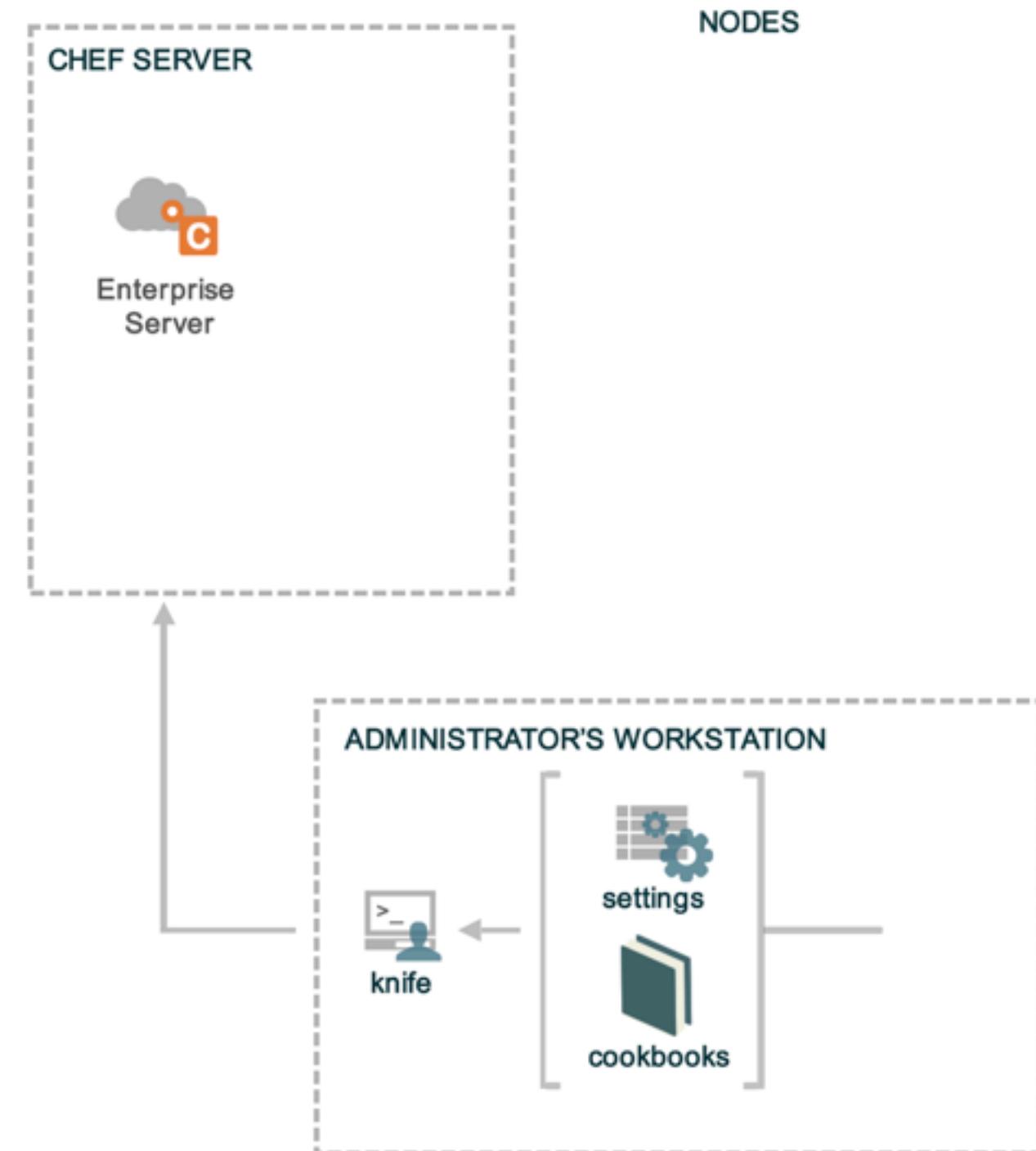
- Download Sublime Text
 - Free trial, not time bound
 - Works on every platform
- sublimetext.com



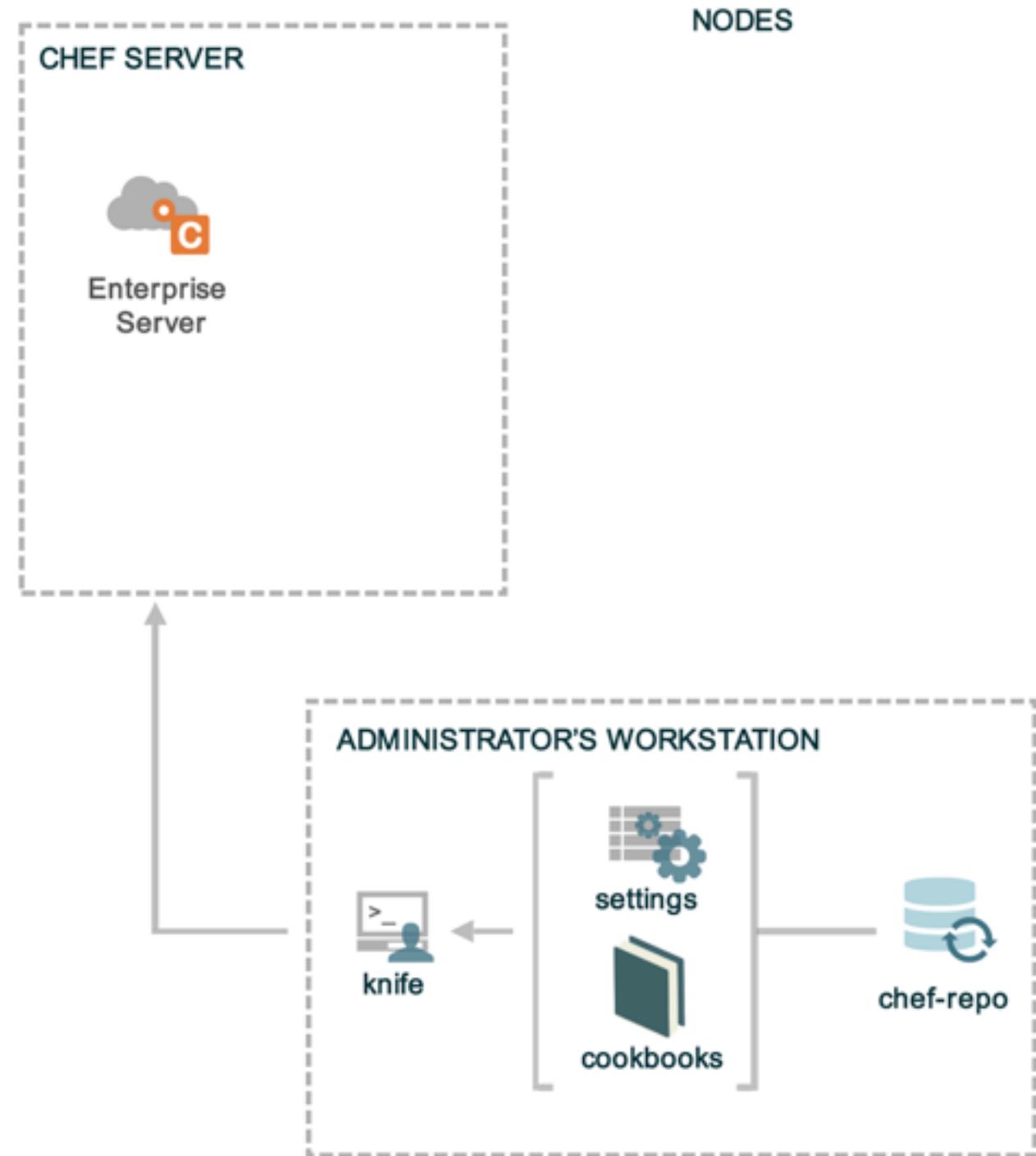
Checkpoint



What's Next?



Source Code Repository



Review Questions

- What is the chef-repo?
- What is knife?
- What is name of the knife configuration file?
- Where is the knife configuration file located?
- What is your favorite text editor? :)

Organization Setup

Setup an Organization

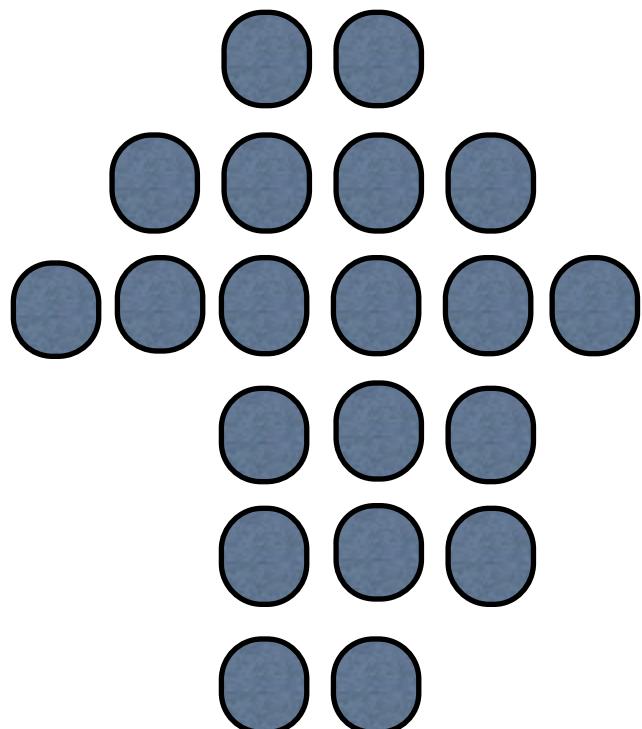
v2.1.1_WIN

Lesson Objectives

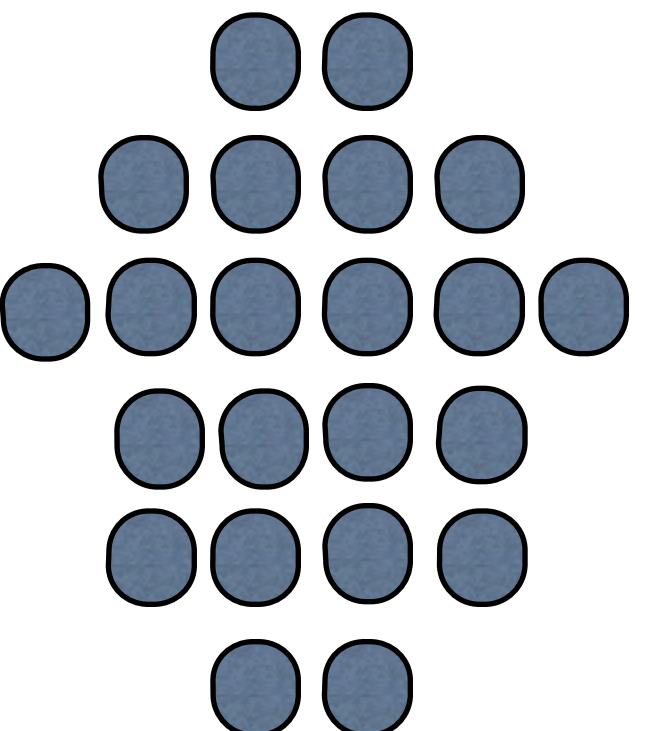
- After completing the lesson, you will be able to
 - Explain the purpose of Organizations
 - Manage your Chef Organization
 - Invite users into your organization
 - Accept invitations into other organizations

Organizations

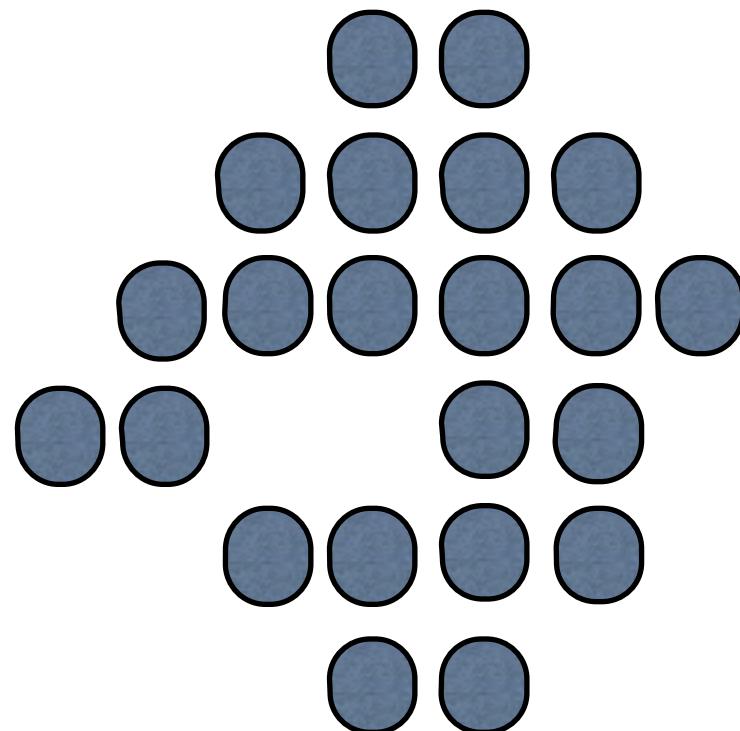
My Infrastructure



Your Infrastructure



Their Infrastructure

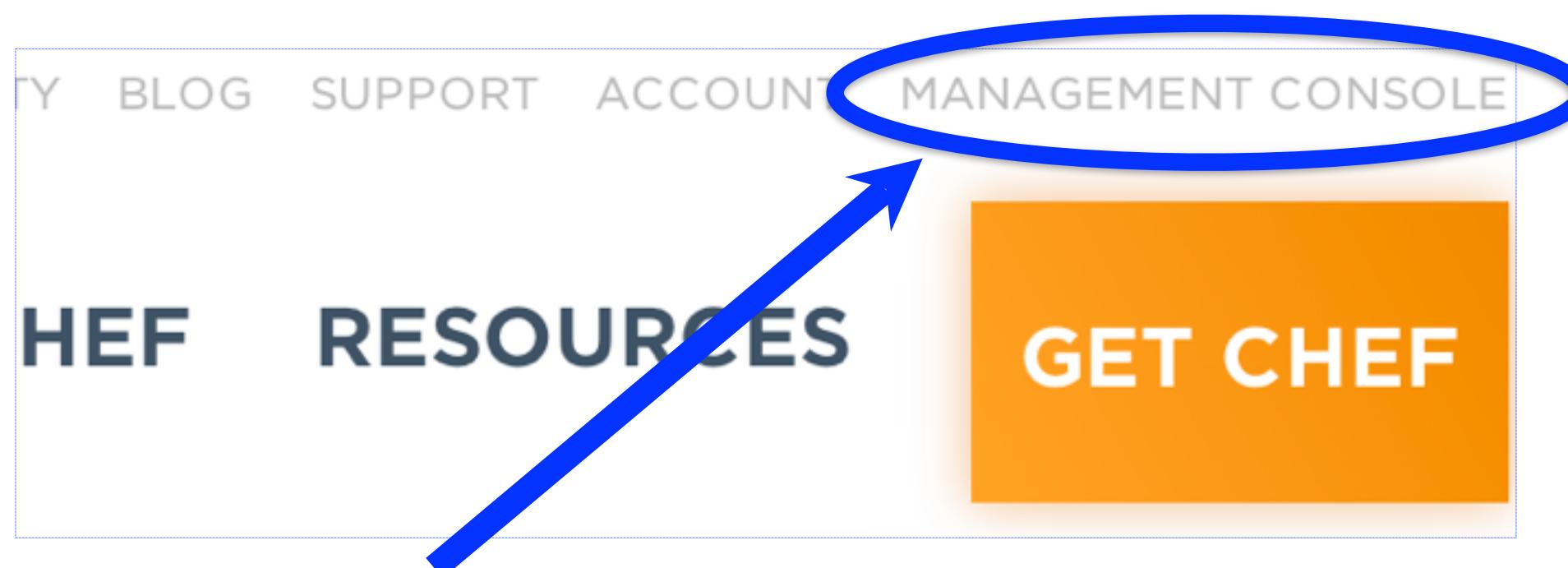


Organizations

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

Manage Organizations

- Login to your Hosted Enterprise Chef (chef.io)
- Select Management Console



Sign In

Username

Password

Sign In

Forgot your password?

Organizations

The screenshot shows the CHEF MANAGE web application interface. At the top, there is a dark header bar with the CHEF MANAGE logo on the left and four navigation tabs: Nodes, Reports, Policy, and Administration. The Administration tab is highlighted with an orange border. Below the header is a sidebar on the left containing links for creating organizations, generating validation keys, knife config files, inviting users, leaving organizations, and a starter kit. The main content area on the right is titled "Showing All Organizations" and lists one organization named "i_watch_dora_the_explorer".

**CHEF
MANAGE**

Nodes Reports Policy Administration

> Organizations

Create
Reset Validation Key
Generate Knife Config
Invite User
Leave Organization
Starter Kit

Users

Groups

Global Permissions

Showing All Organizations

Organization

i_watch_dora_the_explorer

Manage Organizations

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

The screenshot shows the CHEF MANAGE web interface. The top navigation bar includes links for Nodes, Reports, Policy, and Administration, with Administration being the active tab. The top right corner shows a user profile for 'i_watch_dora_the_explorer' and notification counts of 0 for messages and 0 for users.

The main content area is titled 'Showing All Organizations'. A table lists one organization:

Organization	Actions
i_watch_dora_the_explorer	 ▾

A context menu is open over the 'i_watch_dora_the_explorer' row, listing the following actions:

- Reset Validation Key
- Generate Knife Config
- Invite User
- Leave Organization
- Starter Kit

The left sidebar contains a navigation menu with the following items:

- > Organizations
 - Create
 - Reset Validation Key
 - Generate Knife Config
 - Invite User
 - Leave Organization
 - Starter Kit
- Users

Manage Organizations

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

Exercise: Invite users into your org

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

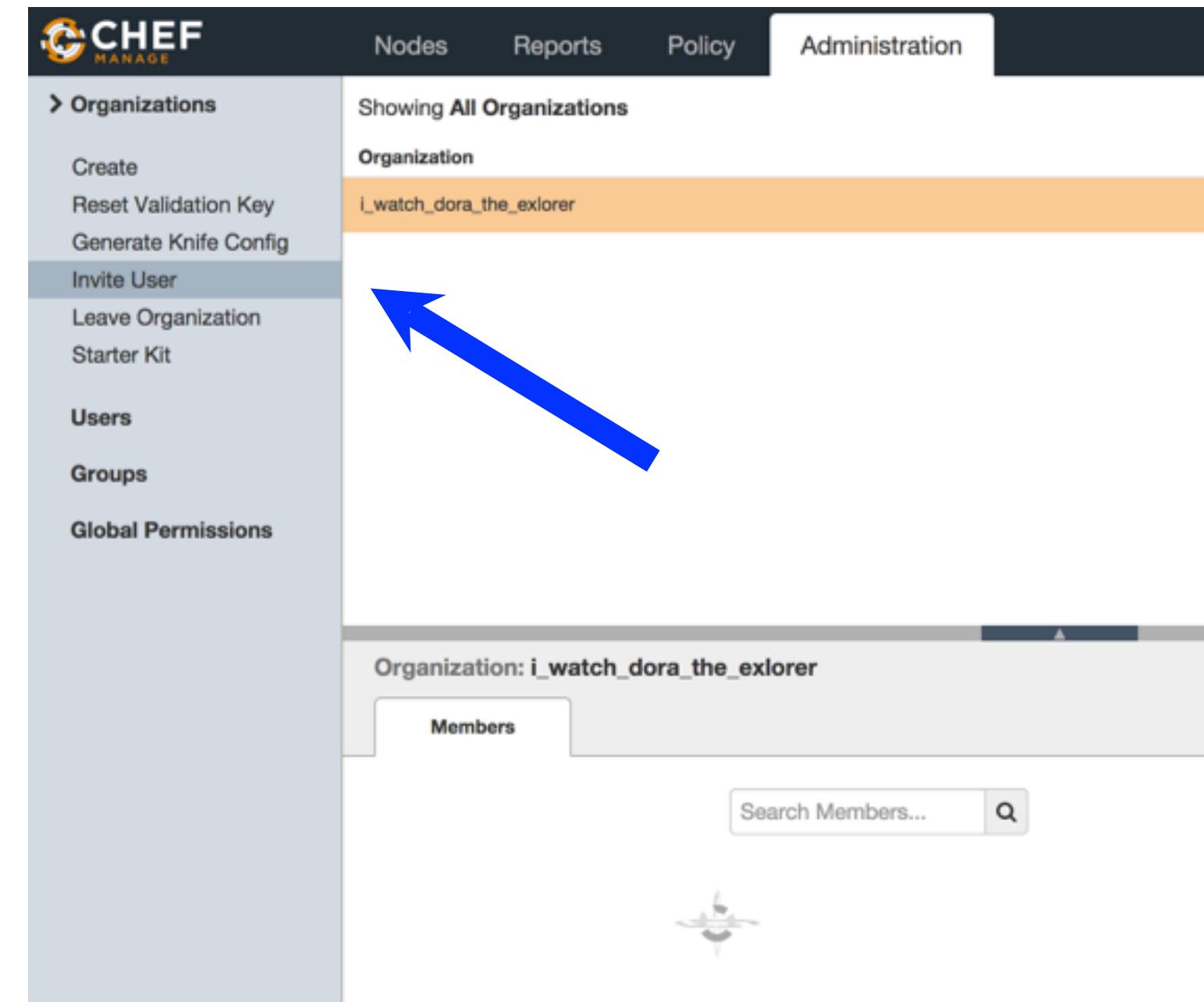
Exercise: Invite Users into your Org

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

Exercise: Invite a classmate into your org

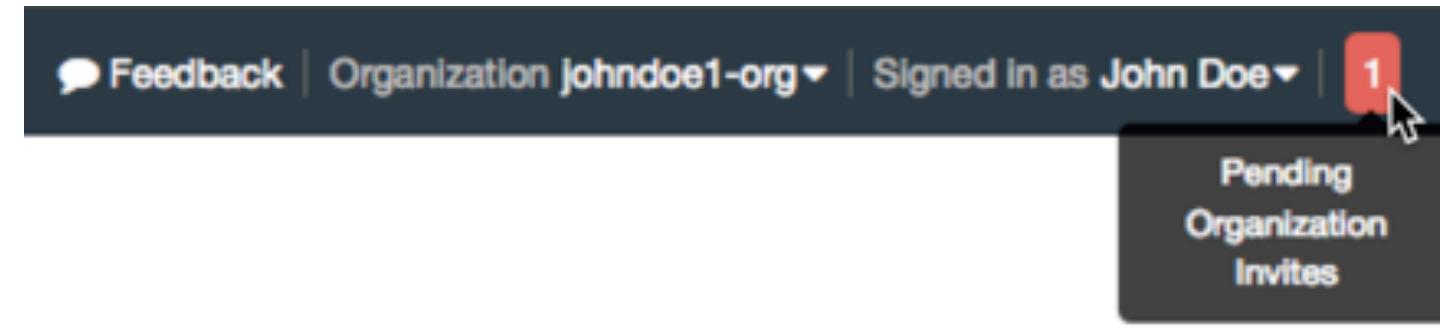
- Navigate to <http://manage.chef.io>

- Click the "Administration" tab,
- Select the appropriate Organization
- Click "Invite User" from the left menu
- Enter your classmate's 'Chef Username' and click Invite

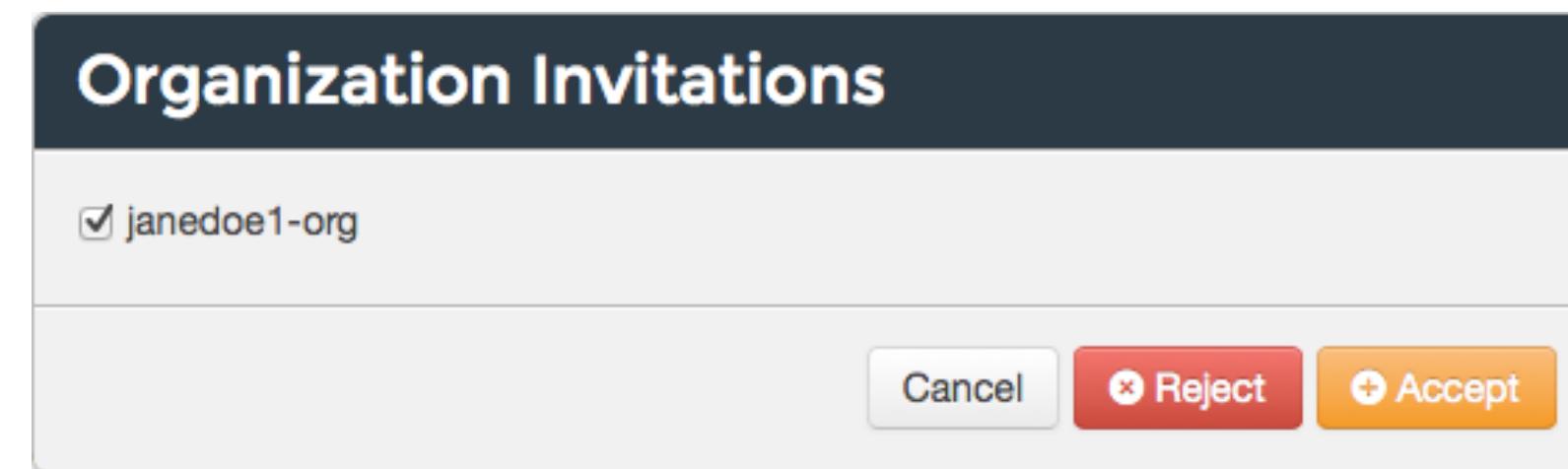


Exercise: Accept an invite

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



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



Exercise: View classmate's org

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

The screenshot shows the Opscode Manage interface for managing organizations. On the left, there's a sidebar with options like 'Organizations', 'Create', 'Reset Validation Key', etc. The main area shows a list of organizations: 'janedoe1-org' and 'johndoe1-org'. The 'johndoe1-org' row is highlighted with an orange background. At the top right, there's a dropdown menu showing 'Organization: johndoe1-org'. This dropdown is circled in orange. Below the list, there's a section for 'Organization: johndoe1-org' with tabs for 'Members' and 'Search Members...'. There's also a table with a single entry: 'Name: johndoe1' with a 'Remove' button.

Copyright © 2012-2013 Opscode, Inc.

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

Feedback What's New? About Opscode Manage

Exercise: Revoke access

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

The screenshot shows the Opscode Manage web interface. The top navigation bar includes links for Nodes, Reports, Policy, and Administration. The Administration tab is selected. On the left, a sidebar under the 'Organizations' heading lists 'Create', 'Reset Validation Key', 'Generate Knife Config', 'Invite User', 'Leave Organization' (which is circled in orange), and 'Starter Kit'. The main content area displays a table titled 'Showing All Organizations' with a single row for 'johndoe1-org'. Below this, a modal window is open for the 'johndoe1-org' organization, specifically the 'Members' tab. It shows a search bar and a table with one entry: 'Name: johndoe1'. To the right of this entry is a 'Remove' button, which is also circled in orange.

Review Questions

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

Test Node Setup

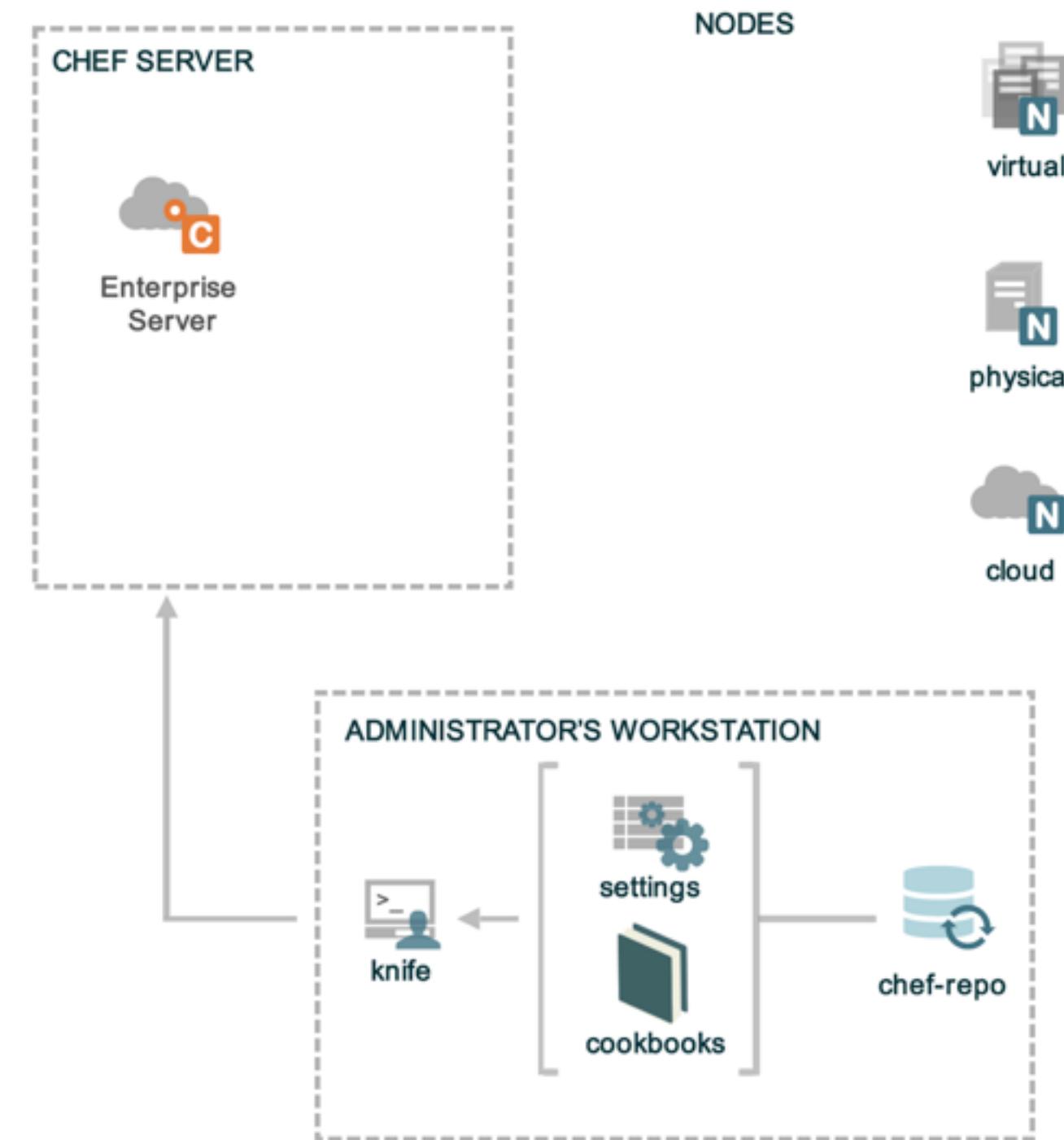
Setup a node to manage

v2.1.1_WIN

Lesson Objectives

- After completing the lesson, you will be able to
 - Install chef-client on a node using "knife bootstrap"
 - Explain how knife bootstrap configures a node to use the Organization created in the previous section
 - Explain the basic configuration needed to run chef-client

Nodes



Install Windows plugin for Knife

```
PS \> gem install knife-windows
```

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

Nodes

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

We Have No Nodes Yet

The screenshot shows the Chef Manage web application. The top navigation bar includes the Chef logo and the word "MANAGE". Below the logo, there is a breadcrumb trail: "Nodes > Nodes". The main content area has a dark header with four tabs: "Nodes" (which is active), "Reports", "Policy", and "Administration". The main body of the page displays the message "Showing All Nodes" and a message box stating "There are no items to display." On the left side, there is a sidebar with several options: "Delete", "Manage Tags", "Reset Key", "Edit Run List", and "Edit Attributes".

CHEF
MANAGE

> Nodes

Nodes Reports Policy Administration

Showing All Nodes

There are no items to display.

Delete
Manage Tags
Reset Key
Edit Run List
Edit Attributes

Training Node

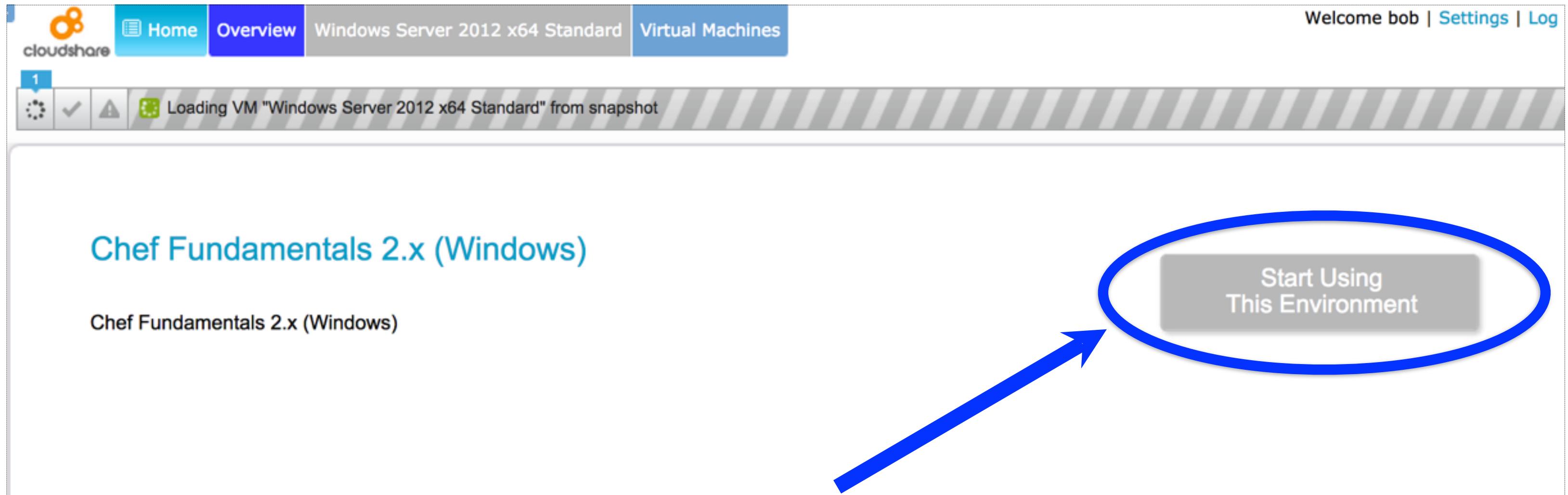
- The labs require a node to be managed
- We use the CloudShare training environment
 - Windows Server 2012 x64
 - 2GB RAM
 - 40GB HDD

CloudShare Node

- Class URL: <Class URL goes here>

CloudShare Node

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



CloudShare Node

1
Environment is ready

Chef Fundamentals 2.x (Windows)

Windows Server 2012

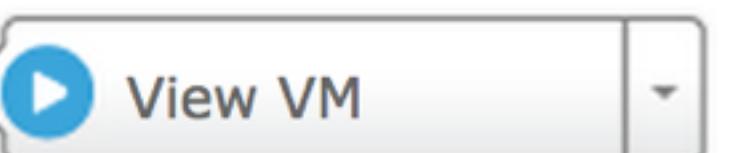
Windows Server 2012 x64 Standard

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

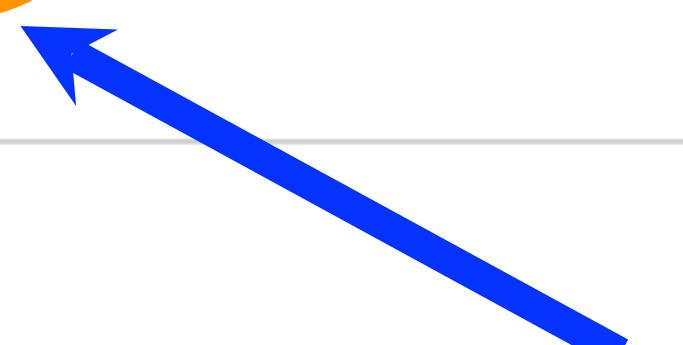
OS: Windows

State: Running

[More details ▶](#)

 View VM

 RDP file  Reboot VM  Revert



CloudShare Node

Chef Fundamentals 2.x (Windows)

Windows Server 2012 x64 Standard

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

OS: Windows
State: Running
[Hide details ▾](#)

VM Details

External Address: [uv01oi149k6q8wf7z27.vm.cld.sr](#) 

Internal IP: 10.160.33.236

Total Memory: 2048 MB

Disk Size: 40 GB

CPU: 1

The machine was prepared in 4 seconds

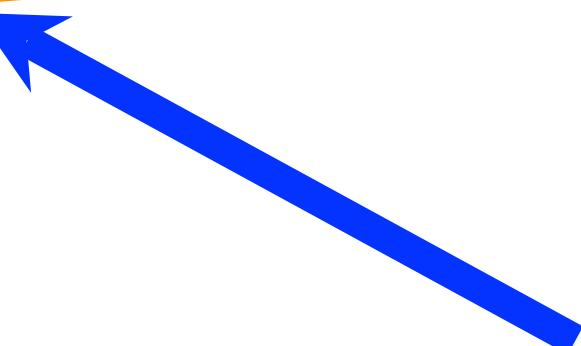
Credentials (show password)

Auto-login: Administrator (local user)

Username: Administrator

Password: *****

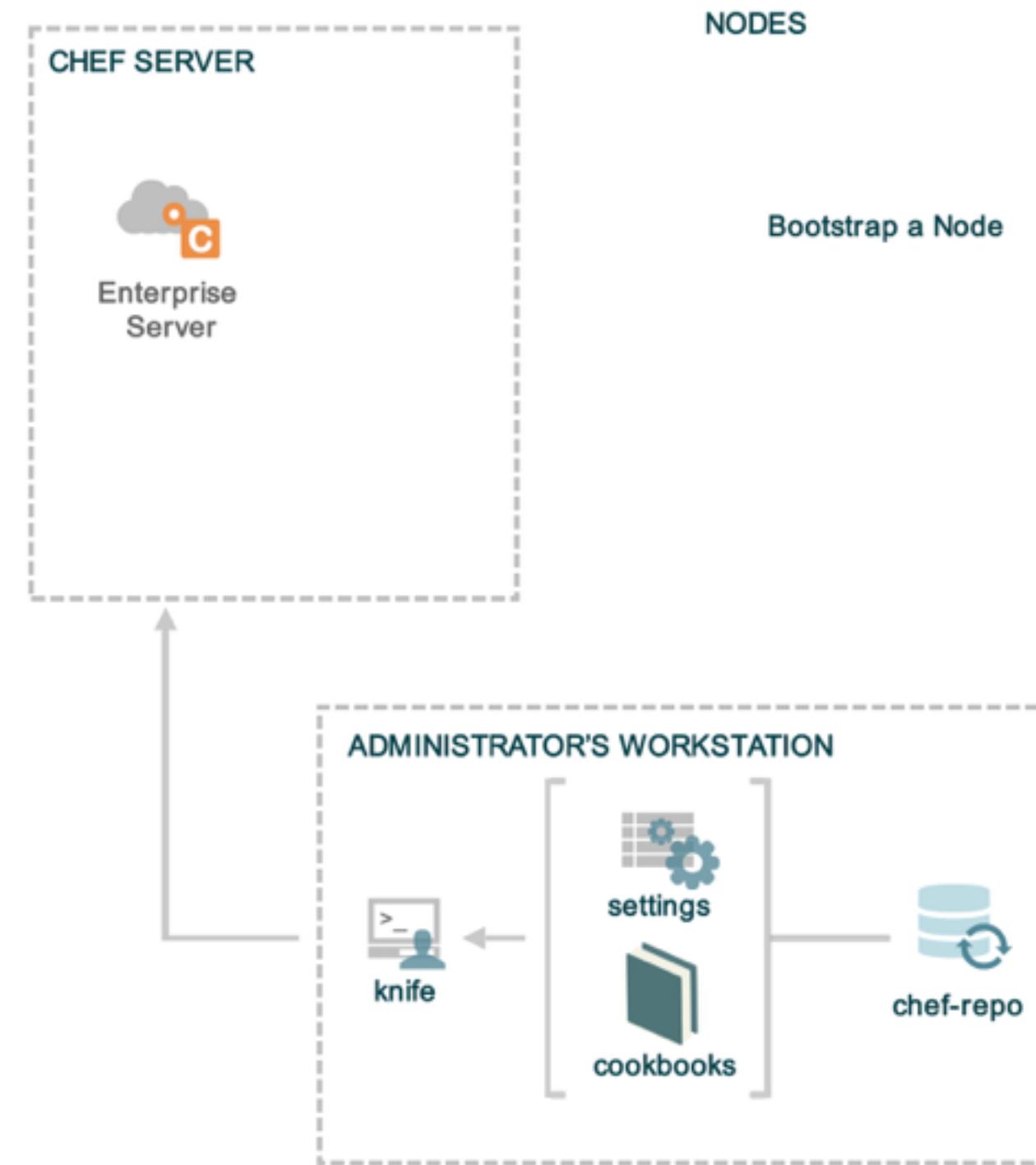
 RDP file  Reboot VM  Revert



Checkpoint

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

Checkpoint



"Bootstrap" the Target Instance

```
PS \> knife bootstrap windows winrm <ETERNAL_ADDRESS> -x chef -P chef -N "node1"
```

```
Bootstrapping Chef on uvolbv4gjw3ktiwifb3.vm.cld.sr
```

```
Enter your password:
```

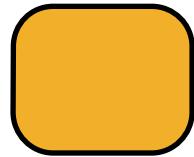
```
uvolbv4gjw3ktiwifb3.vm.cld.sr "Rendering "C:\Users\chef\AppData\Local\Temp\bootstrap-2000-139299860
uvolbv4gjw3ktiwifb3.vm.cld.sr Checking for existing directory "C:\chef"...
uvolbv4gjw3ktiwifb3.vm.cld.sr Existing directory not found, creating.
uvolbv4gjw3ktiwifb3.vm.cld.sr C:\Users\chef>(...
```

knife bootstrap

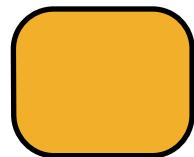
```
knife bootstrap windows winrm HOSTNAME -x ...
```



Chef Server



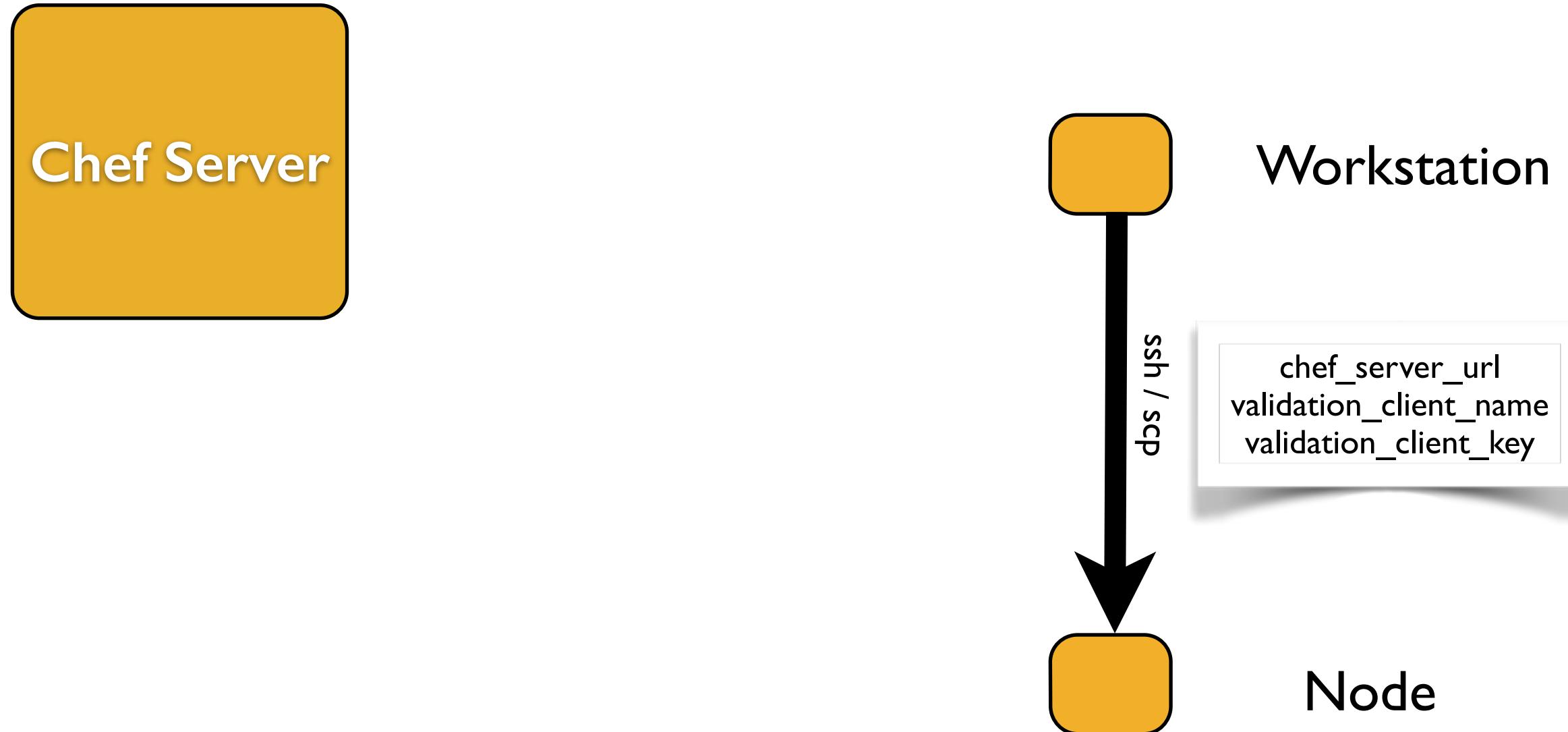
Workstation



Node

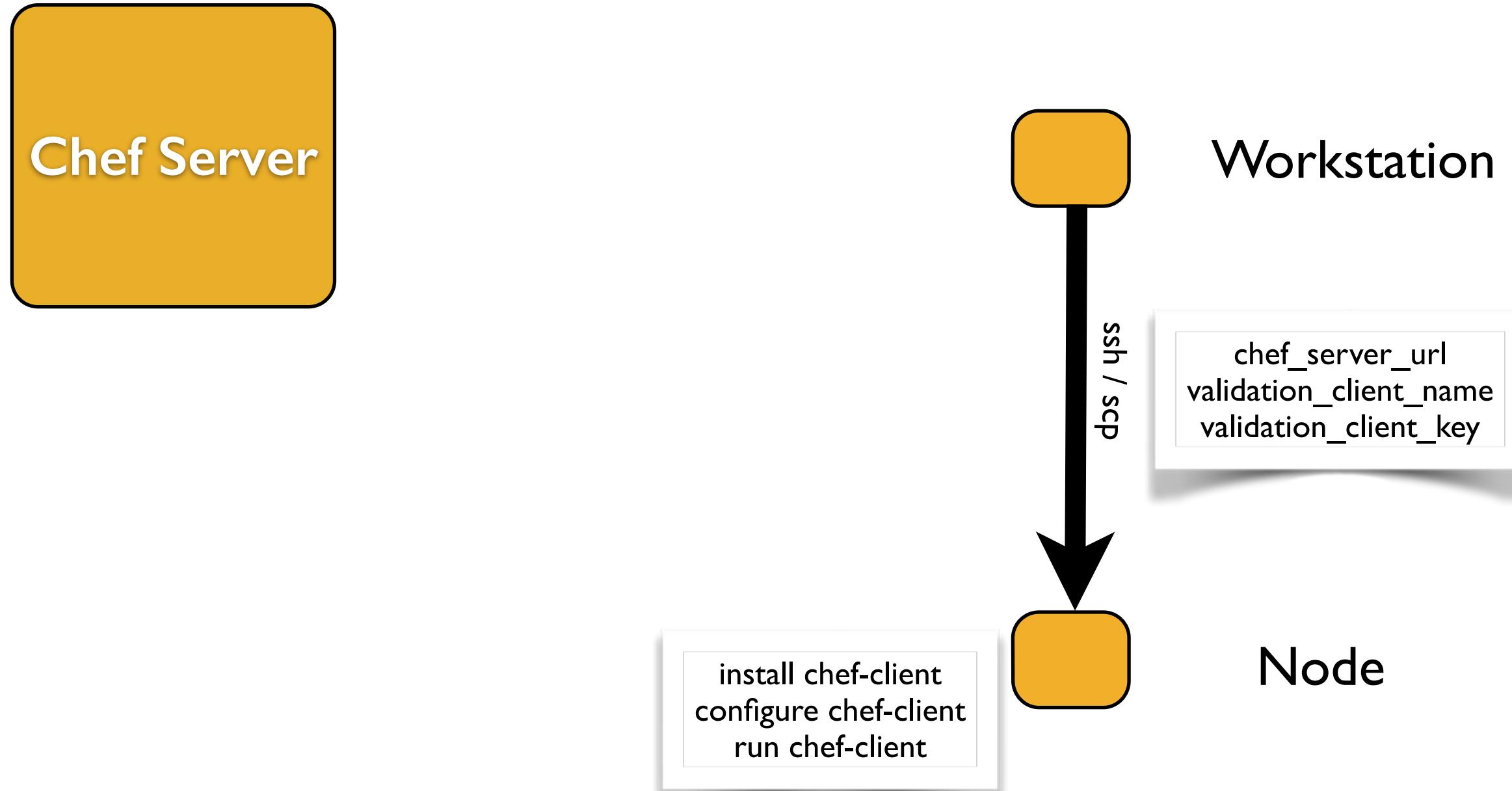
knife bootstrap

```
knife bootstrap windows winrm HOSTNAME -x ...
```



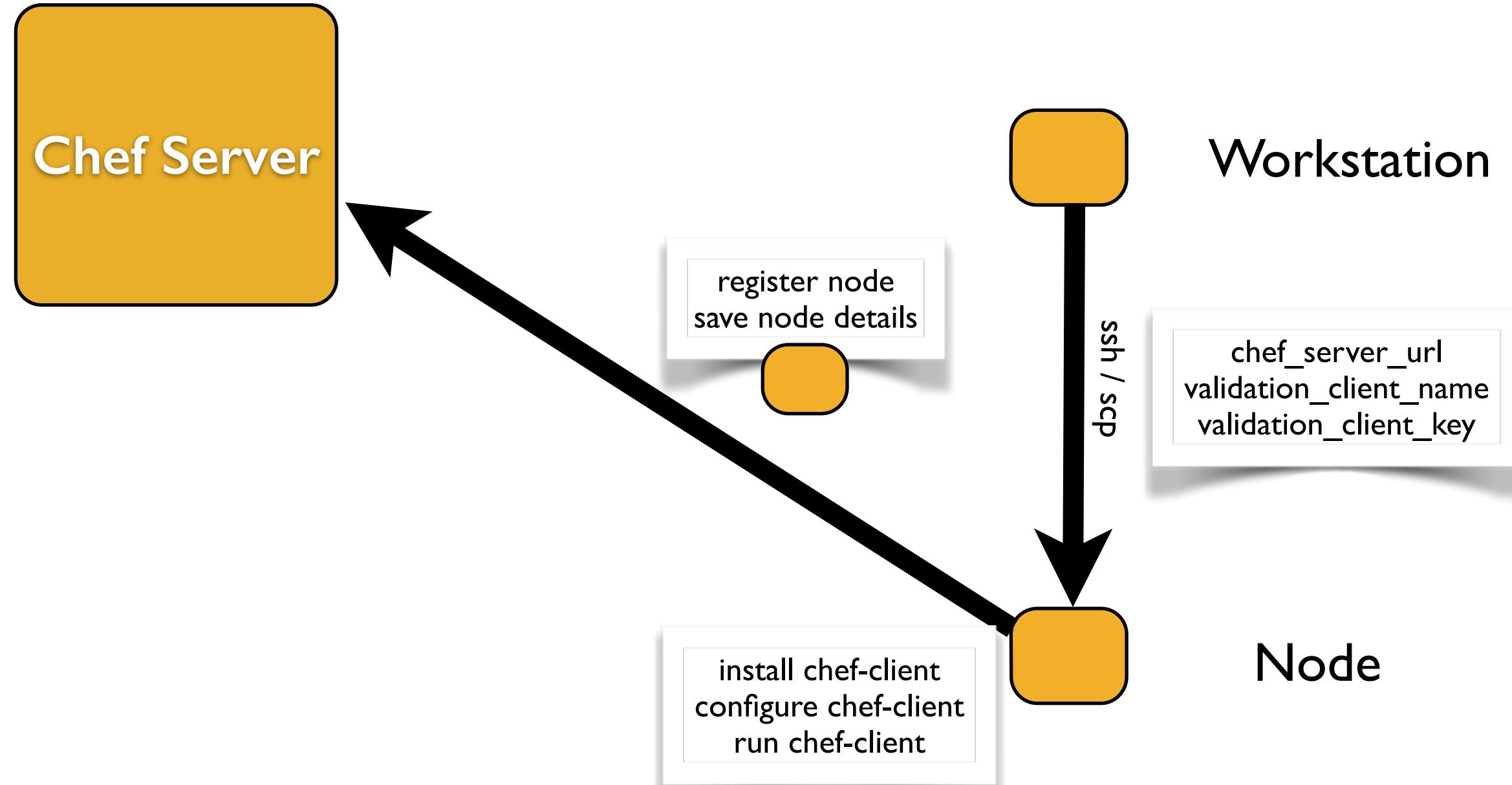
knife bootstrap

```
knife bootstrap windows winrm HOSTNAME -x ...
```



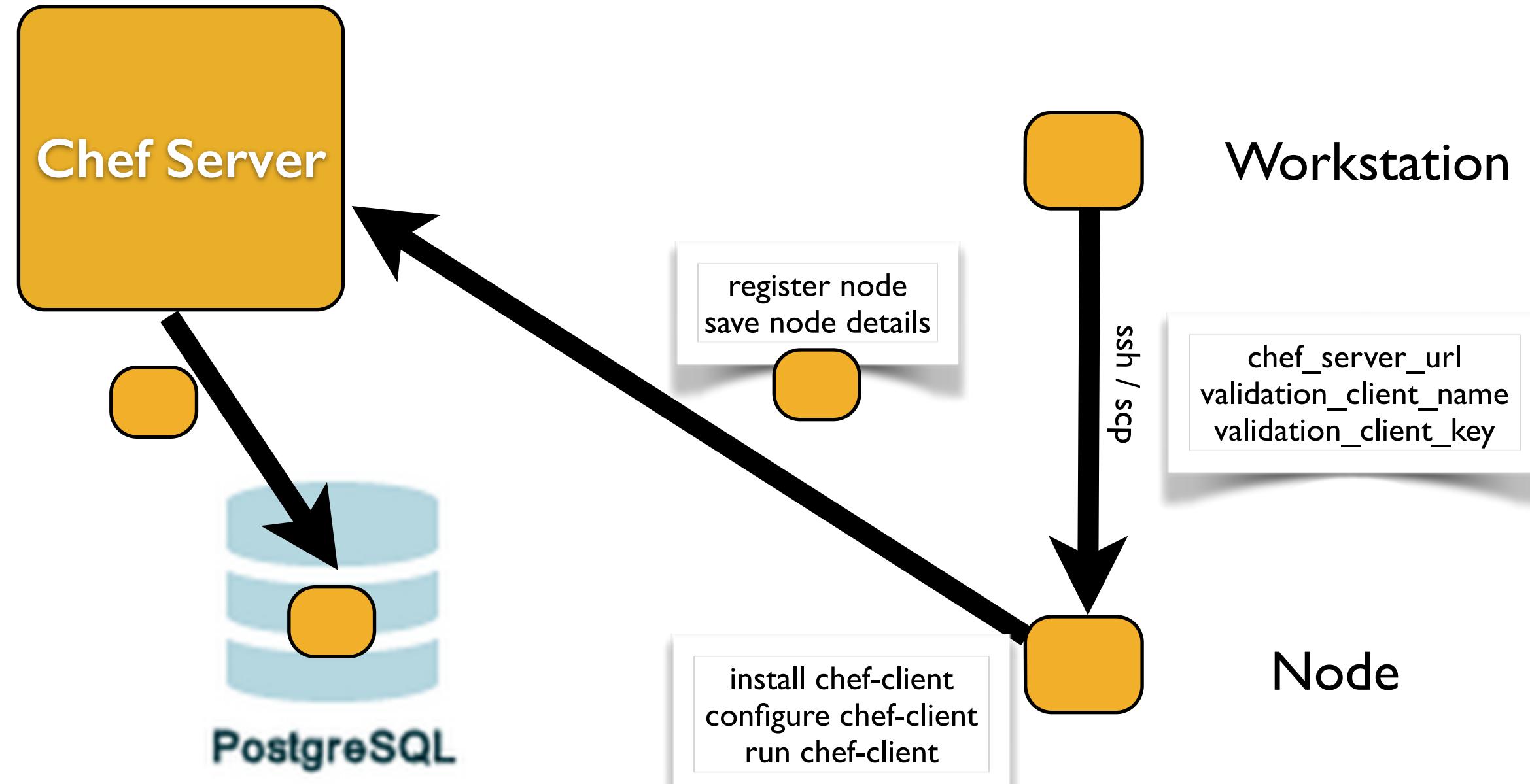
knife bootstrap

```
knife bootstrap windows winrm HOSTNAME -x ...
```



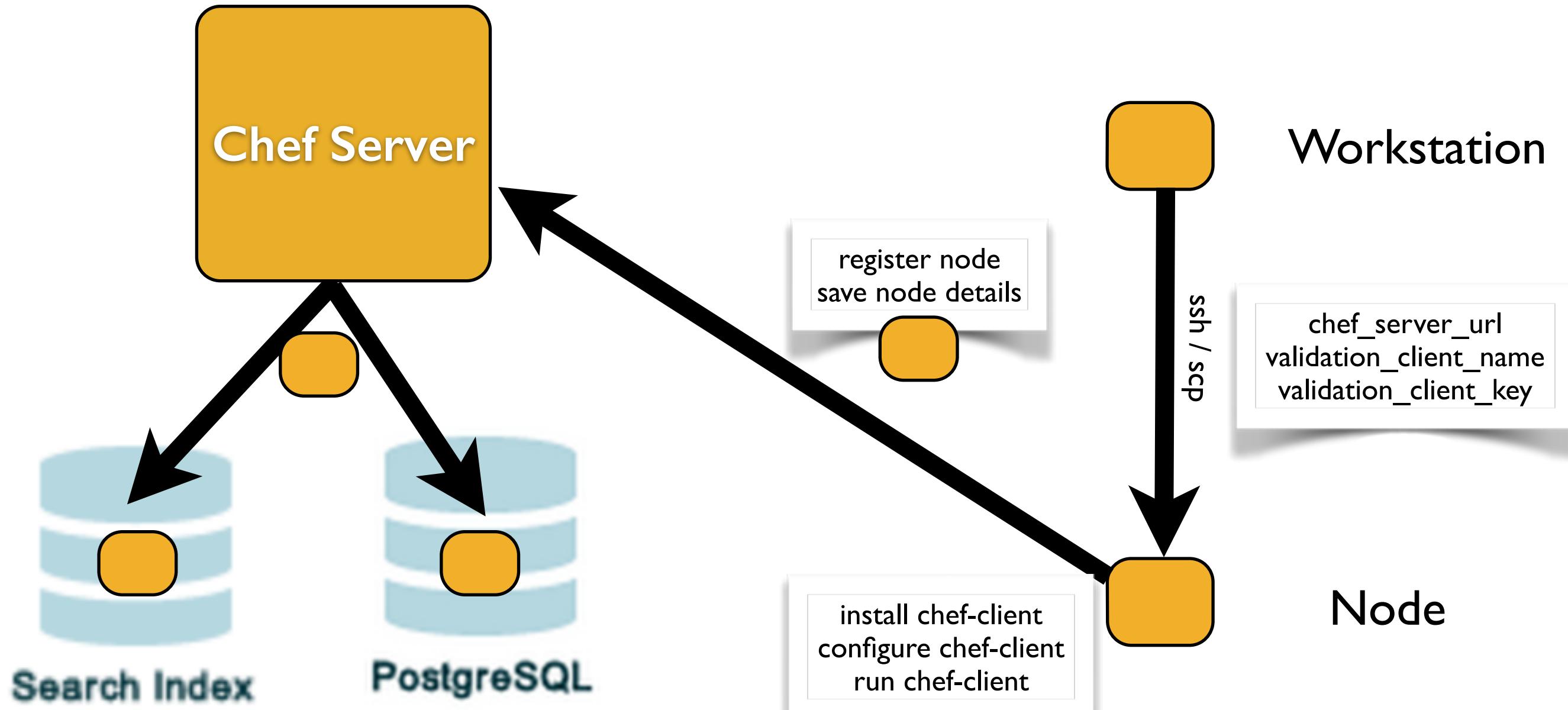
knife bootstrap

```
knife bootstrap windows winrm HOSTNAME -x ...
```



knife bootstrap

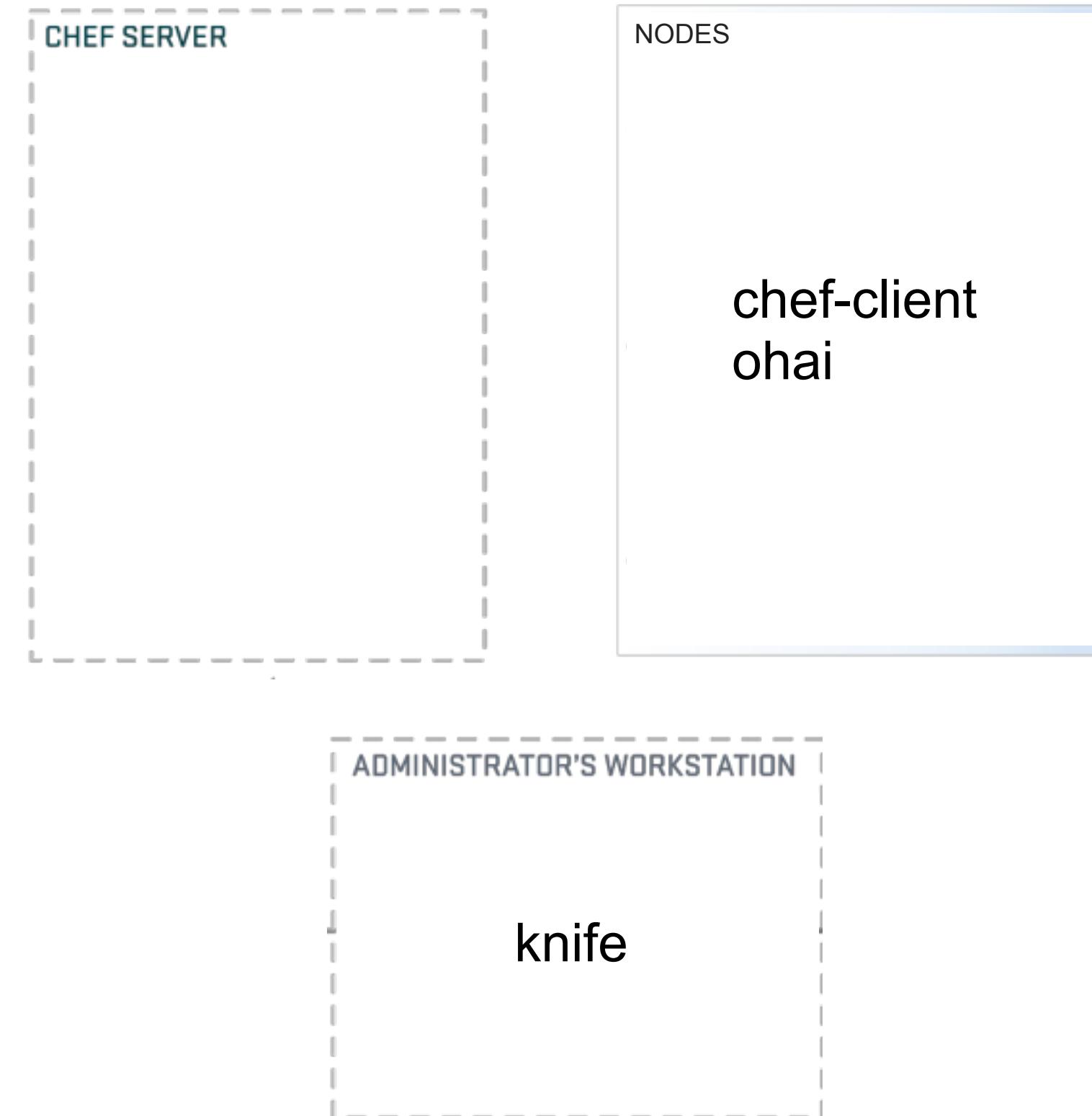
```
knife bootstrap windows winrm HOSTNAME -x ...
```



What just happened?

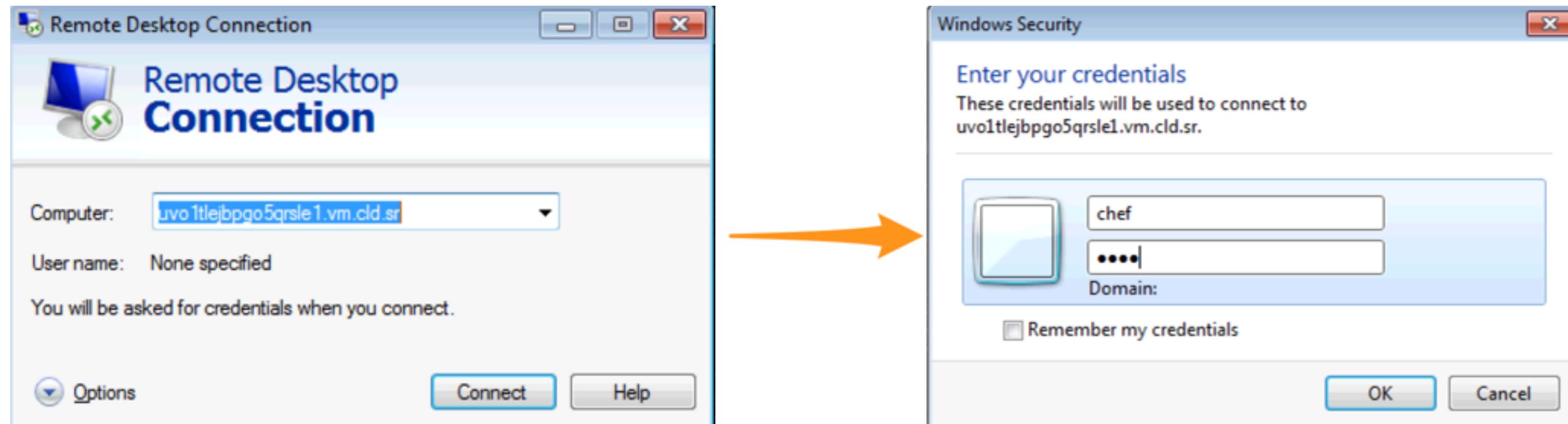
- Chef and all of its dependencies installed via an operating system-specific package ("omnibus installer")
- Installation includes
 - The Ruby language - used by Chef
 - knife - Command line tool for administrators
 - chef-client - Client application
 - ohai - System profiler
 - ...and more

Review - Workstation or Node?



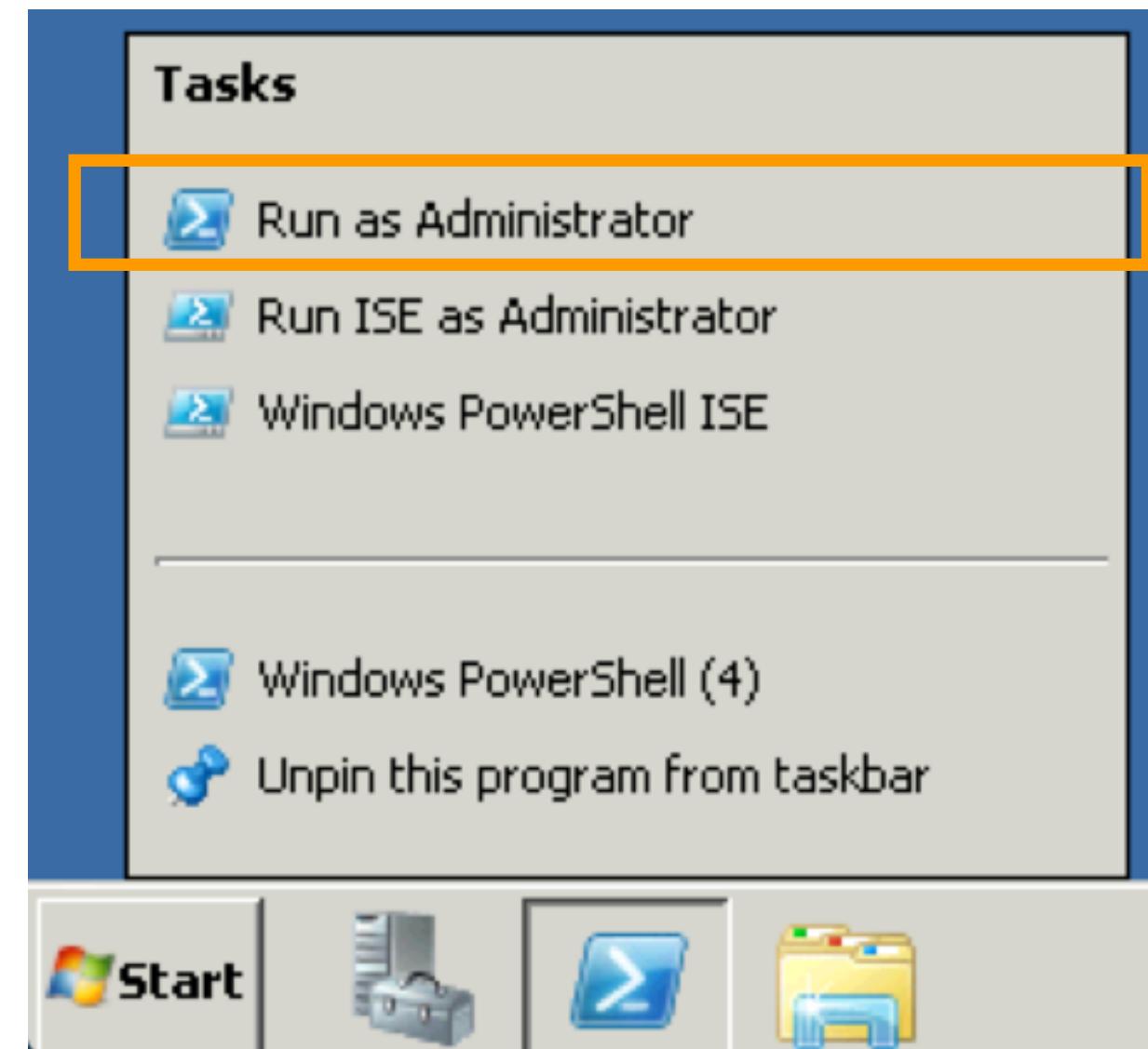
Verify Your Target Instance's Chef-Client is Configured Properly

- Use Remote Desktop to connect to the host.
- User: **chef** Password: **chef**



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

- Open an **Administrator Powershell** on the VM



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

```
remote@PS > 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

Examine C:\chef\client.rb

```
remote@PS > 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 "node1"
```

Verify the log level on your test node

```
remote@PS > 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 "node1"
```

Examine C:\chef\first-boot.json

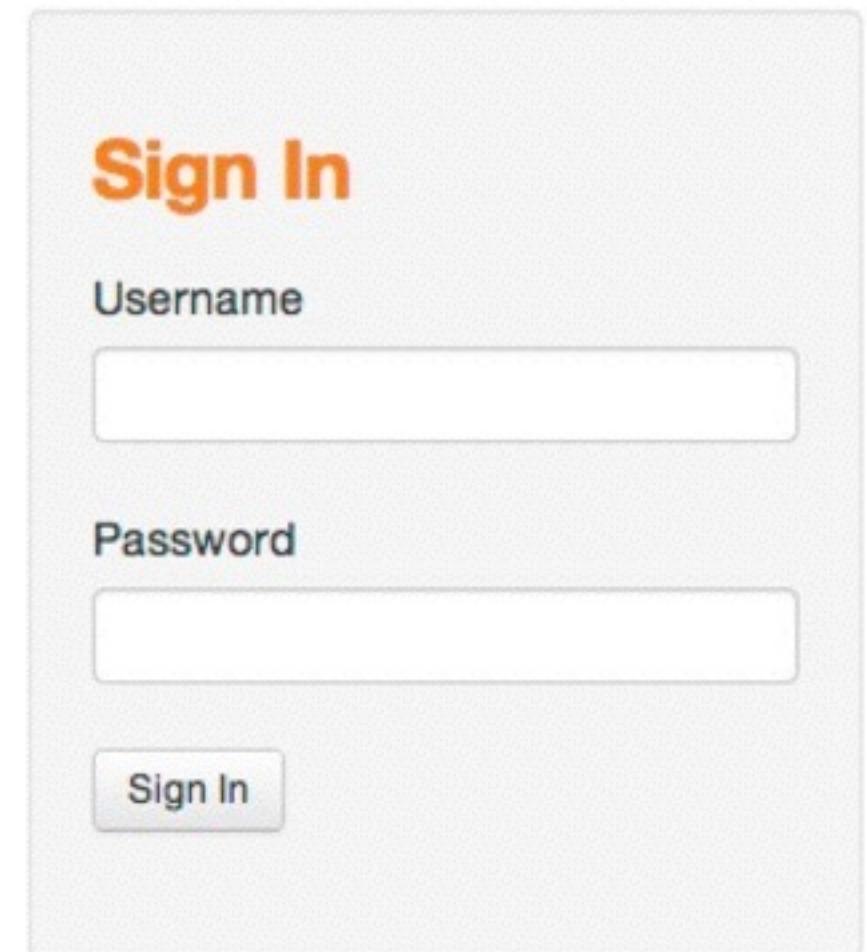
```
remote@PS > gc c:\chef\first-boot.json
```

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

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

View Node on Chef Server

- Login to your Hosted Enterprise Chef
 - via Management Council
 - or <https://manage.chef.io>



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, it says "Showing All Nodes". A table lists nodes with columns: Node Name, Platform, FQDN, IP Address, Uptime, and Last. One row is highlighted for "node1". Below the table, a modal window is open for "Node: node1". This modal has three tabs: Details (which is active and circled in orange), Attributes, and Permissions. Under the Details tab, it shows "Last Check In: 24 Minutes" (2014-02-21 11:11) and "Uptime: 44 Minutes" (Since 2014-02-21 15). To the right of the modal, there are sections for Environment, Platforms, FQDN, and IP Address. At the bottom of the modal, it says "Tags" and "There are no items to display." On the far right, there's a "Run List" section with "Expand All" and "Collapse All" buttons. The footer of the page includes copyright information (Copyright © 2012–2014 Chef, Inc.), help text ("Need help? If you have questions or are stuck, we are here to help."), and a feedback link.

CHEF
MANAGE

Nodes Reports Policy Administration

Showing All Nodes

Node Name	Platform	FQDN	IP Address	Uptime	Last
node1	windows	C1263834251	10.160.33.236	44 minutes	27 m

Node: node1

Details Attributes Permissions

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

Uptime: 44 Minutes
Since 2014-02-21 15

Tags

+ Add

There are no items to display.

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

View Node on Chef Server

- Click the 'Attributes' tab

The screenshot shows the Chef Manage web interface. At the top, there's a navigation bar with tabs: Nodes (which is active), Reports, Policy, and Administration. Below the navigation bar, it says "Showing All Nodes". A table lists nodes with columns: Node Name, Platform, FQDN, IP Address, Uptime, and Last. One node, "node1", is selected and highlighted in orange. In the details panel for "node1", there are three tabs: Details, Attributes (which is highlighted with an orange oval), and Permissions. The "Attributes" tab displays a hierarchical list of node attributes under the heading "Attributes". The attributes listed are:

- tags:
 - + languages
 - + kernel
- os: windows
- os_version: 6.2.9200
- + chef_packages
 - hostname: C1263834251
 - fqdn: C1263834251
 - domain:
- + network
- + counters

At the bottom of the page, there are copyright information, help links, and feedback links.

Copyright © 2012–2014 Chef, Inc.

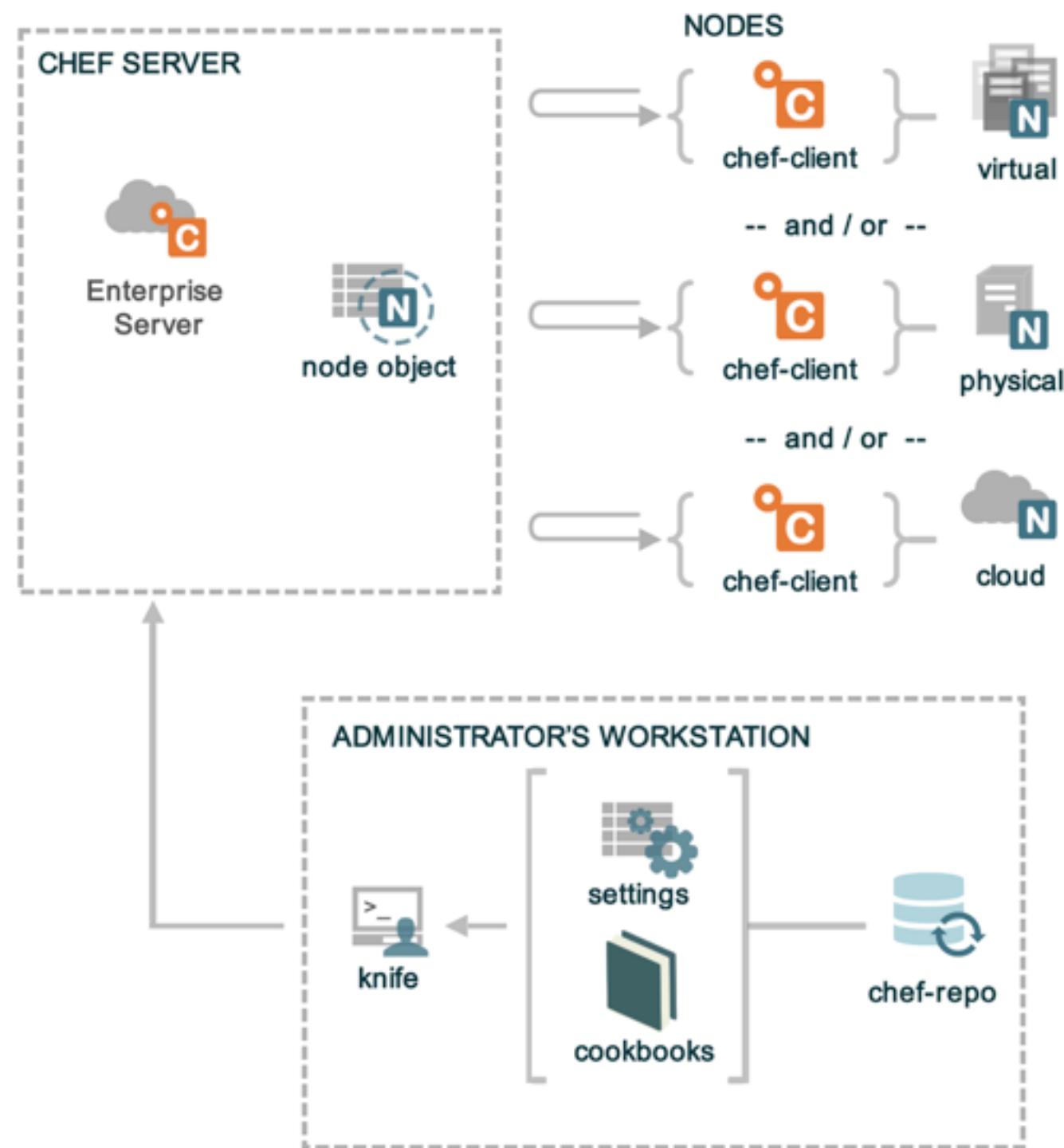
Need help? If you have questions or are stuck, we are [here to help](#).

Feedback

Node

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

Checkpoint



Review Questions

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

Chef Resources and Recipes

Writing an IIS demo cookbook

v2.1.1_WIN

Lesson Objectives

- After completing the lesson, you will be able to
 - Describe in detail what a **cookbook** is
 - Create a new cookbook
 - Explain what a **recipe** is
 - Describe how to use the `powershell_script`, `service`, and `cookbook_file` resources
 - Upload a **cookbook** to the Chef Server
 - Explain what a **run list** is, and how to set it for a node via knife
 - Explain the output of a chef-client run

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.

The Problem and the Success Criteria

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

Required steps

- Install IIS
- Start the service, and make sure it will start when the machine boots
- Write out the home page
- Please note in this course we're teaching Chef primitives, not web server management
- This is probably not the IIS configuration you would use in production

Exercise: Create a new Cookbook

```
PS \> 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

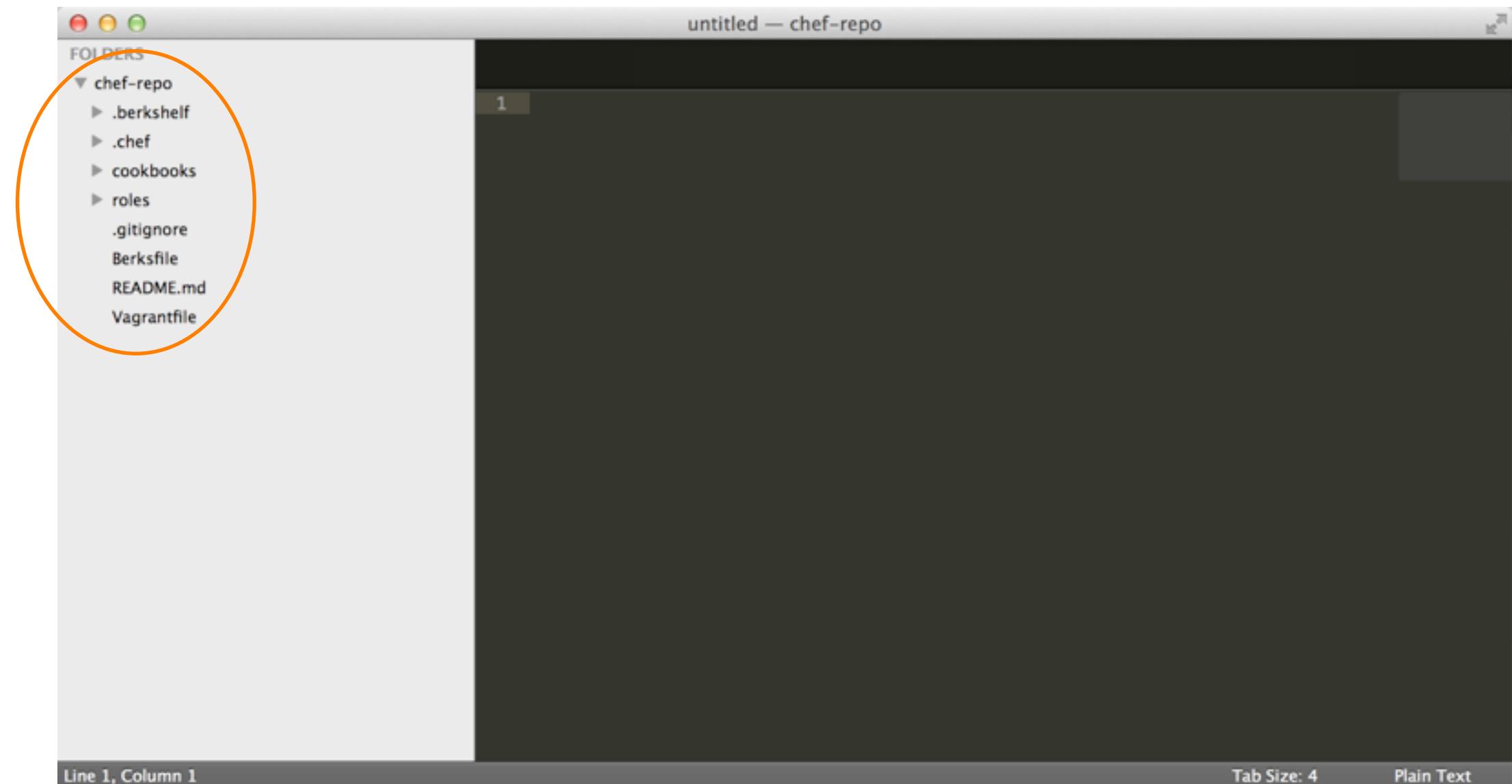
```
PS \> dir cookbooks\iis_demo
```

Mode	LastWriteTime	Length	Name
d----	9/5/2013 1:52 PM		attributes
d----	9/5/2013 1:52 PM		definitions
d----	9/5/2013 1:52 PM		files
d----	9/5/2013 1:52 PM		libraries
d----	9/5/2013 1:52 PM		providers
d----	9/5/2013 1:52 PM		recipes
d----	9/5/2013 1:52 PM		resources
d----	9/5/2013 1:52 PM		templates
-a---	9/5/2013 1:52 PM	472	CHANGELOG.md
-a---	9/5/2013 1:52 PM	287	metadata.rb
-a---	9/5/2013 1:52 PM	1531	README.md

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



Edit the default recipe

OPEN IN EDITOR: cookbooks\iis_demo\recipes\default.rb

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

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
```

Exercise: Add a powershell resource to install IIS

OPEN IN EDITOR: cookbooks\iis_demo\recipes\default.rb

```
#  
# Cookbook Name:: iis_demo  
# Recipe:: default  
#  
# Copyright 2013, 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 code parameter

```
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 code **parameter**
- With a run **action**

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

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

OPEN IN EDITOR: cookbooks\iis_demo\recipes\default.rb

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

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

SAVE FILE!

So the resource we just wrote...

- Is a **service** resource

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

So the resource we just wrote...

- Is a **service** resource
- Whose **name** is w3svc

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

So the resource we just wrote...

- Is a **service** resource
- Whose **name** is `w3svc`
- With two **actions**: **start** and **enable**

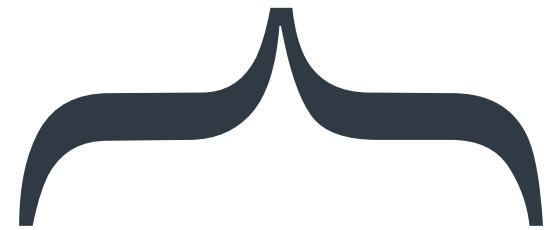
```
service "w3svc" do
  action [ :enable, :start ]
end
```

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

Service Resource

service “w3svc” ...



startsrc - aix

update-rc.d - debian

rc-update - gentoo

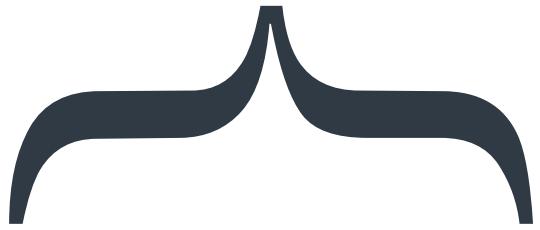
launchctl - osx

svcadm - solaris

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

Package Resource

```
package "git"
```



```
yum install git  
apt-get install git  
pacman sync git  
pkg_add -r git
```

Providers are determined by node's platform

But How does it *really* work?

- Chef is Open Source - All of the source code is on GitHub
- Don't be afraid to Explore
- It's just msieexec under the hood

```
def install_package(name, version)
  ...
  shell_out!("msieexec /qn /i \"#{@new_resource.source}\" ...")
  ...
end
```

Source: <https://github.com/opscode/chef/blob/master/lib/chef/provider/package/windows/msi.rb>

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!

So the resource we just wrote...

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

- Is a **cookbook_file** resource

So the resource we just wrote...

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

- Is a **cookbook_file** resource
- Whose name is `c:\inetpub\wwwroot\Default.htm`

So the resource we just wrote...

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

- Is a **cookbook_file** resource
- Whose **name** is `c:\inetpub\wwwroot\Default.htm`
- With two **parameters**:
 - **source** of `Default.htm`
 - **rights** of `:read, "Everyone"`

Order Matters

- Resources are executed in order

1st

2nd

3rd

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

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

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

Full contents of the IIS_demo recipe

```
# Cookbook Name:: iis_demo
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
powershell_script "Install IIS" do
  code "add-windowsfeature Web-Server"
  action :run
end

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

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

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

SAVE FILE!

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 (windows-6.3)
 - platform-major (windows-6)
 - platform
 - default
- 99% of the time, you will just use **default**

Exercise: Upload the cookbook

```
PS \> knife cookbook upload iis_demo
```

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

Recipe Naming

- Recipes are referenced using the notation ‘*cookbook::recipe*’, where ‘*recipe*’ is the name of the ‘.rb’ file in the “*cookbook/recipe*” directory
- The “*default.rb*” recipe for a given cookbook may be referred to just by the name of the cookbook (e.g. ‘*iis_demo*’)
- However, if we added another recipe to this cookbook named “*createsite.rb*”, we would refer to it as ‘*iis_demo::createsite*’

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]`”

Exercise: Add iis_demo recipe to node's run list

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

```
node1:  
  run_list: recipe[iis_demo]
```

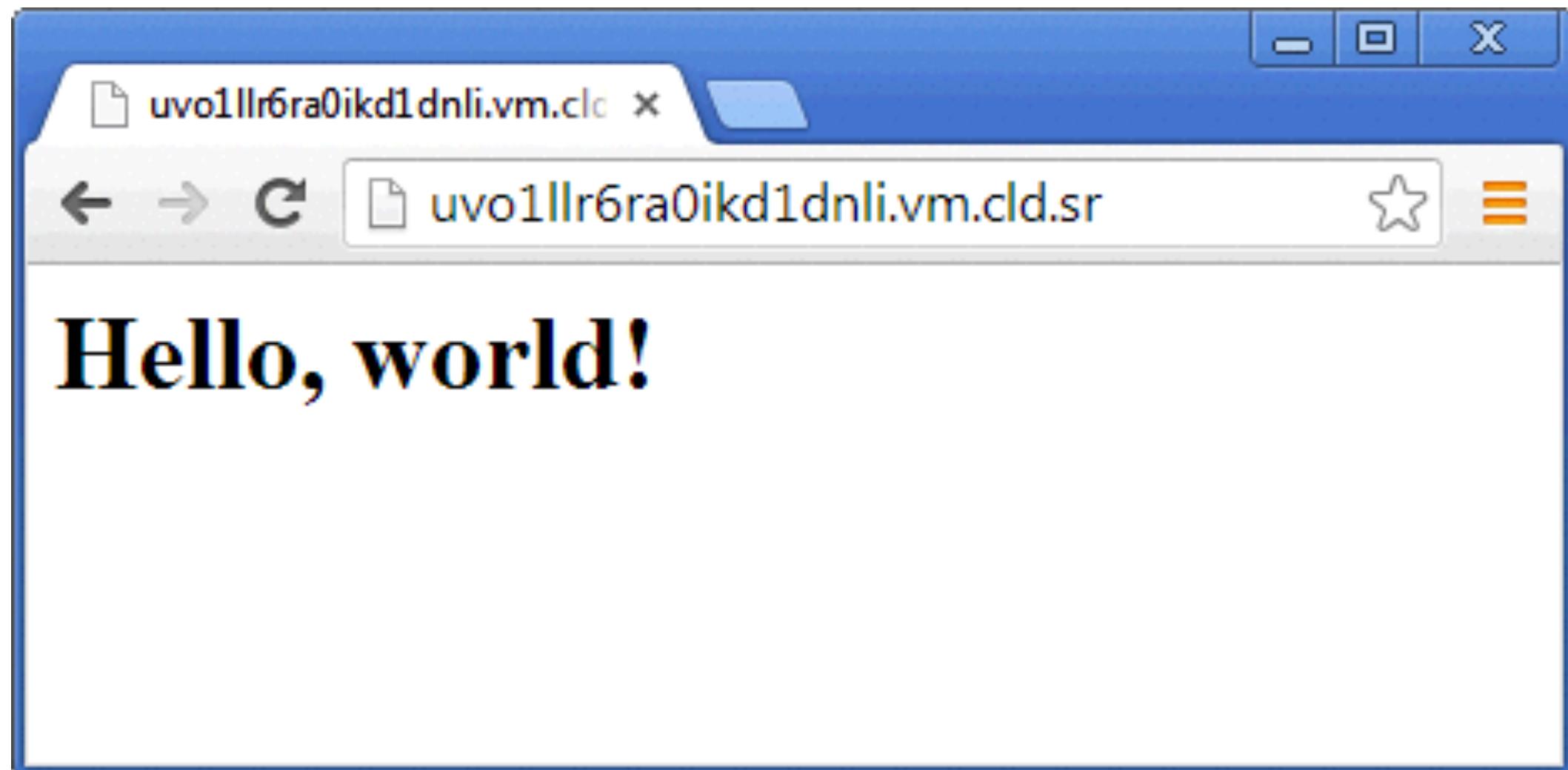
Exercise: Run the chef-client on your test node

```
remote@PS\> 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 node1
[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!)

Take 5



Reading the output of a chef-client run

```
Starting Chef Client, version 11.16.4
[2014-11-23T08:09:34-08:00] INFO: *** Chef 11.16.4 ***
[2014-11-23T08:09:34-08:00] INFO: Chef-client pid: 4996
[2014-11-23T08:09:44-08:00] INFO: Run List is [recipe[iis_demo]]
[2014-11-23T08:09:44-08:00] INFO: Run List expands to [iis_demo]
[2014-11-23T08:09:44-08:00] INFO: Starting Chef Run for node1
[2014-11-23T08:09:44-08:00] INFO: Running start handlers
[2014-11-23T08:09:44-08:00] INFO: Start handlers complete.
```

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

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

```
* 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

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

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

Idempotence

- Actions on resources in Chef are designed to be *idempotent*
 - I.e. they can be applied multiple times but the end result is still the same - like multiplying by 1 in math!
 - Or in mathematical terms, $F(F(x))=F(x) \forall x$
- Chef is a "desired state configuration" system - if a resource is already configured, no action is taken. This is called **convergence**

Exercise: Re-run the Chef Client

```
remote@PS\> 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
...
```

Recap - So resources are grouped into recipes

recipe[iis_demo::default]

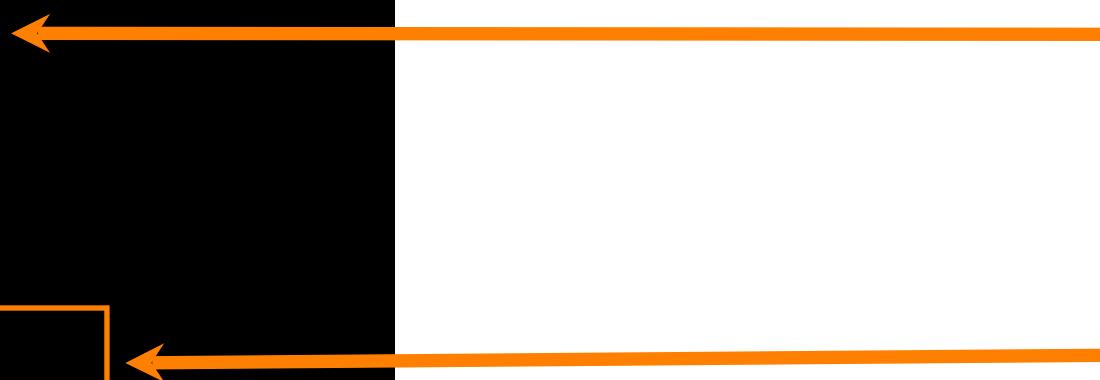
```
powershell_script "Install IIS" do
  action :run
  code "add-windowsfeature Web-Server"
end

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

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

Cookbooks contain recipes and supporting files

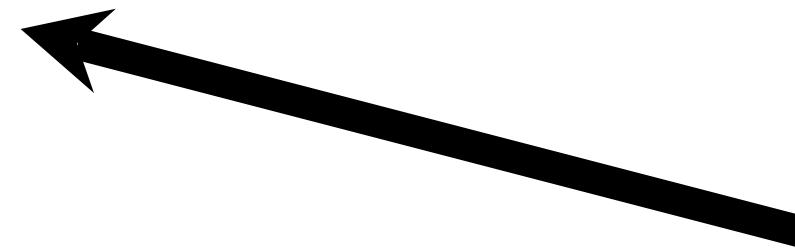
```
...  
└── cookbooks  
    └── iis_demo  
        ├── recipes  
        │   ├── default.rb  
        │   └── server.rb  
        └── files  
            └── default  
                └── index.html  
        └── metadata.rb  
        └── attributes  
            └── default  
...  
...
```



Recipes

Supporting File

Cookbooks are installed as artifacts on Chef Server

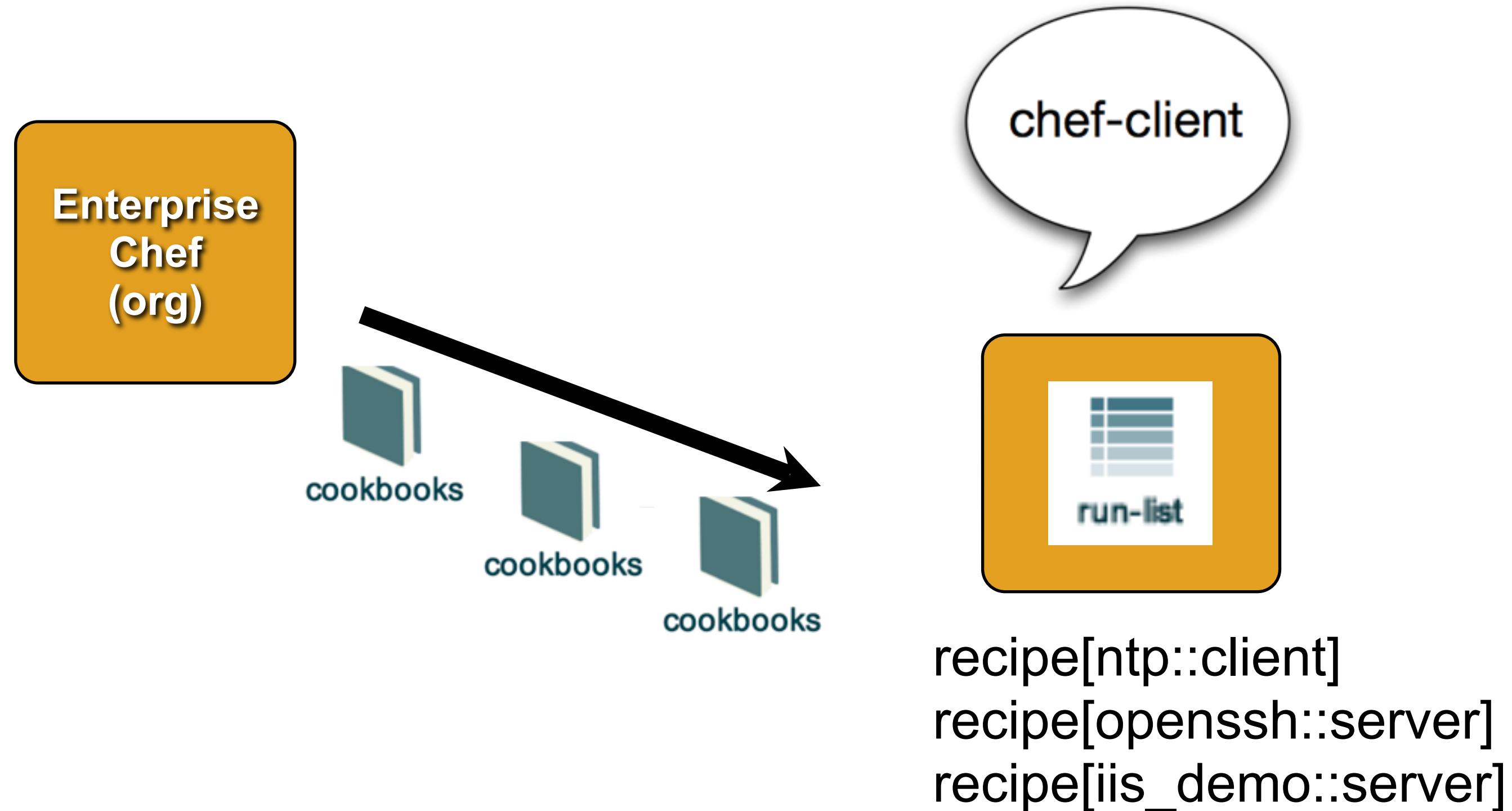


iis_demo v0.1.0

ntp v0.2.0

openssh v2.4.7

Nodes have run_lists made of recipes



Questions

- What makes up a cookbook?
- How do you create a new cookbook?
- What is a recipe?
- What is a resource?
- How do you upload a cookbook to the Chef Server?
- What is a run list?

Dissecting your first chef-client run

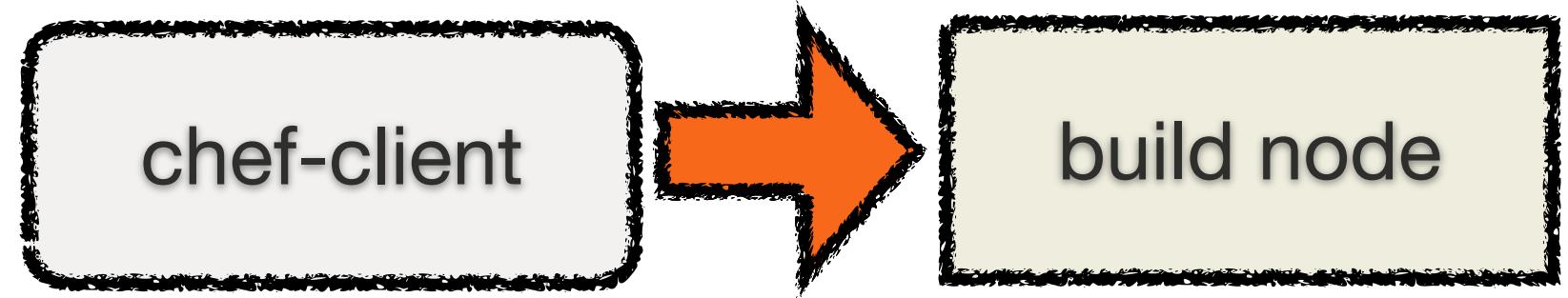
The Anatomy of a Chef run

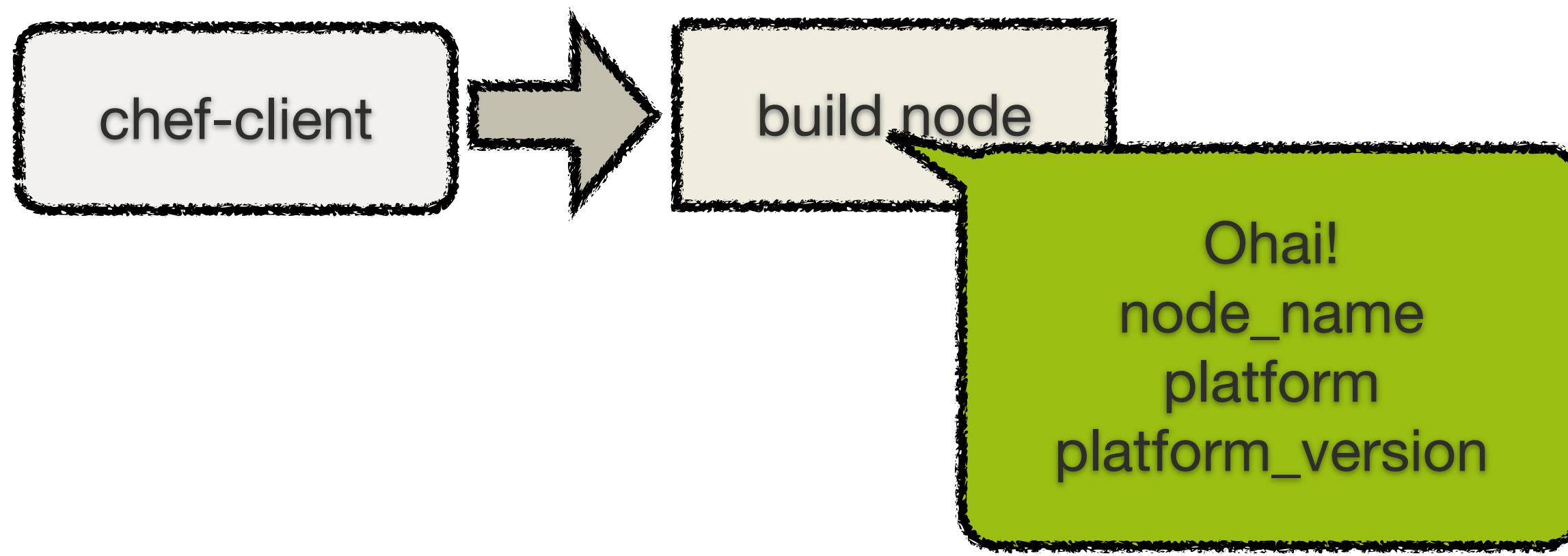
v2.1.1_WIN

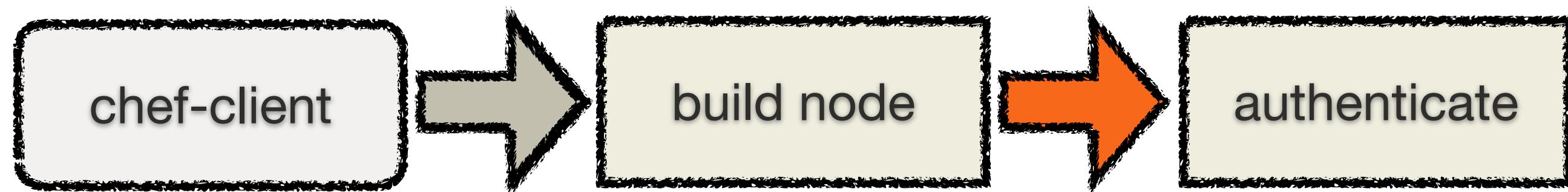
Lesson Objectives

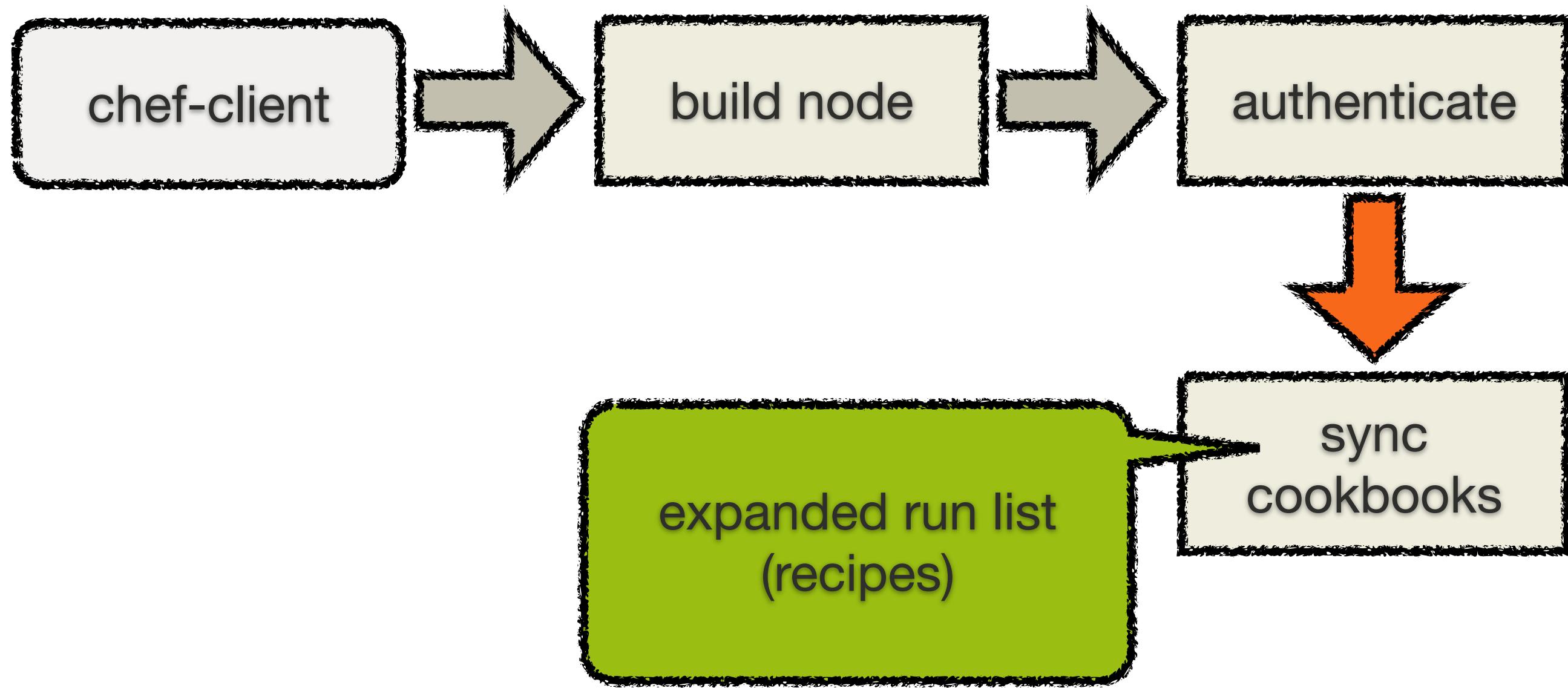
- After completing the lesson, you will be able to
 - List all the steps taken by a chef-client during a run
 - Explain the basic security model of Chef
 - Explain the concepts of the Resource Collection

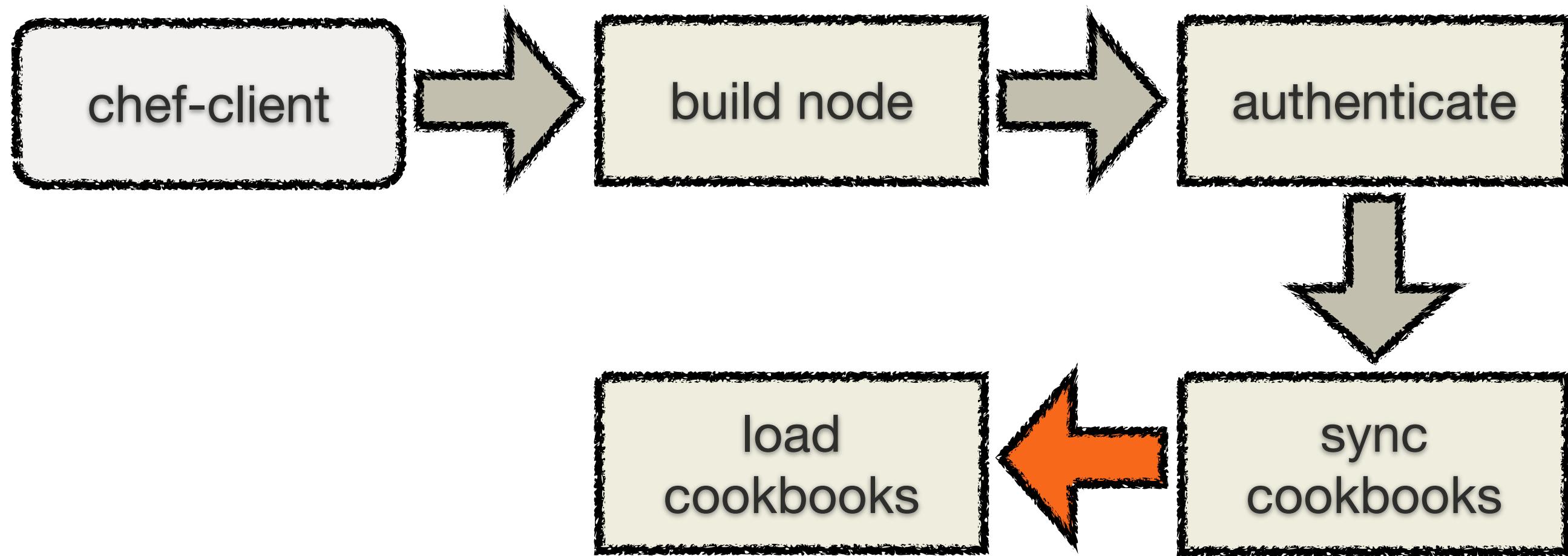
chef-client

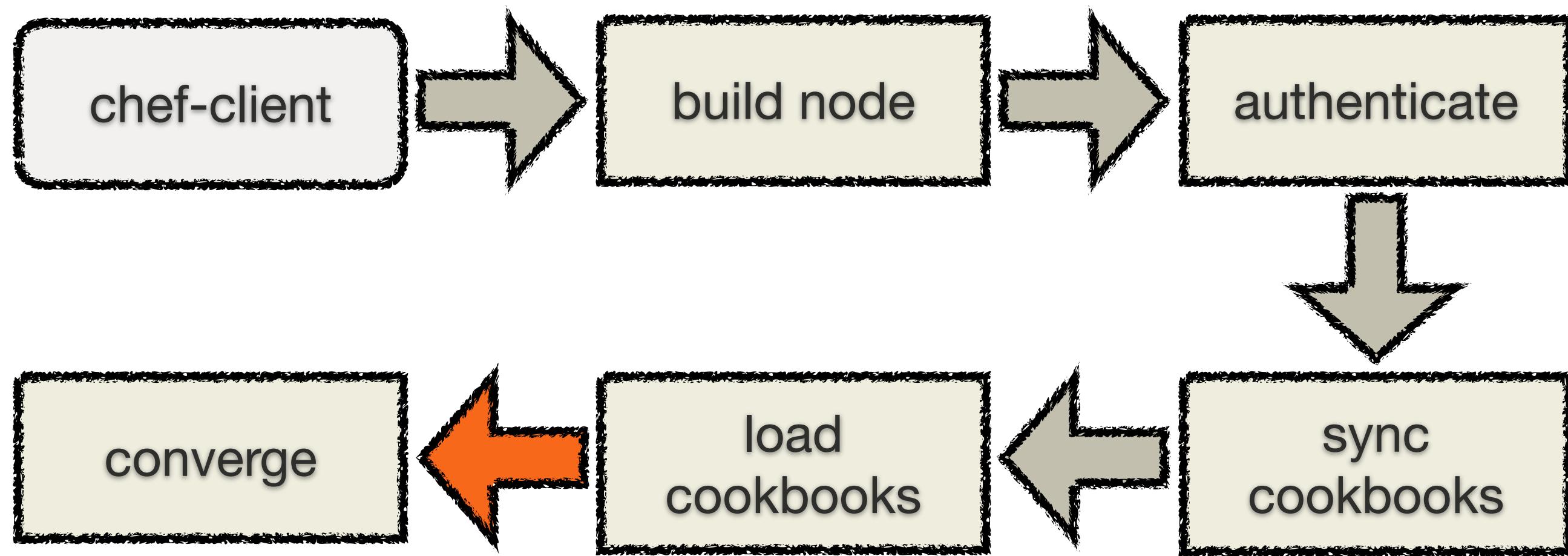


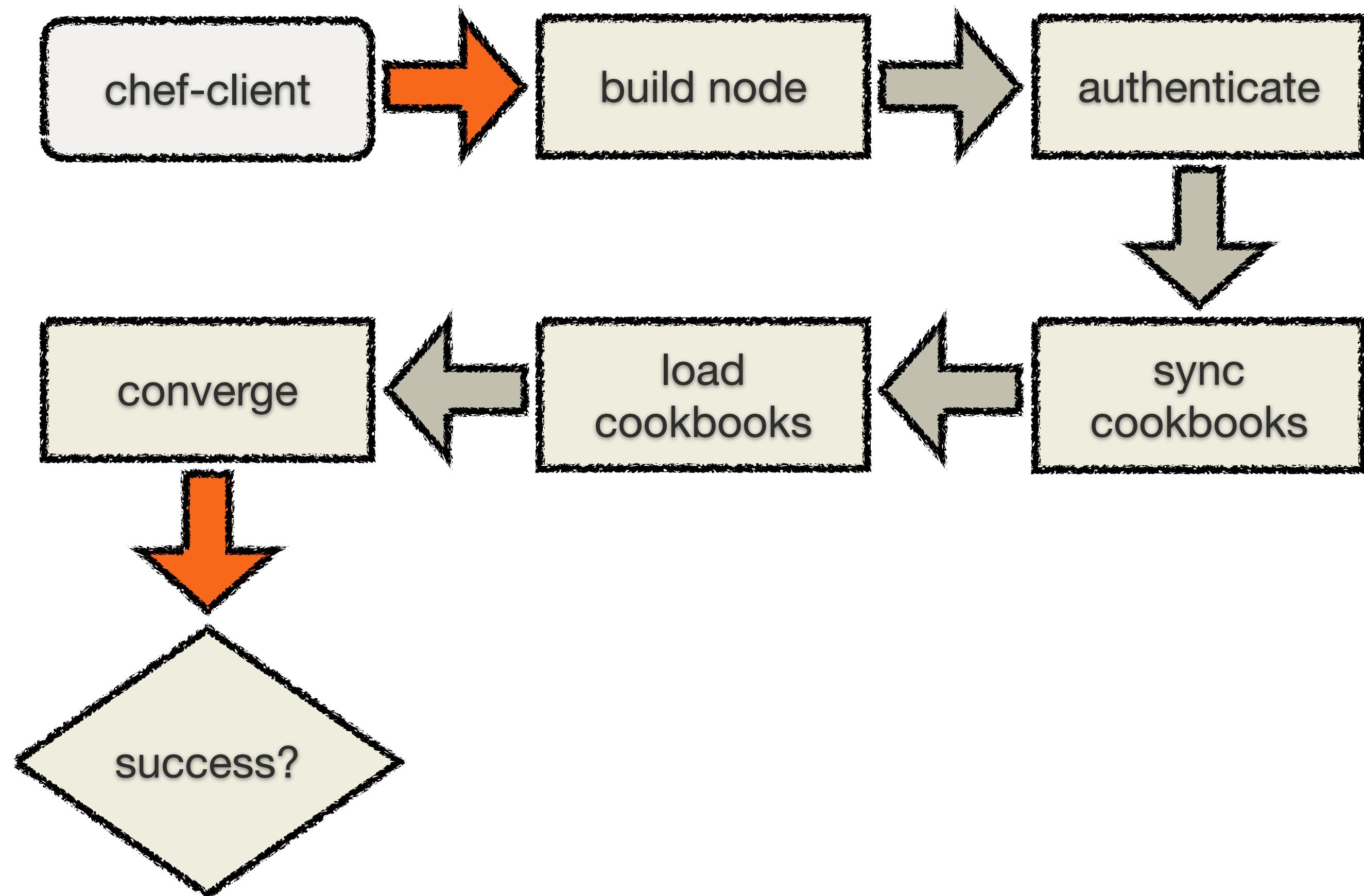


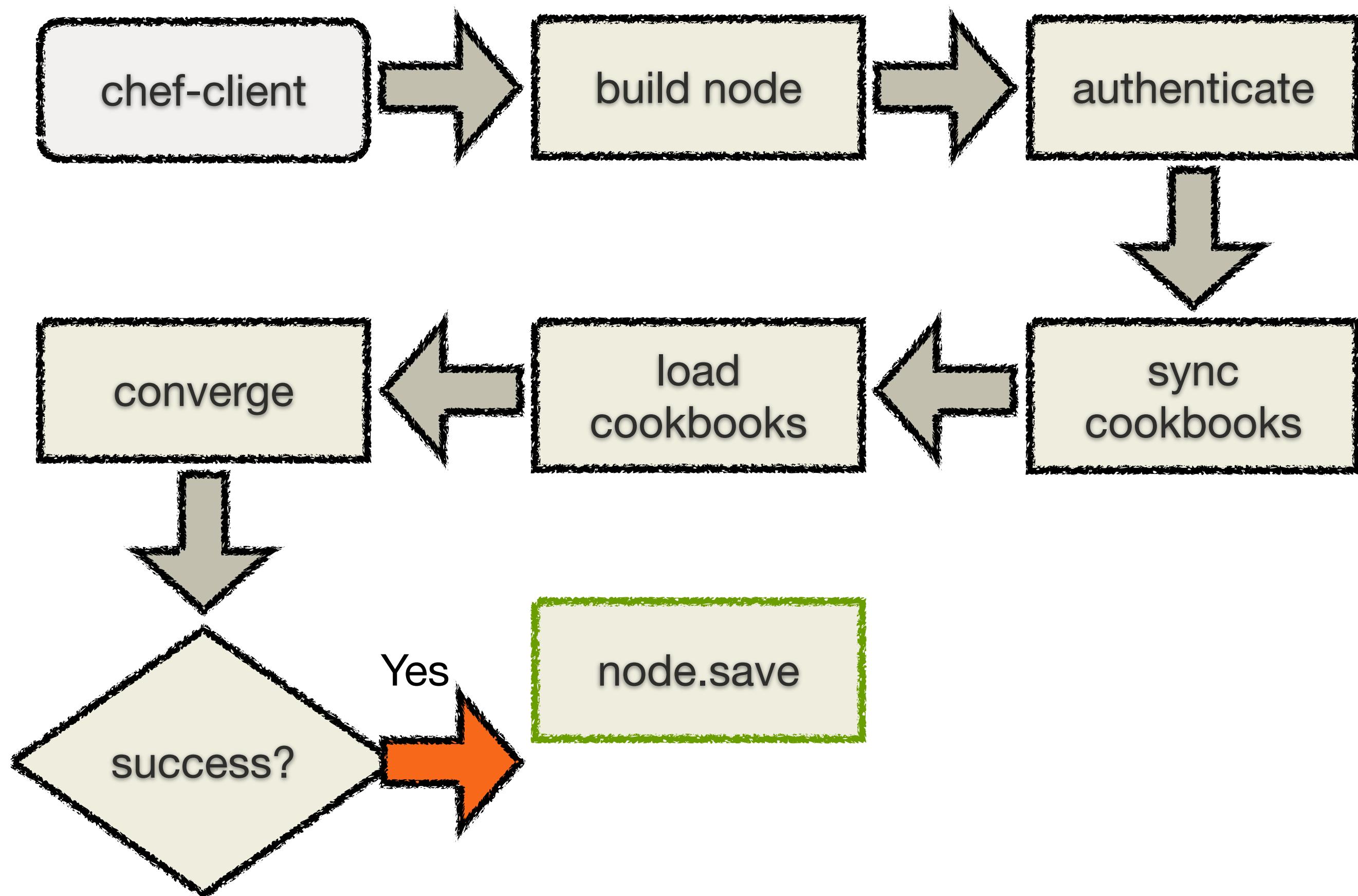


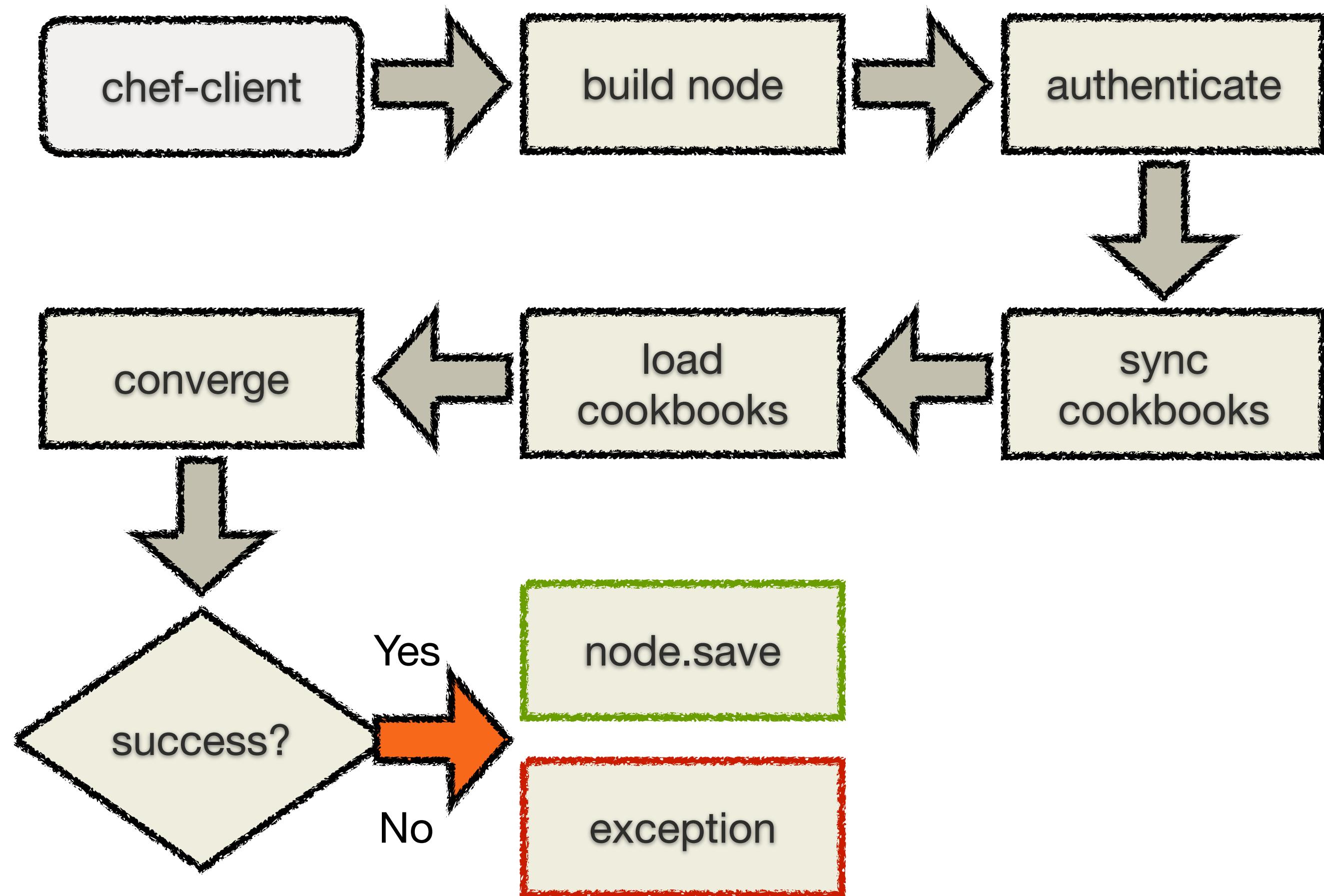


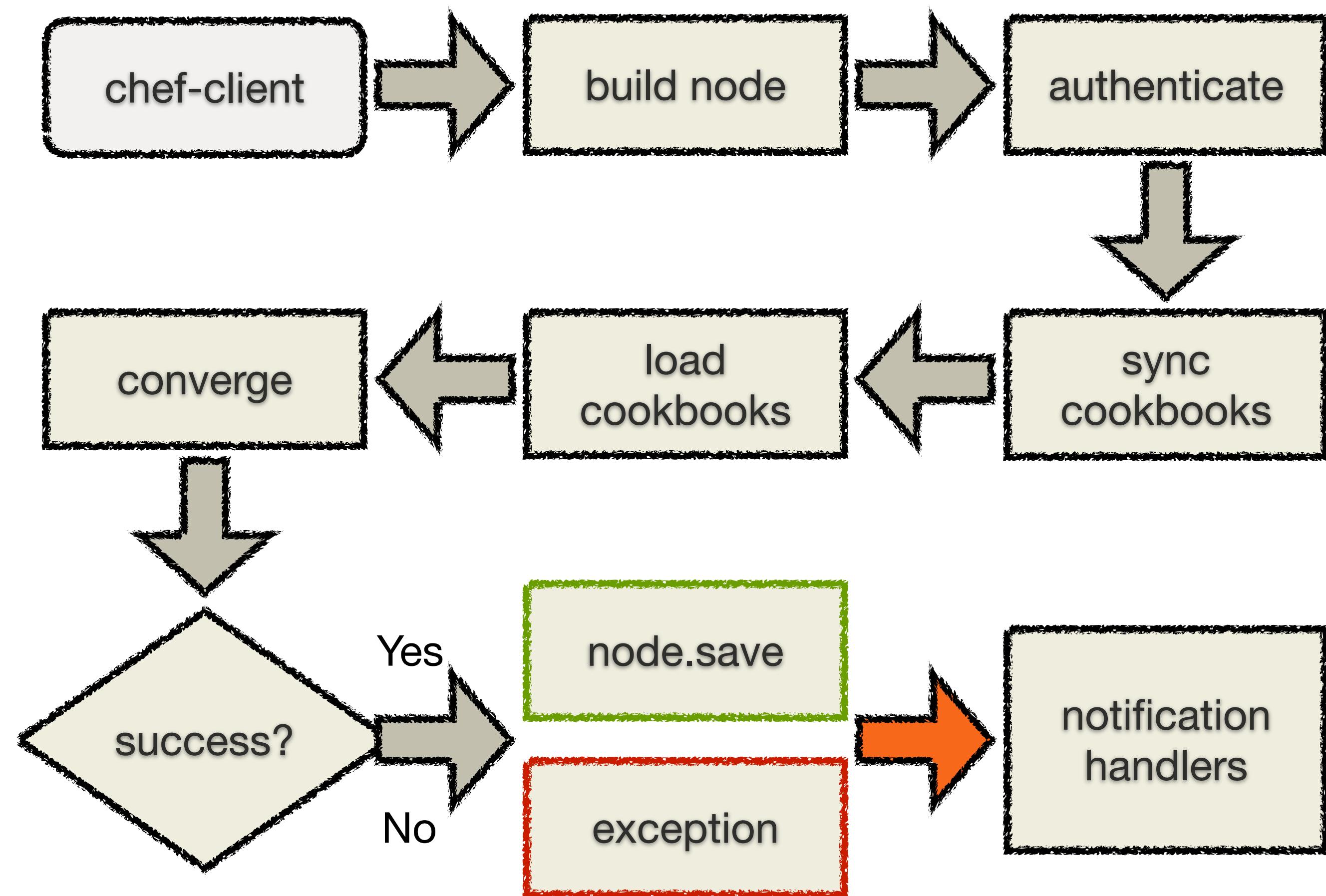










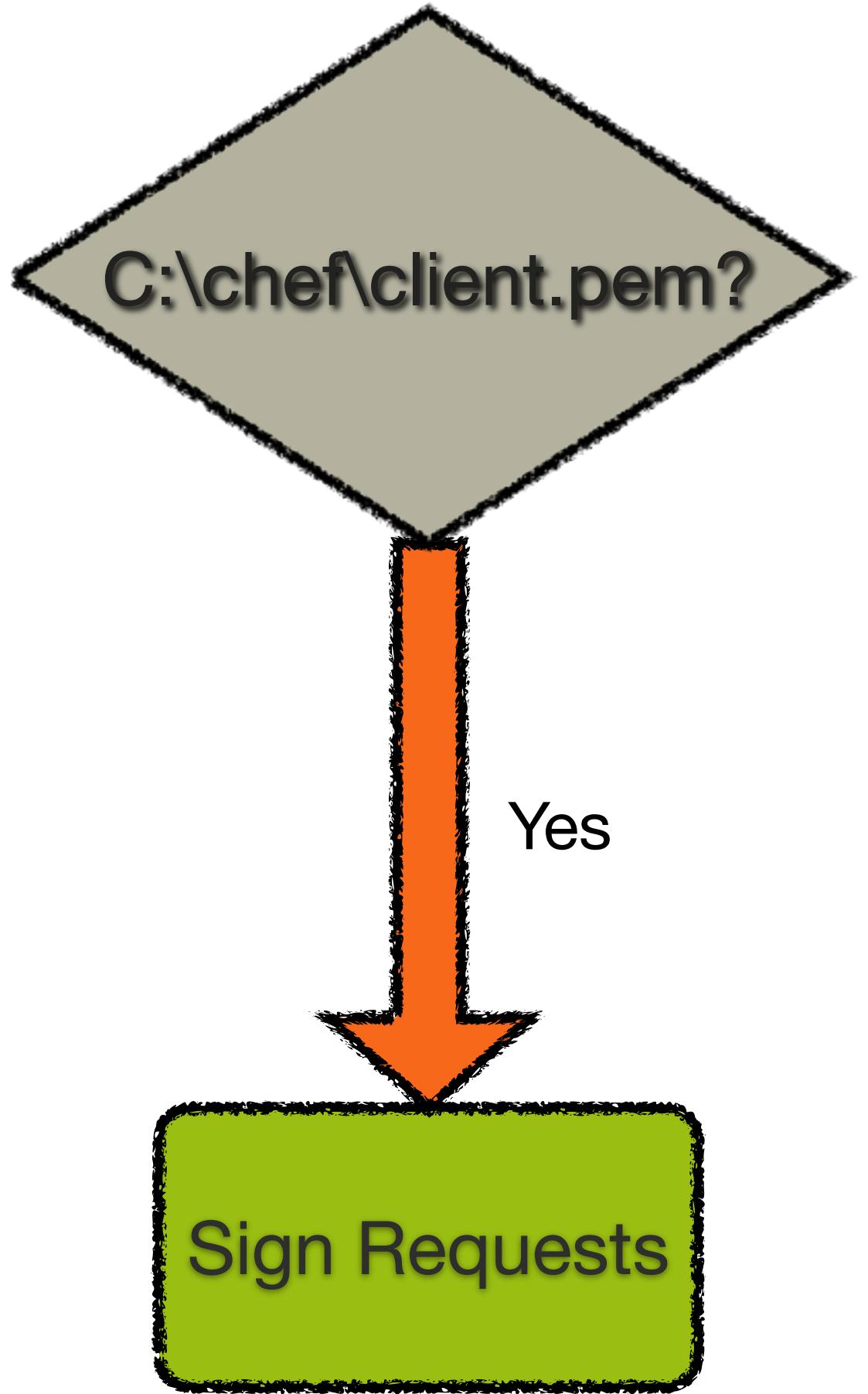


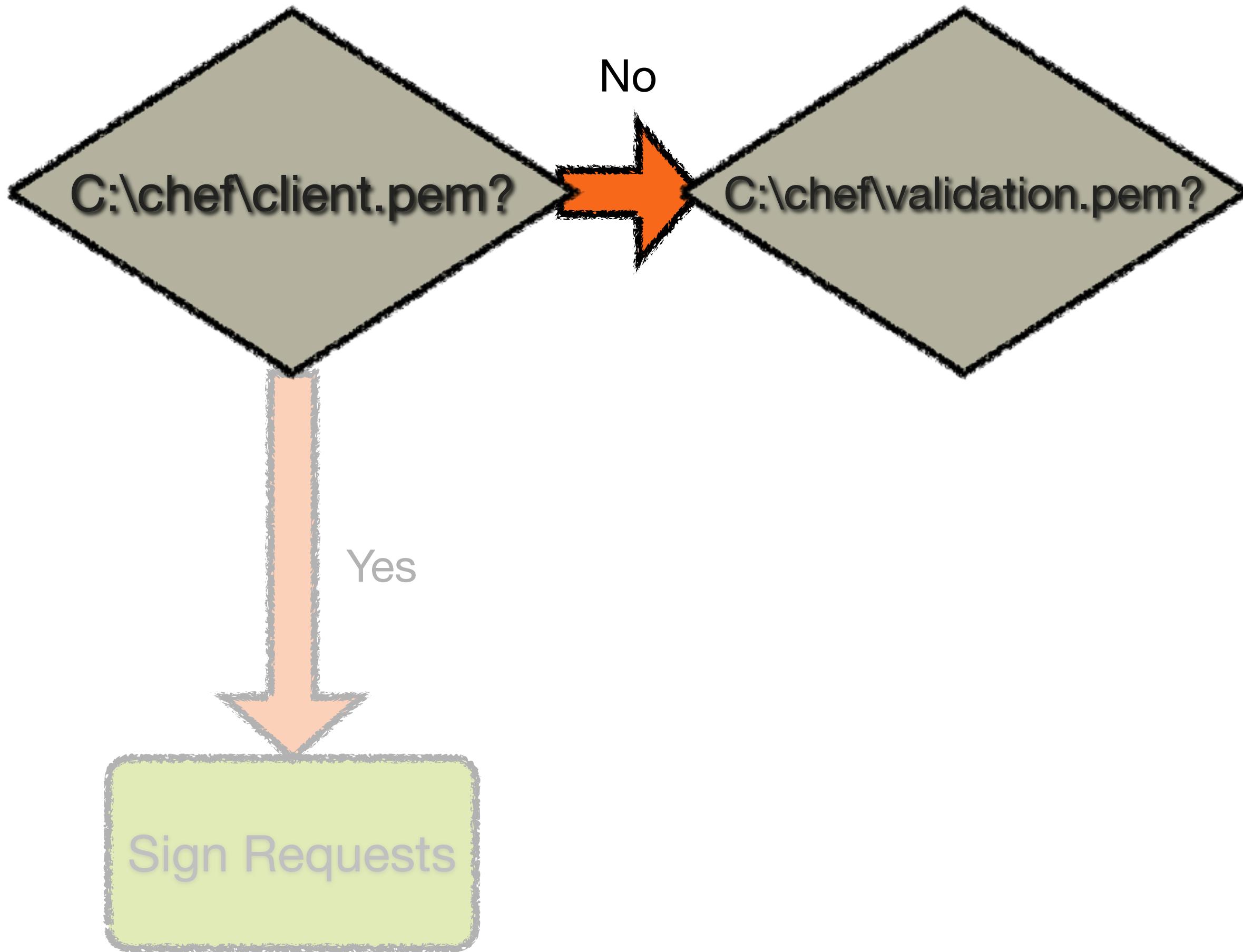
Private Keys

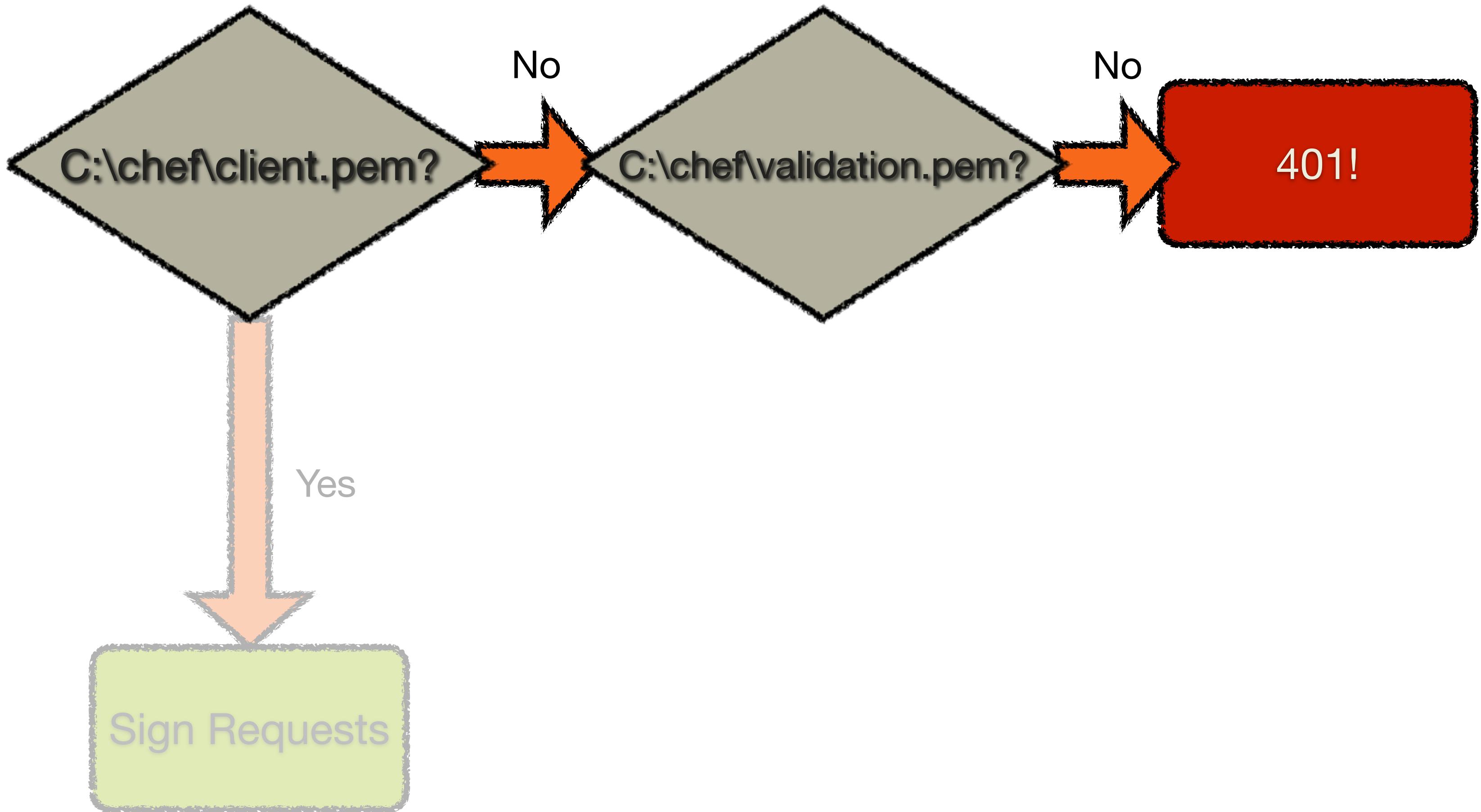
- Chef Server requires keys to authenticate.
 - client.pem - private key for API client
 - validation.pem - private key for ORGNAME-validator
- Next, let's see how those are used...

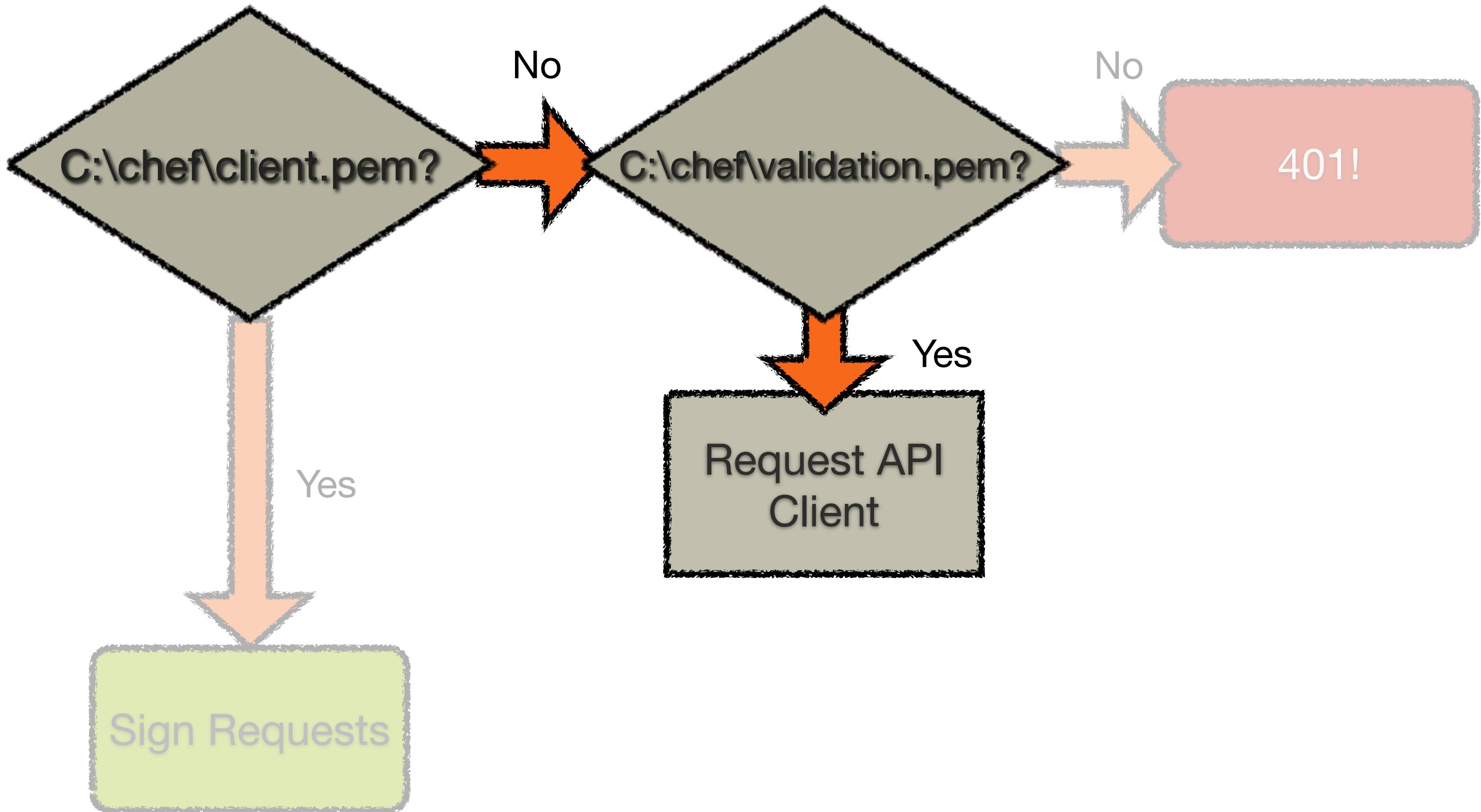


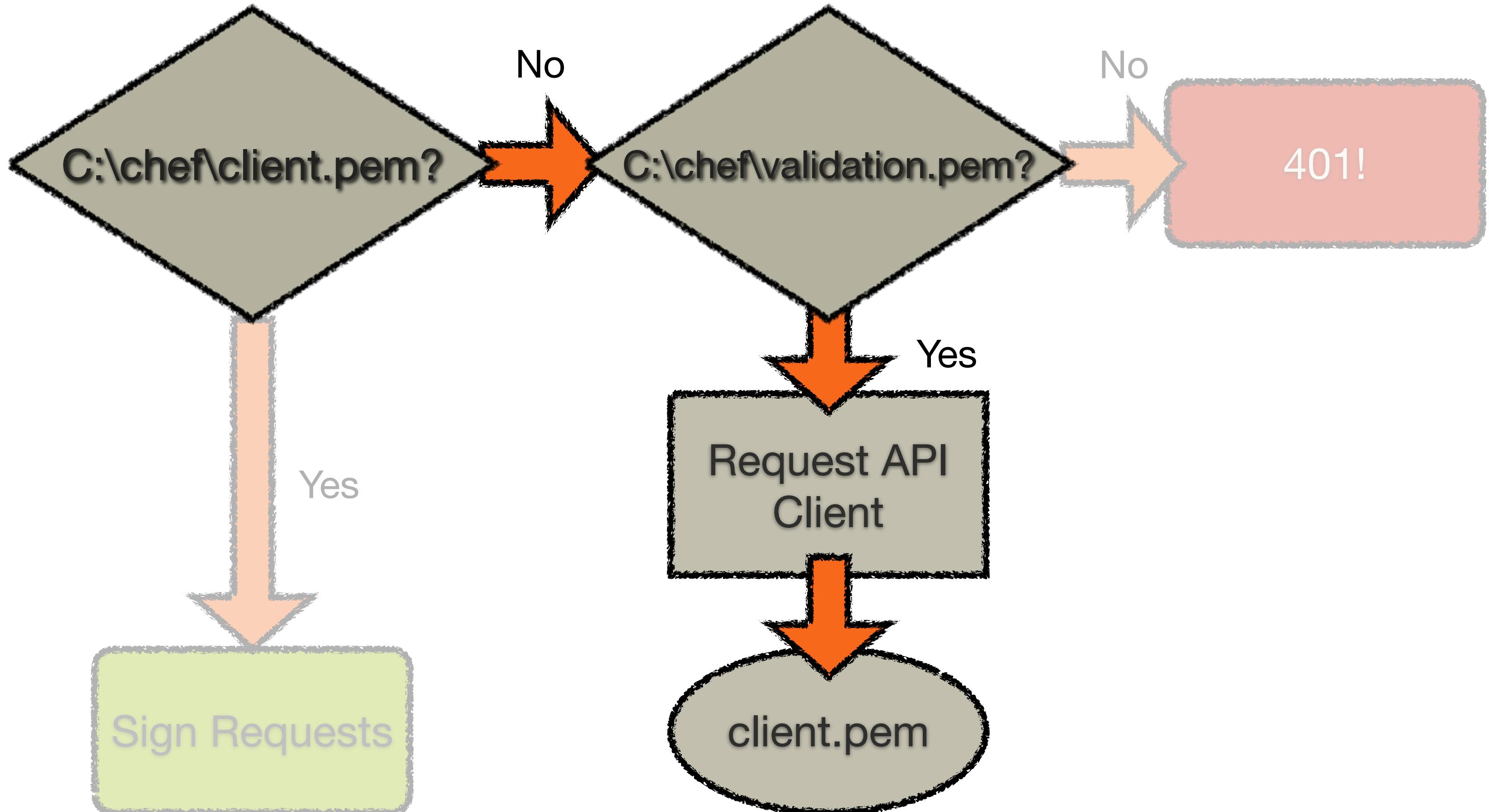
C:\chef\client.pem?

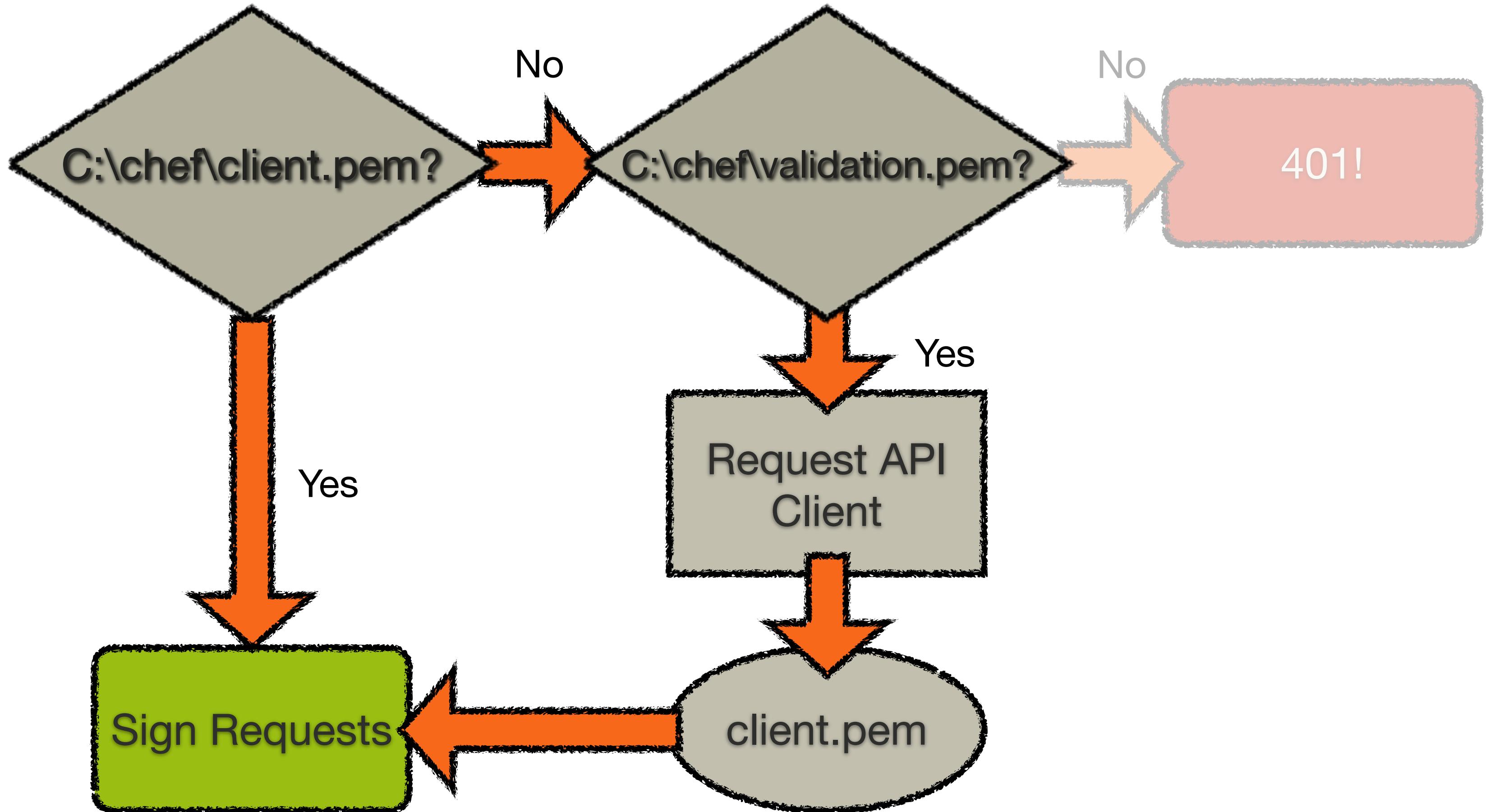












About the resource collection

v2.1.1_WIN

Multiphase Execution - Compile Phase

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

Multiphase Execution - Execute Phase

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

Resource Collection Phase 1 - Compile Phase

Recipe

```
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
```

Resource Collection

```
resource_collection = [
  Powershell_script["Install IIS"],
  service ["w3svc"],
  cookbook_file["c:\inetpub\wwwroot\Default.htm"]
]
```

Resource Collection Phase 1 - Execute Phase

Recipe

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

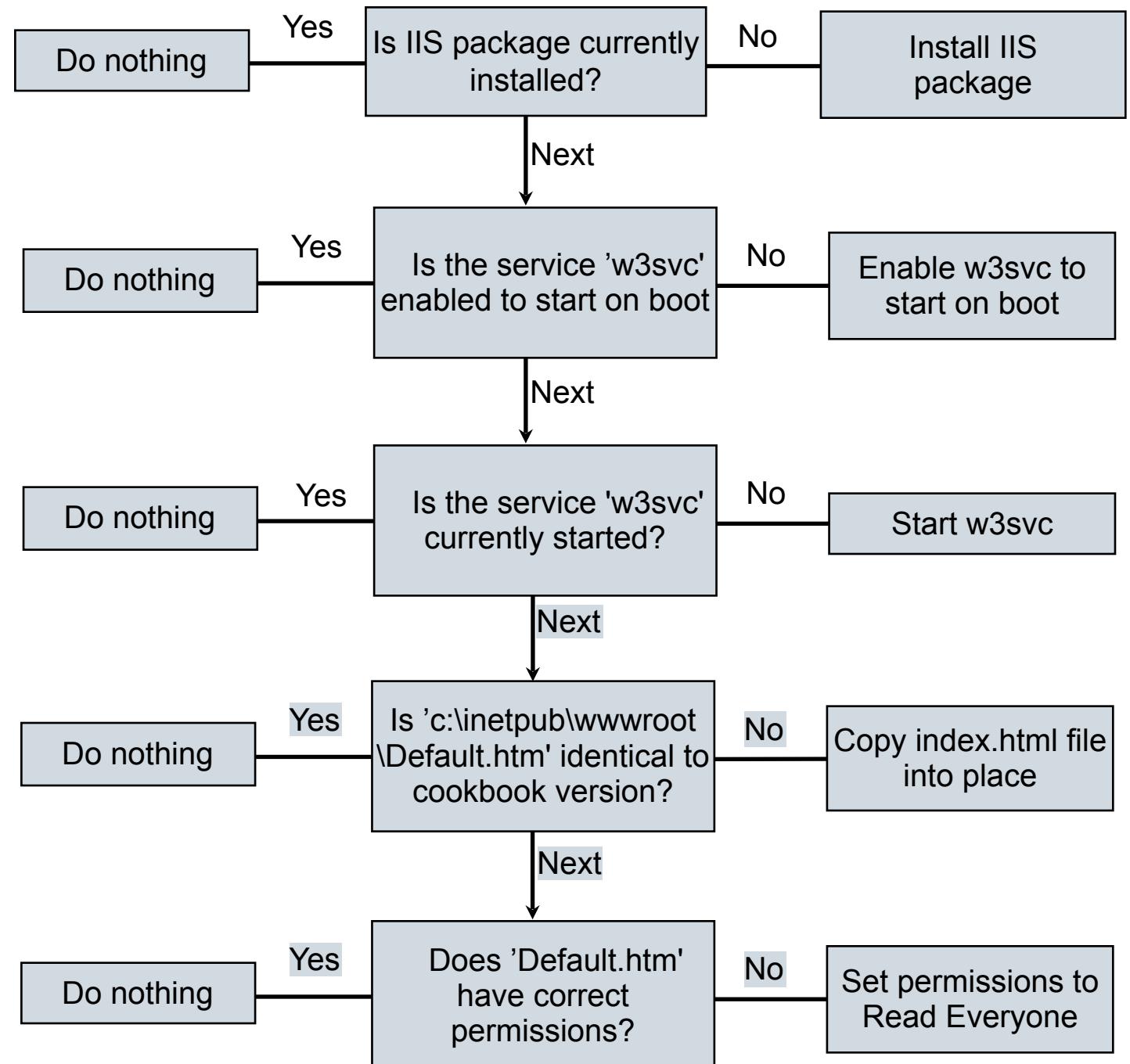
service "w3svc" do
  action [ :enable, :start ]
end

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

Resource Collection

```
resource_collection = [
  package["Install IIS"],
  cookbook_file["c:\inetpub\wwwroot\Default.htm"],
  service ["w3svc"]
]
```

Execution



Recipe order is important!

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

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

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

Resource Collection - Multiple Recipes

1. recipe[ntp::client]

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

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

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

Resource Collection

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

Resource Collection - Multiple Recipes

2. recipe[openssh::client]

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

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

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

Resource Collection

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

Resource Collection - Multiple Recipes

3. recipe[iis_demo::default]

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

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

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

Resource Collection

```
resource_collection [
  package[ntp],
  template[/etc/ntp.conf],
  service[ntp],
  package[openssh],
  template[/etc/sshd/sshd_config],
  service[openssh],
  powershell_script[Install IIS],
  service[w3svc],
  cookbook_file[c:\wwwroot\Default.htm]
]
```

The final resource collection

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

```
package[ntp]
template[/etc/ntp.conf]
service[ntp]
package[openssh]
template[/etc/sshd/sshd_config]
service[openssh]
powershell_script[Install IIS]
service[w3svc]
cookbook_file[c:\wwwroot\Default.htm]
```



Multiphase Execution

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

Recipe

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

Resource Collection

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

Remember - Resource order is important!

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

The diagram illustrates three parallel sequences of operations, each labeled with a name above it:

- recipe-1**: Consists of three actions followed by three tests. The first action is labeled "action", the second "action", and the third "action". The first test is labeled "test", the second "test", and the third "test". The sequence is enclosed in curly braces: $\{O_1, O_2, O_3\}$.
- recipe-2**: Consists of three actions followed by three tests. The first action is labeled "action", the second "action", and the third "action". The first test is labeled "test", the second "test", and the third "test". The sequence is enclosed in curly braces: $\{O_1, O_2, O_3\}$.
- recipe-3**: Consists of three actions followed by three tests. The first action is labeled "action", the second "action", and the third "action". The first test is labeled "test", the second "test", and the third "test". The sequence is enclosed in curly braces: $\{O_1, O_2, O_3\}$.

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

Review Questions

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

Introducing the Node object

Attributes & Search

v2.1.1_WIN

Lesson Objectives

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

What is the Node object

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

Node

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

View Node on Chef Server

- 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, it says "Showing All Nodes". A table lists nodes with columns: Node Name, Platform, FQDN, IP Address, Uptime, and Last. One row is highlighted for "node1". Below the table, a modal window is open for "Node: node1". This modal has three tabs: Details (which is active and circled in orange), Attributes, and Permissions. Under the Details tab, it shows "Last Check In: 24 Minutes" (2014-02-21 11:11) and "Uptime: 44 Minutes" (Since 2014-02-21 15). To the right of the modal, there are sections for Environment, Platforms, FQDN, and IP Address. At the bottom left of the modal, it says "Tags" and "Add". At the bottom right, it says "Run List", "Expand All", and "Collapse All". At the very bottom of the page, there's a footer with copyright information, help text, and a feedback link.

CHEF
MANAGE

Nodes Reports Policy Administration

Showing All Nodes

Node Name	Platform	FQDN	IP Address	Uptime	Last
node1	windows	C1263834251	10.160.33.236	44 minutes	27 m

Node: node1

Details Attributes Permissions

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

Uptime: 44 Minutes
Since 2014-02-21 15

Tags + Add

Run List

Environment: _de

Platforms: windo

FQDN: C126

IP Address: 10.16

Copyright © 2012 – 2014 Chef, Inc.

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

Feedback

View Node on Chef Server

- Click the 'Attributes' tab

The screenshot shows the Chef Manage web interface. At the top, there's a navigation bar with tabs: Nodes (which is active), Reports, Policy, and Administration. Below the navigation bar, it says "Showing All Nodes". A table lists nodes with columns: Node Name, Platform, FQDN, IP Address, Uptime, and Last. One node, "node1", is selected and highlighted in orange. In the details panel for "node1", there are three tabs: Details, Attributes (which is highlighted with an orange oval), and Permissions. The "Attributes" tab displays a hierarchical list of node attributes under the heading "Attributes". The attributes listed are:

- tags:
 - + languages
 - + kernel
- os: windows
- os_version: 6.2.9200
- + chef_packages
 - hostname: C1263834251
 - fqdn: C1263834251
 - domain:
- + network
- + counters

At the bottom of the page, there are copyright information, help links, and feedback links.

Copyright © 2012–2014 Chef, Inc.

Need help? If you have questions or are stuck, we are [here to help](#).

Feedback

Exercise: Listing nodes

```
PS\> knife node list
```

node1

Exercise: Listing clients

```
PS\> knife client list
```

```
ORGNAME-validator  
node1
```

Each node must have a unique name

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

Exercise: Showing details about a node

```
PS\> knife node show node1
```

```
Node Name:      node1
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:
```

What is the Node object

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

Ohai

```
"languages": {  
  "ruby": {  
    "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",  
        "ip": "127.0.0.1",  
        "netmask": "255.0.0.0",  
        "broadcast": "127.255.255.255",  
        "link_layer": "00:0C:29:4D:4D:00",  
        "mtu": 16436, "txqueuelen": 1000, "state": "UP",  
        "flags": 4099, "hwaddr": "00:0C:29:4D:4D:00",  
        "inet": {  
          "family": "inet", "netmask": "255.0.0.0",  
          "broadcast": "127.255.255.255",  
          "state": "UP", "link_layer": "00:0C:29:4D:4D:00",  
          "hwaddr": "00:0C:29:4D:4D:00", "inet6": {},  
          "inet6": {}  
        }  
      }  
    }  
  }  
}
```

Exercise: Run Ohai on Target

```
remote@PS\> ohai | oh -Paging
```

```
{
  "languages": {
    "ruby": {
      "platform": "i386-mingw32",
      "version": "1.9.3",
      "release_date": "2013-11-22",
      "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",
      ...
    }
  }
}
```

Exercise: Showing all the attributes of a node

```
PS\> knife node show node1 -l
```

```
Node Name:      node1
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: Showing the raw format of the node

```
PS\> knife node show node1 -Fj
```

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

Exercise: Show only the fqdn attribute

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

```
node1:  
fqdn: C1263834251
```

Exercise: Use search to find the same data

```
PS\> knife search node "*:*" -a fqdn
```

```
1 items found
```

```
node1:
```

```
fqdn: c1263834251
```

Questions

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

Setting Node Attributes

Setting attributes in recipes and attribute files

v2.1.1_WIN

Lesson Objectives

- After completing the lesson, you will be able to
 - Describe where and how attributes are set
 - Explain the attribute merge order and precedence rules
 - Declare an attribute with a recipe and sets its value

What are Attributes?

- Attributes represent information about your node
- The information can be auto-detected from the node (e.g. # of CPUs, amount of RAM) & populated by Ohai
- You can also set attributes on your node using cookbook recipes & attribute files, roles, environments, etc
- Attributes keep the program code separate from data
- All attributes are set on the "node object", and are indexed for search on the server

Attribute Sources

- Attributes can be set at various levels (in increasing order of precedence)
 - Automagically on the node itself (by Ohai)
 - In roles
 - In environments
 - In cookbook recipes
 - In cookbook attribute files

Ohai

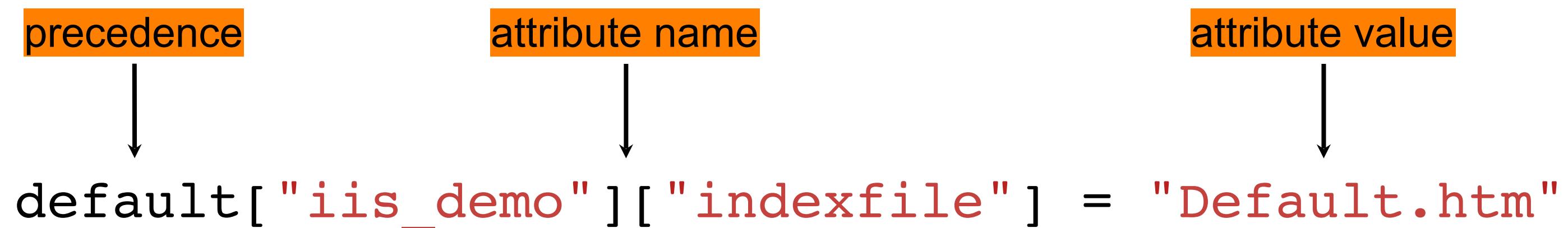
```
"languages": {  
    "ruby": {  
        "platform": "i386-  
mingw32",  
        "version": "1.9.3",  
        "target": "i386-pc-  
mingw32",  
        "target_cpu": "i386",  
        "target_vendor": "pc",  
        "target_os": "mingw32"  
        "host": "i686-pc-  
mingw32",  
        "host_cpu":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

```
"kernel": {
    "os_info": {
        "boot_device": "\Device\HarddiskVolume1",
        "build_number": "9200",
        "build_type": "MultiprocessorFree",
        "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",
                "linklayer": "enp0s3"
            }
        }
    }
}
```

Setting attributes in attribute files

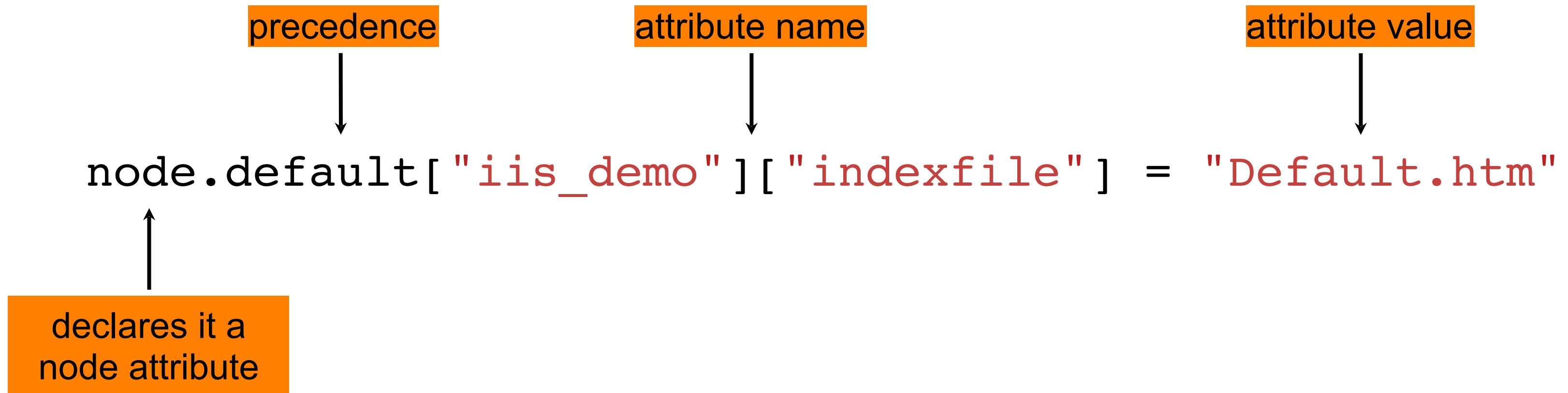
- Attributes can be set in the cookbook's attributes file
 - `./cookbooks/<cookbook>/attributes/default.rb`
 - Format is



- We'll look at precedence later....

Setting Attributes in Recipes

- They can also be set directly in Recipes
- Precede attribute name with 'node.' as follows



The Problem and the Success Criteria

- **The Problem:** We have defined our 'index.html' 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

Exercise: Set attribute in 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

Exercise: Add index1.html 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!

Exercise: Set 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!

Exercise: Upload the cookbook

```
PS\> knife cookbook upload iis_demo
```

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

Exercise: Re-run Chef Client

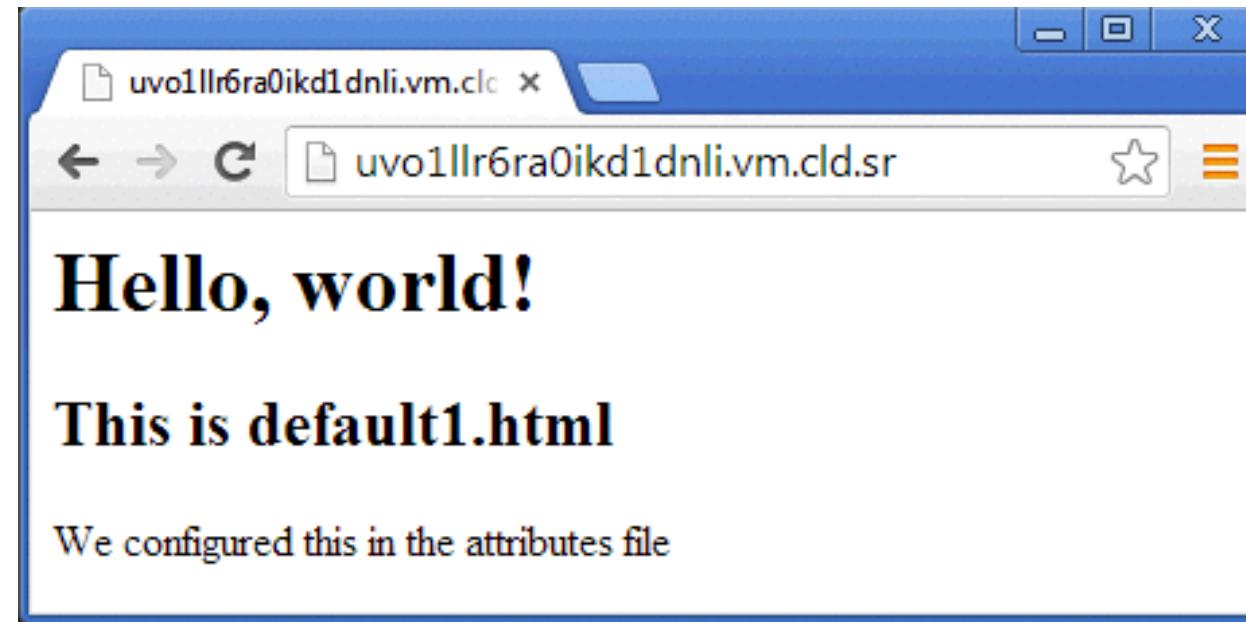
```
remote@PS> chef-client
```

```
...
* 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



Exercise: Set attribute in the recipe

OPEN IN EDITOR: Cookbooks\iis_demo\recipes\default.rb

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

node.default["iis_demo"]["indexfile"] = "Default2.htm"
cookbook_file 'c:\inetpub\wwwroot\Default.htm' do
  source node["iis_demo"]["indexfile"]
  rights :read, "Everyone"
end
```

SAVE FILE!

Exercise: Add index2.html to cookbook's files/default directory

OPEN IN EDITOR: cookbooks\iis_demo\files\default\Default2.htm

```
<html>
<body>
  <h1>Hello, world!</h1>
  <h2>This is Default2.htm</h2>
  <p>We configured this in the recipe</p>
</body>
</html>
```

SAVE FILE!

Exercise: Upload the cookbook

```
PS\> knife cookbook upload iis_demo
```

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

Exercise: Re-run Chef Client

```
remote@PS\> chef-client
```

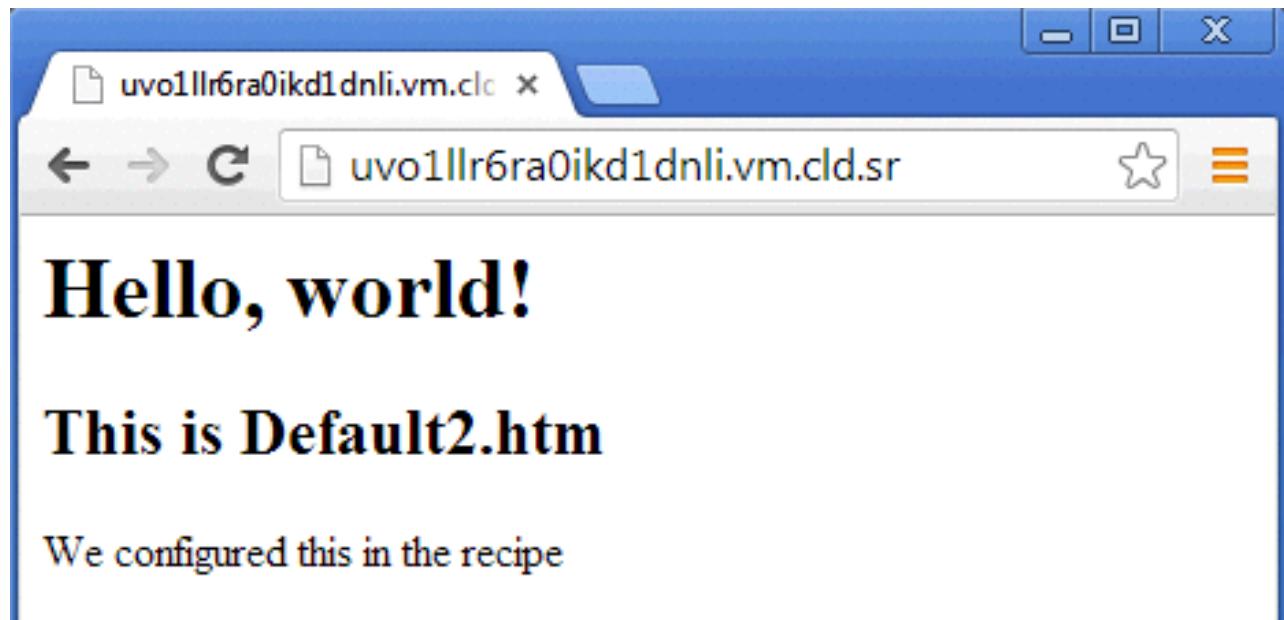
```
* cookbook_file[c:\inetpub\wwwroot\Default.htm] action create[2014-02-25T01:54:29-08:00] INFO: Processing cookbook_file[c:\inetpub\wwwroot\Default.htm] action create (iis_demo::default line 25)
[2014-02-25T01:54:30-08:00] INFO: cookbook_file[c:\inetpub\wwwroot\Default.htm] backed up to c:/chef/backup/inetpub/wwwroot\Default.htm.chef-20140225015430.025195
[2014-02-25T01:54:30-08:00] 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 2d8349 to 355514
  --- c:\inetpub\wwwroot\Default.htm      2014-02-25 01:43:24.000000000 -0800
  +++ C:/Users/chef/AppData/Local/Temp/4/Default.htm20140225-4300-17e2q1j 2014-02-25 01:54:30.000000000 -0800
  @@ -1,8 +1,9 @@
    <html>
      <body>
        <h1>Hello, world!</h1>
        - <h2>This is Default1.htm</h2>
        - <p>We configured this in the attributes file</p>
        + <h2>This is Default2.htm</h2>
        + <p>We configured this in the recipe</p>
      </body>
    </html>
  +
[2014-02-25T01:54:30-08:00] INFO: Chef Run complete in 3.525611 seconds
[2014-02-25T01:54:30-08:00] INFO: Removing cookbooks/iis_demo/files/default/Default1.htm from the cache; it is no longer needed by chef-client.

Chef Client finished, 2/4 resources updated in 16.099269 seconds
```

Exercise: Verify new homepage works

- Open a web browser
- Recipe value has taken precedence



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 this, a detailed view for 'Node: node1' is shown, with tabs for Details, Attributes (which is selected), and Permissions. The 'Attributes' section includes 'Expand All' and 'Collapse All' buttons. A specific attribute, 'iis_demo' with the value 'indexfile: Default2.htm', is highlighted with an orange border. Other visible attributes 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
- + tags:
- + languages
- + kernel
- os: windows
- os_version: 6.2.9200
- + chef_packages
- hostname: C1263834251
- fqdn: C1263834251
- domain:
- + network
- + counters

Exercise: Viewing attributes using knife

```
PS\> knife node show node1 -l | more
```

```
...
Attributes:
tags:

Default Attributes:
iis_demo:
  indexfile: Default2.htm

Override Attributes:

Automatic Attributes (Ohai Data):
chef_packages:
  chef:
    chef_root: C:/opscode/chef/embedded/lib/ruby/gems/1.9.1/gems/chef-11.10.4-x86-mingw32/lib
    version: 11.10.4
  ohai:
    ohai_root: C:/opscode/chef/embedded/lib/ruby/gems/1.9.1/gems/ohai-6.20.0/lib/ohai
    version: 6.20.0
command:
counters:
  network:
...
...
```

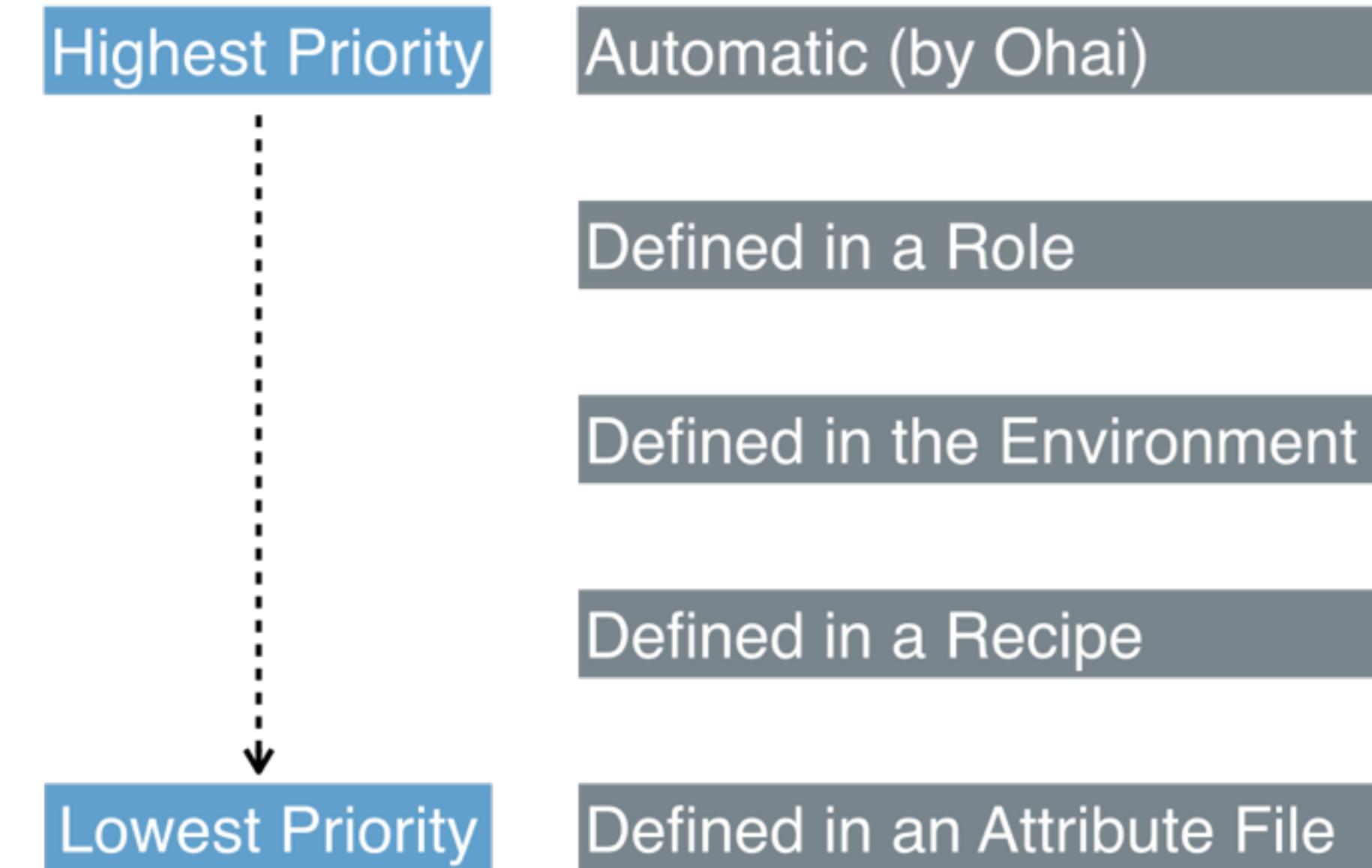
Setting Attributes in Roles and Environments

- Attributes can also be set in Roles and Environments
- These use raw JSON or Ruby format

```
default_attributes({ "iis_demo" => { "port" => 80} })
```

- More later in Roles and Environments sections...

Default Attribute Precedence



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

Questions

- What is an attribute?
- Where can attributes be set?
- What knife command 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 Hosts Cookbook

v2.1.1_WIN

Lesson Objectives

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

The Problem and the Success Criteria

- **The Problem:** We need to block access to the New York Times website from our computers in China
- **How:** We will use the 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 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)

Exercise: Create a cookbook named ‘datacenter’

```
PS\> knife cookbook create datacenter
```

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

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

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

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

SAVE FILE!

- Creates a new Node attribute:

```
node[ "datacenter" ][ "location" ]
```

- Sets the value to the Ruby string – ‘china’
- Note: there is no space between 'default' and '['
- The 'datacenter' in the attribute is just convention so you know the cookbook the attribute was set in. This is not an enforced syntax

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

- When setting an attribute in a cookbook, it should (almost) always be a **default** attribute
- There are exceptions, but they are rare. Take my word for it. :)

Best Practice: Always make your cookbooks have default values

- If a cookbook needs an attribute to exist, it should either define a default value for it in an attribute file, or depend on another cookbook that does
- **Never rely on an attribute being created manually**

Exercise: Upload the Datacenter cookbook

```
PS\> knife cookbook upload datacenter
```

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

Exercise: Create a cookbook named ‘hosts’

```
PS\> knife cookbook create hosts
```

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

Exercise: Open the default recipe in your editor

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

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

What resource should we use?

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

Exercise: Add a template resource for hosts.erb

- Use a **template** resource

- The **name** is

```
"#{ENV['windir']}\\System32\\drivers\\etc\\hosts"
```

- The resource has one parameter

- **source "hosts.erb"**

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!

Exercise: Open hosts.erb in your Editor

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

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

Exercise: Open host.erb in your Editor

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

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

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

Exercise: Open host.erb in your Editor

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

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

- You can use any Ruby construct in a template
 - Starting with <% will evaluate the expression, but not insert the result
 - Ending with -%> will not insert a line in the resulting file

Templates Are Used For Almost All Configuration Files

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

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

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

Exercise: Upload the hosts cookbook

```
PS\> knife cookbook upload hosts
```

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

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

```
PS\> knife node run_list add node1 'recipe[hosts]'
```

```
node1:
  run_list:
    recipe[iis_demo]
    recipe[hosts]
```

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

```
Template Context:  
-----  
on line #2  
1: #This file is      ged by 'server at node[  
2: <% case node[      'fqdn' ] %>  
3:   <% when "chin      ce' or 'location'  
4:     0.0.0.0          '-%>  
5:   <% end -%>  
  
[2014-02-25T02:49:26-08:00] INFO: Running chef-client in a new process  
[2014-02-25T02:49:26-08:00] FATAL: Chef::Mixin::TemplateError: undefined method `[]' for nil:NilClass  
Chef Client failed. 1 resources updated in 16.114976 seconds  
[2014-02-25T02:49:26-08:00] INFO: Sending resource update report (run-id: d659518e-d981-49dc-9735-9b3cf5324be2)  
[2014-02-25T02:49:26-08:00] FATAL: Chef::Mixin::TemplateError: undefined method `[]' for nil:NilClass
```



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

```
Chef::Mixin::Template::TemplateError
-----
undefined method `[]' for nil:NilClass
Resource Declaration:
-----
# In C:/chef/cache/cookbooks/host/recipes/default.rb
10: template "C:\\Windows\\System32\\drivers\\etc\\hosts" do
11:   source "host.erb"
12: end
Compiled Resource:
-----
# Declared in C:/chef/cache/cookbooks/host/recipes/default.rb:9:in `from_file'
template("C:\\Windows\\System32\\drivers\\etc\\hosts") do
  provider Chef::Provider::Template
  action "create"
  retries 0
.....
.....
Template Context:
-----
on line #2
1: #This file is managed by server at <%= node[ "fqdn" ] %>
2: <% case node[ "datacenter" ][ "location" ] %>
3: <% when 'china' %>
4:   0.0.0.0    nytimes.com
5: <% else %>
[2014-02-25T02:49:26-08:00] INFO: Running queued delayed notifications before re-raising exception
[2014-02-25T02:49:26-08:00] FATAL: Stacktrace dumped to c:/chef/cache/chef-stacktrace.out
Chef Client failed. 1 resources updated in 16.114976 seconds
[2014-02-25T02:49:26-08:00] INFO: Sending resource update report (run-id: d659518e-d981-49dc-9735-9b3cf5324be2)
[2014-02-25T02:49:26-08:00] FATAL: Chef...Mixin...TemplateError: undefined method `[]' for nil:NilClass
```

Stack Traces

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

Scroll up

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

```
=====
Error executing action `create` on resource 'template[C:\Windows\System32\drivers\etc\hosts]'

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

Resource Declaration:
-----
# In C:/chef/cache/cookbooks/host/recipes/default.rb

10: template "C:\\Windows\\System32\\drivers\\etc\\hosts" do
11:   source "host.erb"
12: end
```

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

```
[2014-02-25T02:49:22-08:00] INFO: Run List expands to [iis_demo, hosts]
[2014-02-25T02:49:22-08:00] INFO: Starting Chef Run for node1
[2014-02-25T02:49:22-08:00] INFO: Running start handlers
[2014-02-25T02:49:22-08:00] INFO: Start handlers complete.
resolving cookbooks for run list: ["iis_demo", "hosts"]
[2014-02-25T02:49:23-08:00] INFO: Loading cookbooks [hosts, iis_demo]
```

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

nil:NilClass error

- The bottom line is when you see this error

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

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

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'
```

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'
```

Cookbook Metadata

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'
```

SAVE FILE!

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

Cookbook Attributes are applied for all downloaded cookbooks!

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

Exercise: Upload the hosts cookbook

```
PS\> knife cookbook upload hosts
```

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

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

```
+#This file is managed by the server c1263834251
+ 0.0.0.0      nytimes.com

-# localhost name resolution is handled within DNS itself.
-#
-# ...
[2014-02-25T03:14:51-08:00] INFO: Local node identified as: nytimes.com
[2014-02-25T03:14:58-08:00] INFO: Running cookbooks: iis_iis_webfarm
it is no longer needed to specify --fqdn for this node
[2014-02-25T03:14:58-08:00] INFO: Running handlers: [2014-02-25T03:14:58-08:00] INFO: Running handlers complete
[2014-02-25T03:14:58-08:00] INFO: Report handlers complete
Chef Client finished, 2/5 resources updated in 15.412866 seconds
[2014-02-25T03:14:58-08:00] INFO: Sending resource update report (run-id: fcfcd6d8-3bf6-4965-b71c-
caae1ef53ebf)
```

Exercise: Check your work

```
remote@PS\> gc C:\Windows\System32\drivers\etc\hosts
```

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

Exercise: Check automated backups

```
remote@PS\> gci C:\chef\backup\Windows\System32\drivers\etc
```

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

Exercise: Show your test node's datacenter attribute

```
PS\> knife search node "*:*" -a datacenter
```

```
1 items found
target1:
  datacenter:
    location: china
```

Exercise: Change your node's Location

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

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

SAVE FILE!

Exercise: Upload the datacenter cookbook

```
PS\> knife cookbook upload datacenter
```

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

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

```
* template[C:\Windows\System32\drivers\etc\hosts] action create[2014-02-25T03:26:02-08:00]
INFO: Processing template[C:\Windows\System32\drivers\etc\hosts] action create (hosts::default
line 9)
[2014-02-25T03:26:02-08:00] INFO: template[C:\Windows\System32\drivers\etc\hosts] backed up to
c:/chef/backup\Windows\System32\drivers\etc\hosts.chef-20140225032602.580921
[2014-02-25T03:26:02-08:00] INFO: template[C:\Windows\System32\drivers\etc\hosts] updated file
contents C:\Windows\System32\drivers\etc\hosts

- update content in file C:\Windows\System32\drivers\etc\hosts from 94a261 to 1f8bdf
  --- C:\Windows\System32\drivers\etc\hosts          2014-02-25 03:14:57.000000000 -0800
  +++ C:/Users/chef/AppData/Local/Temp/4/chef-rendered-template20140225-4940-1y6uwwv
2014-02-25 03:26:02.000000000 -0800
    @@ -1,4 +1,3 @@
      #This file is managed by server at C1263834251
    - 0.0.0.0      nytimes.com

[2014-02-25T03:26:03-08:00] INFO: Chef Run complete in 3.338407 seconds
```

Exercise: Check your work

```
remote@PS\> gc C:\Windows\System32\drivers\etc\hosts
```

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

Exercise: Show your test node's datacenter attribute

```
PS\> knife node show node1 -a datacenter
```

```
target1:  
  datacenter:  
    location: russia
```

Congratulations!

- You now know 3 very important resources for Windows configuration management
 - Powershell_script
 - Template
 - Service

Questions

- What goes in a cookbook's attribute files?
- What are the 4 different levels of precedence?
- When do you need to specify a cookbook dependency?
- What does <% mean, and where will you encounter it?
- What are the 3 important resources in Windows configuration management?

Template Variables, Notifications, and Controlling Idempotency

Refactoring the iis_demo Cookbook

v2.1.1_WIN

Lesson Objectives

- After completing the lesson, you will be able to
 - Use the powershell resources
 - Control idempotence manually with not_if and only_if
 - Navigate the Resources page on docs.opscode.com
 - Describe the Directory resource
 - Implement resource notifications
 - Explain Template Variables, and how to use them
 - Use Ruby variables, loops, and string expansion

The Problem and the Success Criteria

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

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/>

Exercise:

Create a default.rb attribute file in the iis_demo cookbook

OPEN IN EDITOR: \cookbooks\iis_demo\attributes\default.rb

```
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

Exercise: Open the default iis_demo recipe in your editor

OPEN IN EDITOR: \cookbooks\iis_demo\recipes\default.rb

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

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

node.default["iis_demo"]["indexfile"] = "Default2.htm"
cookbook_file "c:\\inetpub\\wwwroot\\Default.htm" do
  source node["iis_demo"]["indexfile"]
  rights :read, "Everyone"
end
```

SAVE FILE!

Execute 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 execute resources idempotent

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

Building the resource collection - revisited

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

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

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

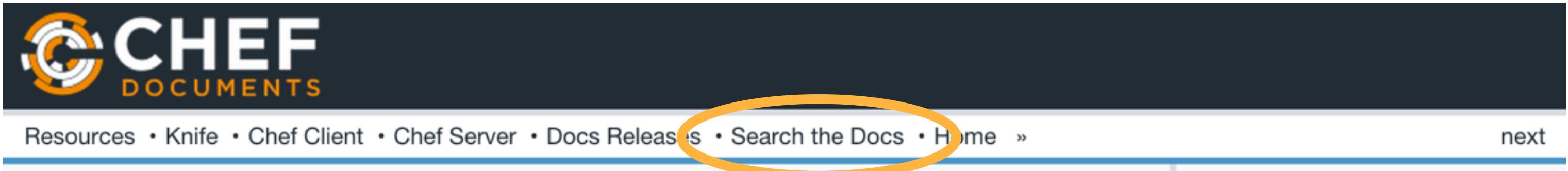
```
if !<code to check if IIS is running>
  execute "start-iis" do
    notifies :start, "service[iis]"
  end
end
```

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

Best Practice: The Chef Docs Site

- The Chef Docs Site is the home for all of the documentation about Chef.
 - It is very comprehensive
 - It has a page on every topic
- <http://docs.chef.io>
- Let's use the docs to learn more about **not_if** and **only_if**

Exercise: Search for more information about Resources



The screenshot shows the top navigation bar of the Chef Documentation website. The navigation links include: Resources • Knife • Chef Client • Chef Server • Docs Releases • **Search the Docs** • Home ». The link "Search the Docs" is circled in yellow.

All about Chef ...

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

Getting Started

From the beginning: [An Overview of Chef](#) | [About Workstations](#) | [About the Chef Server](#) | [About Nodes](#) | [About Cookbooks](#) | [About Attributes](#) | [About Resources and Providers](#) | [About LWRPs \(Custom Resources\)](#) | [About Knife](#) | [About Chef for Windows](#)

Table Of Contents

- All about Chef ...
 - Getting Started
 - The Community
 - The Workstation
 - Chef DK
 - The Chef Server
 - Server Essentials
 - Chef Analytics
 - The Nodes
 - Cookbooks

Exercise: Search for more information about Resources

Search the Documentation for Chef

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

The screenshot shows a search interface for the Chef Documentation. At the top is a search bar containing the word "guards". To the right of the search bar are a blue "X" button and an orange search icon. Below the search bar are three tabs: "All", "Chef Documentation" (which is selected), and "Cookbooks". Underneath the tabs, it says "About 44 results (0.31 seconds)". To the right of this, there is a "Sort by:" dropdown set to "Relevance". The main content area displays two search results. The first result is circled in orange. It has a blue link "Common Functionality — Chef Docs" and a URL "https://docs.getchef.com/resource_common.html". Below the link, a snippet of text reads: "A **guard** attribute can be used to evaluate the state of a node during the execution phase of the chef-client run. Based on the results of this evaluation, a **guard** ...". The second result is partially visible below it.

guards

All Chef Documentation Cookbooks

About 44 results (0.31 seconds) Sort by: Relevance

[Common Functionality — Chef Docs](#)
https://docs.getchef.com/resource_common.html

A **guard** attribute can be used to evaluate the state of a node during the execution phase of the chef-client run. Based on the results of this evaluation, a **guard** ...
Labeled Chef ...

[powershell_script — Chef Docs](#)
https://docs.getchef.com/resource_powershell_script.html

Table Of Contents. powershell_script. Syntax; Actions; Attributes. **Guards**. Providers; Examples ... Use

The Resources Page



[Resources](#) • [Knife](#) • [Chef Client](#) • [Chef Server](#) • [Docs Releases](#) • [Search the Docs](#) • [Home](#) »

[previous](#) | [next](#)

Common Functionality

All resources (and lightweight resources) share a set of common actions, attributes, conditional executions, notifications, and relative path options.

Actions

The following actions are common to every resource:

Action	Description
<code>:nothing</code>	Use to define a resource that does nothing. This action is often used to define a resource that is later notified by other resources.

Table Of Contents

Common Functionality

- Actions
 - Examples
- Attributes
 - Examples
- Guards
 - Attributes
 - Arguments
 - `not_if` Examples
 - `only_if` Examples
- Guard Interpreters
 - Attributes
 - Inheritance
 - Examples
- Lazy Attribute Evaluation
- Notifications
 - Notifications Timers

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...

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
```

```
#node.default["iis_demo"]["indexfile"] = "Default2.htm"
#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!

- Comment out the cookbook_file resource from recipe
- 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
```

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

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}"

  directory site_dir
```

SAVE FILE!

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

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 }`

Exercise: Iterate over each IIS site

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

  directory site_dir
```

- Store directory path in a variable called **site_dir**
- **directory** resource creates a directory
- **#{ENV['SYSTEMDRIVE']}** – insert local environment variable
- **#{site_name}** means “insert the value of **site_name** here”

Exercise: Iterate over each IIS site

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

- First pass
 - The value of `site_dir` is the string “`c:\\inetpub\\wwwroot\\clowns`”
- Second pass
 - The value of `site_dir` is the string “`c:\\inetpub\\wwwroot\\bears`”

Pop Quiz: 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 ...
```

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

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

c:\foo\bar

c:\site_dir

c:\foo\bar

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.

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})){
    $NewWebsiteParams = @{
      Name= '#{site_name}';
      Port= #{site_data["port"]};
      PhysicalPath= '#{site_dir}';
      ApplicationPool= '#{site_name}'
    }
    New-Website @NewWebsiteParams
  }
  elseif ((Get-WebBinding -Name #{site_name}).bindingInformation -ne '*:#{site_data["port"]}:') {
    $CurrentBinding = (Get-WebBinding -Name #{site_name}).bindingInformation
    $BindingParameters = @{
      Name= '#{site_name}';
      Binding= $CurrentBinding;
     PropertyName= 'Port';
      Value = #{site_data["port"]}
    }
    Set-WebBinding @BindingParameters
  }
  Get-Website -Name #{site_name} | Where {$_.state -like 'Stopped'} | Start-Website
  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

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: 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
```

Exercise: Add a file resource to set the permissions on the Default.htm files

```
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

- Don't forget the last **end**
- Delete the rest of the lines below

The entire recipe



WHO'S AWESOME?

You're Awesome

<http://tinyurl.com/pelqotr>

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 the two **template variables** are prefixed with an @ symbol

Exercise: Upload the iis_demo cookbook

```
PS\> knife cookbook upload iis_demo
```

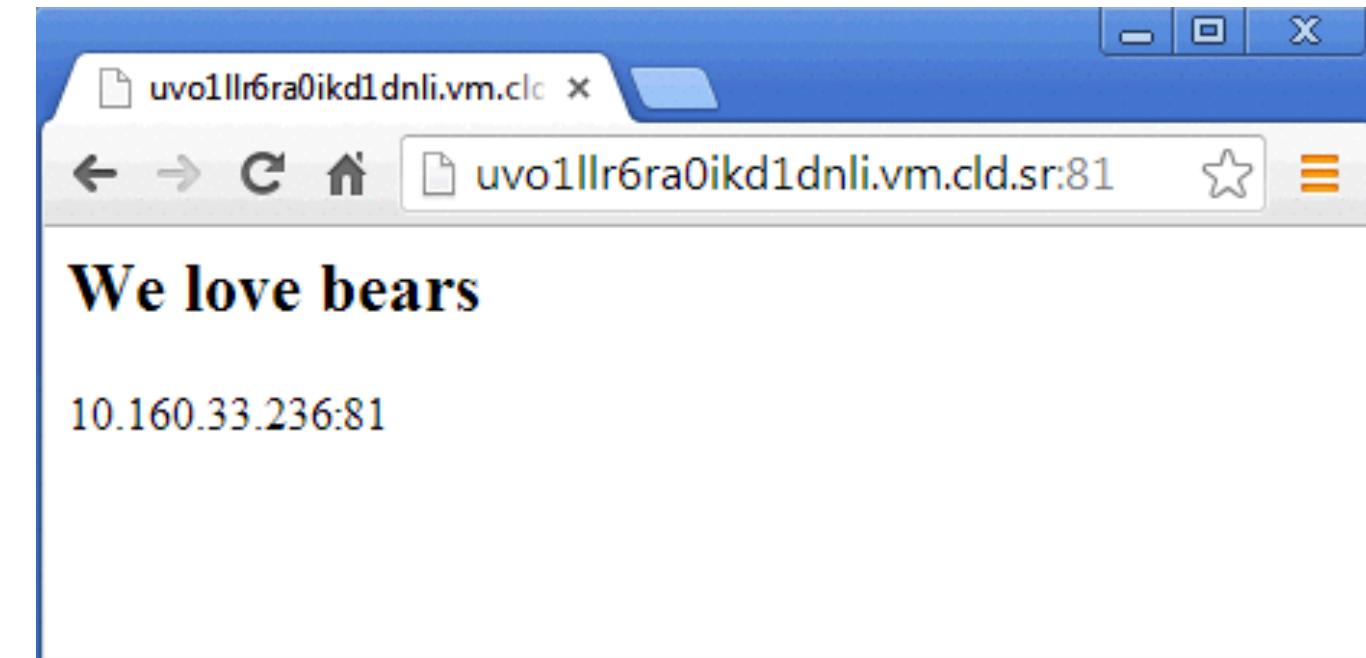
```
Uploading iis_demo [ 0.2.0 ]
Uploaded 1 cookbook.
```

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

```
"C:/Users/ADMINI~1/AppData/Local/Temp/2/chef-script20130918-1312-ebgd1i.ps1"
 * template[C:/inetpub/wwwroot/bears/Default.htm] action create
INFO: Processing template[C:/inetpub/wwwroot/bears/Default.htm] action create
(iis_demo::default line 45)
  (up to date)
Recipe: hosts::default
 * template[C:/Windows/System32/drivers/etc/hosts] action create
INFO: Processing template[C:/Windows/System32/drivers/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!



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

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

Questions

- How do you control the idempotence of an `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?

Search

A deeper dive....

v2.1.1_WIN

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

What is search?

- Use search to query data indexed on the chef server
- 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 server and is **invoked from** within a recipe, using knife or via WebUI

Exercise: Use knife search

```
PS\> knife search node "*:*"
```

```
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
Platform:       windows 6.2.9200
Tags:
```

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?"
```

Indexing and search delay

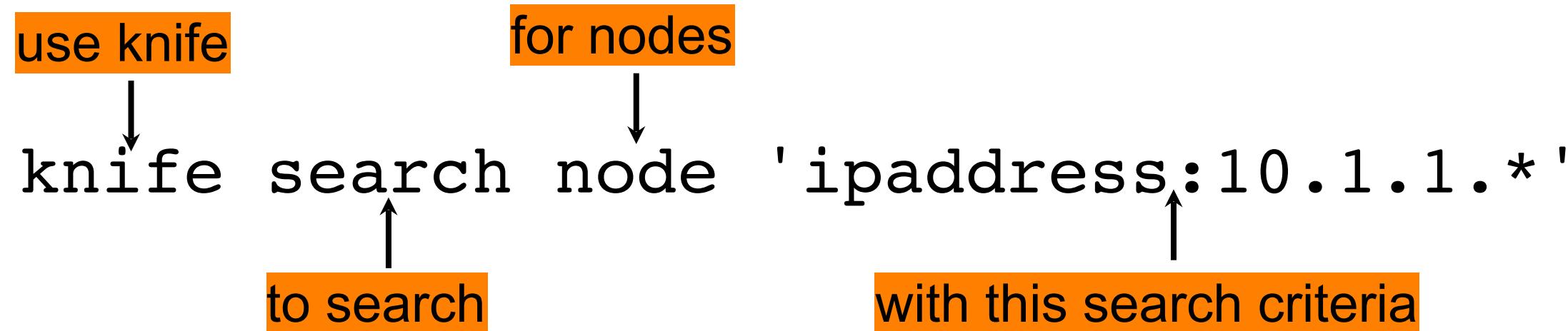
- Search is based on an index, so if you write to a node it may not be immediately available

```
node.default["serveradmin"] = "Jane"  
search("node", "*:*").each do |servers|  
  user servers["serveradmin"]  
end
```

BAD PRACTICE!

- Here, since the attribute value has just been written the search will not pick it up until the index is rebuilt
- Also, node attributes changed during a Chef run aren't posted to the server until the entire Chef run is successful
- But the value is stored in memory - so no need to search!

Remember what you're searching for!



- A successful search returns **all objects**, not just the attributes, that satisfy the search criteria
- In other words 'search for nodes containing this attribute', **NOT** 'search for this attribute across nodes'!

Exercise: Find a node with the specific attribute value

```
PS\> knife search node "ipaddress:10.*"
```

```
1 items found
```

```
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
Platform:       windows 6.2.9200
Tags:
```

Exercise: Using Knife to find an attribute value

```
PS\> knife search node "*:*" -a ipaddress
```

```
1 items found
```

```
node1:
```

```
  ipaddress: 10.160.33.236
```

Exercise: Return just the attribute value

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

```
1 items found
```

```
node1:
```

```
ipaddress: 10.160.33.236
```

Exercise: Boolean matching

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

```
1 items found

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
Platform:       windows 6.2.9200
Tags:
```

Exercise: Boolean matching

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

```
1 items found
```

```
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
Platform:       windows 6.2.9200
Tags:
```

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\iis_demo\recipes\ip-logger.rb

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

end
```

SAVE FILE!

Exercise: Upload the iis_demo cookbook

```
PS\> knife cookbook upload iis_demo
```

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

Recipe Naming

- Recipes are referenced using the notation ‘**cookbook::recipe**’, where ‘recipe’ is the name of the ‘**.rb**’ file in the “**cookbook/recipe**” directory
- The “**default.rb**” recipe for a given cookbook may be referred to just the cookbook name (e.g. ‘*mysql*’)
- If we added another recipe to this cookbook named ‘**backup**’, we would refer to it as ‘*mysql::backup*’

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

```
PS\> knife node run_list add node1 'recipe[iis_demo::ip-logger]'
```

```
node1:  
  run_list:  
    recipe[iis_demo]  
    recipe[hosts]  
    recipe[iis_demo::ip-logger]
```

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

```
Recipe: iis_demo::ip-logger
  * log[The Windows servers in your organization have the following FQDN/
IP Addresses:- C1263834251/10.160.33.236] action
write[2014-02-25T06:19:43-08:00] INFO: Processing log[The Windows servers
in your organization have the following FQDN
/IP Addresses:- C1263834251/10.160.33.236] action write (iis_demo::ip-
logger line 2)
[2014-02-25T06:19:43-08:00] INFO: The Windows servers in your
organization have the following FQDN/IP Addresses:-
C1263834251/10.160.33.236
```

Exercise: Remove the recipe to your test node's run list

```
PS\> knife node run_list remove node1 'recipe[iis_demo::ip-logger]'
```

```
node1:  
  run_list:  
    recipe[iis_demo]  
    recipe[hosts]
```

Questions

- What search engine is chef search built upon?
- What is searchable in chef?
- What character 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

Writing a Users cookbook

v2.1.1_WIN

Lesson Objectives

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

The Problem and the Success Criteria

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

Where should we store the user data?

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

Introducing Data Bags

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

Make a data_bags directory

```
PS \> md 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

```
PS \> md data_bags\users  
PS \> knife data_bag create users
```

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

Created data_bag[users]

Exercise: Create a user item in the users data bag

OPEN IN EDITOR: data_bags\users\bobo.json

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

SAVE FILE!

Exercise: Create the data bag item

```
PS\> knife data_bag from file users bobo.json
```

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

Exercise: Create a user item in the users data bag

OPEN IN EDITOR: data_bags\users\frank.json

```
{  
  "id": "frank",  
  "comment": "Frank Belson",  
  "password": "sP3nC3r4Hire!"  
}
```

SAVE FILE!

Exercise: Create the data bag item

```
PS\> knife data_bag from file users frank.json
```

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

Exercise: Show all the users in Chef

```
PS\> knife search users '*:*'
```

```
2 items found
```

```
chef_type: data_bag_item
comment: Frank Belson
data_bag: users
id: frank
password: sP3nC3r4Hire!
```

```
chef_type: data_bag_item
comment: Bobo T. Clown
data_bag: users
id: bobo
password: iAm@Clo0wN!!!
```

Exercise: Find Bobo's id in Chef

```
PS\> knife search users 'comment:Bobo T. Clown' -a id
```

```
1 items found
```

```
data_bag_item_users_bobo:  
  id: bobo
```

Exercise: Create a data bag named groups

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

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

Created data_bag[groups]

Exercise: Create a group item in the group data bag

OPEN IN EDITOR: \data_bags\groups\clowns.json

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

SAVE FILE!

Exercise: Create the data bag item

```
PS\> knife data_bag from file groups clowns.json
```

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

Exercise: Show all the groups in Chef

```
PS\> knife search groups '*:*'
```

```
1 items found

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

Exercise: Create a cookbook named ‘users’

```
PS\> knife cookbook create users
```

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

Exercise: Open the default recipe in your editor

OPEN IN EDITOR: \cookbooks\users\recipes\default.rb

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

SAVE FILE!

Exercise: Open the default recipe in your editor

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

include_recipe "users::groups"
```

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

Exercise: Open the default recipe in your editor

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

include_recipe "users::groups"
```

- `include_recipe` ensures another recipes resources are complete before we continue
- Chef will only include each recipe once

Best Practice: Use `include_recipe` and `include_attribute` liberally

- If there is a pre-requisite for your recipe that resides in another recipe (the ASP.NET 4.5 runtime existing for your ASP.NET application, for example)
 - Always use `include_recipe` to include it specifically, even if you put it in a run list

Exercise: Open the users::group recipe in your editor

OPEN IN EDITOR: \cookbooks\users\recipes\groups.rb

```
search( :groups, "*:*").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

Exercise: Upload the users cookbook

```
PS\> knife cookbook upload users
```

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

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

```
PS\> knife node run_list add node1 'recipe[users]'
```

```
node1:  
  run_list:  
    recipe[iis_demo]  
    recipe[hosts]  
    recipe[users]
```

Exercise: Re-run the Chef Client

```
remote@PS\> 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[clowns] action create[2014-02-25T07:56:24-08:00] INFO: Processing group[clowns]
action create (users::groups line 2)
[2014-02-25T07:56:24-08:00] INFO: group[clowns] created
    - create group[clowns]
[2014-02-25T07:56:25-08:00] INFO: Chef Run complete in 6.203709 seconds
```

Exercise: Verify the users and groups exist

```
remote@PS\> net users
```

```
User accounts for \\C1263834251
-----
Administrator          bobo
frank                  Guest
The command completed successfully.
```

```
remote@PS\> net localgroup
```

```
Aliases for \\WIN-BBNT36Q0RB2
*Administrators
...
*circus
*Performance Log Users
```

Let's review real quick..

- We just created a centralized user and group repository, from scratch in about ~40 lines.
 - (That's kind of like what LDAP and Active Directory do, only they are, well, fancier.)
- Between Data Bags and Node Attribute precedence, Chef provides a plethora of ways to inform the patterns you use to configure your infrastructure

Questions

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

Roles

Role-based Attributes and Merge Order Precedence

v2.1.1_WIN

Lesson Objectives

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

What is a Role?

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

What is a Role?

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

Best Practice: Roles live in your chef-repo

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

Exercise: Create a role named webserver

```
PS\> md roles
```

Mode	LastWriteTime	Length	Name
-----	-----	-----	-----
d----	25-Feb-14 3:07 PM		roles

Exercise: Create the webserver role

OPEN IN EDITOR: roles\webserver.rb

- A Role has a:
 - name
 - description
 - run_list

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

SAVE FILE!

Exercise: Create the webserver role

OPEN IN EDITOR: roles\webserver.rb

- You can set default node attributes within a role.

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

SAVE FILE!

Exercise: Create the role

```
PS\> knife role from file webserver.rb
```

Updated Role webserver!

Exercise: Show the role with knife

```
PS\> knife role show webserver
```

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

Exercise: Search for the roles that have recipe[iis_demo] in their run list

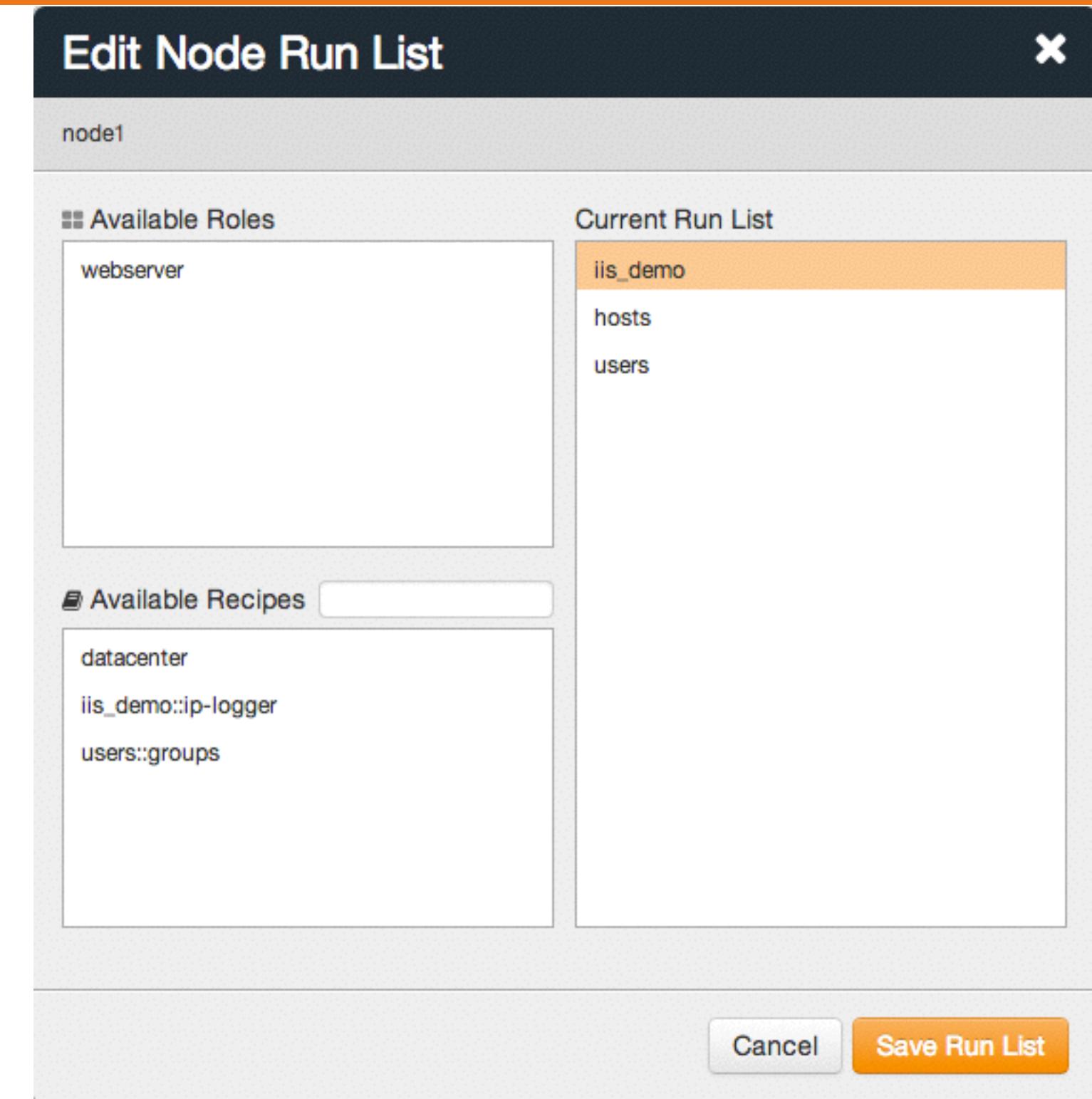
```
PS\> knife search role 'run_list:recipe\[iis_demo\]'
```

```
1 items found

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

Exercise: Replace `recipe[iis_demo]` with `role[webserver]` in run list

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



Exercise: Re-run the Chef Client

```
Remote@PS\> chef-client
```

```
[2014-02-25T09:44:14-08:00] INFO: Run List is [role[webserver],  
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
```

Exercise: Re-run the Chef Client

Remote@PS\> chef-client

```
[2014-02-25T09:44:18-08:00] INFO: directory[C:\inetpub\wwwroot\admin] created directory C:\inetpub\wwwroot\admin
  - create new directory C:\inetpub\wwwroot\admin
* powershell_script[create app pool for admin] action run[2014-02-25T09:44:18-08:00] INFO: Processing
powershell_script[create app pool for admin] action run (iis_demo::default line 24)

  Name          State       Applications
  ----          -----      -----
  admin         Started

[2014-02-25T09:44:19-08:00] INFO: powershell_script[create app pool for admin] ran successfully

  - execute "powershell.exe" -NoLogo -NonInteractive -NoProfile -ExecutionPolicy RemoteSigned -InputFormat None -
File"C:/Users/chef/AppData/Local/Temp/2/chef-script20140225-2140-1dltsrt.ps1"
* powershell_script[new website for admin] action run[2014-02-25T09:44:19-08:00] INFO: Processing
powershell_script[new website for admin] action run (iis_demo::default line 29)

  Name          ID  State       Physical Path           Bindings
  ----          --  -----      -----
  admin         8718 Started    C:\inetpub\wwwroot\admin
  7409

7

[2014-02-25T09:44:20-08:00] INFO: powershell_script[new website for admin] ran successfully
```

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 }
```

- While our role has...

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

Exercise: Search for the IIS sites attribute on all nodes with the role webserver

```
PS\> knife search node 'role:webserver' -a iis_demo.sites
```

```
1 items found
```

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

Exercise: Edit the webserver role

OPEN IN EDITOR: roles\webserver.rb

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

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

SAVE FILE!

Exercise: Create the role

```
PS\> knife role from file webserver.rb
```

Updated Role webserver!

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

Exercise: Search for the apache sites attribute on all nodes with the role webserver

```
PS\> knife search node 'role:webserver' -a iis_demo.sites
```

```
1 items found
```

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

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

Merge Order and Precedence

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

Best Practice: Roles get default attributes

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

Best Practice: Have “base” roles

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

Exercise: Create the base role

OPEN IN EDITOR: roles\base.rb

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

SAVE FILE!

Exercise: Create the role

```
PS\> knife role from file base.rb
```

Updated Role base!

Exercise:

Add the base role to the webserver role's run list

OPEN IN EDITOR: roles\webserver.rb

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

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

SAVE FILE!

Exercise: Update the role

```
PS\> knife role from file webserver.rb
```

Updated Role webserver!

Exercise: Re-run the Chef Client

```
remote@PS\> chef-client
```

```
...
[2014-02-25T10:05:36-08:00] INFO: Run List is [role[webserver], recipe[hosts], recipe[users]]
[2014-02-25T10:05:36-08:00] INFO: Run List expands to [hosts, users, iis_demo]
[2014-02-25T10:05:36-08:00] INFO: Starting Chef Run for node1
[2014-02-25T10:05:36-08:00] INFO: Running start handlers
[2014-02-25T10:05:36-08:00] INFO: Start handlers complete.
resolving cookbooks for run list: ["hosts", "users", "iis_demo"]
[2014-02-25T10:05:37-08:00] INFO: Loading cookbooks [datacenter, hosts, iis_demo, users]
Synchronizing Cookbooks:
 - hosts
 - datacenter
 - users
 - iis_demo
Compiling Cookbooks...
...
```

Best Practice: Be explicit about what you need or expect

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

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

- Remove all the entries in the run list other than `role[webserver]`, then **save** and **close**.

Questions

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

Environments

Cookbook Version Constraints, Override Attributes, and Per Environment Run Lists

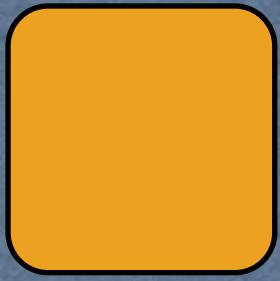
v2.1.1_WIN

Lesson Objectives

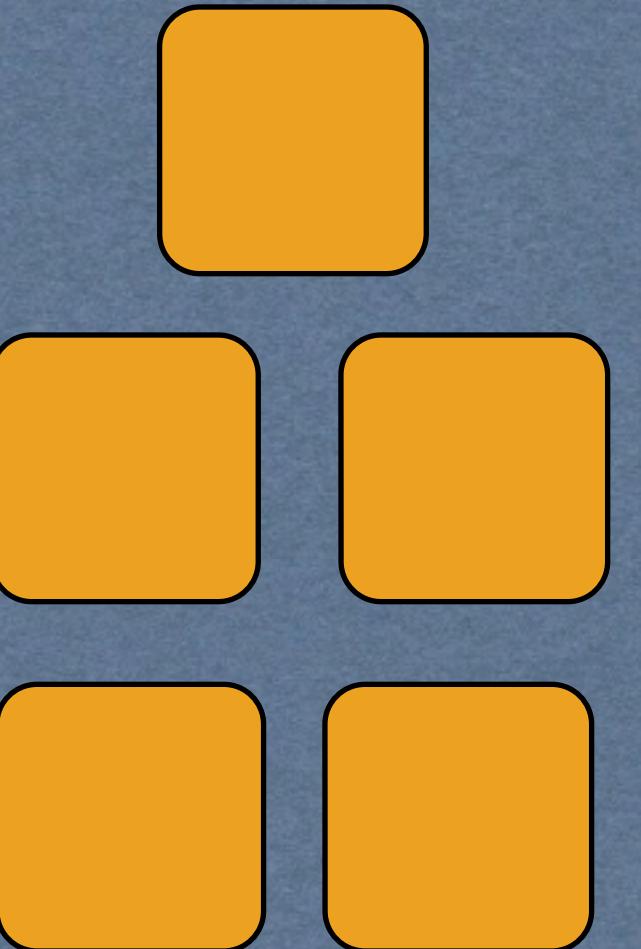
- After completing the lesson, you will be able to
 - Describe what an Environment is, and how it is different from an Organization
 - Set cookbook version constraints
 - Explain when to set attributes in an environment

Environments

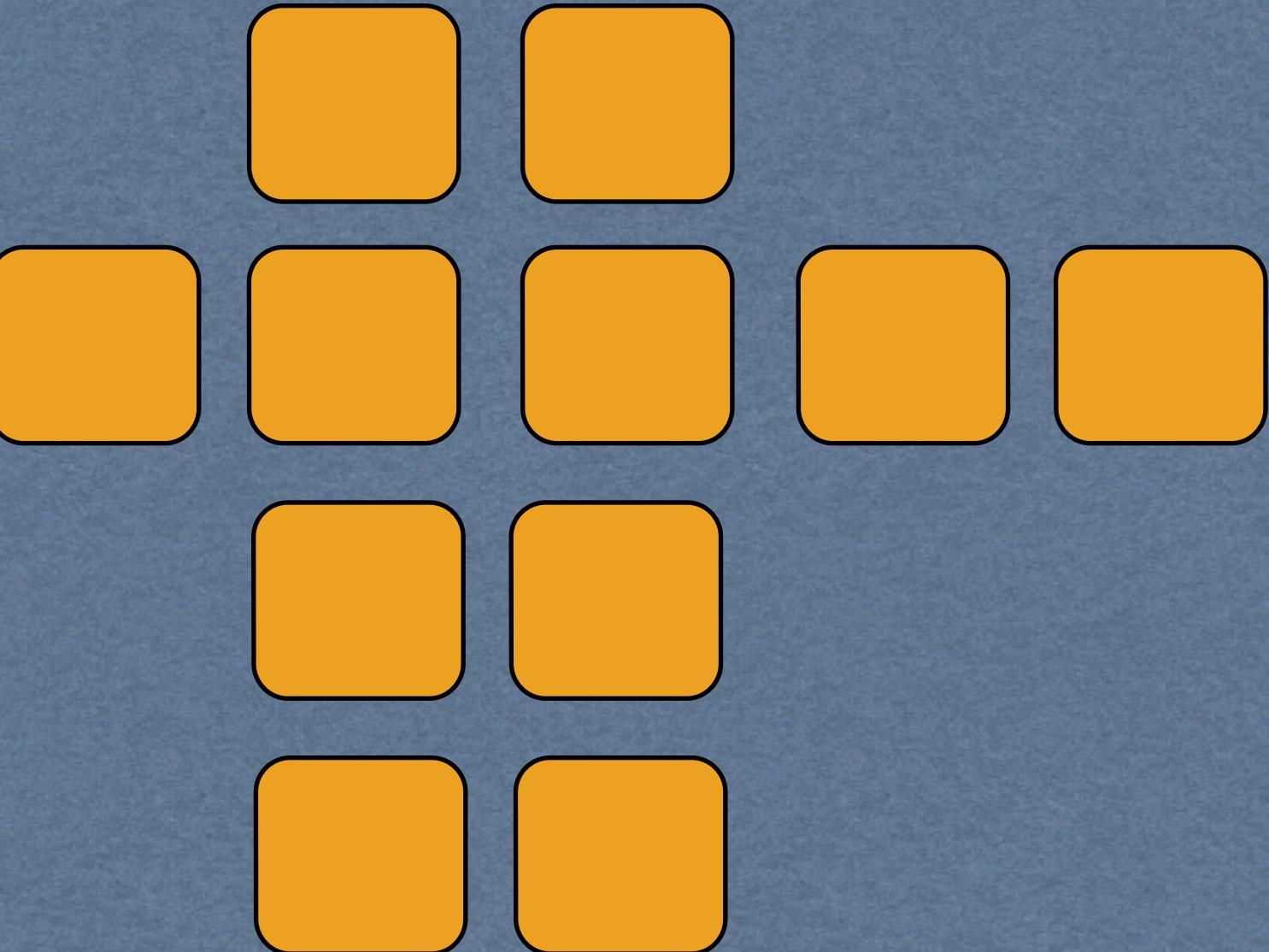
Development



Staging



Production



Organization

Environments

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

Environments Define Policy

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

Environment Best Practice

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

Exercise:

Use knife to show the available cookbook versions

```
PS\> knife cookbook show iis_demo
```

iis_demo	0.2.0	0.1.0
----------	-------	-------

Exercise: List the current environments

```
PS\> knife environment list
```

```
_default
```

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

Make an environments directory

```
PS\> mkdir environments
```

Mode	LastWriteTime	Length	Name
-----	-----	-----	-----
d----	26-Feb-14 9:36 AM		environments

Exercise: Create a dev environment

OPEN IN EDITOR: environments\dev.rb

```
name "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

Cookbook Version Constraints

- = Equal to
- There are other options but equality is the recommended practice.
- Learn more at http://docs.chef.io/chef/essentials_cookbook_versions.html

Exercise: Create the dev environment

```
PS\> knife environment from file dev.rb
```

Updated Environment dev

Exercise: Show your Chef dev environment

```
PS\> knife environment show 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 ‘node1’
- 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. On the left, there's a sidebar with options like Delete, Manage Tags, Reset Key, Edit Run List, and Edit Attributes. The main content area shows a table titled 'Showing All Nodes' with columns: Node Name, Platform, FQDN, IP Address, Uptime, Last Check-In, Environment, and Actions. One row is highlighted for 'node1'. Below the table, a specific node details panel is open for 'node1'. 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). The 'Environment:' dropdown is set to 'dev'. A yellow message box says 'Node environment changed from _default to dev.' with 'Cancel' and 'Save' buttons. At the bottom, it lists 'Platforms: windows', 'FQDN: C1263834251', and 'IP Address: 10.160.33.236'.

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

```
PS: \> knife node environment set node1 dev
```

```
node1:  
  chef_environment: dev
```

Exercise: Re-run the Chef Client

```
Remote@PS\> chef-client
```

```
[2014-02-26T01:46:16-08:00] INFO: Chef Run complete in  
7.988069 seconds
```

```
Running handlers:
```

```
[2014-02-26T01:46:16-08:00] INFO: Running report handlers
```

```
Running handlers complete
```

```
[2014-02-26T01:46:16-08:00] INFO: Report handlers complete
```

Exercise: Create a production environment

OPEN IN EDITOR: environments\production.rb

- Make sure the `iis_demo` cookbook is set to version 0.1.0
- Set an override attribute for being in china for our DC environment no matter what.

```
name "production"
description "For Prods!"
cookbook "iis_demo", "= 0.1.0"
override_attributes({
  "datacenter" => {
    "location" => "china"
  }
})
```

SAVE FILE!

Exercise: Create the production environment

```
PS\> knife environment from file production.rb
```

Updated Environment production

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

```
PS:\> knife node environment set node1 production
```

```
node1:  
  chef_environment: production
```

Exercise: Re-run the Chef Client

```
Remote@PS\> chef-client
```

```
* template[C:\Windows\System32\drivers\etc\hosts] action create[2014-02-26T01:57:44-08:00]
INFO: Processing template[C:\Windows\System32\drivers\etc\hosts] action create
(hosts::default line 9)
[2014-02-26T01:57:44-08:00] INFO: template[C:\Windows\System32\drivers\etc\hosts] backed up
to c:/chef/backup\Windows\System32\drivers\etc\hosts.chef-20140226015744.972901
[2014-02-26T01:57:44-08:00] INFO: template[C:\Windows\System32\drivers\etc\hosts] updated
file contents C:\Windows\System32\drivers\etc\hosts
```

```
- update content in file C:\Windows\System32\drivers\etc\hosts from 1f8bdf to 94a261
  --- C:\Windows\System32\drivers\etc\hosts          2014-02-25 03:26:02.000000000 -0800
  +++ C:/Users/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 C1263834251
  +  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 `iis_demo` cookbook
- While the recipe applied fine, investigating the system will reveal IIS is still configured as it was in the 0.2.0 cookbook
- A better way to ensure a smooth rollback: write contra-resources to clean up, and have a new version of the cookbook.

Exercise: Create a production environment

OPEN IN EDITOR: environments\production.rb

- Make sure the `iis_demo` cookbook is set to version 0.2.0

Exercise: Upload the updated production environment

```
PS\> knife environment from file production.rb
```

Updated Environment production

Questions?

- What is an Environment?
- How is it different from an Organization?
- What is a cookbook version constraint?
- Which cookbook constraint should we most likely be using?
- What kind of node attributes do you typically set from an Environment?
- What is a per-environment run list?

Using Community Cookbooks

Open Source: Saving you time!

v2.1.1_WIN

Lesson Objectives

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

The easy way...

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

Exercise: Find and preview cookbooks on the site

The screenshot shows the homepage of the Chef Supermarket website. 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 is a search bar with a blue button labeled "GO". The main content area features a dark header with the text "Welcome to Supermarket. Find, explore and view Chef cookbooks for all of your ops needs." Below this, there are three buttons: "Sign Up For a Chef Account" and "Sign In With Your Chef Account". Underneath these buttons is a horizontal line separating them from the main content. The main content is divided into three columns: "Explore", "Learn", and "Share". Each column contains a list of links. At the bottom of the page, there is a section titled "Community Stats" with the numbers "1,644 Cookbooks" and "58,717 Chefs". The footer of the page includes a copyright notice: "© 2008–2014 Chef Software, Inc. All Rights Reserved."

COOKBOOKS CONTRIBUTORS TOOLS & PLUGINS

CREATE ACCOUNT SIGN IN

Cookbooks GO

Welcome to Supermarket. Find, explore and view Chef cookbooks for all of your ops needs.

Sign Up For a Chef Account Sign In With Your Chef Account

Explore

- [Browse Cookbooks](#)
- [Read the Chef Blog](#)

Learn

- [Learn Chef](#)
- [Read the Chef Docs](#)
- [Community Guidelines](#)
- [How to Contribute](#)

Share

- [Share your cookbooks](#)
- [Chat on IRC at #chef on irc.freenode.net](#)
- [Join the Supermarket Mailing List](#)
- [Contribute to Supermarket](#)

Community Stats

1,644 Cookbooks 58,717 Chefs

© 2008–2014 Chef Software, Inc. All Rights Reserved.

Exercise: Search for a chef-client cookbook

The screenshot shows a web browser displaying the Chef Supermarket at <https://supermarket.getchef.com>. The search bar contains the query "Search for: chef-client". The results page features a welcome message, account creation and sign-in buttons, and sections for Explore, Learn, and Share. Key statistics at the bottom indicate 1,644 Cookbooks and 58,717 Chefs.

https://supermarket.getchef.com

CHEF
SUPERMARKET

COOKBOOKS CONTRIBUTORS TOOLS & PLUGINS

CREATE ACCOUNT SIGN IN

Cookbooks ▾ Search ← **Search for: chef-client**

Welcome to Supermarket. Find, explore and view Chef cookbooks for all of your ops needs.

[Sign Up For a Chef Account](#) [Sign In With Your Chef Account](#)

Explore

- [Browse Cookbooks](#)
- [Read the Chef Blog](#)

Learn

- [Learn Chef](#)
- [Read the Chef Docs](#)
- [Community Guidelines](#)
- [How to Contribute](#)

Share

- [Share your cookbooks](#)
- [Chat on IRC at #chef on irc.freenode.net](#)
- [Join the Supermarket Mailing List](#)
- [Contribute to Supermarket](#)

Community Stats

1,644 Cookbooks 58,717 Chefs

© 2008–2014 Chef Software, Inc. All Rights Reserved.

Search Results...

The screenshot shows the Chef Supermarket search results page for the query "chef-client". The search bar at the top contains "chef-client". Below the search bar, there are 26 cookbooks listed. The first result is "chef-client" by "chef" (version 3.8.0, updated September 5, 2014). It is described as managing client.rb configuration and the chef-client service. The cookbook link is highlighted with a blue arrow pointing to it from the text "We're probably looking for this one" which is overlaid on the page. The second result is "chef-client_syslog" by "sawanoiboly" (version 0.1.1, updated September 6, 2014), which manages chef-client log to syslog.

https://supermarket.getchef.com/cookbooks?utf8=%E2%9C%93&q=chef-client

CHEF SUPERMARKET

COOKBOOKS CONTRIBUTORS TOOLS & PLUGINS

CREATE ACCOUNT SIGN IN

Cookbooks ▾ chef-client GO

26 Cookbooks RSS Sort by Most Downloaded Most Followed

chef-client 3.8.0 Updated September 5, 2014

Manages client.rb configuration and chef-client service

cookbook 'chef-client', '~> 3.8.0'

SUPPORTED PLATFORMS

We're probably looking for this one

chef-client_syslog 0.1.1 Updated September 6, 2014

chef-client log to syslog

cookbook 'chef-client_syslog', '~> 0.1.1'

SUPPORTED PLATFORMS

12,365,357 TOTAL DOWNLOADS 149 FOLLOWERS Follow

299,428 TOTAL DOWNLOADS 1 FOLLOWER Follow

Viewing a cookbook

The screenshot shows a web browser displaying the Chef Supermarket at <https://supermarket.getchef.com/cookbooks/chef-client>. The page is for the 'chef-client' cookbook, version 3.8.0. The interface includes a navigation bar with links for COOKBOOKS, CONTRIBUTOR, and TOOLS & PLUGINS, and buttons for CREATE ACCOUNT and SIGN IN. A search bar with a 'GO' button is present. The main content area features a summary of the cookbook, including its name, version, maintainer (Berkshelf), and a snippet of the code. Below this are tabs for README, Dependencies, Changelog, and Foodcritic. The 'README' tab is highlighted with an orange arrow pointing to it from the left. To the right, there's a sidebar with details about the cookbook's author (Chef Software) and a 'View Source' button, which is also highlighted with an orange arrow and a dashed orange box labeled 'Browse Source Code'.

README displayed on the page (if it has one)

chef-client

Manages client.rb configuration and chef-client service

Berkshelf Librarian Knife

cookbook 'chef-client', '~> 3.8.0'

README Dependencies Changelog Foodcritic

chef-client Cookbook

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

Requirements

- Chef 0.10.14+
- Dhais 0.6.12+

Chef 10.14.0 or greater is recommended to make use of the `client_fork` configuration option.

Platforms

The following platforms are tested directly under test-kitchen; see `.kitchen.yml` and `TESTING.md` for details.

- Ubuntu 10.04, 12.04
- CentOS 5.9, 6.4
- Debian 6.0.7
- SUSE (SLES) 11-sp2
- OmniOS r151006c

The following platforms are known to work:

- Debian family (Debian, Ubuntu etc)
- Red Hat family (Redhat, CentOS, Oracle etc)

View Source

UPATED ON SEPTEMBER 5, 2014
Created on December 16, 2010

Platforms

LICENSE Apache 2.0

14,033 DOWNLOADS OF VERSION 12

Download Cookbook

Browse Source Code

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

v2.1.1_WIN

Exercise: Use knife to search the community site

```
PS\> knife cookbook site search chef-client
```

```
chef:
  cookbook:          http://cookbooks.opscode.com/api/v1/cookbooks/chef
  cookbook_description: Installs and configures Chef for chef-client and chef-server
  cookbook_maintainer: chef
  cookbook_name:      chef

chef-client:
  cookbook:          http://cookbooks.opscode.com/api/v1/cookbooks/chef-client
  cookbook_description: Manages client.rb configuration and chef-client service
  cookbook_maintainer: chef
  cookbook_name:      chef-client

chef-client-cron:
  cookbook:          http://cookbooks.opscode.com/api/v1/cookbooks/chef-client-cron
  cookbook_description: Manages aspects of only chef-client
  cookbook_maintainer: bryanwb
  cookbook_name:      chef-client-cron

chef-client_syslog:
.....
```

Exercise: Use knife to show the chef-client cookbook on the community site

```
PS\> knife cookbook site show chef-client
```

```
average_rating:
category:          Other
created_at:        2010-12-16T23:00:45.000Z
deprecated:        false
description:       Manages client.rb configuration and chef-client service
external_url:      http://github.com/opscode-cookbooks/chef-client
foodcritic_failure: true
latest_version:    http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/3.9.0
maintainer:         chef
metrics:
downloads:
  total: 24629844
versions:
  0.99.0: 562401
...
  3.9.0: 132395
```

Exercise: Download the chef-client cookbook

```
PS\> knife cookbook site download chef-client
```

```
Downloading chef-client from the cookbooks site at
version 3.9.0 to /Users/YOU/chef-repo/chef-
client-3.9.0.tar.gz
```

```
Cookbook saved: /Users/YOU/chef-repo/chef-
client-3.9.0.tar.gz
```

Exercise: Extract chef-client cookbook tarball

```
PS\> 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 internet.
- 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::delete_validation]", "recipe[hosts]",
"recipe[users]"
```

SAVE FILE!

- Add the **delete_validation** recipe

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

- Once Chef enters the actual run, synchronizing cookbooks, it has 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

```
PS\> knife cookbook upload chef-client
```

```
Uploading chef-client [3.0.2]
ERROR: Cookbook 'chef-client' depends on cookbook 'curl' version '1.2.0',
ERROR: which is not currently uploaded to the server.
```



Exercise: Download the cron cookbook

```
PS\> 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

```
PS\> 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

```
PS\> knife cookbook upload cron
```

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

Exercise: Upload the chef-client cookbook

```
PS\> knife cookbook upload chef-client
```

```
Uploading chef-client [3.0.2]
ERROR: Cookbook 'chef-client' depends on cookbook 'curl' version '1.2.0',
ERROR: which is not currently uploaded to the server.
```



Exercise: Download the logrotate cookbook

```
PS\> 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

```
PS\> 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

```
PS\> knife cookbook upload logrotate
```

```
Uploading logrotate [1.3.0]
Uploaded 1 cookbook.
```

Exercise: ...windows

```
PS\> knife ...
```

Downloading ...

Extracting ...

Uploading ...

Exercise: Upload the chef-client cookbook

```
PS\> knife cookbook upload chef-client
```

Uploading chef-client [3/3] [3/3] [3/3]
ERROR: Cookbook 'chef-client' depends on cookbook 'curl' version '>= 1.2.0',
ERROR: which is not currently being loaded. Could not find 'curl' on the
server.



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

```
PS\> knife role from file base.rb
```

Updated Role base!

Exercise: Re-run the Chef Client

```
remote@PS\> 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_demo]
[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]
...
```

Exercise: Verify chef-client is running

```
Remote@PS\> get-service chef*
```

Status	Name	DisplayName
-----	----	-----
Running	chef-client	Chef Client Service

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, totaling 134. A search bar labeled "Filter" is at the top of the list. The columns in the table are Server Name, Display Name, Service Name, Status, and Start Type. One service, "Chef Client Service" (Service Name: chef-client), is highlighted in blue, indicating it is currently running automatically. 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

BEST PRACTICES ANALYZER
Warnings or Errors | 0 of 0 total

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 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?

WORKING WITH CHEF, THE COMPANY

v2.1.1_WIN



Chef Status

- Status for Chef related systems can be found by browsing to
 - <http://status.chef.io>, or
 - 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 <http://www.getchef.com/support/>
 - By sending an email to 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

To re-open a closed ticket

- To re-open a closed ticket either
 - Click on the closed ticket in the web interface, and reply in the box provided, or
 - Reply to the email thread
- This will restart the ticket and move it back into the active queue

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 starts a support contract for an org, then user2@domain.com and user3@domain.com can also submit tickets on behalf of "domain.com" for that org
- Any user email address be added explicitly to the Support Tool for your org by emailing sales@chef.io or submitting a ticket

New releases

- Release Notes can be found here
 - <http://docs.chef.io/>

Further Resources

v2.1.1_WIN

Chef Fundamentals - Q&A Forum

- Chef Fundamentals Google Group Q&A Forum
- <http://bit.ly/ChefFundamentalsForum>
- Join the group and post questions

Further Resources: Cookbooks and Plugins

- Useful cookbooks
 - DNS: djbdns, pdns, dnsimple, dynect, route53
 - Monitoring: nagios, munin, zenoss, zabbix
 - Package repos: yum, apt, freebsd
 - Security: ossec, snort, cis_benchmark
 - Logging: rsyslog, syslog-ng, logstash, logwatch
- Application cookbooks:
 - application, database
 - python, java, php, ruby
- Plugins
 - Cloud: knife-ec2, knife-rackspace, knife-openstack, knife-hp
 - Windows: knife-windows
- More listed on docs.opscode.com

Further Resources

- <http://chef.io/>
- <http://supermarket.chef.io/>
- <http://docs.chef.io/>
- <http://learn.chef.io>
- <http://lists.chef.io>
- <http://www.youtube.com/user/getchef>
- irc.freenode.net #chef, #chef-hacking, #learnchef
- Twitter @chef #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





CHEF PRESENTS

#ChefConf 2015

March 31 – April 2 Santa Clara, CA
Santa Clara Convention Center



ChefConf 2015 is the largest, most vibrant gathering of web-scale IT and DevOps leaders, practitioners, and innovators. Featuring three days of inspired discussions, collaborative presentations, technical training, and hands-on labs focused on automating infrastructure and the continuous delivery of applications.

The definitive assembly for the tens of thousands-strong Chef community will convene IT leaders representing some of the most dynamic companies and thinking in the world.

Local Meetup Groups

- Fill in local meetup groups here
<http://opscode.meetup.com>
- <https://wiki.opscode.com/display/chef/Resources+for+Community+Organizers>

Just enough Ruby for Chef

A quick & dirty crash course

v2.1.1_WIN

Objectives

- Know what irb is, and how to use it.
- Become familiar with:
 - Variable Assignment
 - Basic Arithmetic
 - Strings
 - Truthiness
 - Operators
 - Arrays
 - Hashes
 - Regular Expressions
 - Conditionals
 - Method Declaration
 - Classes
 - Objects

irb - the interactive ruby shell

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

Exercise: Start irb on your ‘target’ node

```
remote@PS\> c:\opscode\chef\embedded\bin\irb
```

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

Exercise: Variable assignment

```
remote@PS\> c:\opscode\chef\embedded\bin\irb
```

```
irb(main):001:0> x = "hello"  
=> "hello"  
irb(main):002:0> puts x  
Hello  
=> nil
```

Exercise: Arithmetic

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

Exercise: Arithmetic

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

Exercise: Arithmetic

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

Exercise: Arithmetic

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

Exercise: Arithmetic

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

Exercise: Arithmetic

```
irb(main):008:0> 5.class  
=> Fixnum  
irb(main):009:0> 5.0.class  
=> Float
```

Exercise: Arithmetic

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

Exercise: Strings

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

Exercise: Strings

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

Exercise: Strings

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

Exercise: Strings

```
irb(main):014:0> "\"so easy\""  
=> "\"so easy\""  
irb(main):015:0> puts "\"so easy\""  
"so easy"  
=> nil
```

Exercise: Strings

```
irb(main):016:0> x = "pretty"
=> "pretty"
irb(main):017:0> "#{x} nice"
=> "pretty nice"
irb(main):018:0> '#{x} nice'
=> "\#{x} nice"
```

Exercise: Truthiness

```
irb(main):019:0> true
=> true
irb(main):020:0> false
=> false
irb(main):021:0> nil
=> nil
irb(main):022:0> !!nil
=> false
irb(main):023:0> !!0
=> true
irb(main):024:0> !!x
=> true
```

Exercise: Operators

```
irb(main):025:0> 1 == 1
=> true
irb(main):026:0> 1 == true
=> false
irb(main):027:0> 1 != true
=> true
irb(main):028:0> !!1 == true
=> true
```

Exercise: Operators

```
irb(main):029:0> 2 < 1
=> false
irb(main):030:0> 2 > 1
=> true
irb(main):031:0> 4 >= 3
=> true
irb(main):032:0> 4 >= 4
=> true
irb(main):033:0> 4 <= 5
=> true
irb(main):034:0> 4 <= 3
=> false
```

Exercise: Operators

```
irb(main):035:0> 5 <=> 5  
=> 0  
irb(main):036:0> 5 <=> 6  
=> -1  
irb(main):037:0> 5 <=> 4  
=> 1
```

Exercise: Arrays

```
irb(main):038:0> x = [ "a", "b", "c" ]  
=> [ "a", "b", "c" ]  
irb(main):039:0> x[0]  
=> "a"  
irb(main):040:0> x.first  
=> "a"  
irb(main):041:0> x.last  
=> "c"
```

Exercise: Arrays

```
irb(main):042:0> x + [ "d" ]  
=> [ "a", "b", "c", "d" ]  
irb(main):043:0> x  
=> [ "a", "b", "c" ]  
irb(main):044:0> x = x + [ "d" ]  
=> [ "a", "b", "c", "d" ]  
irb(main):045:0> x  
=> [ "a", "b", "c", "d" ]
```

Exercise: Arrays

```
irb(main):046:0> x << "e"
=> [ "a", "b", "c", "d", "e" ]
irb(main):047:0> x
=> [ "a", "b", "c", "d", "e" ]
```

Exercise: Arrays

```
irb(main):048:0> x.map { |i| "the letter #{i}" }
=> ["the letter a", "the letter b", "the letter c",
    "the letter d", "the letter e"]
irb(main):049:0> x
=> [ "a", "b", "c", "d", "e"]
```

Exercise: Arrays

```
irb(main):050:0> x.map! { |i| "the letter #{i}" }
=> [ "the letter a", "the letter b", "the letter c",
  "the letter d", "the letter e"]
irb(main):051:0> x
=> [ "the letter a", "the letter b", "the letter c",
  "the letter d", "the letter e"]
```

Exercise: Hashes

```
irb(main):052:0> h = {  
irb(main):053:1* "first_name" => "Gary",  
irb(main):054:1* "last_name" => "Gygax"  
irb(main):055:1> }  
=> {"first_name"=>"Gary", "last_name"=>"Gygax"}
```

Exercise: Hashes

```
irb(main):056:0> h.keys
=> [ "first_name", "last_name" ]
irb(main):057:0> h["first_name"]
=> "Gary"
irb(main):058:0> h["age"] = 33
=> 33
irb(main):059:0> h.keys
=> [ "first_name", "last_name", "age" ]
```

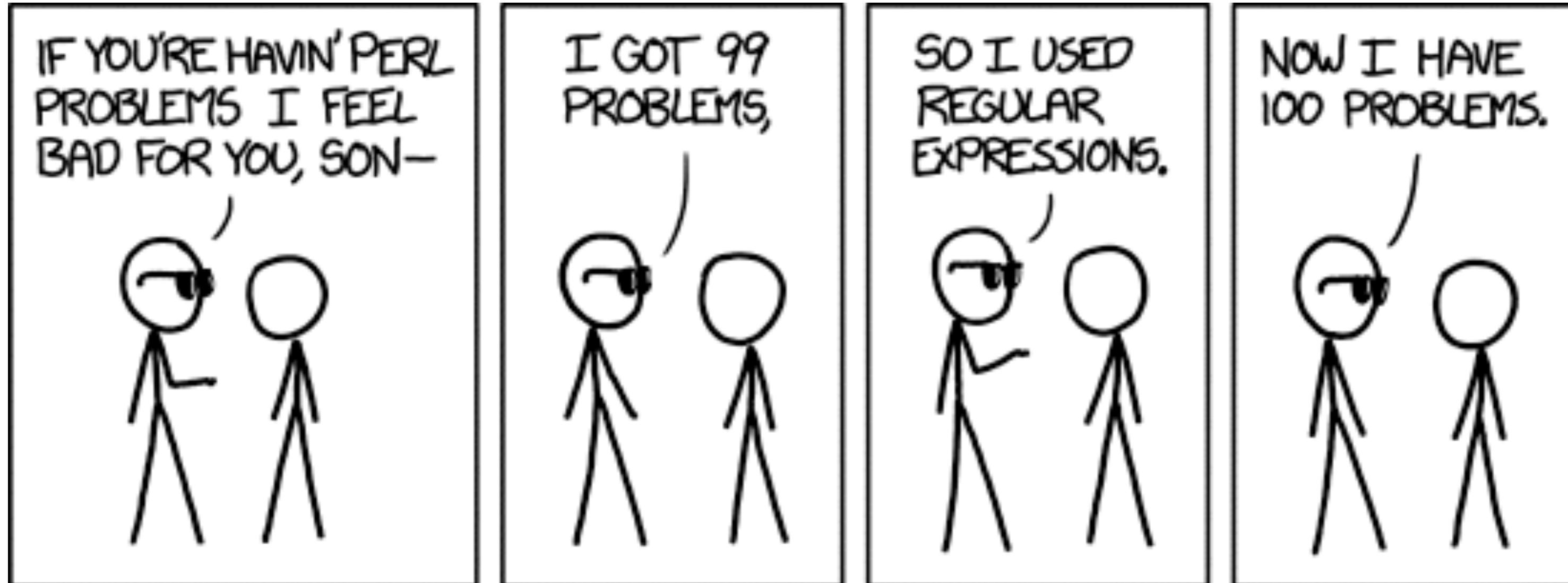
Exercise: Hashes

```
irb(main):060:0> h.values  
=> [ "Gary", "Gygax", 33 ]
```

Exercise: Hashes

```
irb(main):061:0> h.each { |k, v| puts "#{k}: #{v}" }
first_name: Gary
last_name: Gygax
age: 33
=> {"first_name"=>"Gary", "last_name"=>"Gygax",
"age"=>33}
```

We all love Regular Expressions



www.rubular.com

Exercise: Regular Expressions

```
irb(main):062:0> x = "I want to believe"
=> "I want to believe"
irb(main):063:0> x =~ /I/
=> 0
irb(main):064:0> x =~ /lie/
=> 12
irb(main):065:0> x =~ /smile/
=> nil
irb(main):066:0> x !~ /smile/
=> true
```

Exercise: Regular Expressions

```
irb(main):067:0> x.sub(/t/, "T")
=> "I wanT to believe"
irb(main):068:0> puts x
I want to believe
=> nil
irb(main):069:0> x.gsub!(/t/, "T")
=> "I wanT To believe"
irb(main):070:0> puts x
I wanT To believe
=> nil
```

Exercise: Conditionals

```
irb(main):071:0> x = "happy"
=> "happy"
irb(main):072:0> if x == "happy"
irb(main):073:1>   puts "Sure am!"
irb(main):074:1> elsif x == "sad"
irb(main):075:1>   puts "Boo!"
irb(main):076:1> else
irb(main):077:1*>   puts "Therapy?"
irb(main):078:1> end
Sure am!
=> nil
```

Exercise: Conditionals

```
irb(main):079:0> case x
irb(main):080:1> when "happy"
irb(main):081:1>   puts "Sure Am!"
irb(main):082:1>   1
irb(main):083:1> when "sad"
irb(main):085:1>   puts "Boo!"
irb(main):086:1>   2
irb(main):087:1> else
irb(main):088:1>   puts "Therapy?"
irb(main):089:1>   3
irb(main):090:1> end
Sure Am!
=> 1
```

Exercise: Method Definition

```
irb(main):091:0> def metal(str)
irb(main):092:1>   puts "!!#{str} is metal!!"
irb(main):093:1> end
=> nil
irb(main):094:0> metal("ozzy")
!!ozzy is metal!!
=> nil
```

Exercise: Classes

```
irb(main):095:0> class Person
irb(main):096:1>   attr_accessor :name, :is_metal
irb(main):097:1>
irb(main):098:1>   def metal
irb(main):099:2>     if @is_metal
irb(main):100:3>       puts "!!#{@name} is metal!!"
irb(main):101:3>   end
irb(main):102:2> end
irb(main):103:1> end
=> nil
```

Exercise: Classes

```
irb(main):104:0> p = Person.new
=> #<Person:0x891ab4c>
irb(main):105:0> p.name = "Adam Jacob"
=> "Adam Jacob"
irb(main):106:0> p.is_metal = true
=> true
irb(main):107:0> p.metal
!!Adam Jacob is metal!!
=> nil
irb(main):108:0> p.is_metal = false
=> false
irb(main):109:0> p.metal
=> nil
```

Questions

- What is irb?
- What is true in ruby? What is false?
- How do you press for the truth?
- What does `>=` do?
- How do you define a method?
- What is a class?
- What is an object?
- The book you want: “Programming Ruby 1.9” <http://pragprog.com/book/ruby3/programming-ruby-1-9>