Chef Fundamentals

training@opscode.com

Introductions

OPSCODE

RULE THE CLOUD

Chef

Instructor Introduction

Introduce yourselves

Course Objectives and Style

OPSCODE
RULE THE CLOUD

Chef

Course Objectives

- Write cookbooks to automate common infrastructure tasks
- Understand Chef's architecture
- Be familiar with Chef's various tools
- Understand how to 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

This is Chef Bootcamp

- 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 - if everyone gets it already, we'll skip it!
- Ask questions when they come to you
- Ask for help when you need it

Agenda

OPSCODE

RULE THE CLOUD

Chef

OPSCODE

Topics

- Overview of Chef
- Workstation Setup
- Test Node Setup
- Dissecting your first Chef run
- Introducing the Node object
- Writing an Apache cookbook
- Writing an MOTD cookbook
- Refactoring the Apache Cookbook
- Writing a Users Cookbook

RULE THE CLOUD

Topics

- Roles
- Environments
- Using community cookbooks
- If we have time
  - Just enough Ruby for Chef
  - Building LWRPs
  - Shef
  - Working a use case

Breaks!

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

Overview of Chef

What is this thing again?

OPSCODE

Objectives

- Understand how Chef thinks about Infrastructure Automation
- Understand the following concepts:
  - Node
  - Resource
  - Recipe
  - Cookbook
  - Run List
  - Roles
  - Search

RULE THE CLOUD

Evolving towards Configuration Management

- Just build it
- Keep notes in server.txt
- Move notes to the wiki
- Custom scripts (in scm?!)
- Snapshot & Clone

OPSCODE

Applications

http://www.flickr.com/photos/steffenz/337700069/
http://www.flickr.com/photos/kky/704056791/

OPSCODE

Infrastructure

RULE THE CLO

http://www.flickr.com/photos/sbh/462754460/

Collection of Resources

- Nodes
- Networking
- Files
- Directories
- Symlinks
- Mounts

- Routes
- Users
- Groups
- Tasks
- Packages
- Software
- Services
- Configurations
- Stuff

http://www.flickr.com/photos/stevekeys/3123167585/

OPSCODE

Acting in concert

RULE THE

http://www.flickr.com/photos/glowjangles/4081048126/

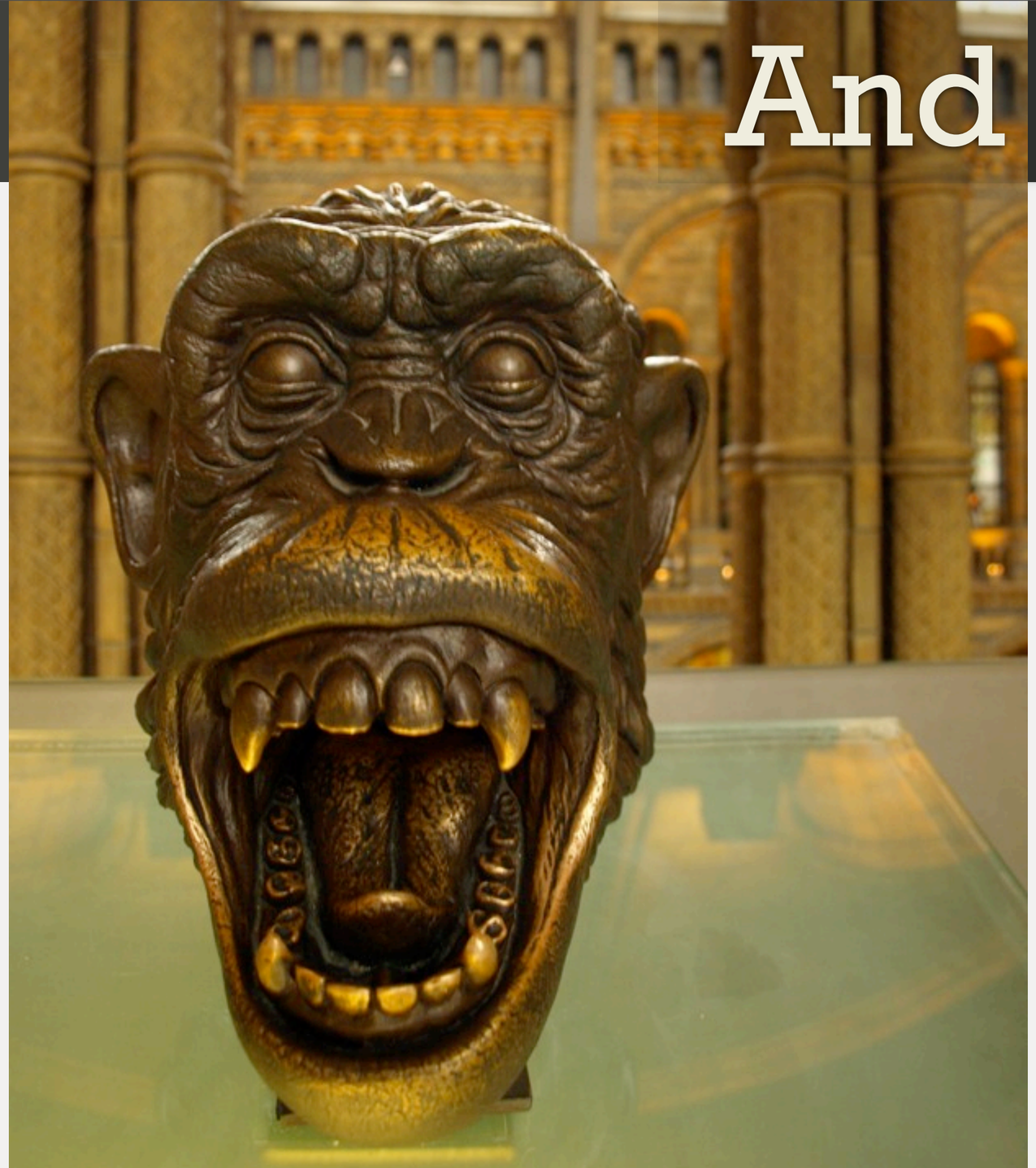
OPSCODE

To provide a Service

http://www.flickr.com/photos/28309157@N08/3743455858/

OPSCODE

And it *evolves*



http://www.flickr.com/photos/16339684@N00/2681435235/

OPSCODE

See Node

Application

RULE THE CLOUD

OPSCODE

See Nodes

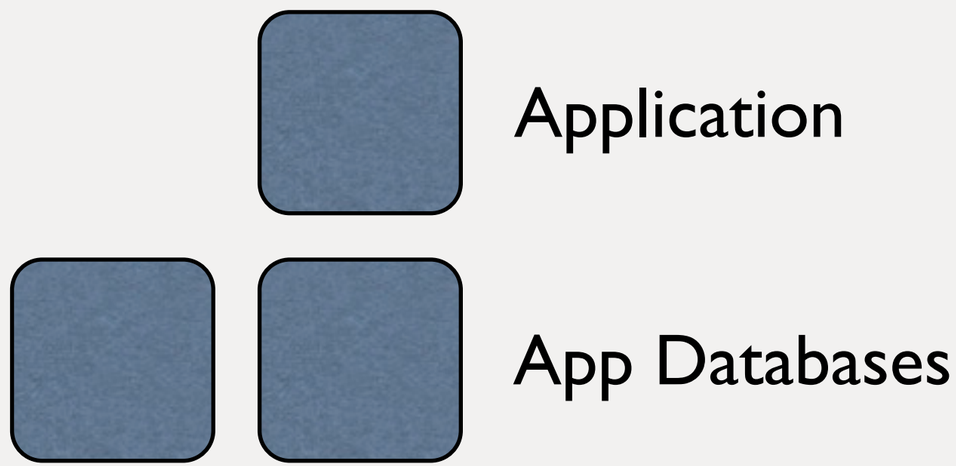
Application

Application Database

RULE THE CLOUD

OPSCODE

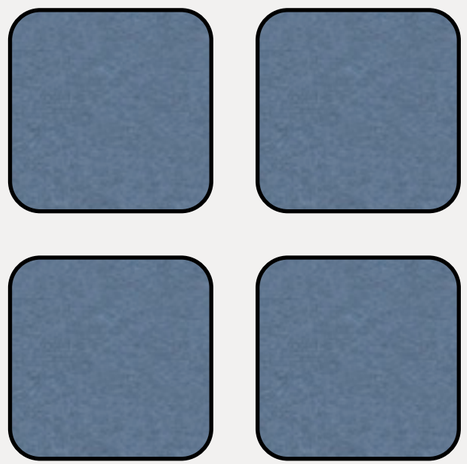
See Nodes Grow



RULE THE CLOUD

OPSCODE

See Nodes Grow



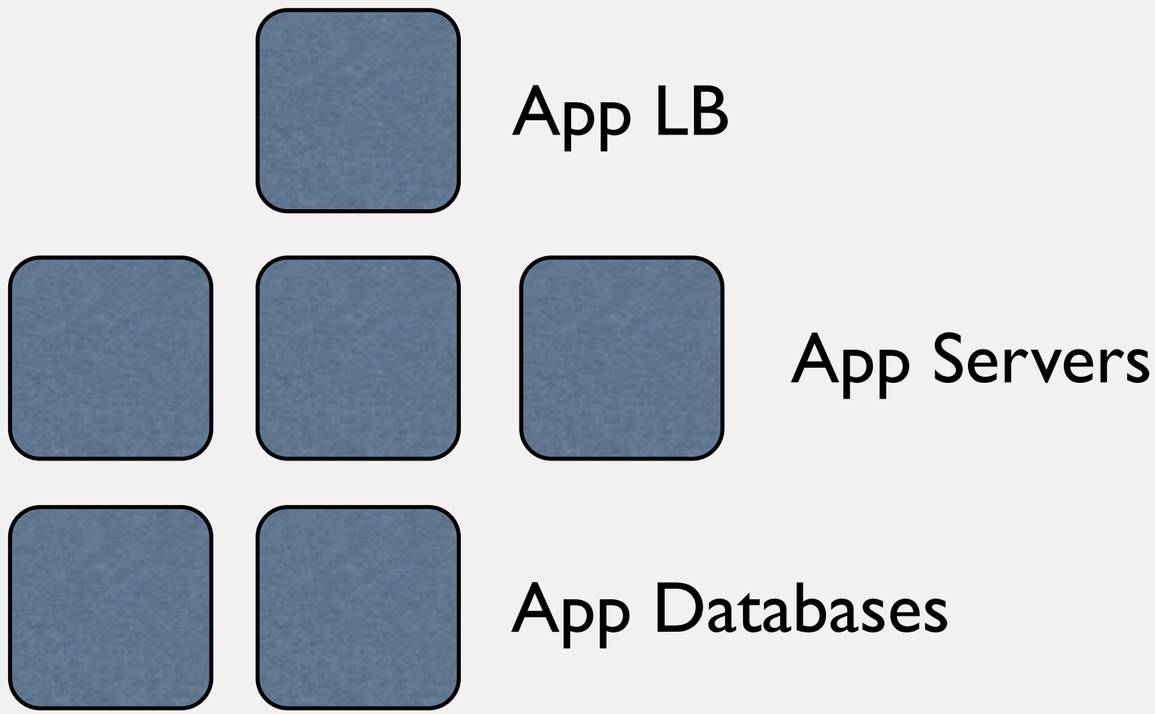
App Servers

App Databases

RULE THE CLOUD

OPSCODE

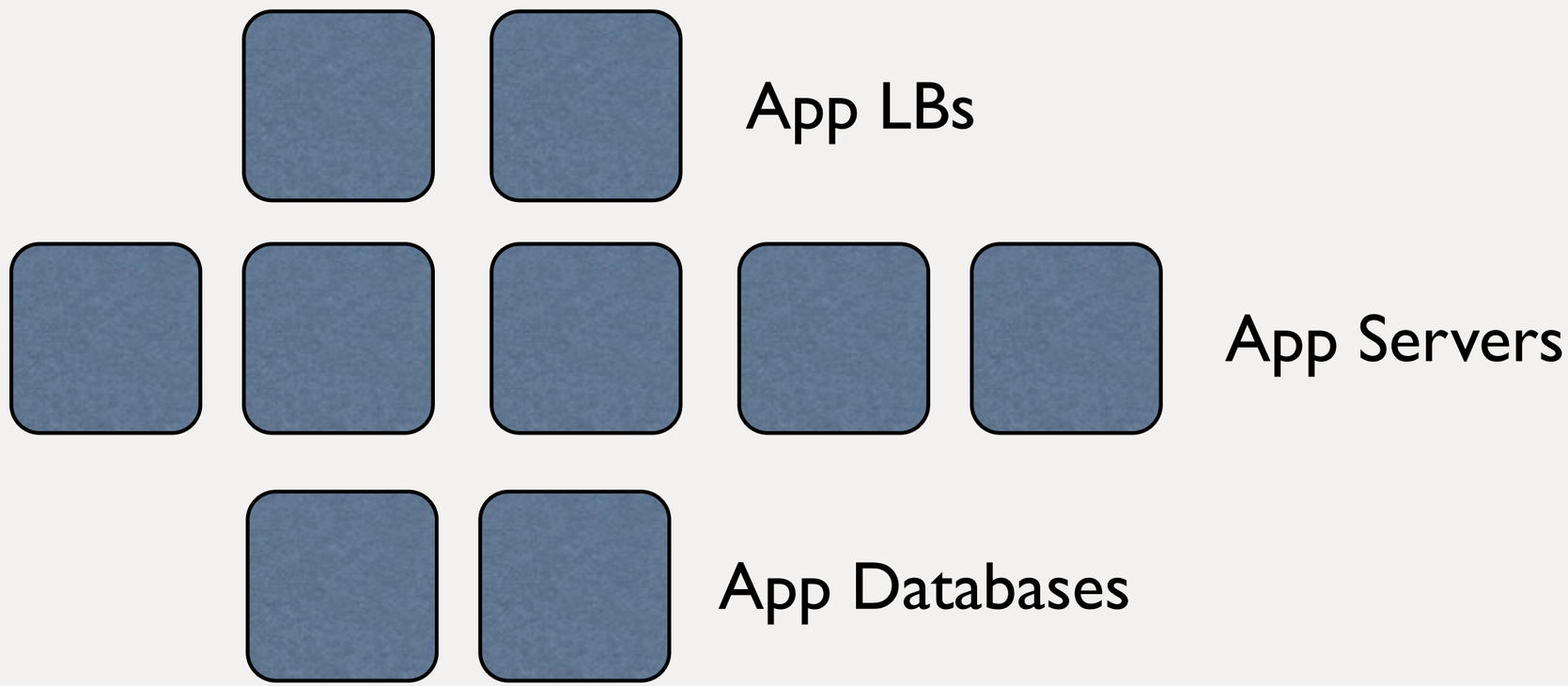
See Nodes Grow



RULE THE CLOUD

OPSCODE

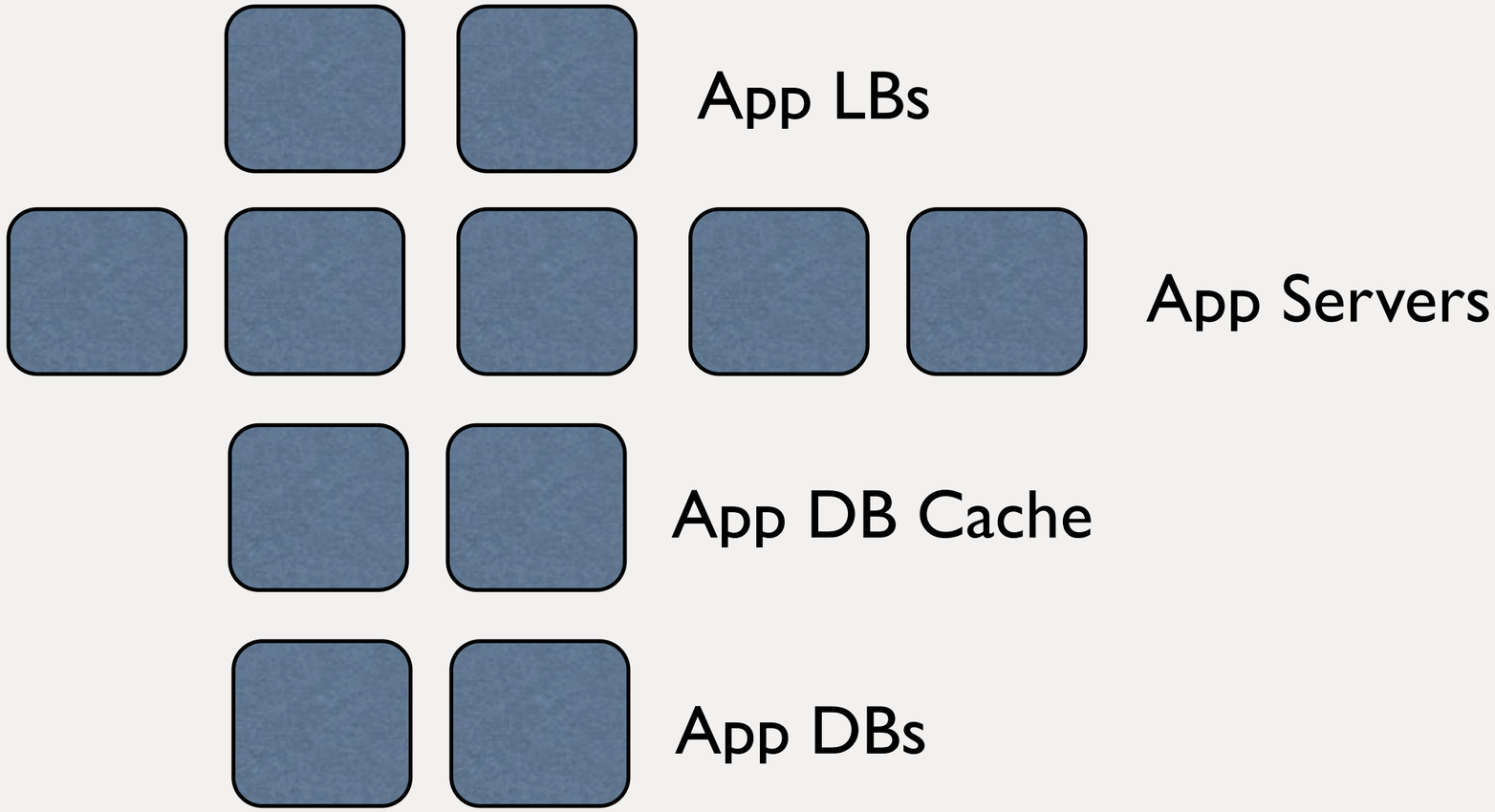
See Nodes Grow



RULE THE CLOUD

OPSCODE

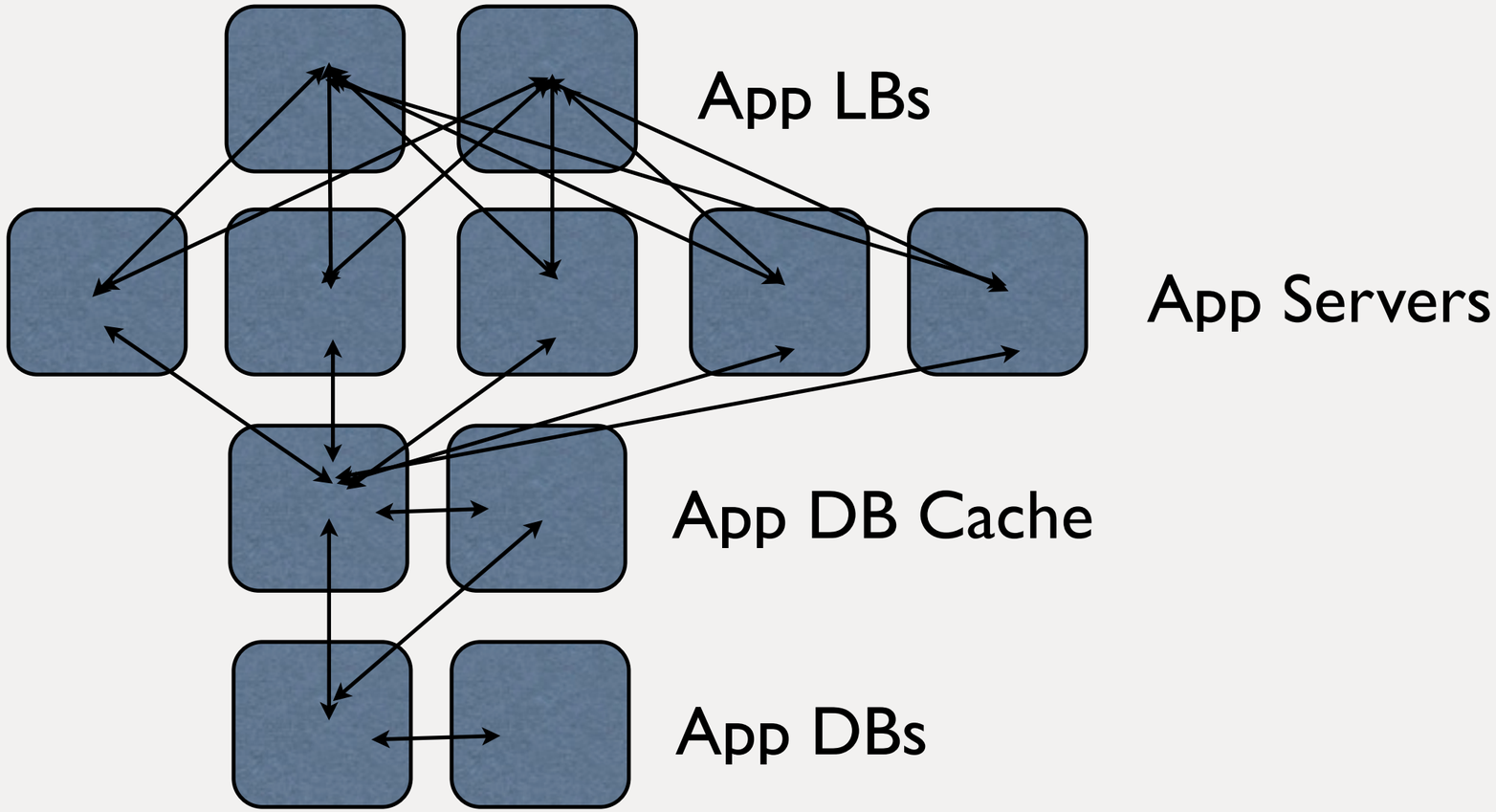
See Nodes Grow



RULE THE CLOUD

OPSCODE

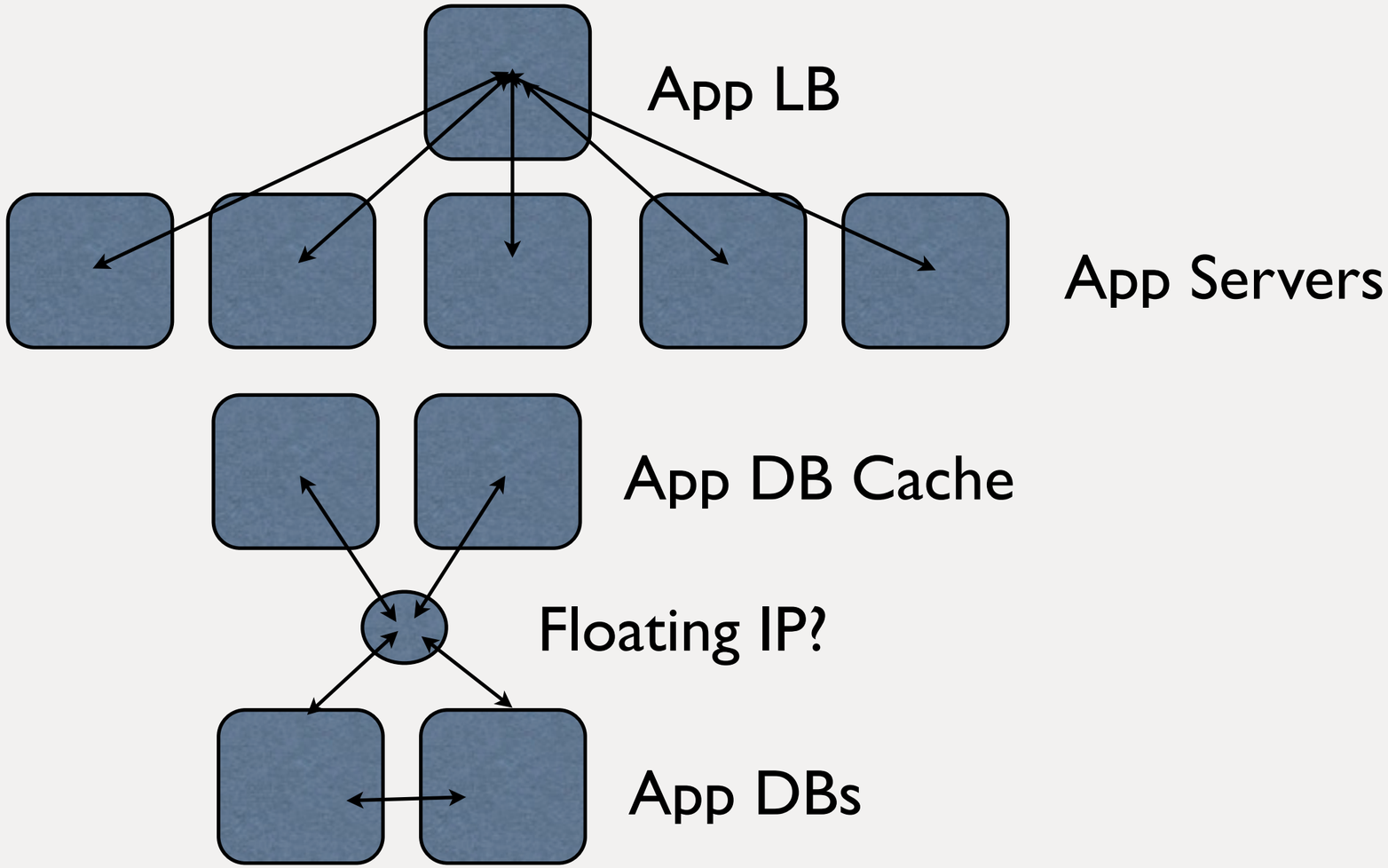
Stitched together with configs



RULE THE CLOUD

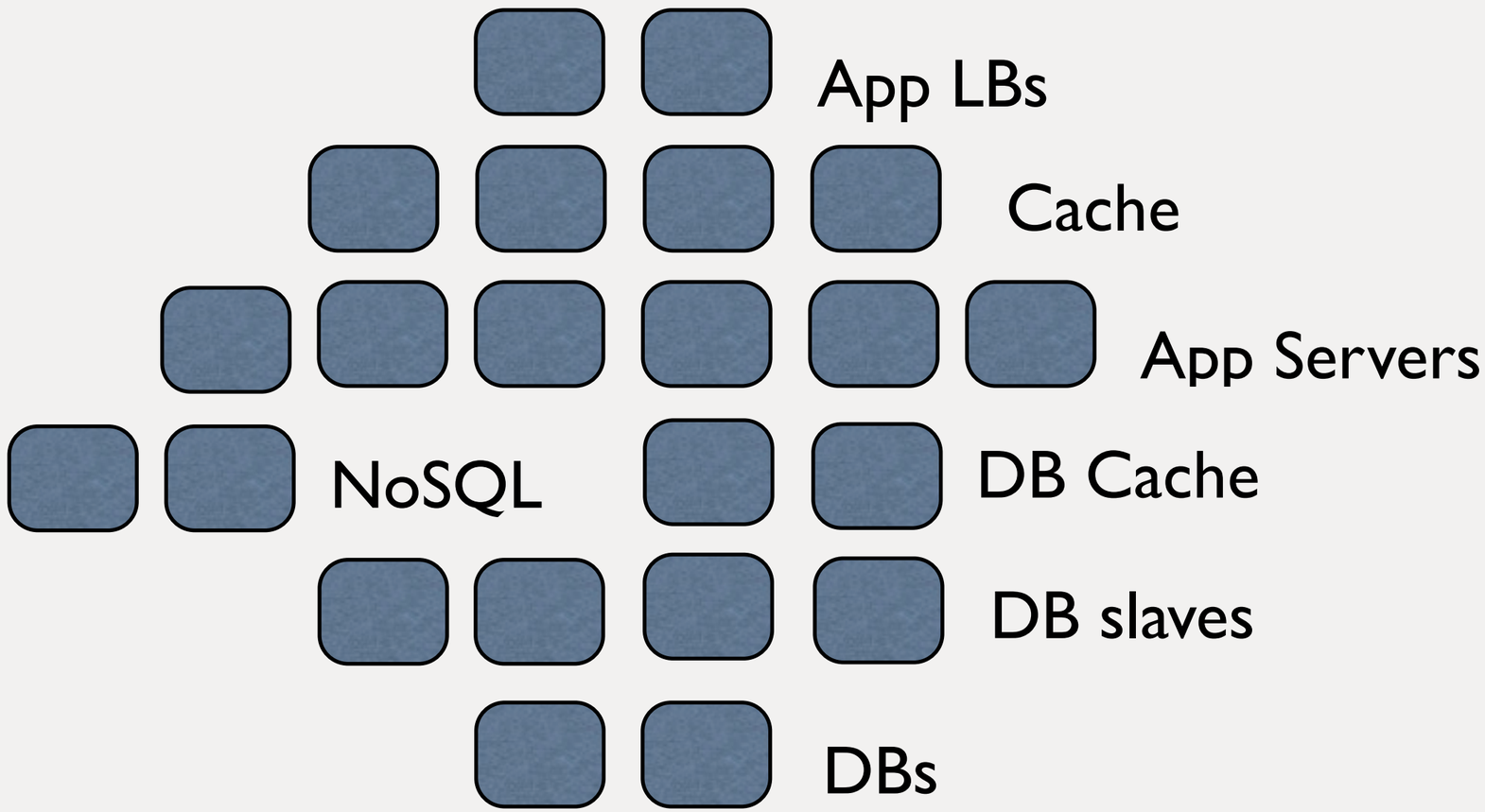
OPSCODE

Your Infrastructure is a snow flake

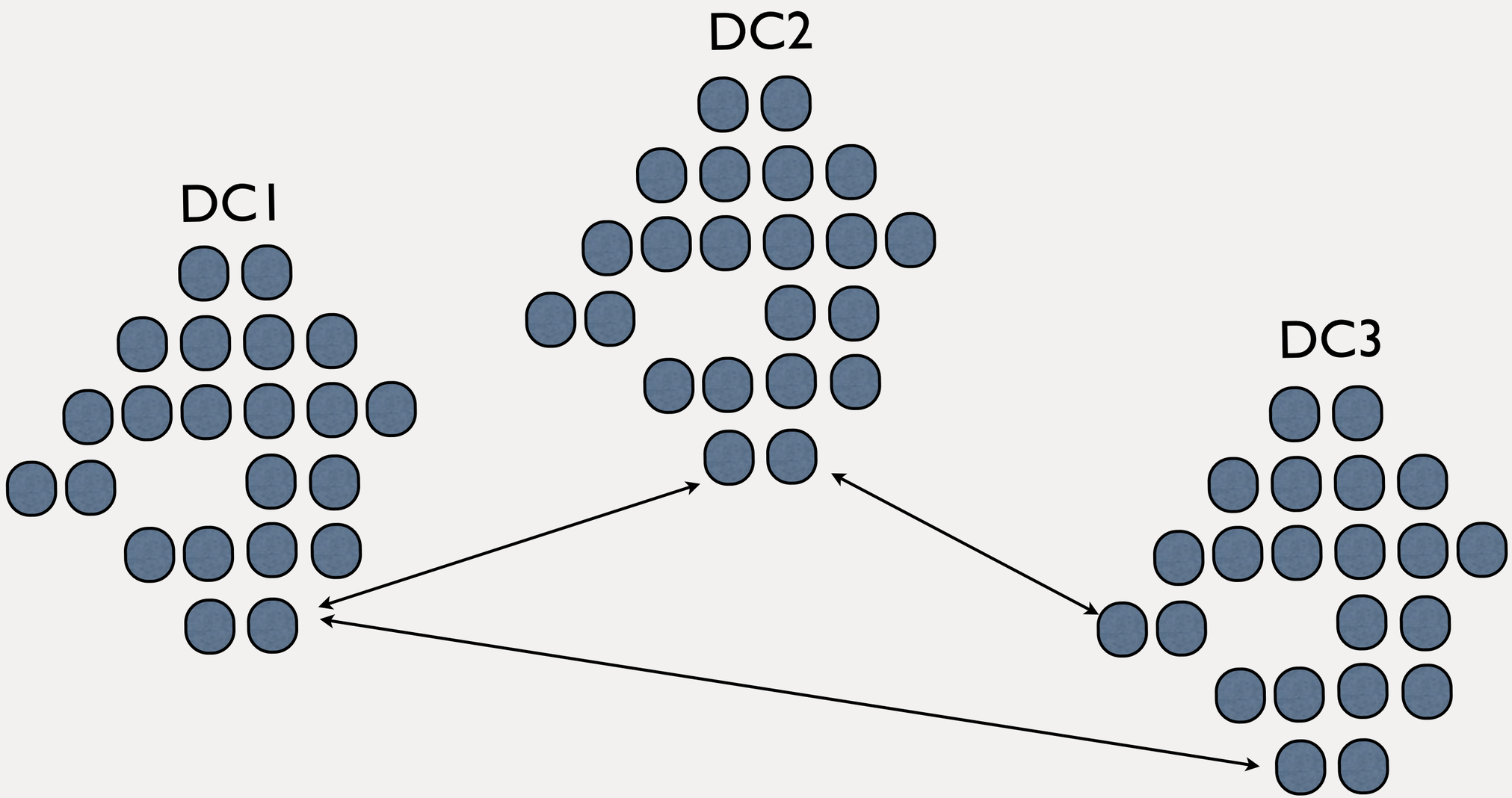


RULE THE CLOUD

Complexity increases quickly



Complexity increases very quickly



Configuration Management

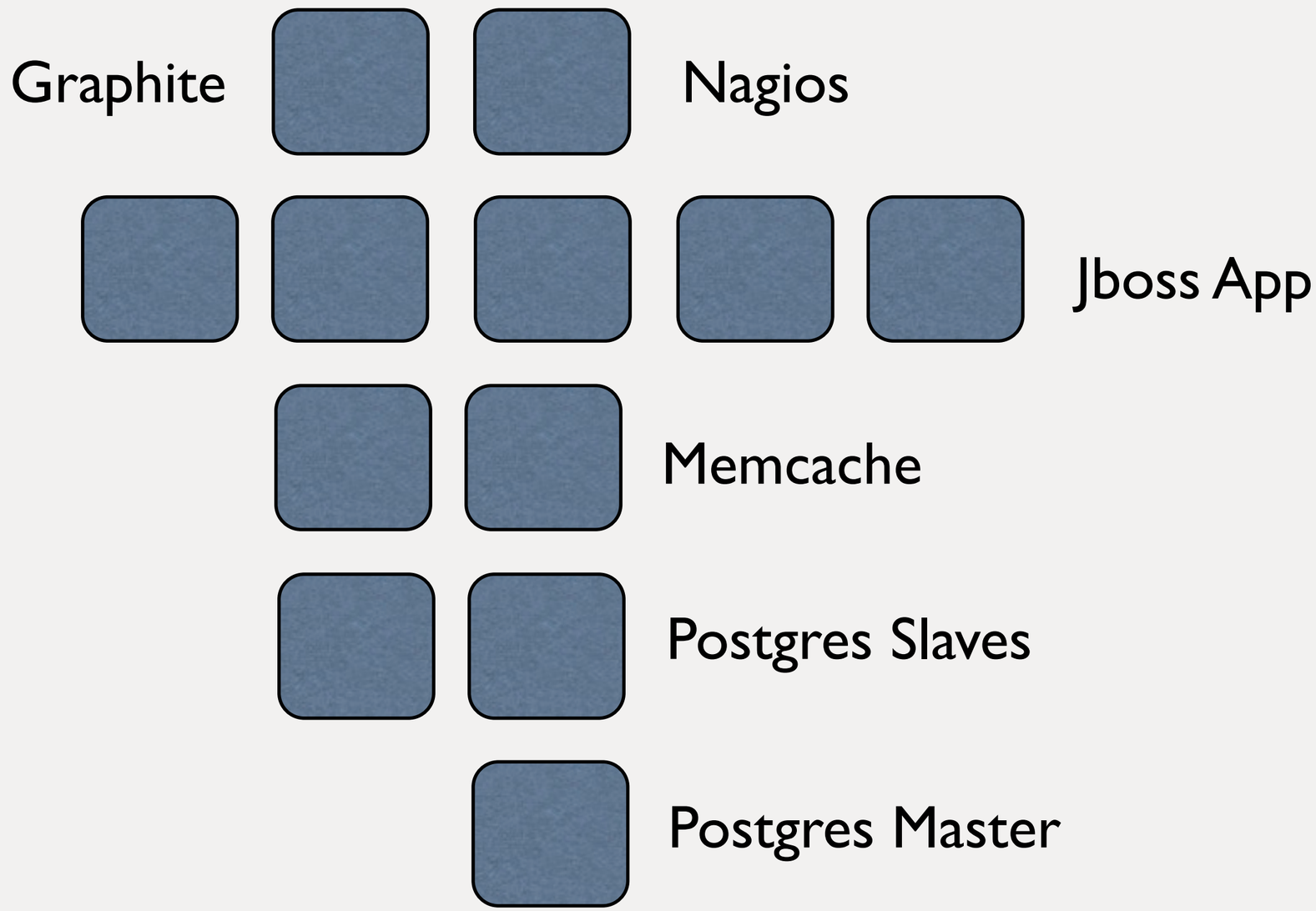
http://www.flickr.com/photos/philliecasablanca/3354734116/

Golden Images are not the answer

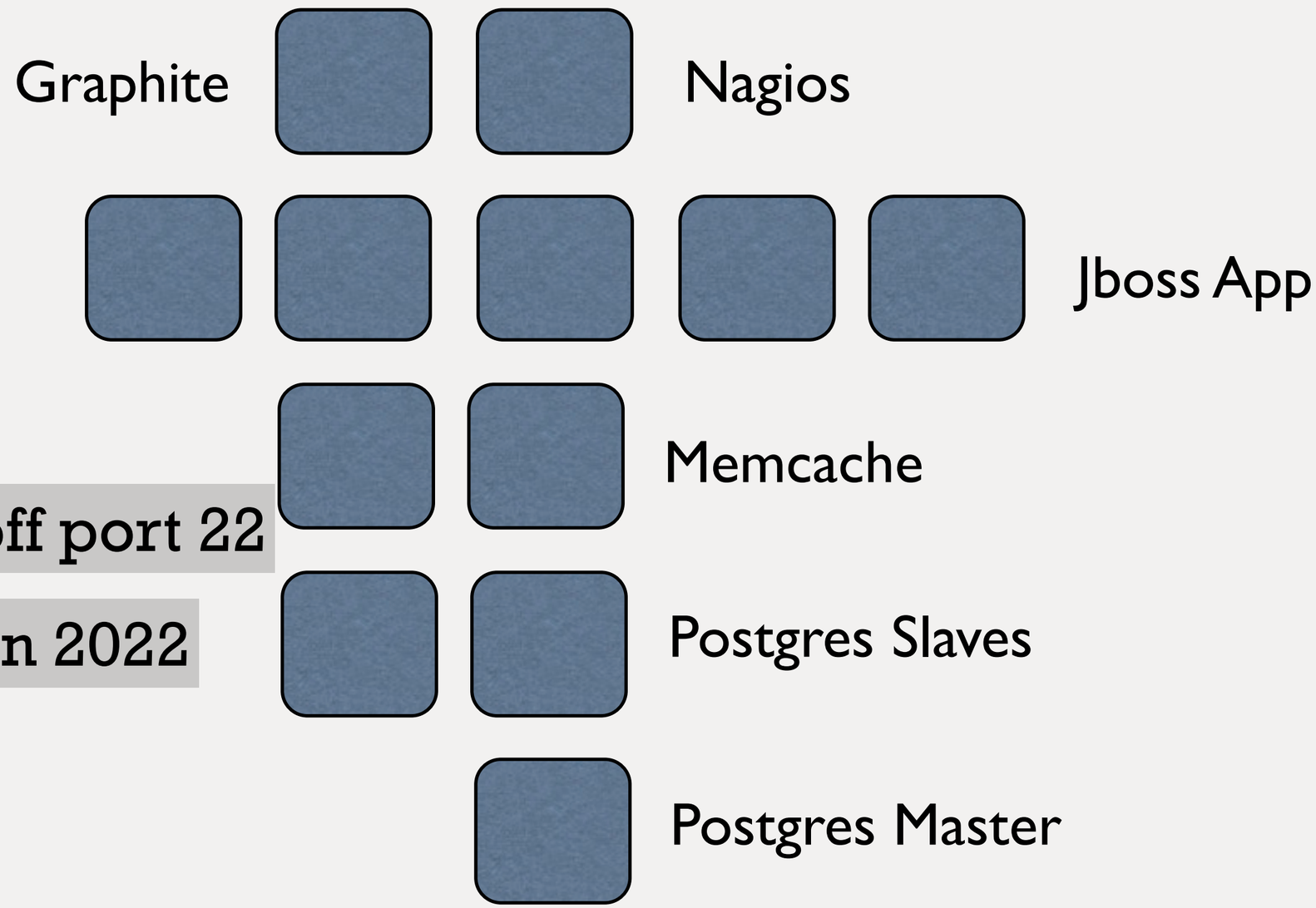
- Gold is heavy
- Hard to transport
- Hard to mold
- Easy to lose configuration detail

http://www.flickr.com/photos/garysoup/2977173063/

Typical Boring Infrastructure

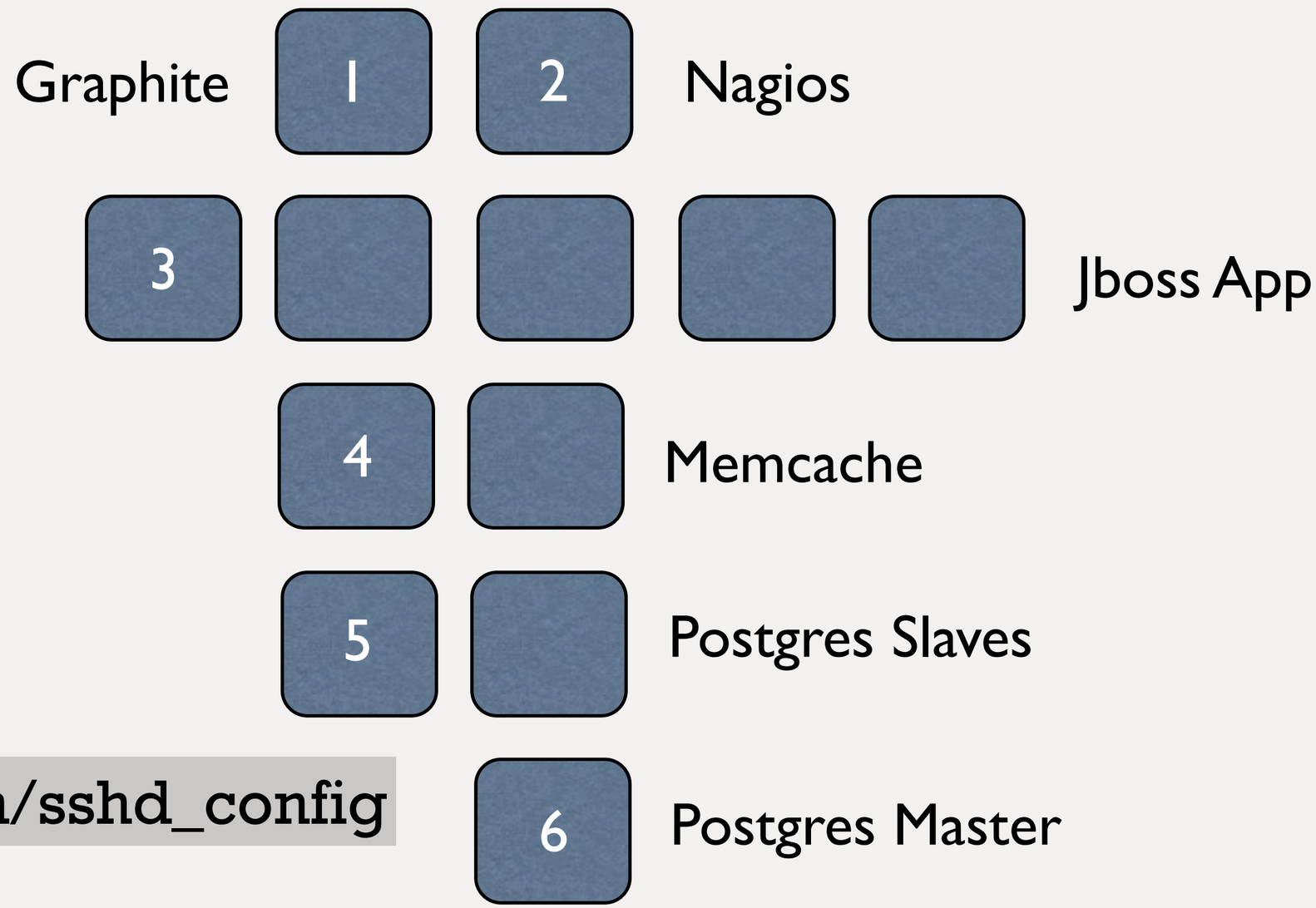


New Compliance Mandate



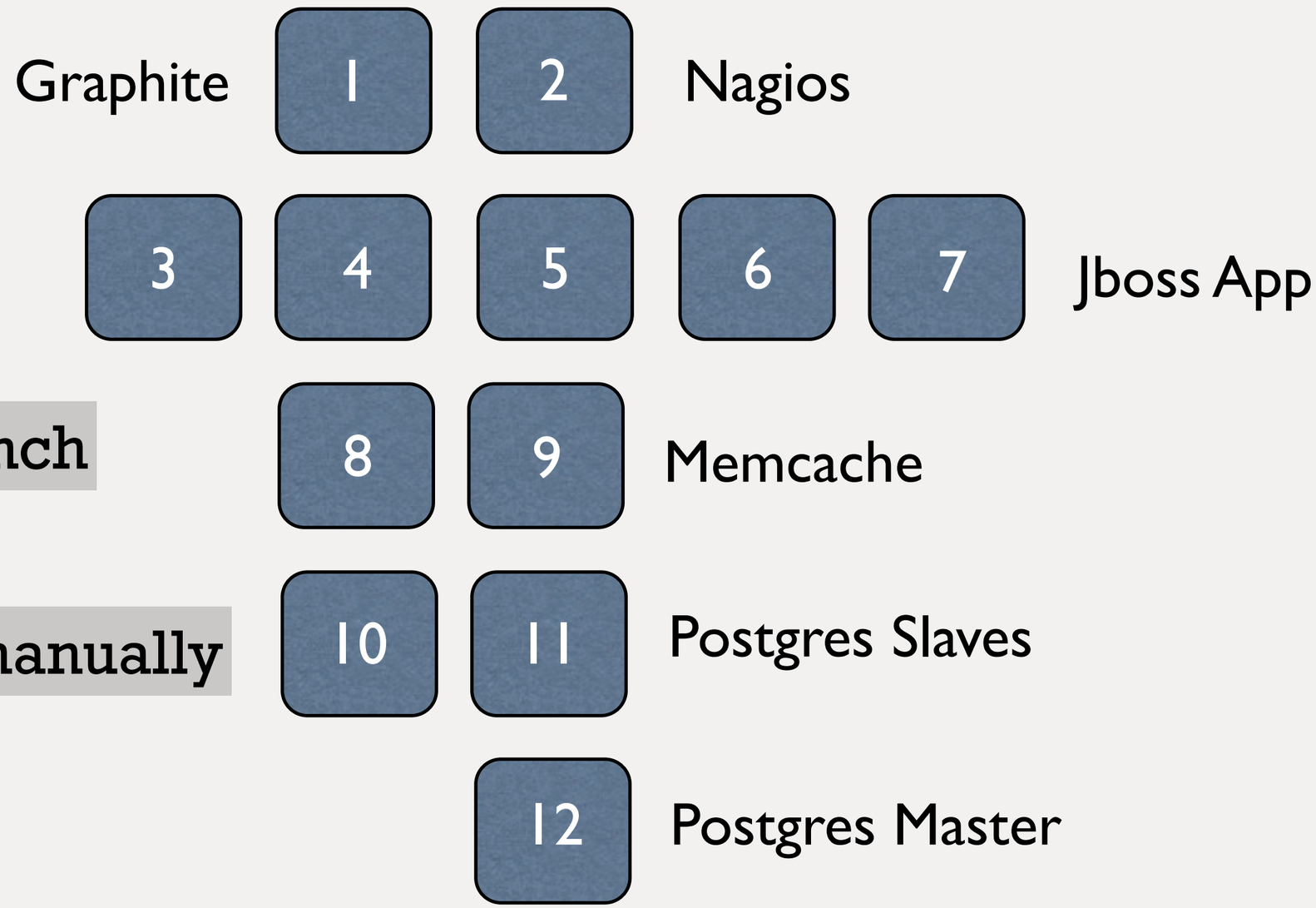
- Move SSH off port 22
- Lets put it on 2022

6 Golden Image Updates



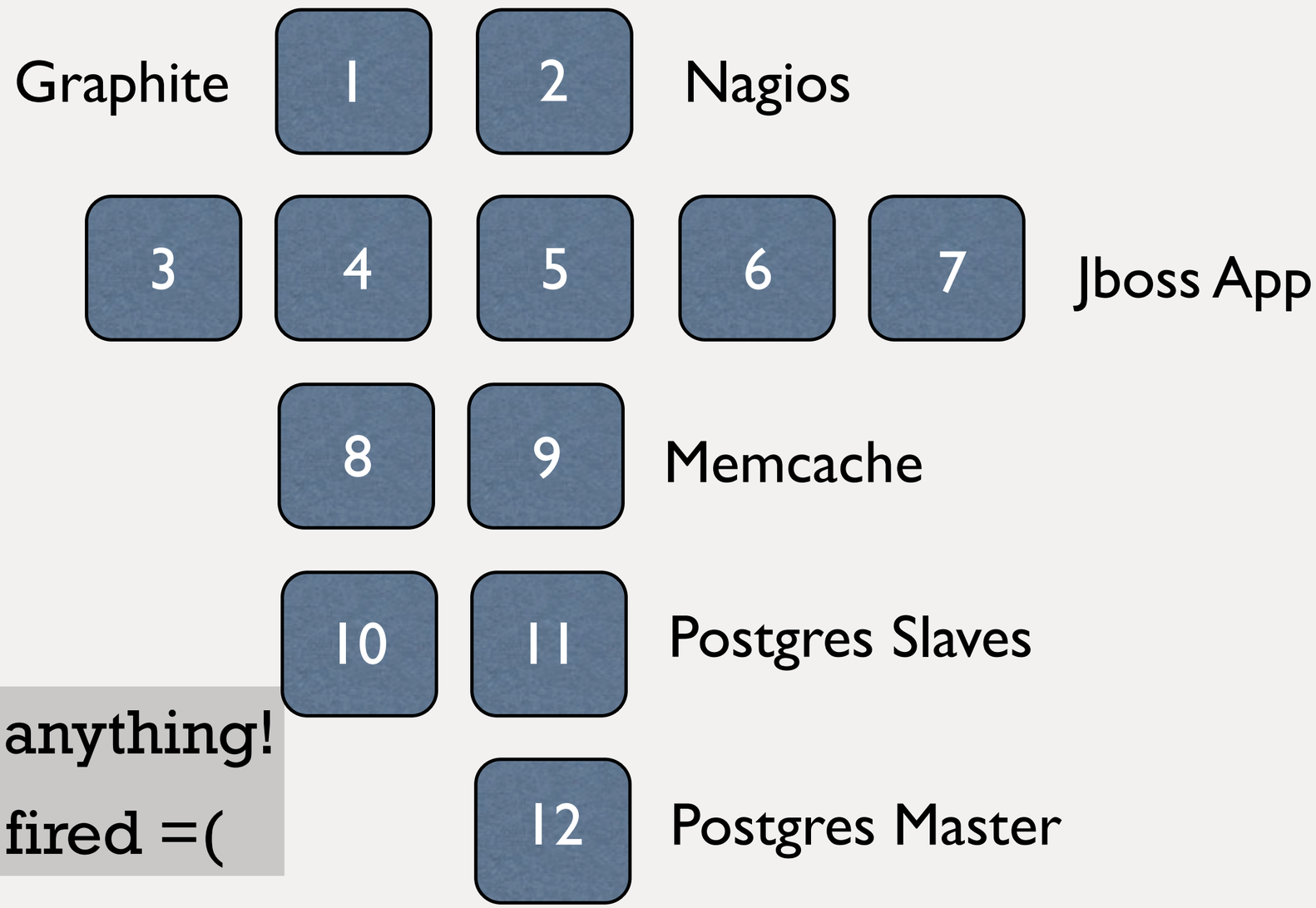
- edit /etc/ssh/sshd_config

12 Instance Replacements



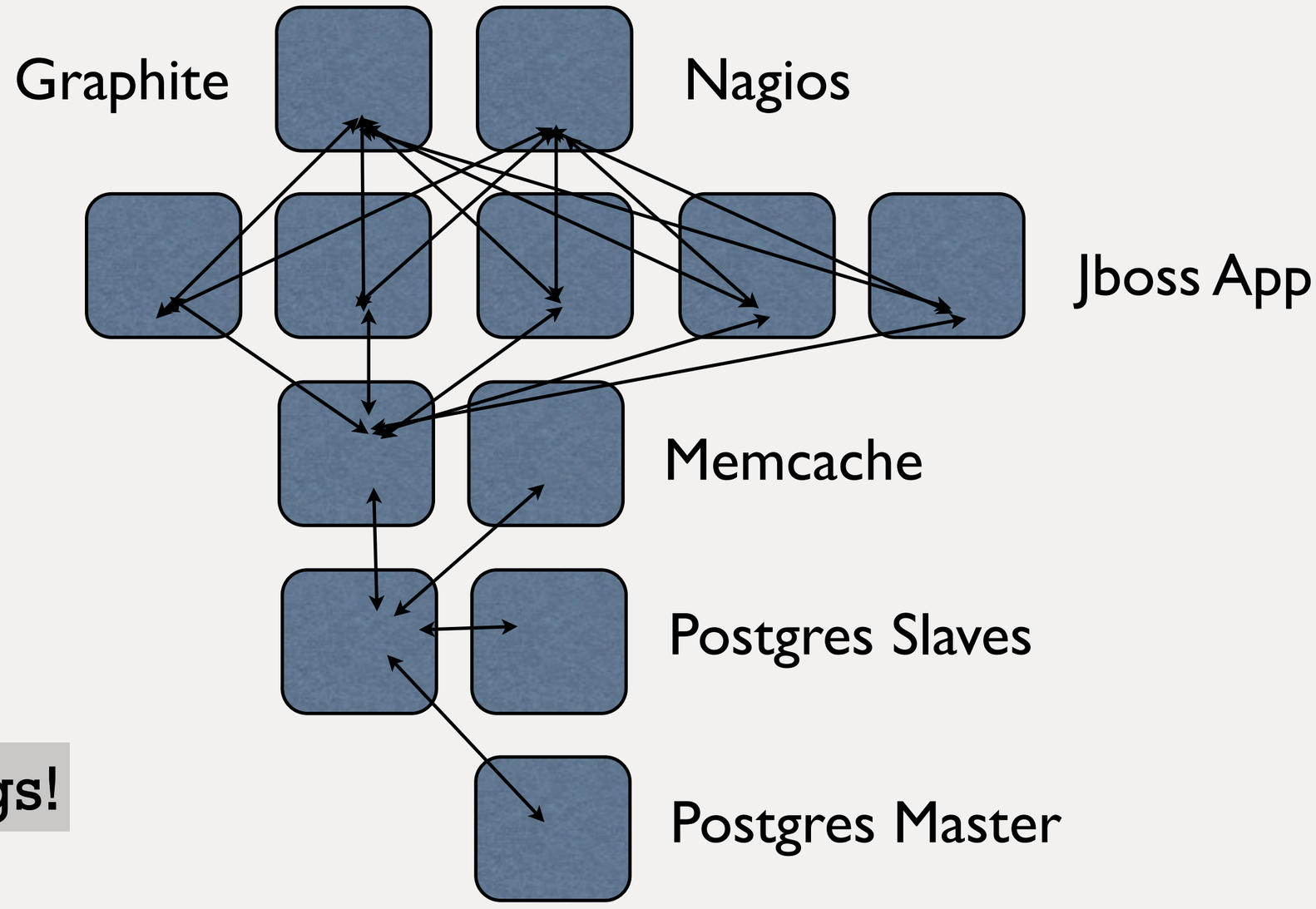
- Delete, launch
- Repeat
- Typically manually

Done in Maintenance Windows



- Don't break anything!
- Bob just got fired =(

Different IP Addresses?



- Invalid configs!

OPSCODE

Configuration Desperation

http://www.flickr.com/photos/francoforeshock/5716969942/

RULE THE

Chef Solves This Problem

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

OPSCODE

Programs!

- Generate configurations directly on nodes
- Reduce management complexity
- Version control the programs

http://www.flickr.com/photos/ssoosay/5126146763/

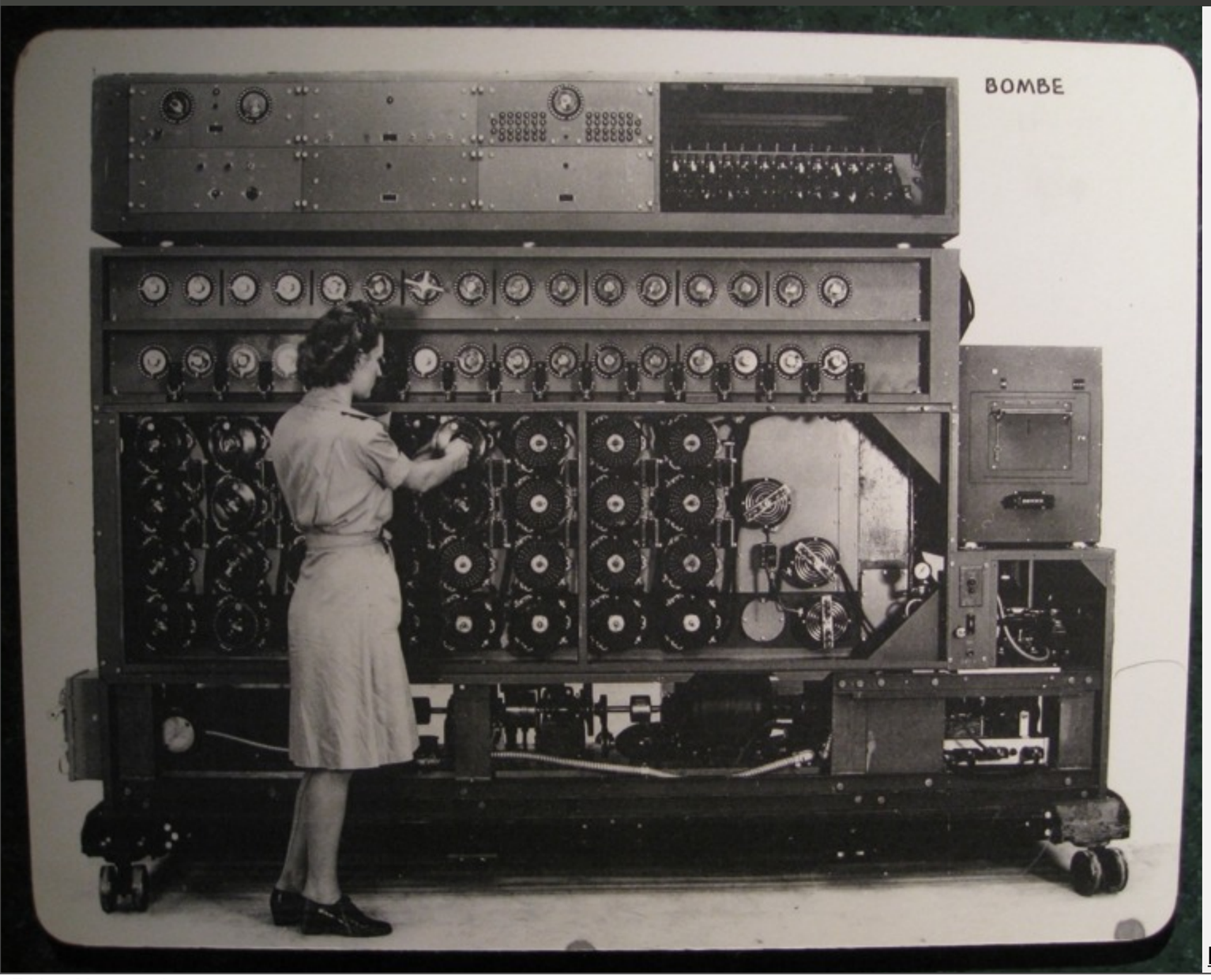
Declarative Interface to Resources

- Define policy
- Say what, not how
- Pull not Push

http://www.flickr.com/photos/bixentro/2591838509/

OPSCODE

Chef is Infrastructure as Code



- Programmatically provision and configure
- Treat like any other code base
- Reconstruct business from code repository, data backup, and bare metal resources.

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

OPSCODE

That looks like this

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

```
template "/etc/ntpd.conf" do
  source "ntpd.conf.erb"
  owner "root"
  group "root"
  mode 0644
  action :create
  variables(:time_server => “time.example.com”)
  notifies :restart, “service[ntpd]”
end
```

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

OPSCODE

Or this

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

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

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

Ohai!


```
"hostname": "server-1",
"fqdn": "server-1.example.com",
"domain": "example.com",
"network": {
  "interfaces": {
    "eth0": {
      "type": "eth",
      "number": "0",
      "encapsulation": "Ethernet",
      "addresses": {
        "00:0C:29:43:26:C5": {
          "family": "lladdr"
        },
        "192.168.177.138": {
          "family": "inet",
          "broadcast": "192.168.177.255",
          "netmask": "255.255.255.0"
        },
        "fe80::20c:29ff:fe43:26c5": {
          "family": "inet6",
          "prefixlen": "64",
          "scope": "Link"
        }
      },
```

```
"memory": {
  "swap": {
    "cached": "0kB",
    "total": "4128760kB",
    "free": "4128760kB"
  },
  "total": "2055676kB",
  "free": "1646524kB",
  "buffers": "35032kB",
  "cached": "210276kB",
  "active": "125336kB",
  "inactive": "142884kB",
  "dirty": "8kB",
  "writeback": "0kB",
  "anon_pages": "22976kB",
  "mapped": "8416kB",
  "slab": "121512kB",
  "slab_reclaimable": "41148kB",
  "slab_unreclaim": "80364kB",
  "page_tables": "1784kB",
  "nfs_unstable": "0kB",
  "bounce": "0kB",
  "commit_limit": "5156596kB",
  "committed_as": "74980kB",
  "vmalloc_total": "34359738367kB",
  "vmalloc_used": "274512kB",
  "vmalloc_chunk": "34359449936kB"
},
```

```
"block_device": {
  "ram0": {
    "size": "32768",
    "removable": "0"
  },
  "ram1": {
    "size": "32768",
    "removable": "0"
  },
  "ram2": {
    "size": "32768",
    "removable": "0"
  },
```

OPSCODE

Decide what to declare

```
execute "load sysctl" do
  command "/sbin/sysctl -p"
  action :nothing
end

bytes = node['memory']['total'].split("kB")[0].to_i * 1024 / 3,
pages = node['memory']['total'].split("kB")[0].to_i * 1024 / 3 / 2048

# adjust shared memory and semaphores
template "/etc/sysctl.conf" do
  source "sysctl.conf.erb"
  variables(
    :shmmax_in_bytes => bytes,
    :shmall_in_pages => pages
  )
  notifies :run, "execute[load sysctl]", :immediately
end
```

Multiphase Execution

```
size = ((2 * 3) * 4) / 2

99.downto(1) do |i|
  beer_bottle "bottle-#{i}" do
    oz size
    action [ :take_down, :pass_around ]
  end
end
```

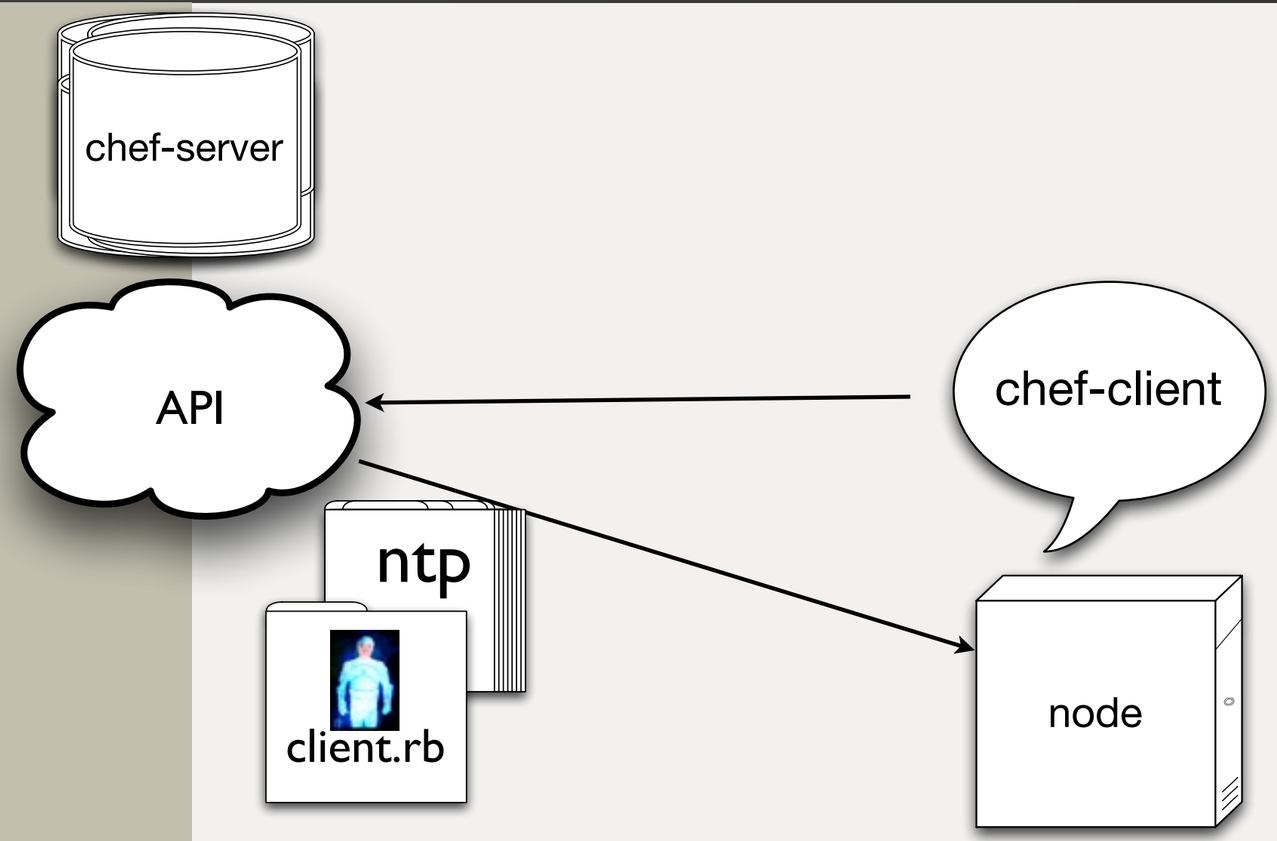
OPSCODE

Recipes and Cookbooks

- Recipes are collections of Resources
- Cookbooks contain recipes, templates, files, custom resources, etc
- Code re-use and modularity

http://www.flickr.com/photos/shutterhacks/4474421855/

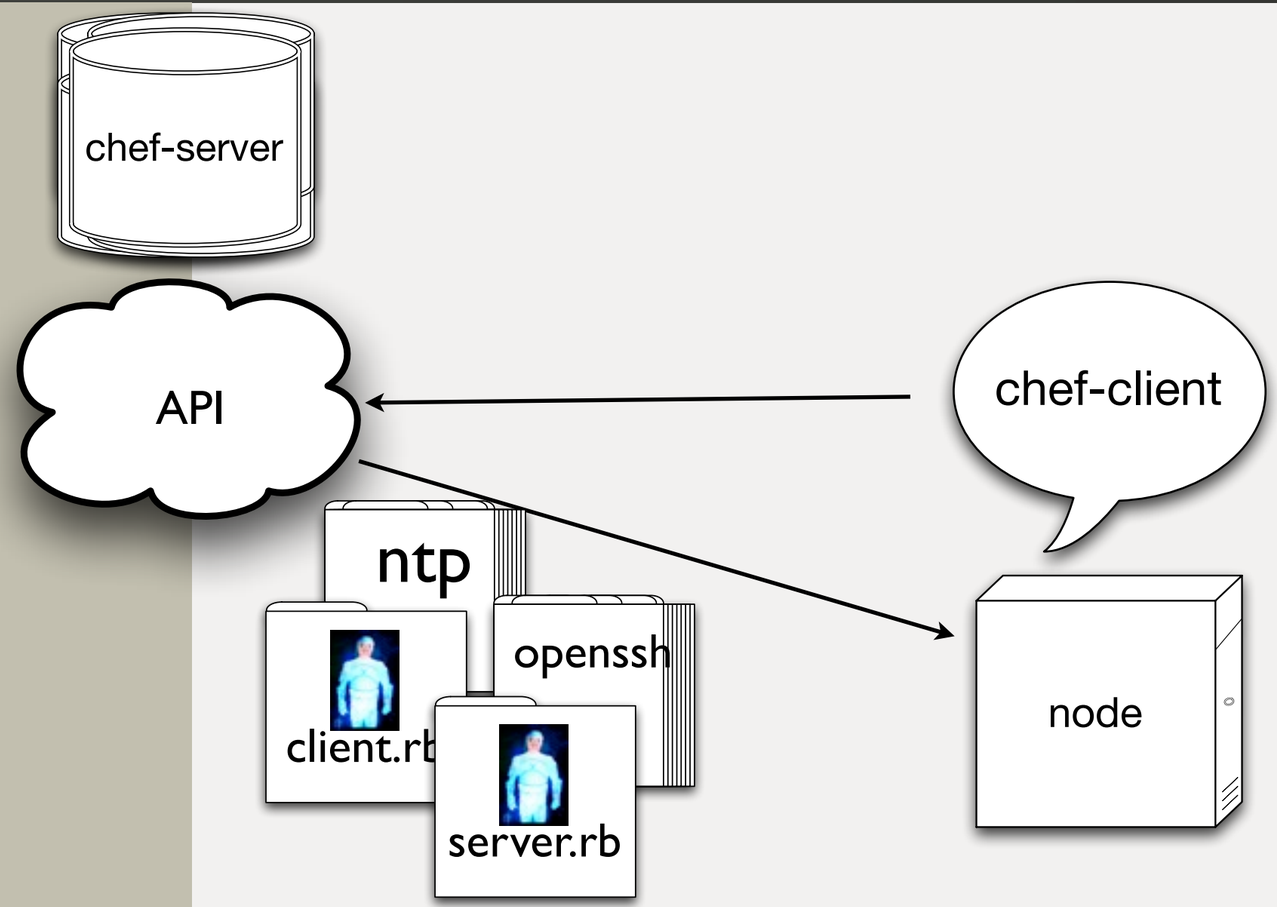
Run Lists



recipe[ntp::client]

OPSCODE

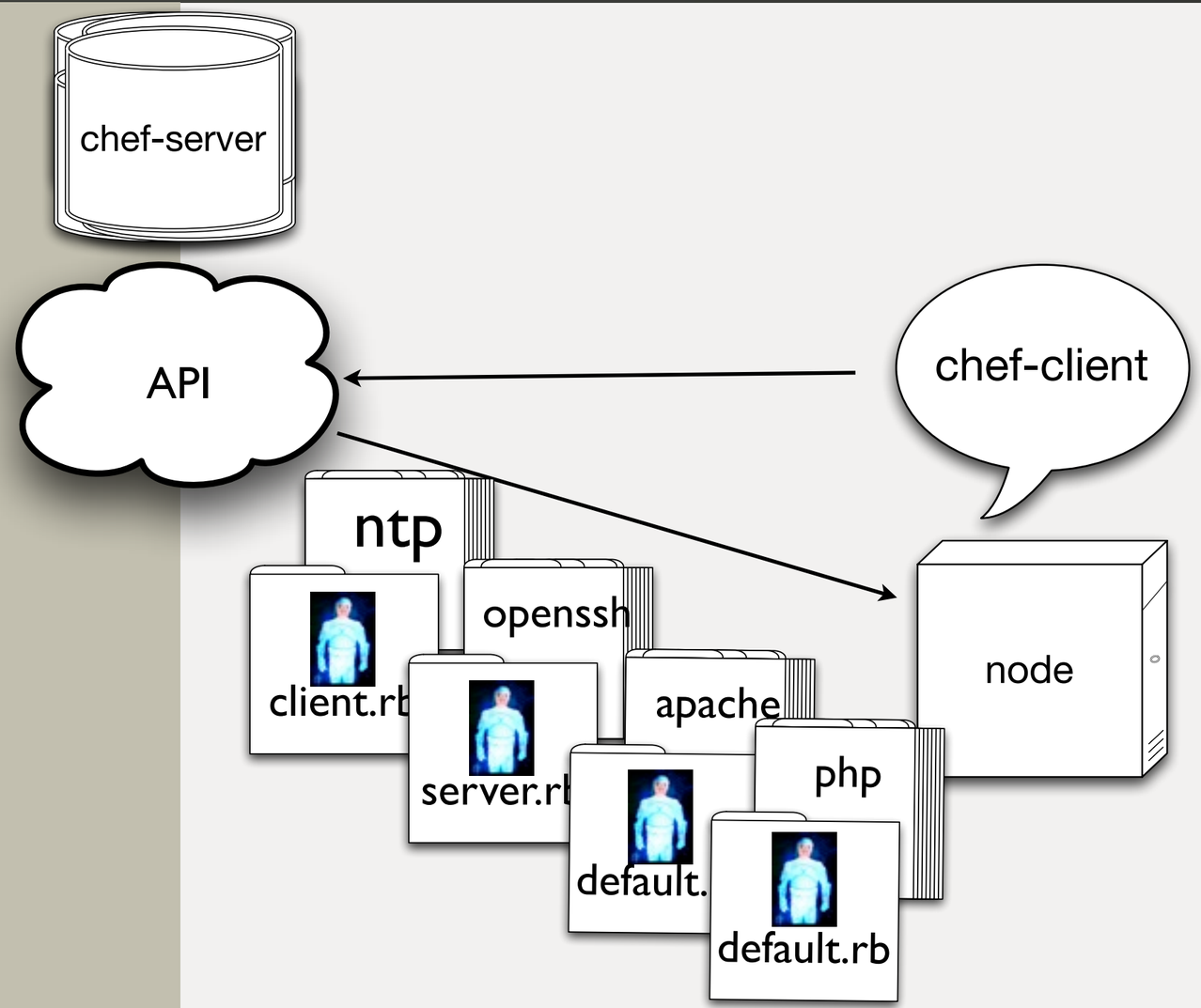
Run Lists



"recipe[ntp::client]",
"recipe[openssh::server]"

RULE THE CLOUD

Run Lists



"recipe[ntp::client]",
"recipe[openssh::server]",
"recipe[apache]",
"recipe[php]"

OPSCODE

Roles

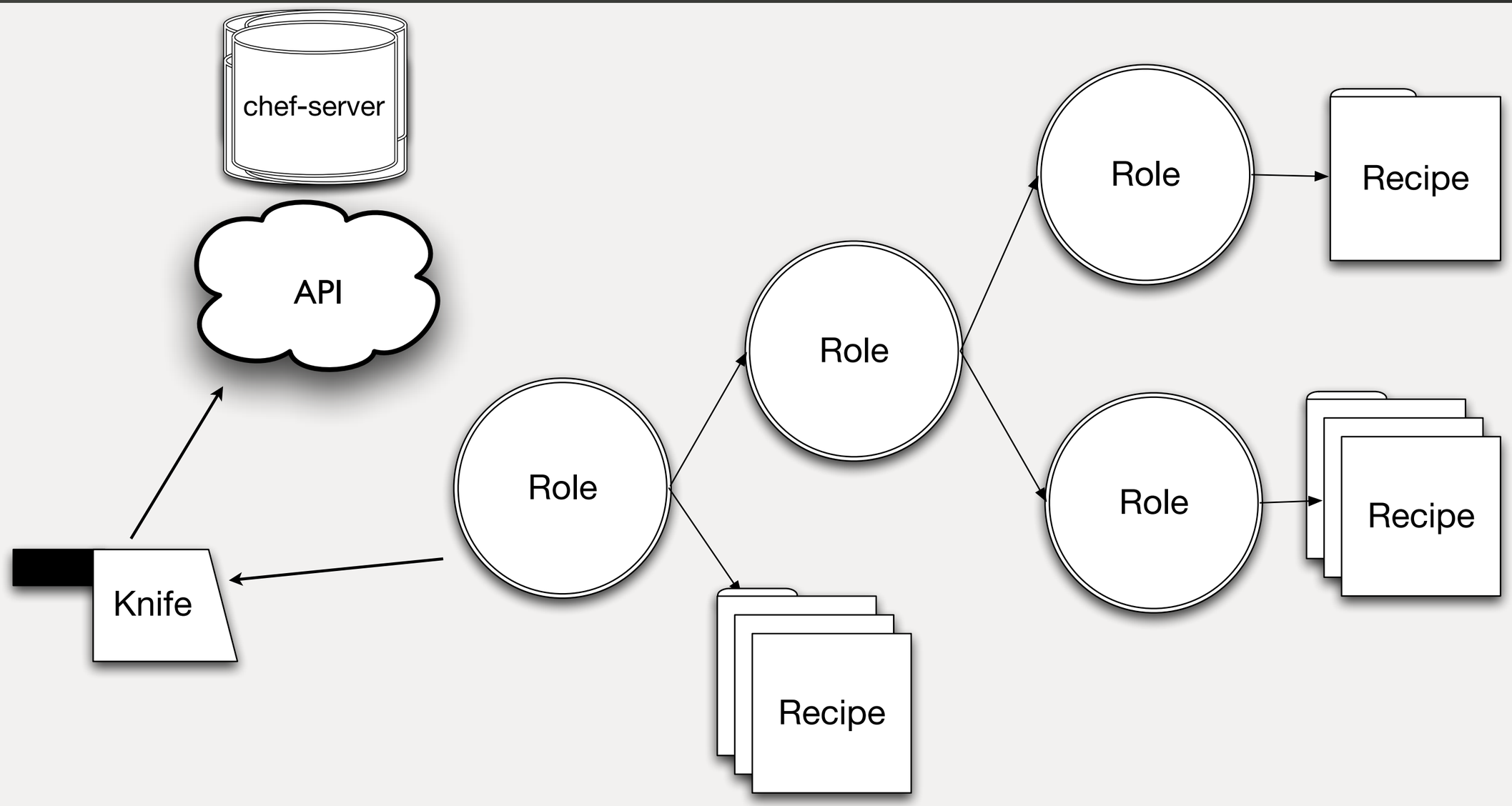
```
name "webserver"
description "webserver server"
run_list [
  "role[base]",
  "recipe[nginx::server]"
]
```

```
name "base"
description "base"
run_list [
  "recipe[selinux::disabled]",
  "recipe[etchosts]",
  "recipe[yum::epel]",
  "recipe[debugtools]"
]
```

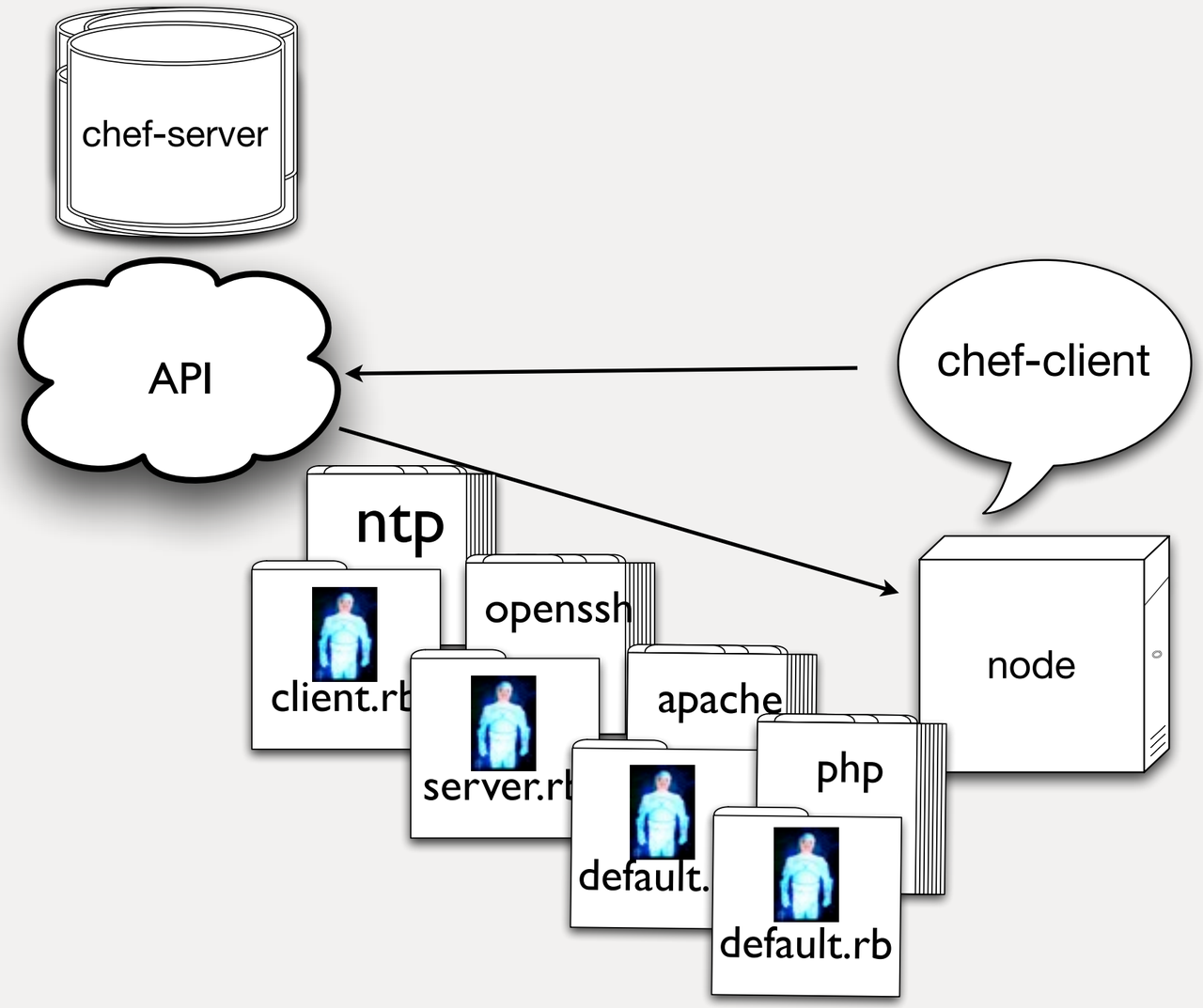
RULE THE CLOUD

OPSCODE

Roles

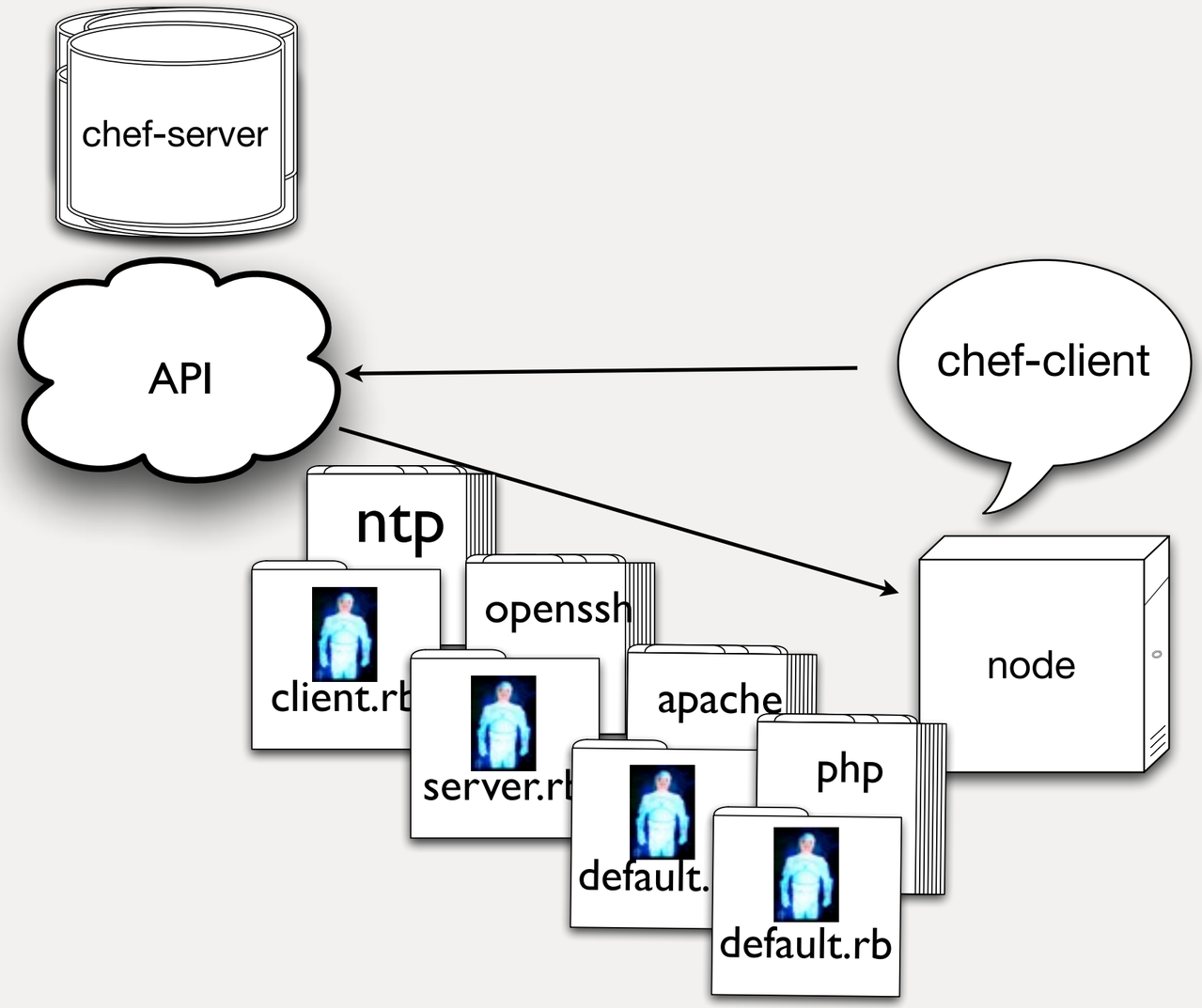


Run Lists



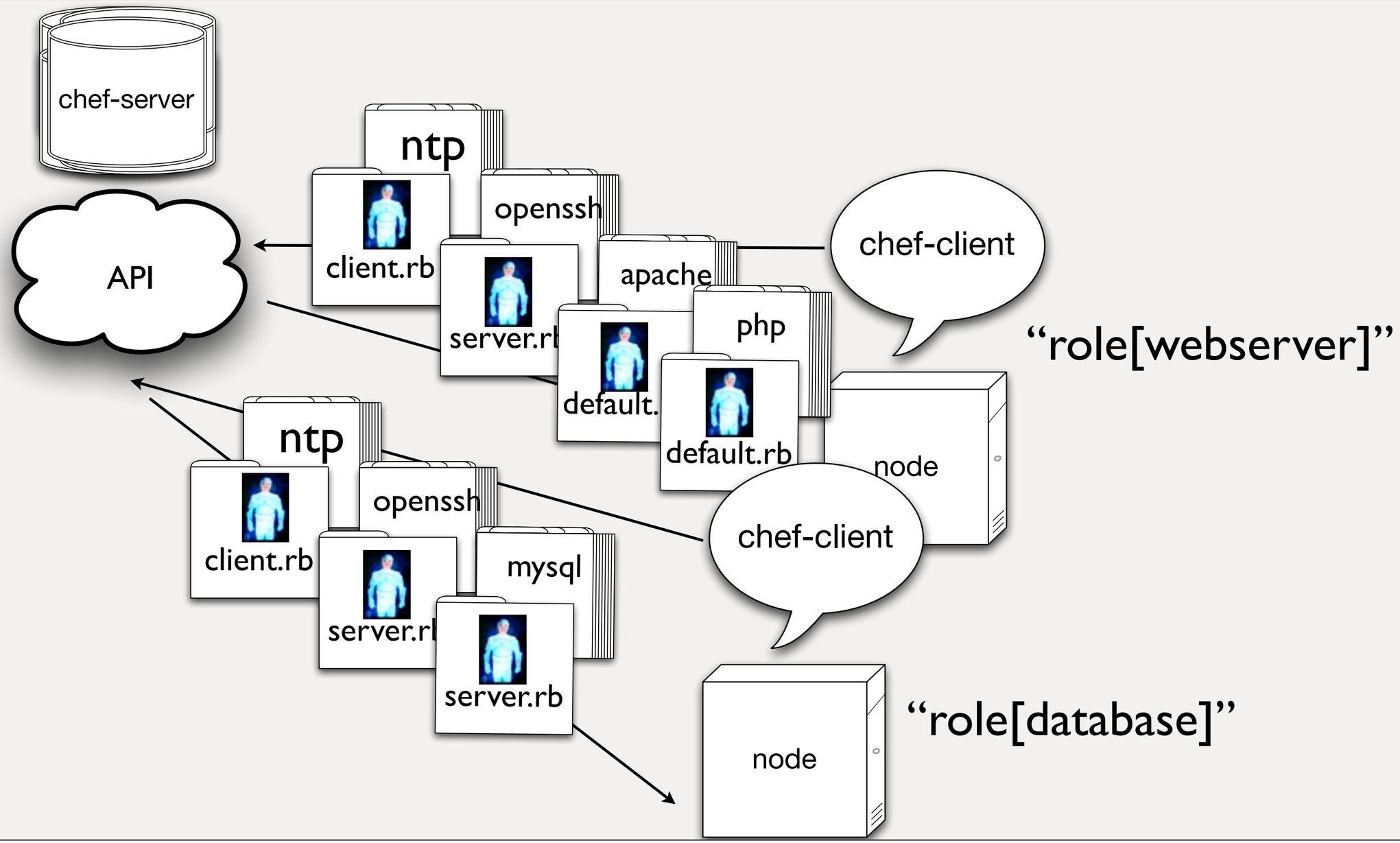
"recipe[ntp::client]",
"recipe[openssh::server]",
"recipe[apache]",
"recipe[php]"

Roles



"role[base]",
"role[webserver]"

Roles


chef-server
API
ntp
client.rb
openssh
server.rb
apache
default.
php
default.rb
chef-client
node
"role[webserver]"
ntp
client.rb
openssh
server.rb
mysql
server.rb
chef-client
node
"role[database]"

OPSCODE

Search

- Search for nodes with Roles
- Find configuration data

- IP addresses
- Hostnames
- FQDNs

http://www.flickr.com/photos/kathycsus/2686772625

OPSCODE

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 -%>
```

OPSCODE

munin::server example

```
node.set[:munin][:server] = true
munin_clients = search(:node, "munin_client:true")

cookbook_file "/etc/cron.d/munin" do
  source "munin-cron"
  mode "0644"
  owner "root"
  group "root"
end

template "/etc/munin/munin.conf" do
  source "munin.conf.erb"
  mode 0644
  variables(:munin_clients => munin_clients)
end
```

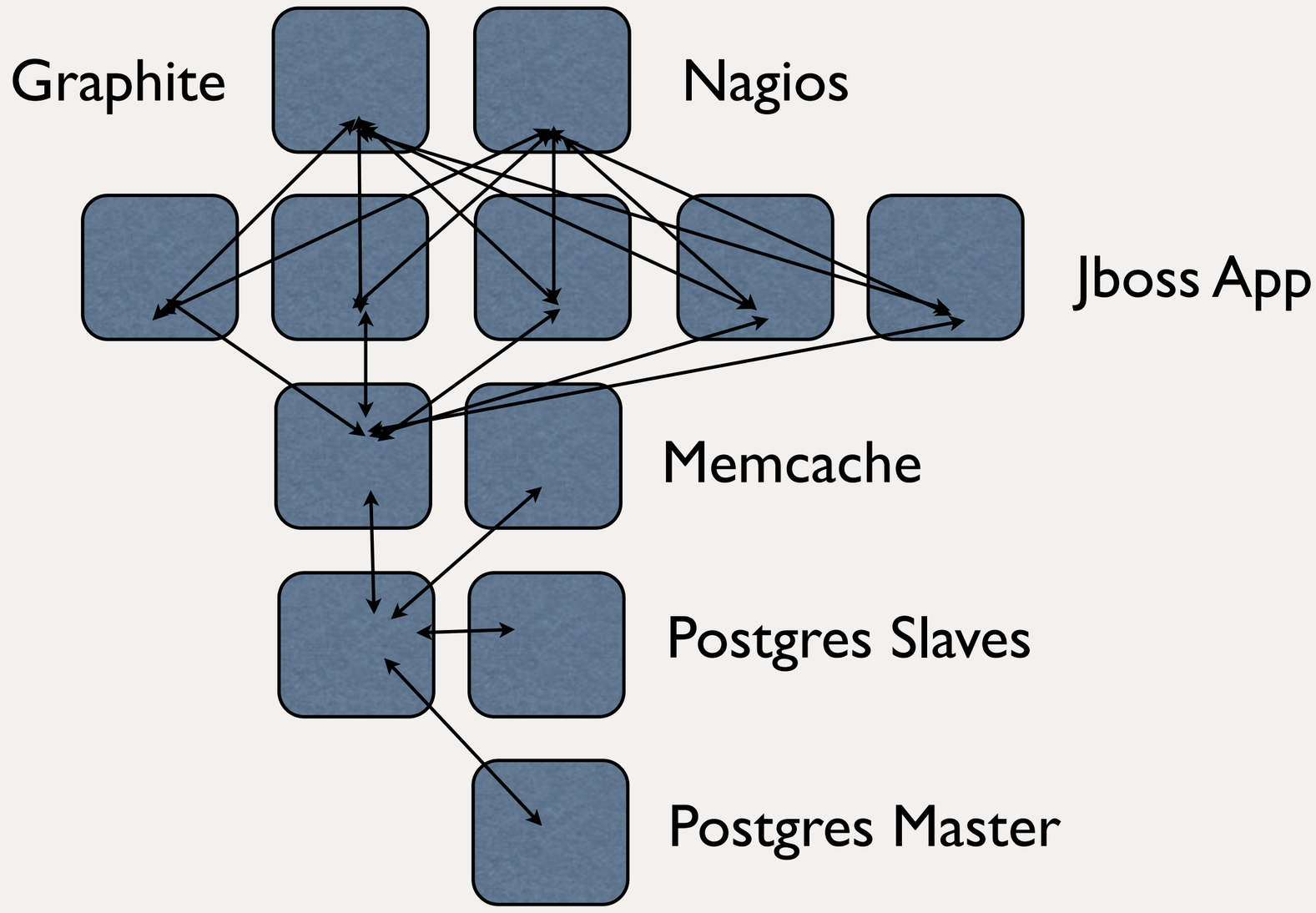
OPSCODE

munin::client example

```
node.set[:munin][:client] = true
munin_servers = search(:node, "munin_server:true")

unless munin_servers.empty?
  package "munin-node" do
    action :install
  end

  template "/etc/munin/munin-node.conf" do
    source "munin-node.conf.erb"
    mode 0644
    variables :munin_servers => munin_servers
    notifies :restart, "service[munin-node]"
  end

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

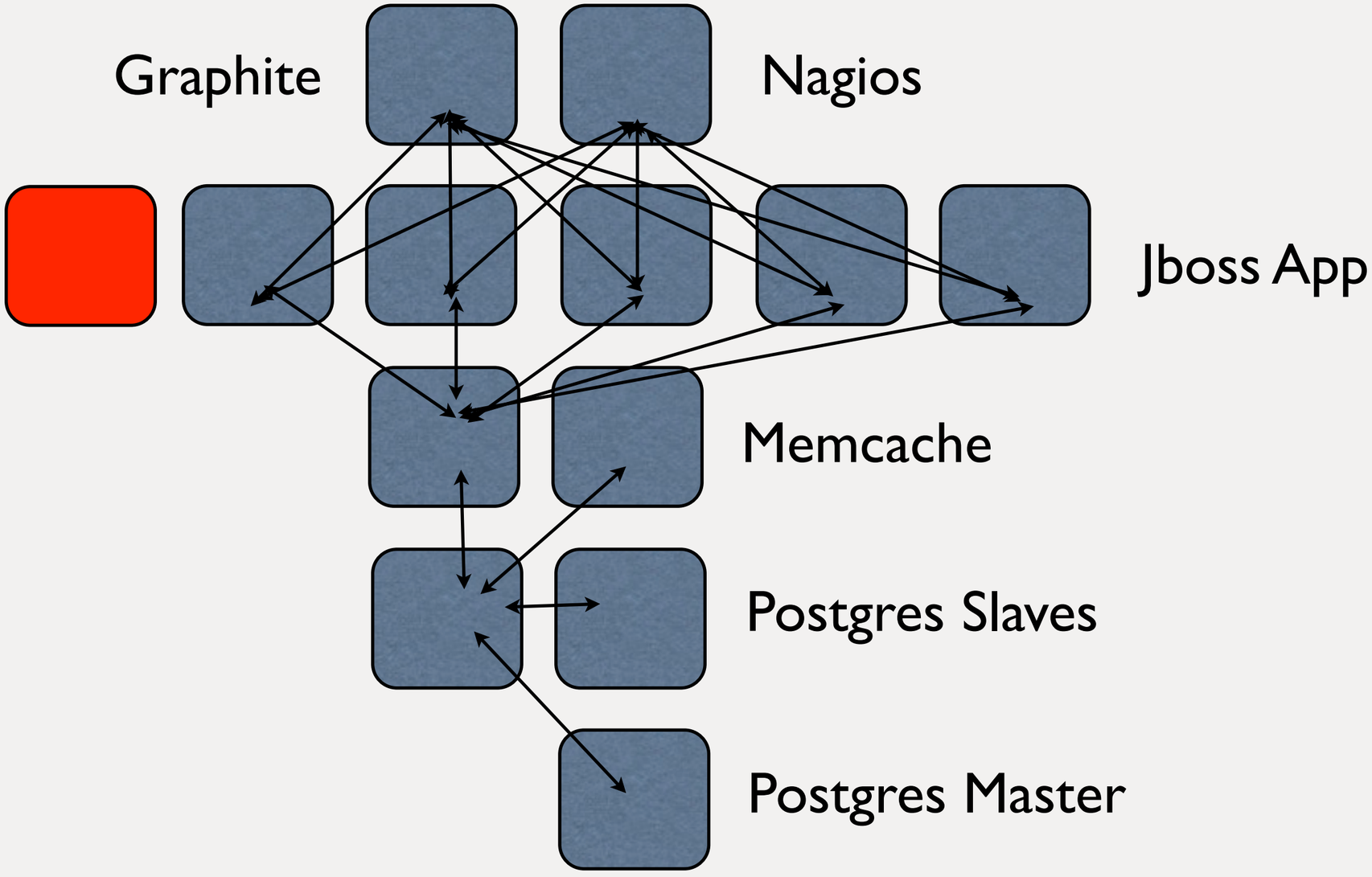
OPSCODE

So when this

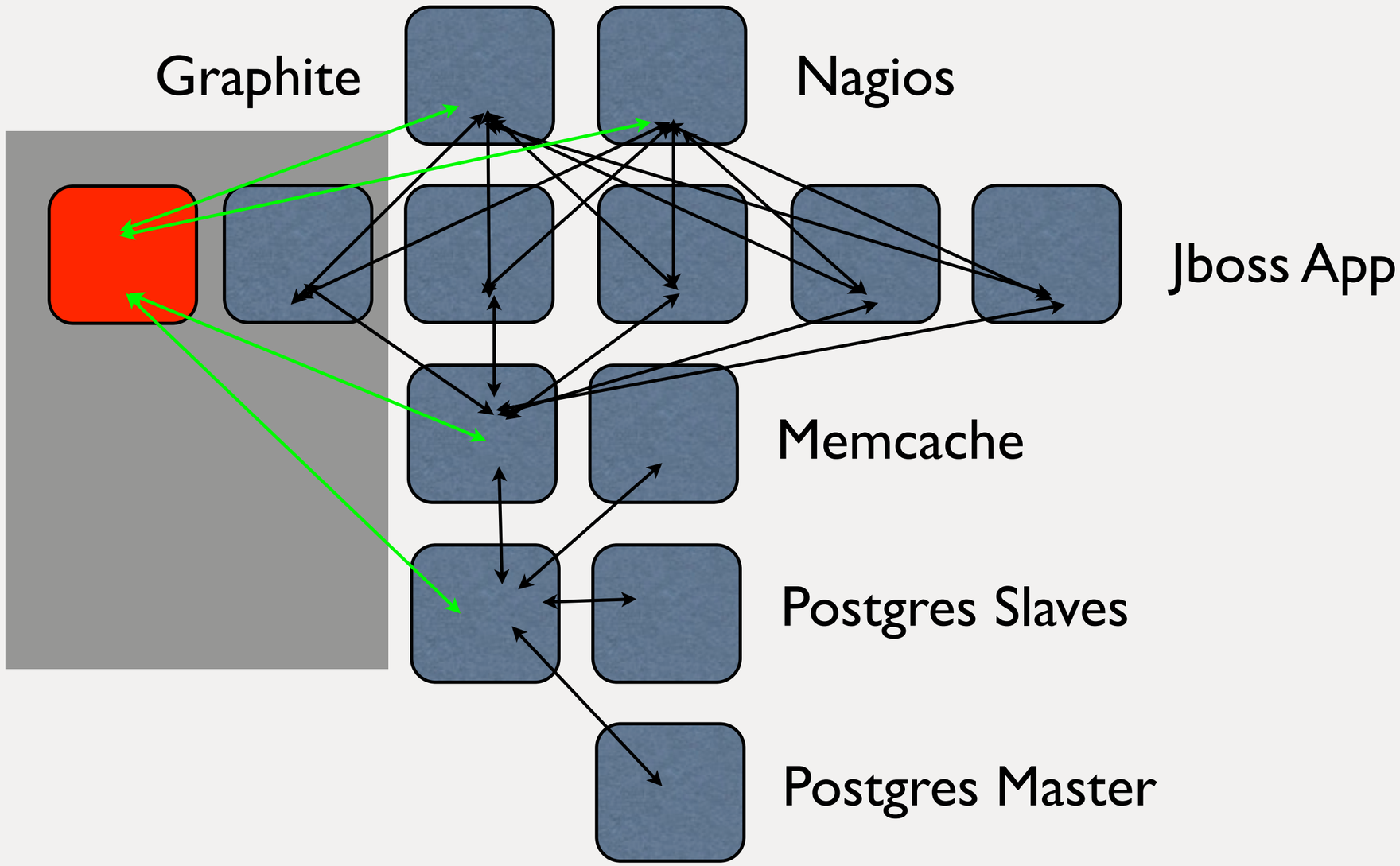


OPSCODE

Becomes this

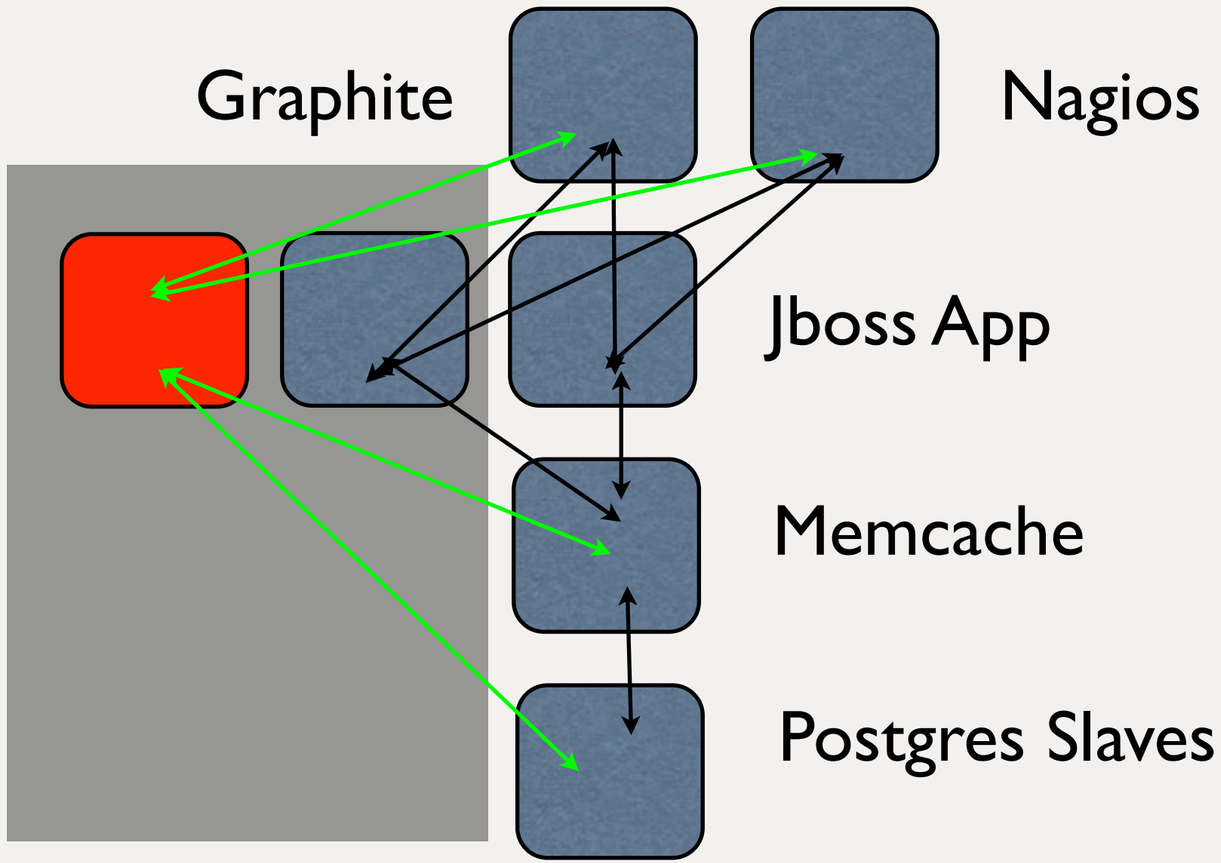


This can happen automatically



OPSCODE

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
- Graphite CPU
- Graphite Memory
- Graphite Disk
- Graphite SNMP
- Memcache firewall
- Postgres firewall
- Postgres authZ config

Questions!

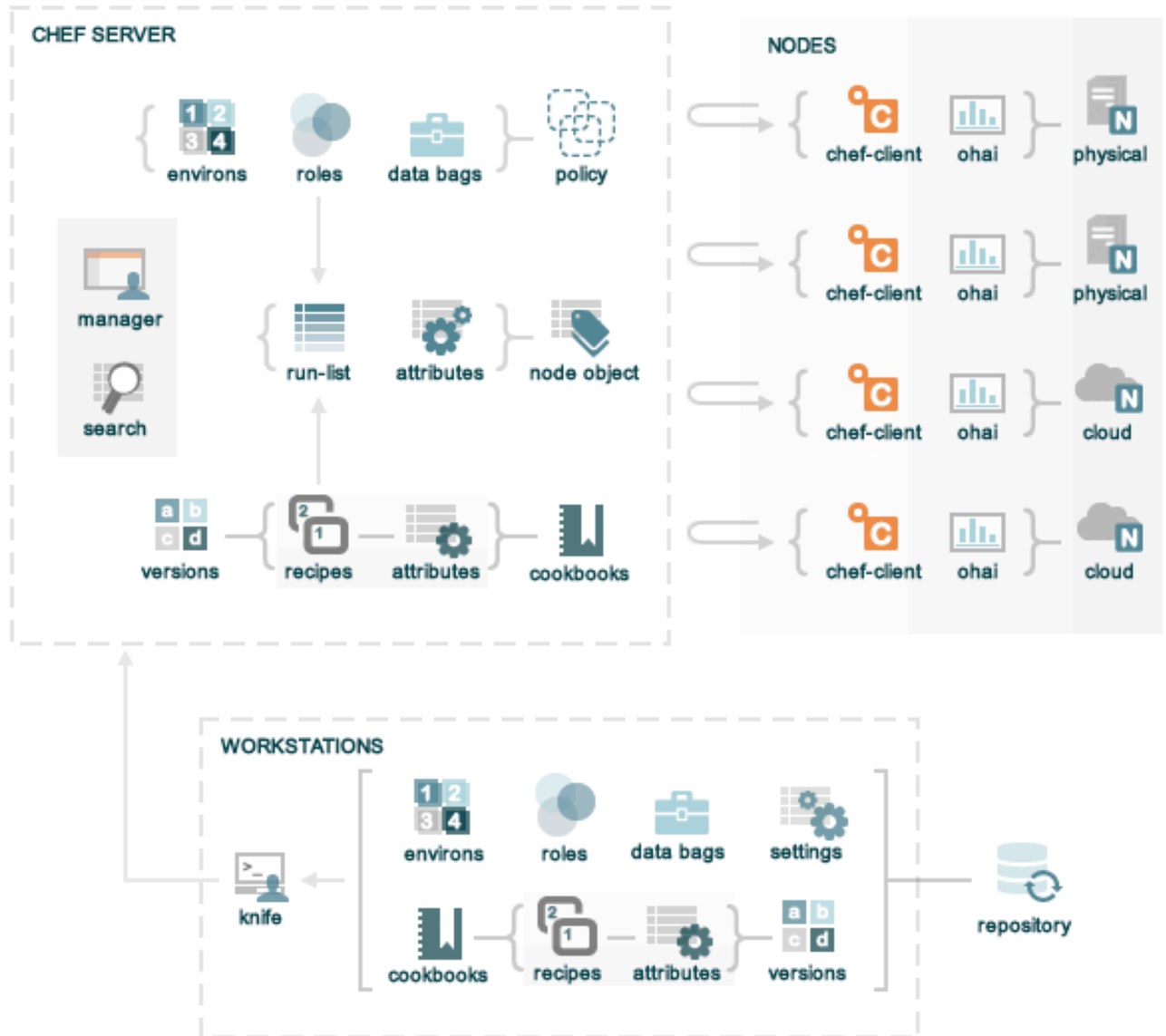
Workstation Setup

Objectives

- Have Chef installed on your Workstation
- Have an empty chef-repo
- Have a unique User created
- Have an Organization created for use during training
- Understand what Knife is
- Have a working Knife configuration
- Ensure a properly configured text editor is available

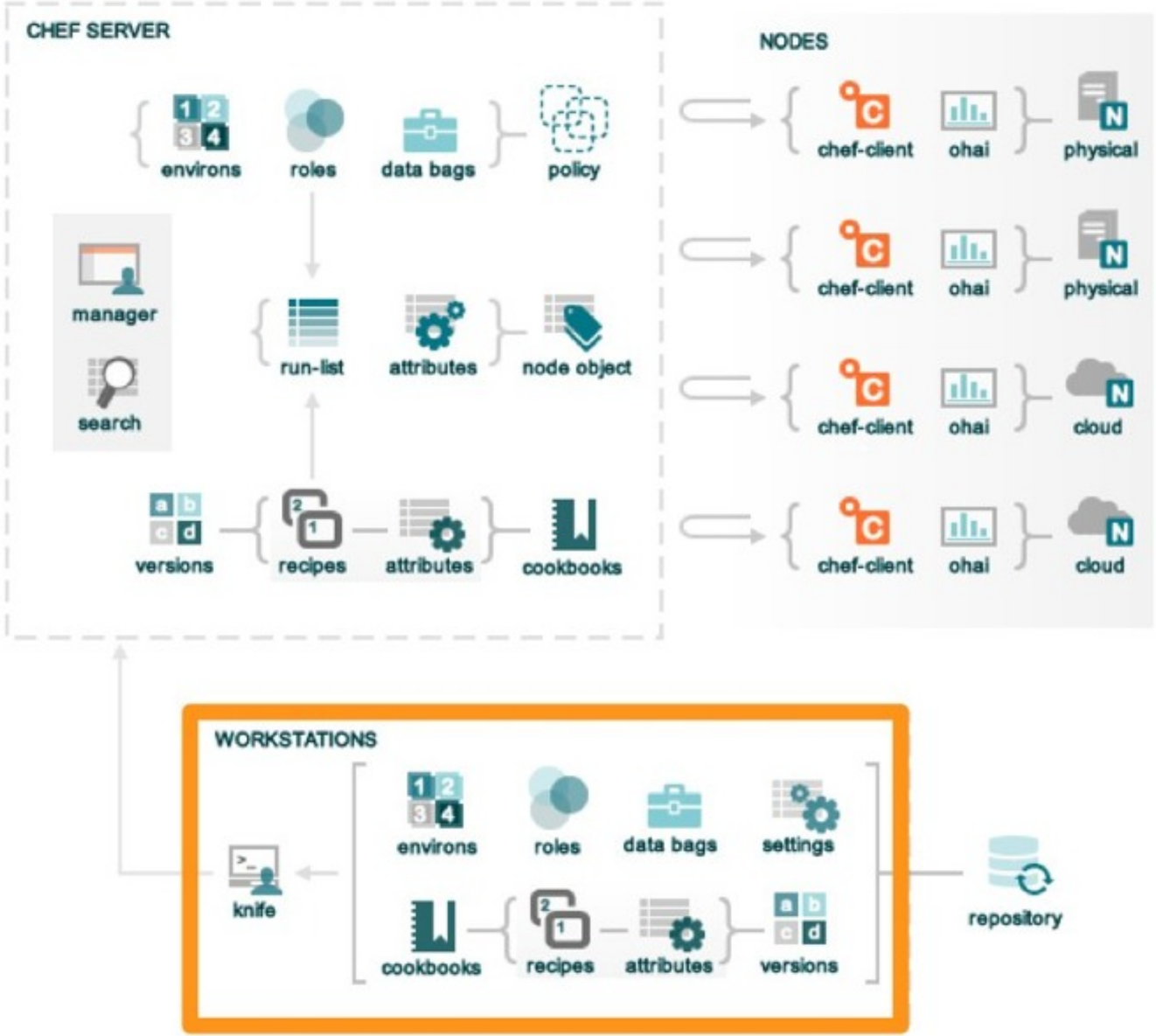
OPSCODE

Landscape of Chef-managed Infrastructure



Workstation Setup

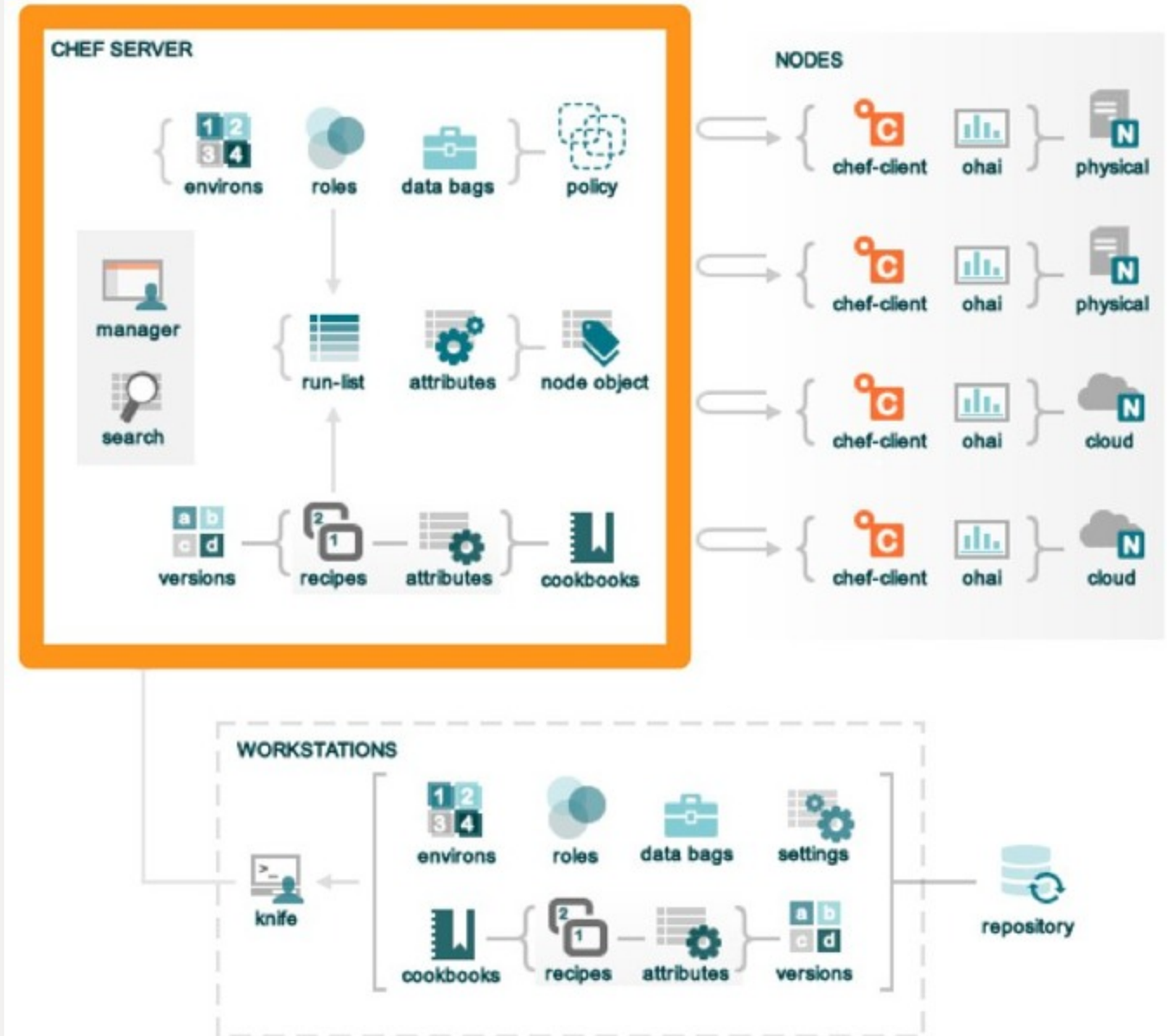
- Install Chef (if not already installed)
- https://www.opscode.com/chef/install/



OPSCODE | Chef

Your Chef Server for this class...

- Set up Chef Server Account
- Opscode Hosted Chef
- https://manage.opscode.com



OPSCODE | Chef

Sign-up for Hosted Chef

Hosted Chef

Instant access to a highly available, dynamically scalable, fully managed and supported automation environment

Opscode's Hosted Chef combines the freedom and flexibility of Chef with the reliability and speed of Opscode. No matter how complex the realities of your business, Hosted Chef makes it easy to deploy servers and scale applications throughout your entire infrastructure. Because it combines the fundamental elements of configuration management and service oriented architectures with the full power of Ruby, Hosted Chef makes it easy to create a fully automated infrastructure.

Download the fact sheet

Plans & Pricing

	Trial
Monthly Fees	$0
Nodes	5
Users	5
Standard Support	Included
Onsite Training	Available

Free Trial ›

About you

1 2 3

First Name * | Middle Name | Last Name *

Username * | Email Address *

Password * | Confirm Password *

Organization Name * | Organization Short Name *

Phone Number * | Company *

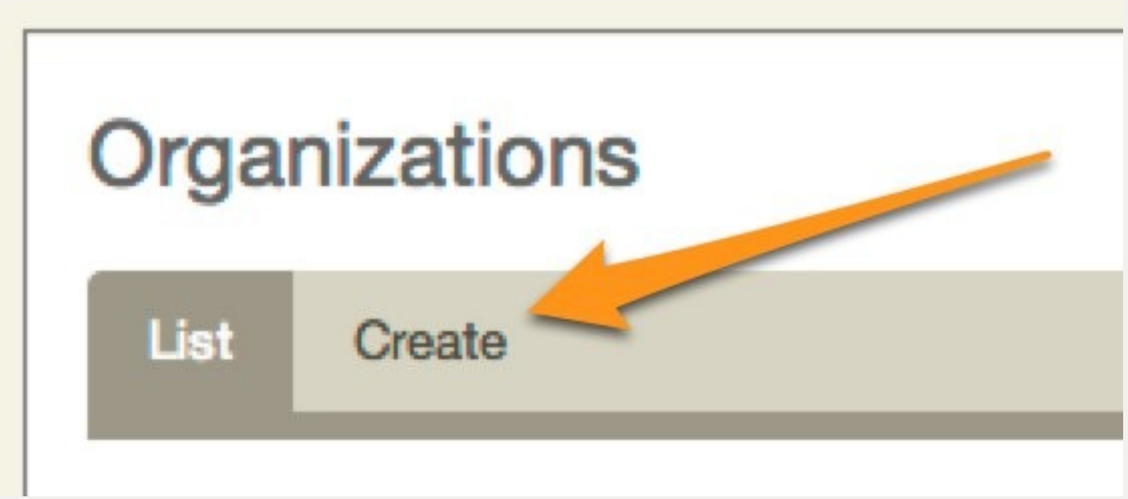
Country * | State *

☐ I agree to the Terms of Service, Opscode Platform Customer Agreement, and Opscode Service Level Agreement.

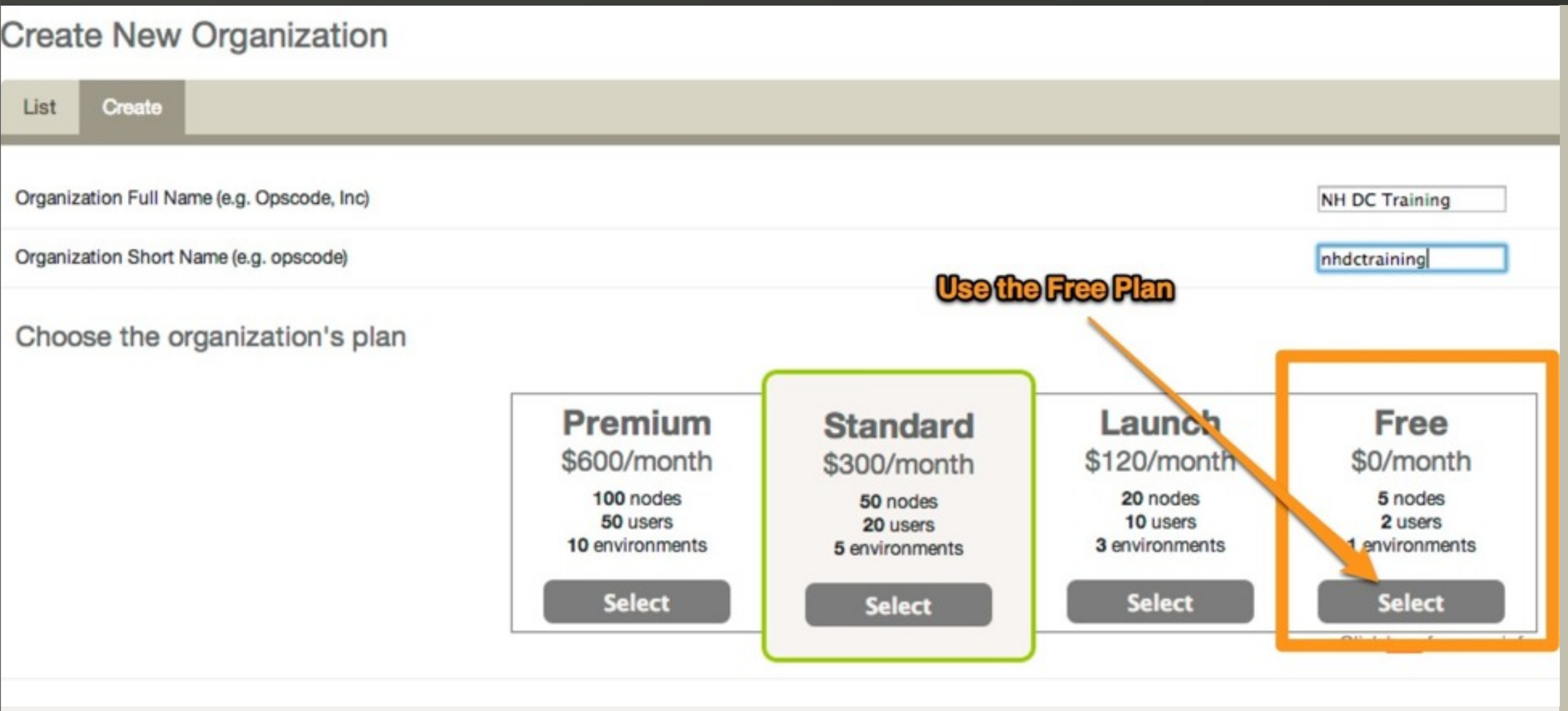
‹ Back | Next ›

OPSCODE | Chef

Create an Organization



Create New Organization



Organization Short Name must be GLOBALLY unique!

Download the Validation Key and Knife Config

Organization: training

List Create

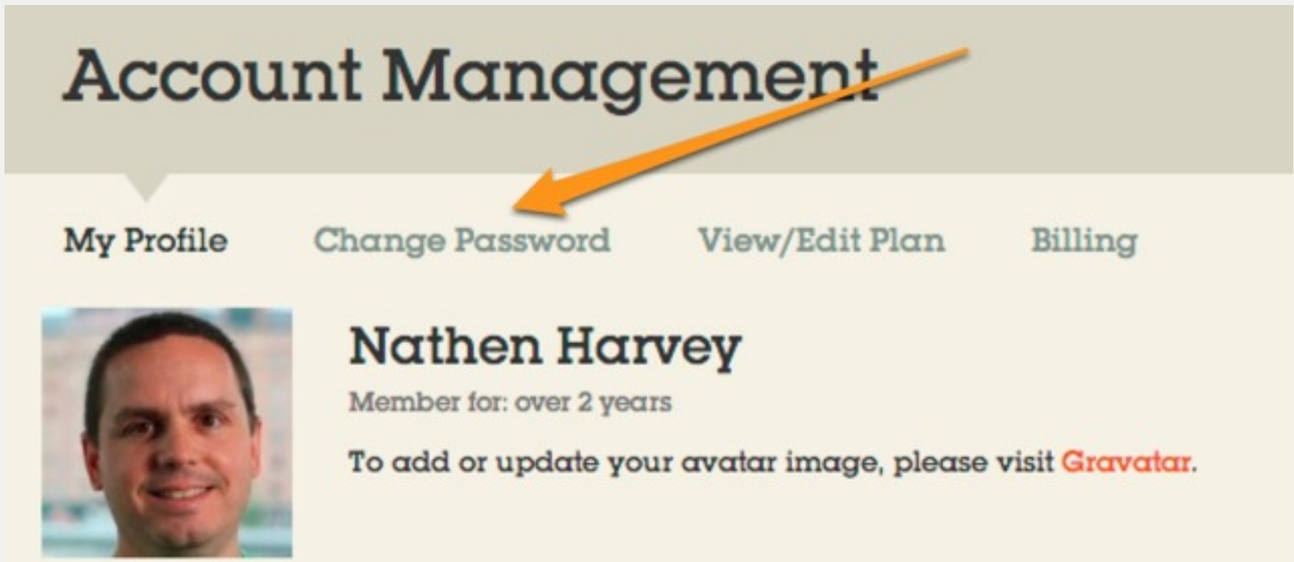
Save the validation key and Knife configuration file

This private key will be required when you interact with Opscode Platform APIs. Opscode does not keep a copy so please store it somewhere safe.

Knife is Chef's command-line tool. You can download a pre-configured copy of the configuration file.

Download validation key | Generate knife config

Get a New User Key



- Only if you don't have your user key with you today!

OPSCODE

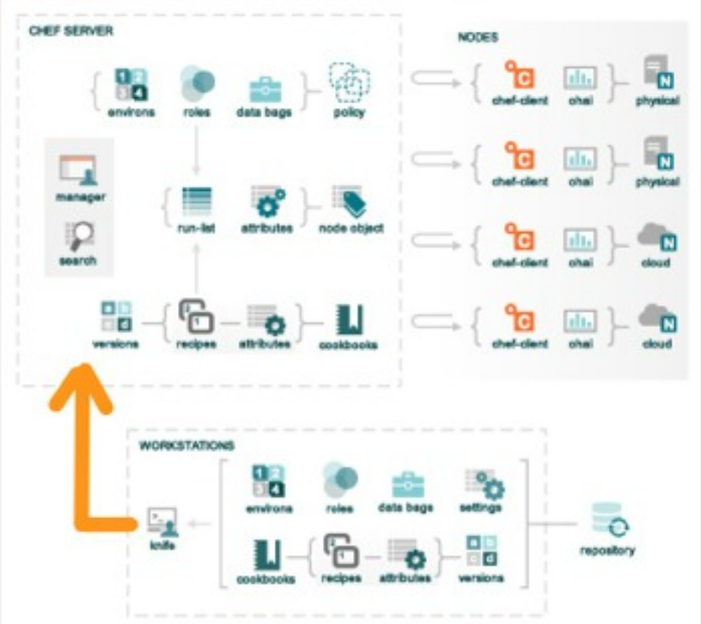
Exercise: Create your Chef Repository

- Every Chef installation has (at least one) Chef Repository.
- Make a chef-repo directory somewhere (e.g. C:\Users\yourname\chef-repo)
- Make the following empty directories in it:
  - cookbooks
  - data_bags
  - environments
  - roles

RULE THE CLOUD

Copy Chef Server Files

```
# copy your user key, validation key and knife config:
> cd chef-repo
> mkdir .chef
> cp ~/Downloads/ORGNAME-validator.pem .chef
> cp ~/Downloads/USERNAME.pem .chef
> cp ~/Downloads/knife.rb .chef

> ls .chef
ORGNAME-validator.pem
USERNAME.pem
knife.rb
```



Verify Knife

```
> knife --version
Chef: 11.4.4
```

Your version may differ, that's okay!

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

Knife is the command-line tool for Chef

Exercise: Run 'knife help list'

```
$ knife help list
```

```
Available help topics are:
  bootstrap
  client
  configure
  cookbook
  cookbook-site
  data-bag
  environment
  exec
  index
  knife
  node
  role
  search
  shef
  ssh
  status
  tag
```

Exercise: Run 'knife client list'

```
$ knife client list
```

```
opscode-training-validator
```

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++, Eclipse, etc.

Best Practice: Set your EDITOR

Linux/OSX/Unix, in your ~/.bash_profile

```
export EDITOR=/path/to/emacs
source ~/.bash_profile
```

Windows in cmd.exe

```
setx EDITOR "\"%ProgramFiles%\Windows NT\Accessories\wordpad.exe\""
```

or

```
setx EDITOR "\"%programfiles%\Sublime Text 2\sublime_text.exe\" --wait“
```

close cmd.exe and start a new cmd.exe session

Using Notepad++ on Windows

```
setx EDITOR "\"C:\Program Files (x86)\Notepad++\notepad++.exe\" -nosession -multiInst"

set EDITOR="C:\Program Files (x86)\Notepad++\notepad++.exe" -nosession -multiInst
```

Exercise: Verify a working EDITOR is configured

```
$ knife client create editor-test
```

knife-edit-0402462757132607373 50.js (/private/var/folders/0c/j_mymdt9063_195pf830x70r0000gn/T) - VIM

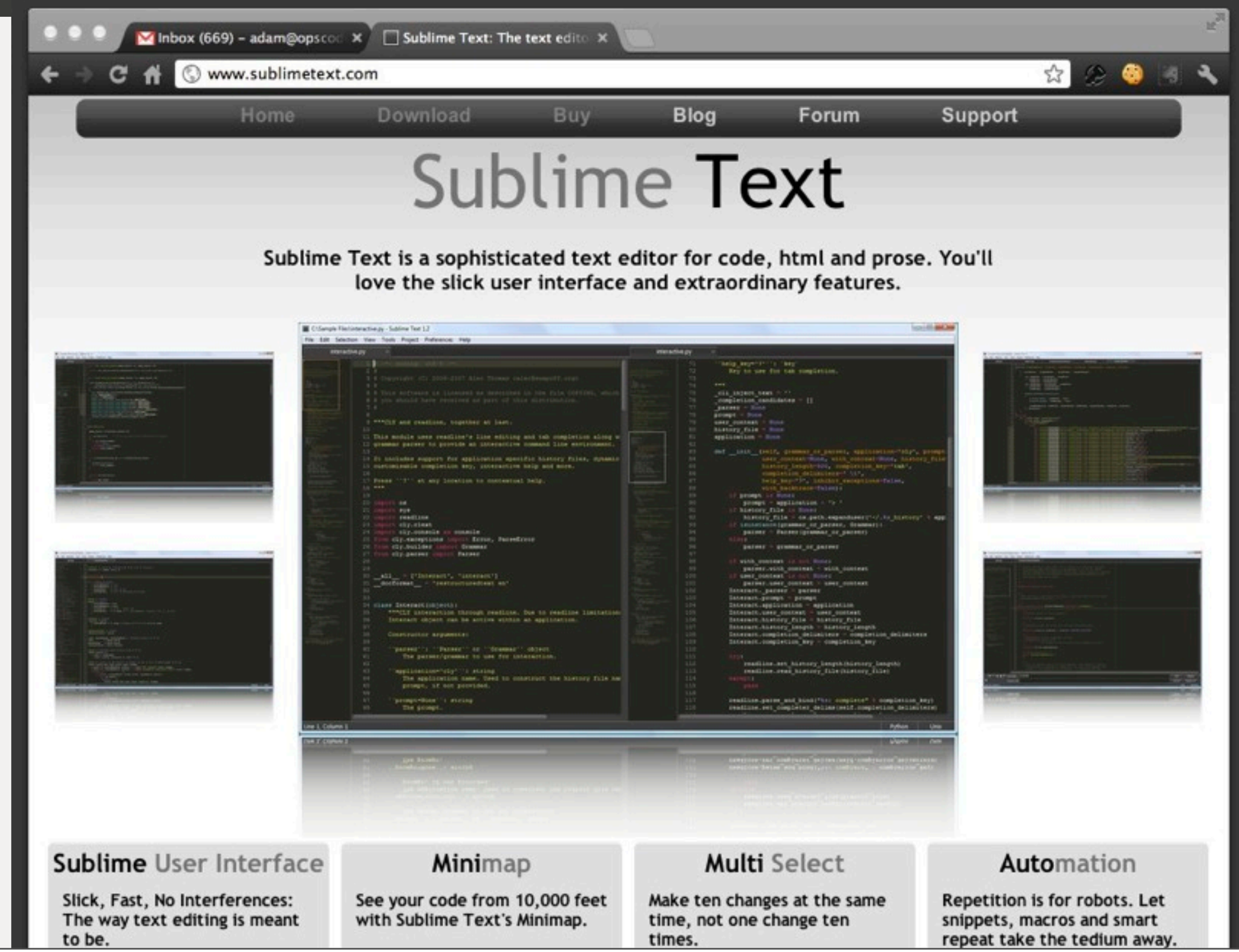
```
1
2   "name": "editor-test",
3   "public_key": null,
4   "admin": false,
5   "json_class": "Chef::ApiClient",
6   "chef_type": "client"
7
~
~
N  /private/var/folders/0c/j_mymdt9063_195pf830x70r0000gn/T/knife-edit-0402462757132607373 50.js
```

Save the file and **Exit** your editor.

OPSCODE

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
- http://sublimetext.com



Using Sublime on Mac OS X

```
export EDITOR=’/Applications/Sublime\ Text\ 2.app/
Contents/SharedSupport/bin/subl -w’
```

Using Sublime on Windows

```
setx EDITOR "\"%programfiles%\Sublime Text 2\sublime_text.exe\" --
wait"
```

Exercise: Verify a working EDITOR is configured

```
$ knife client create editor-test
```

knife-edit-0402462757132607373 50.js (/private/var/folders/0c/j_mymdt9063_195pf830x70r0000gn/T) - VIM

```
1
2   "name": "editor-test",
3   "public_key": null,
4   "admin": false,
5   "json_class": "Chef::ApiClient",
6   "chef_type": "client"
7
~
~
N  /private/var/folders/0c/j_mymdt9063_195pf830x70r0000gn/T/knife-edit-0402462757132607373 50.js
```

Save the file and **Exit** your editor.

Exercise: Verify a working EDITOR is configured

```
Created client[editor-test]
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAv313jX0crYxdNsqXTflh2N6e3kGSUTCJRVwCXIFh6YmdoY3c
bcIWESebcL5y5Vta3HjTSmohkSIW3JF0BLK39fQ8V0dM0wSBmZm9NvJwKucuwRes
u8EszMeLY/9tGFD9TK0iFlPN1jtRE1Ym61QRziRyAvkbkPEZacIVdayNM5TS8LuN
BSdYIZM8vnqRr6cf7WJn0m+lIKdcR1plHWtQT9EgYyPBVlwW9R8fR1i/QHeOyGz2
RTlYYAq+qvuUimv3OQoZXca/nkxroFWj80s9RJCgPdI44n51ws/PPmPQYeWykZFt
9+pg5dud+Q1WbxR5yH0lFDGVGaoFBB6mny+JaQIDAQABAoIBAQCeKdC8gfFU+1Hm
trEAi5IhIcfQxhJHkzJeop+kro0j6zHSxayiz6OQacl+9x9JiAplfjrvBPhSuvht
GIrecot10FSitlajGN6+8vgRUXAKT8cTYC2hKu6I+eyNHOWMJyEA2yQDkxa582aE
9SzSRS8ruHifMght0GZwLwmwl2RNjBpbqWK6YI24yv6ljgqkcUoC7gGECMKWFWoE
4DLYKnmza+0WeVe+xeC9R9gCz4qqq56U06RptcJljLQNzjLsHv/1x1tRgyiy1k7Q
INmMPYcproh2Frmx5sipBml2nxGUNfJncMMeFV9AGddIdvPYzY2Wp5vZukTUVTxB
pO03T2sBAoGBAOFRX6cEAvh4P8qZ3UqzDbBuuREJ6XGWNOdMjk4CgJn7owGXBige
XIiB4du3OlCfD2G7p2FFipzaxylEVHTd9djT5NA6FBvUL6ATo0cS6lteTwixwbtH
1RhM2im8DYx+RR2PjpsaerQw3DBVx19EUaU7LS5W7CSMuHuEjHpru1QhAoGBANmQ
2c3NGshBy06dOfKSXIU5h2nyM7N33tsj08upgerA1QXxcRLjWJ2Xr12W+npeFtTV
LtSsBV9HirayhfVqiSiLj1ulS+GbFHMZy8i9vuOO9JM+ysI0NY1XPhxcNjpqRZLn
/ppAbBaZCunx9S7zLRN/YQZOdsScRoYkH//UcwxJAoGBAMiuVHaKESdC2vZVco6s
yZ2CEJ1Hab98XGRCgGCkw+vh/z5UR+yFlj8TB5pNKMhT0zxmCd+OM7Ye+dIchOiO
JQDZQWvgvrZGl6CIvR15kHiOl/ATeoyWnwqEsK8JSjv+3wpEKWyl+fHxrrvyOp/0
Vo/HFCe4zZyEJXAGD6SdvXxhAoGBAKfUWF+sV3uhXW7QVFIt21cd8LqmjoFj97K+
KXRS1xg7FljWTi6O194Bk9KzU5cvm5ckuFJwPFiqfHPAtRuCyjlPpw/ALA/lrFm5
zXyV+nnZ3gr0Bj7XPXRBl3UTIPXg4riXY8yj431vI38iGcvU5LHEshjGUFIMZ060
8UZNIU6ZAoGAGXIrVnC/vCSmt8wEPl3F5xfp9P3w+7gZExl8lo1O4bSxtTIF3+BP
M/OQbtBm294EBzyyRZzTEXhaAByGI07vXeRuv9H8cpFnSlrMaAXB+WV+drhOlL3J
AWlIT8ILi3VhbmG+1Kx7CsK8JinBey3dPvfJ8L9l+gbsEU+3Cv8+U8M=
-----END RSA PRIVATE KEY-----
```

Exercise: Clean up after our EDITOR test

```
$ knife client delete editor-test
```

```
Do you really want to delete editor-test? (Y/N) y
Deleted client[editor-test]
```

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 shows just options

Questions

- What is the chef-repo?
- What is an organization?
- What is knife?
- What is your favorite text editor? :)

Test Node Setup

Objectives

- Have Chef installed on your test node
- Have the Chef Client configured to use the organization created in the previous section
- Successfully execute your first chef-client run.

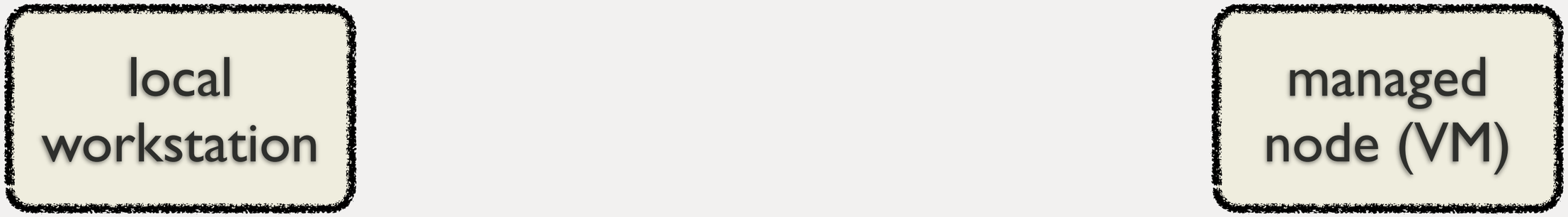
Target Instances

ec2-based Instance

- ec2-STUDENT_ID.compute-1.amazonaws.com
- Ubuntu 12.04
- SSH
  - *Username*: **opscode**
  - *Password*: **opscode**

"Bootstrap" the Target Instance

```
> knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD

Bootstrapping Chef on target
target Starting Chef Client, version 11.4.0
```


local
workstation
managed
node (VM)

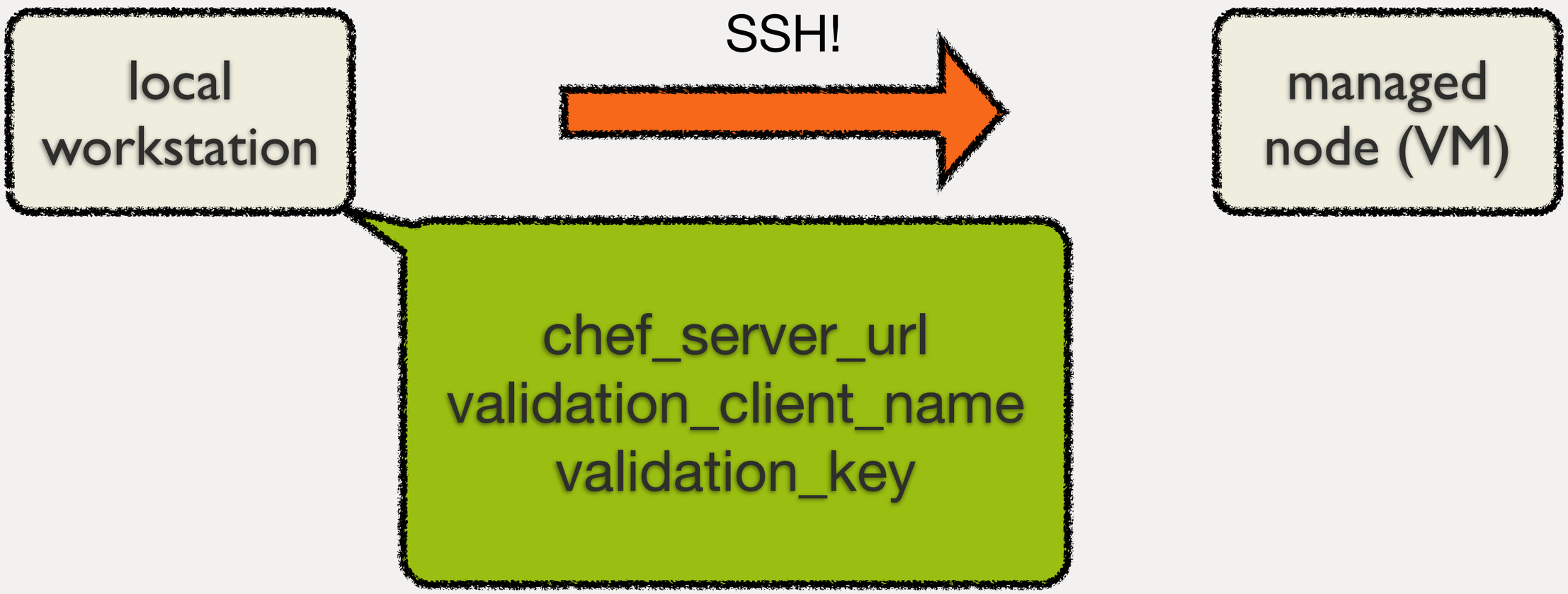
```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```

local workstation

managed node (VM)

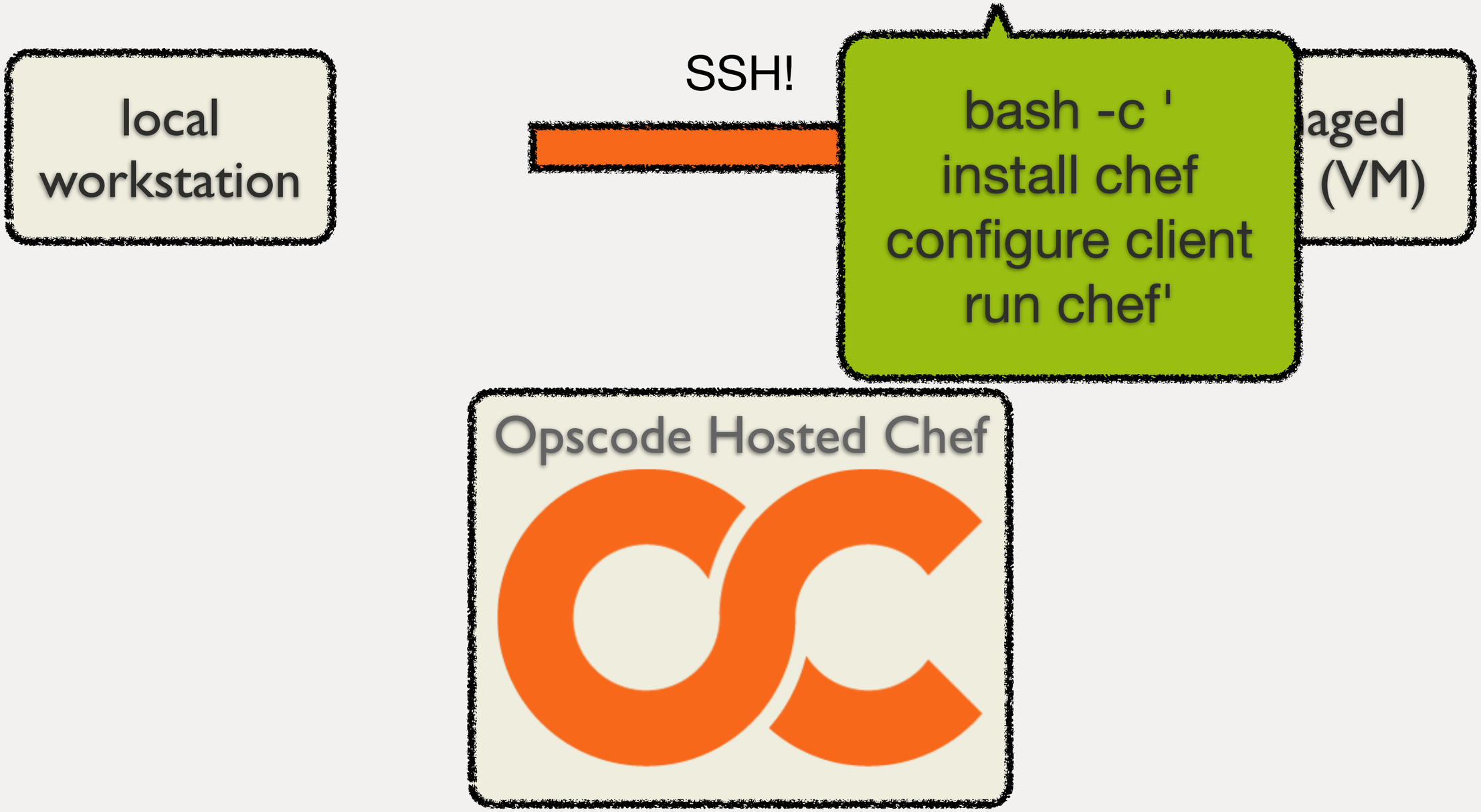
```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```

```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```

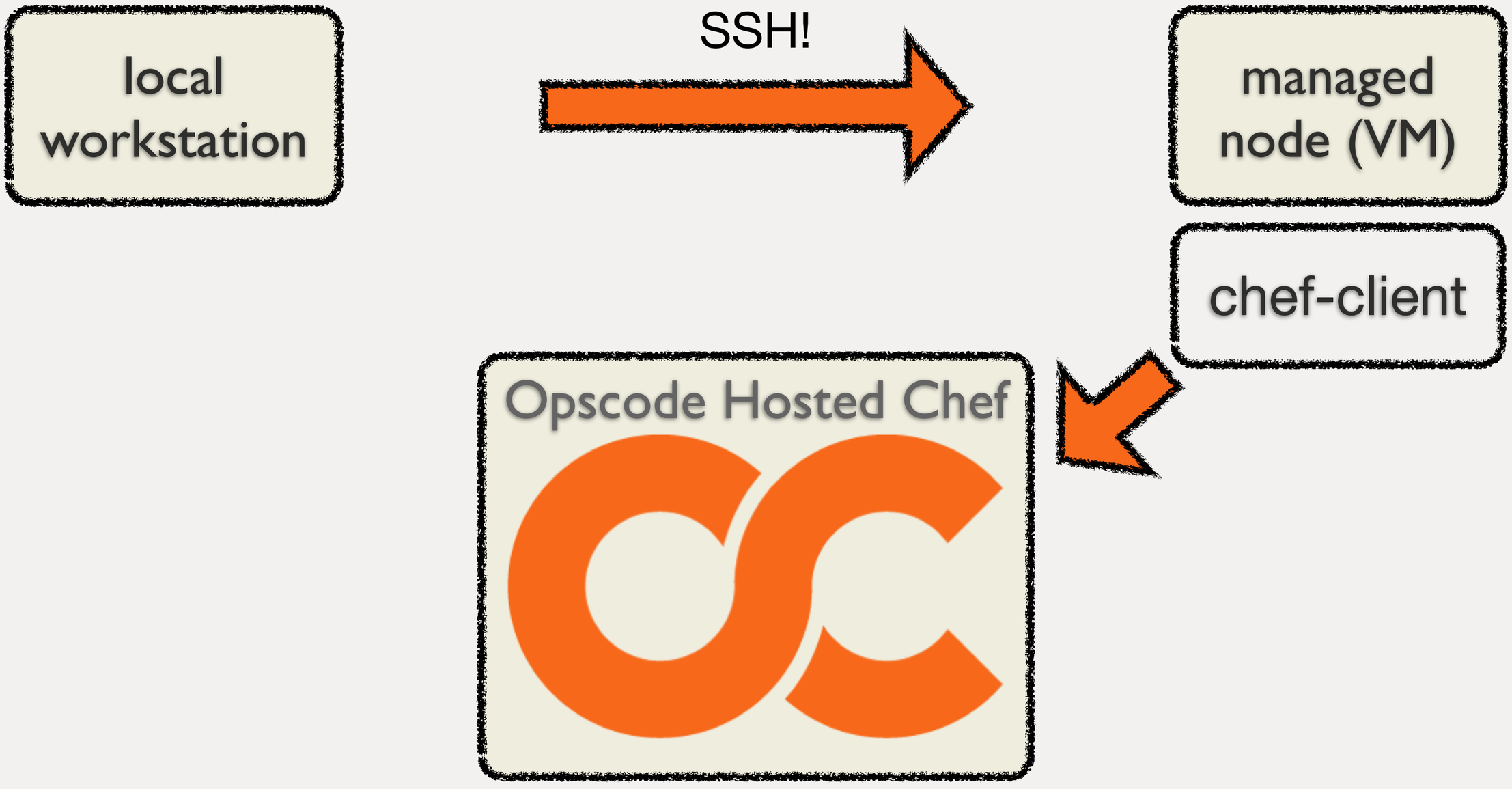


```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```

```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```



```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```



Questions

- Where is the chef-client configuration file?
- What is the command to run chef?
- What is the validation key used for?

Chef 101 Terminology

OPSCODE

RULE THE CLOUD

Chef

chef-client runs on your systems

chef-client talks to a Chef Server

API Clients authenticate with RSA keys

The server has the public key

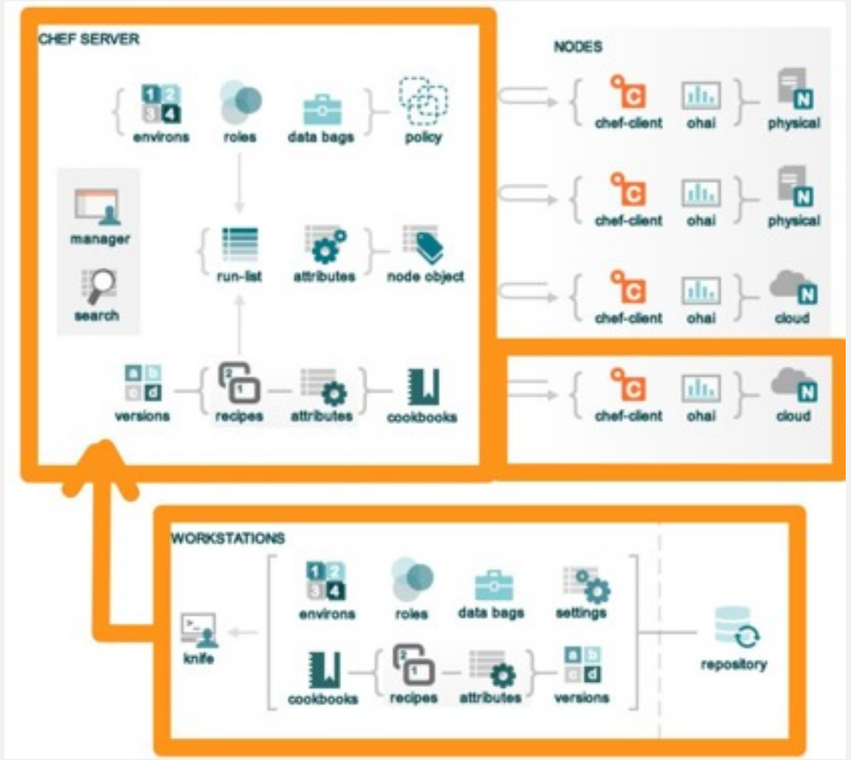
Configured, or managed systems are called **Nodes**

Knife is the command-line user's tool for Chef.

Current Status - Managed Node

Current Status:

```
> knife node list
target1

> knife client list
  target1
  ORGNAME-validator

> knife node show target1
Node Name:   target1
Environment: _default
FQDN:        target1
IP:          10.12.13.201
Run List:
Roles:
Recipes:
Platform:    ubuntu 12.04
```



Knife's commands have built-in help

```
> knife node show --help

> knife help node
```

What did Knife Bootstrap Create?

```
> ssh opscode@target

opscode@target1:~$ ls /etc/chef
client.pem  client.rb  first-boot.json  validation.pem
```

/etc/chef/client.rb

```
$ cat /etc/chef/client.rb
log_level         :auto
log_location      STDOUT
chef_server_url  "https://chef.local/organizations/ORGNAME"
validation_client_name "ORGNAME-validator"
# Using default node name (fqdn)
```

/etc/chef/first-boot.json

```
$ cat /etc/chef/first-boot.json
{"run_list":[]}

$ chef-client -h | grep -i json
    -j JSON_ATTRIBS,  Load attributes from a JSON file or URL
        --json-attributes
```

Chef Server Authentication

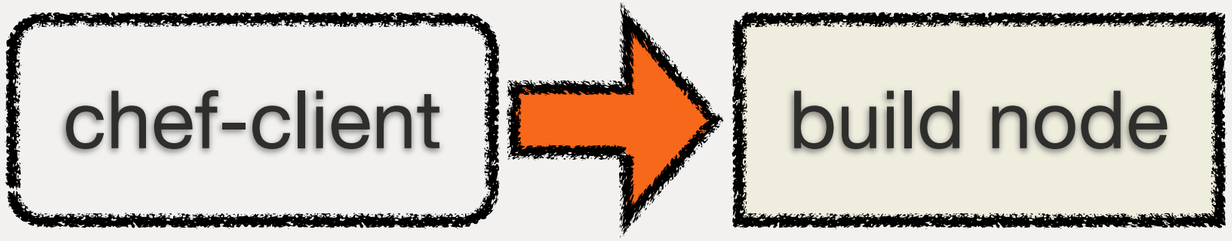
Objectives

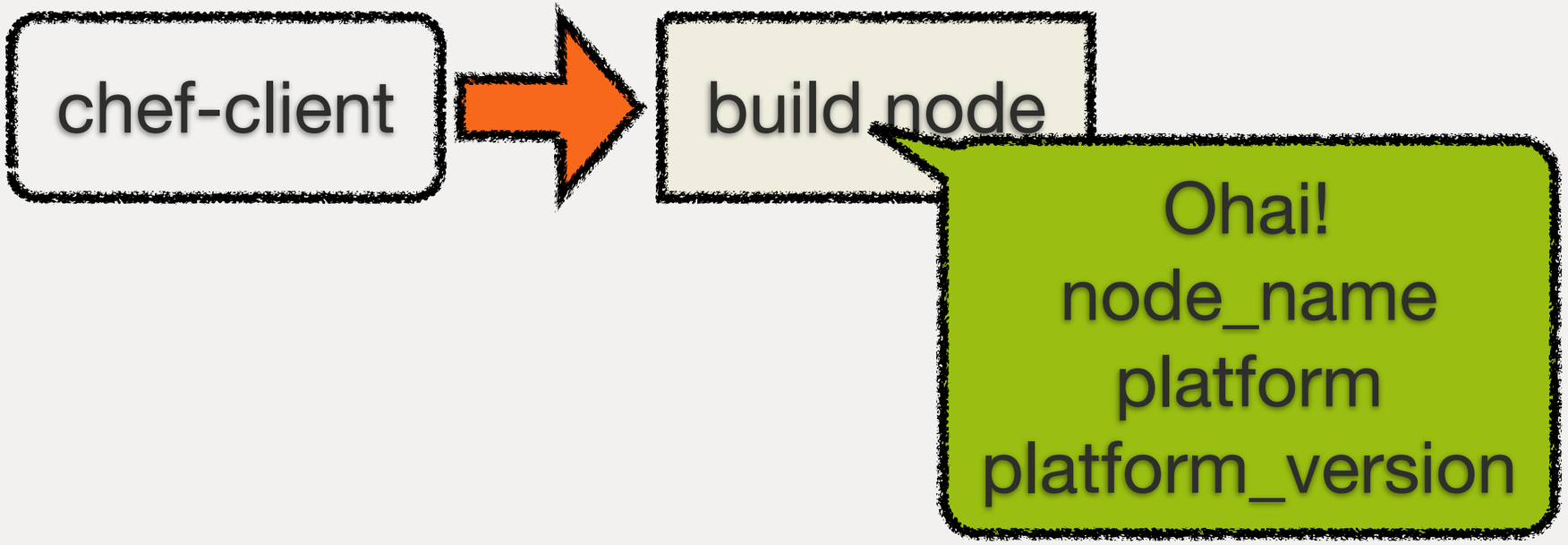
- Understand the steps taken by a chef-client during a run
- Understand the basic security model of Chef

Private Keys

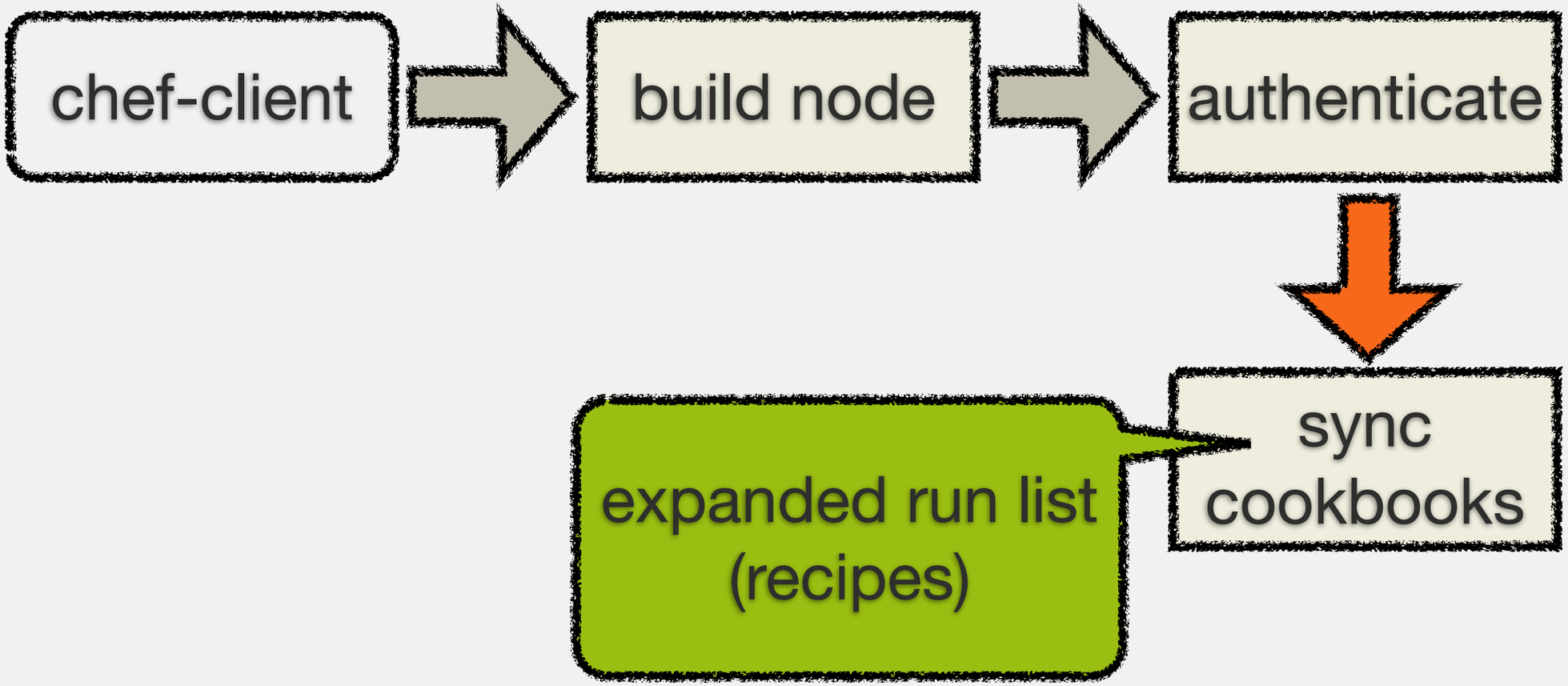
- Chef Server requires keys to authenticate.
  - client.pem - private key for API client
  - validation.pem - private key for ORGNAME-validator
- More on authentication in the next section

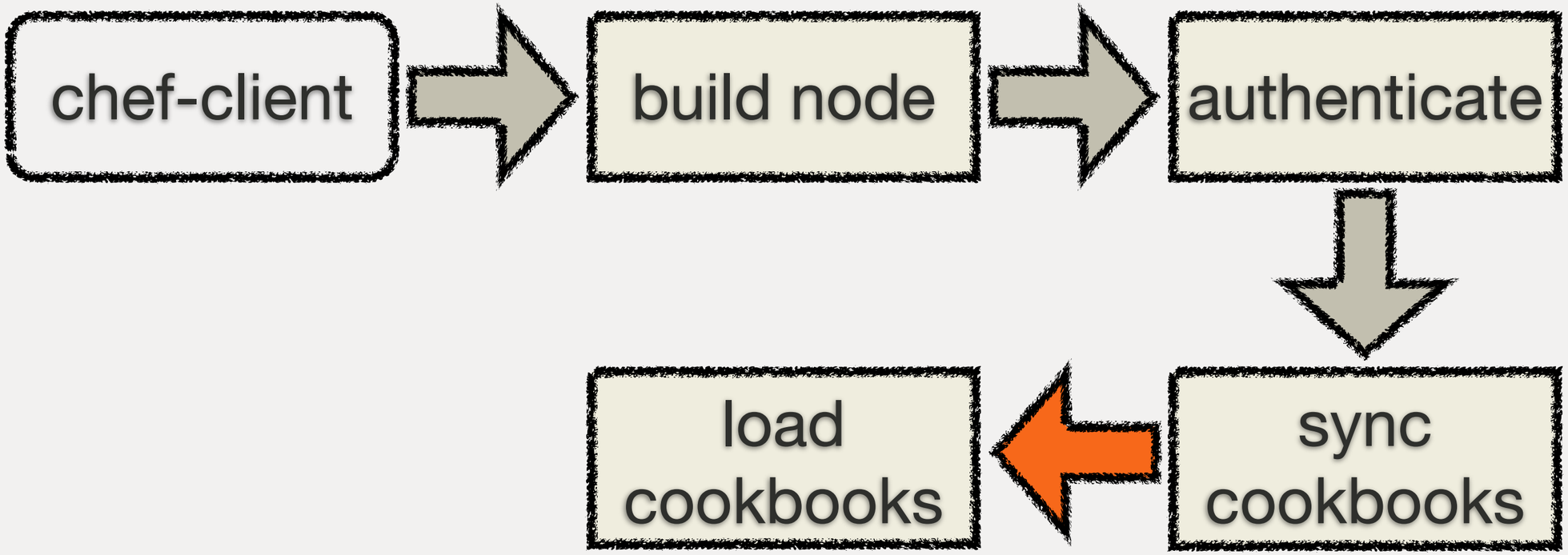
chef-client

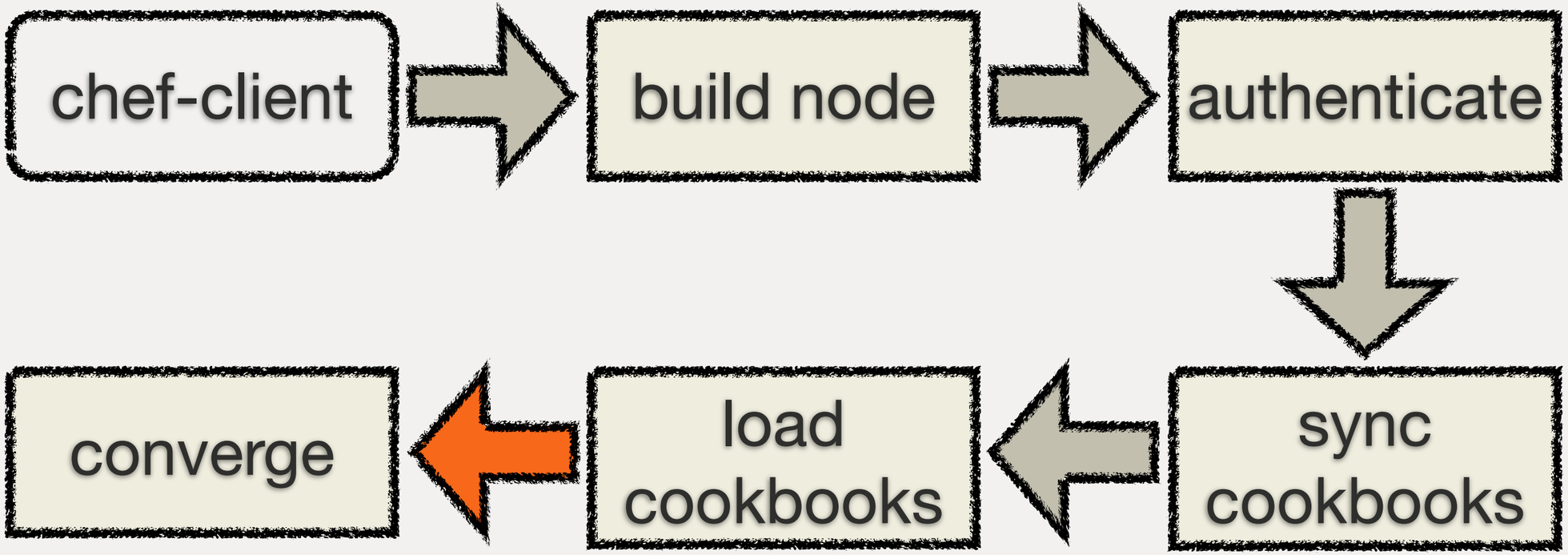

chef-client
build node

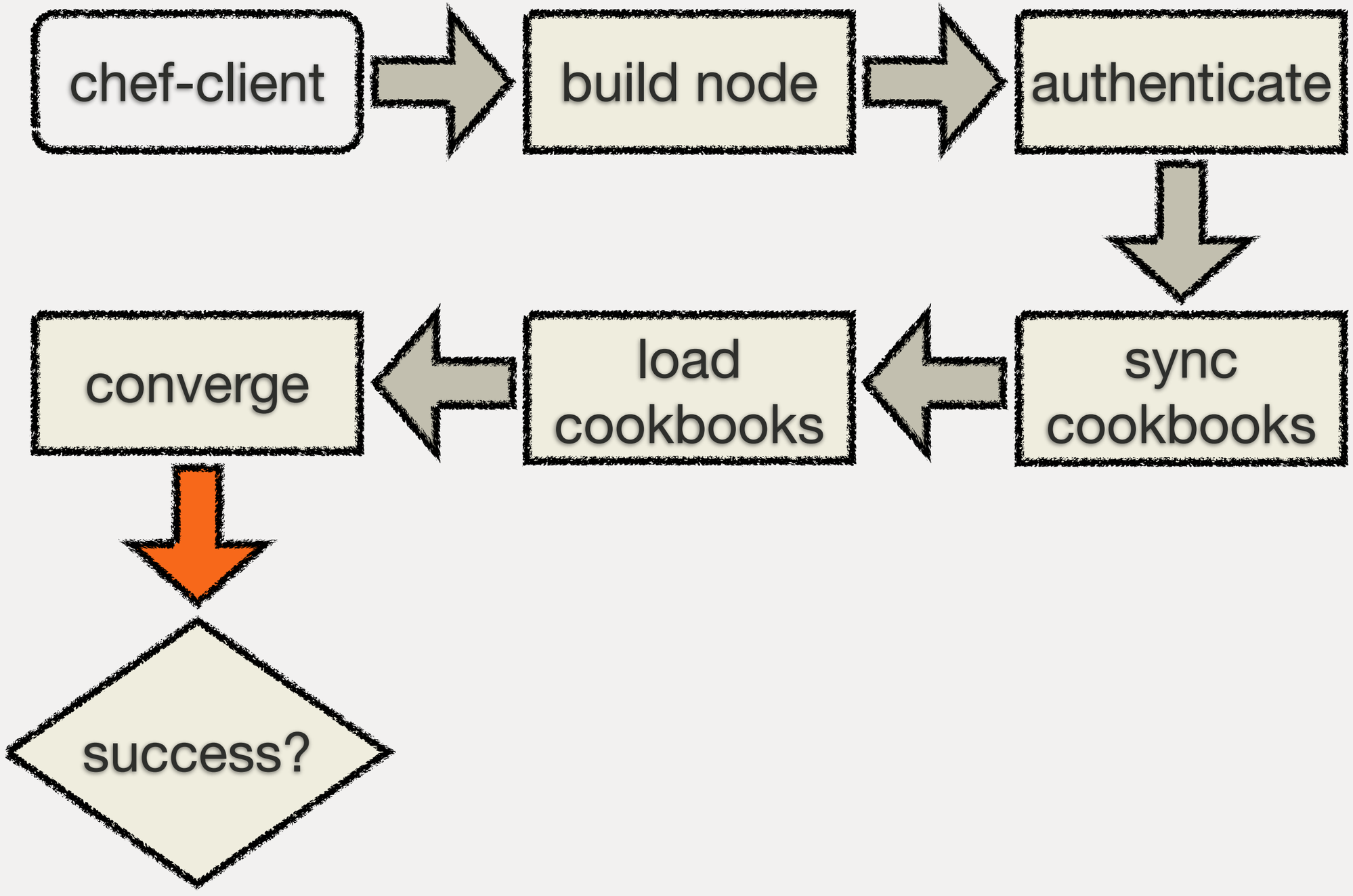

chef-client
build node
Ohai!
node_name
platform
platform_version

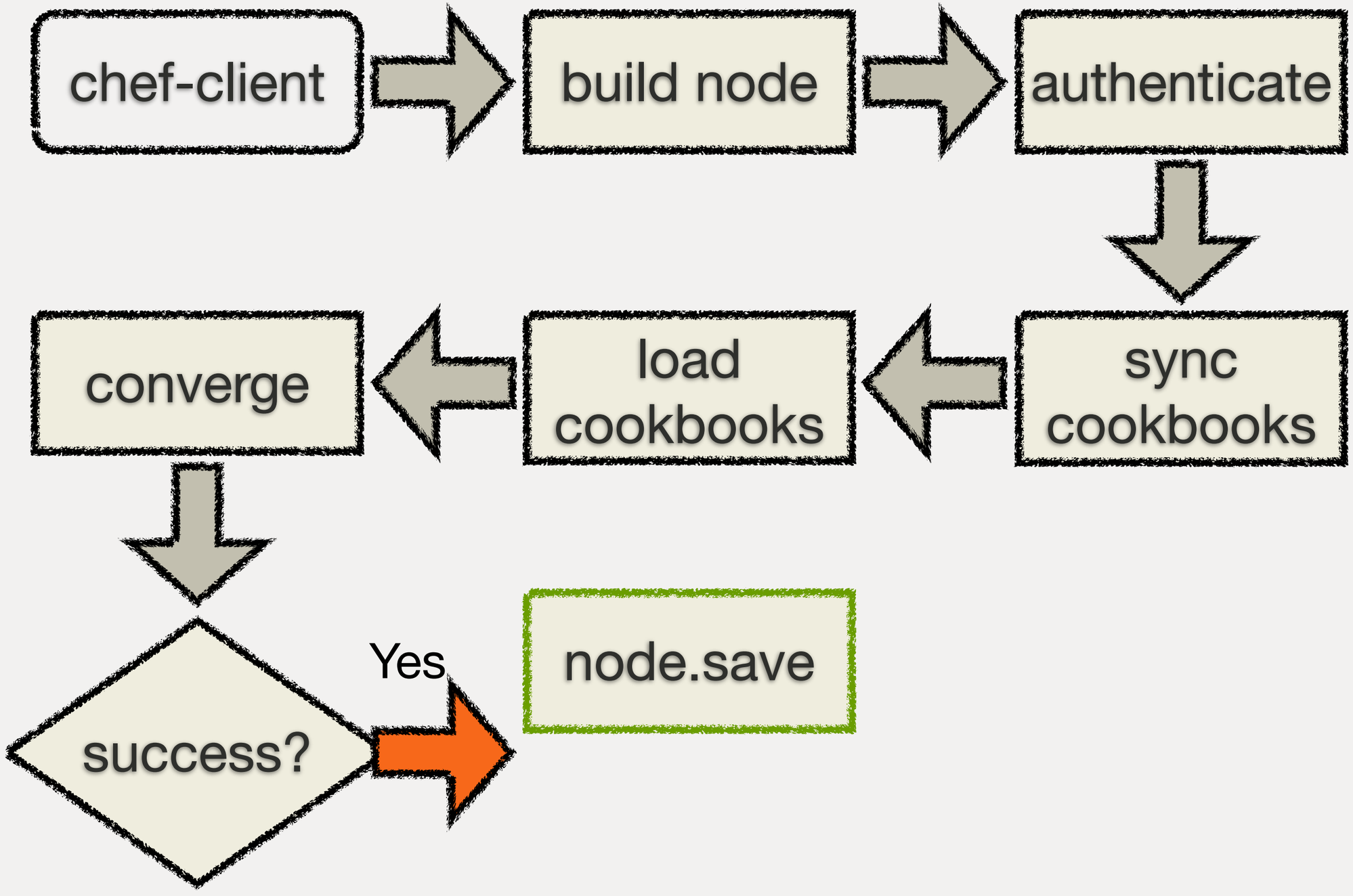
chef-client
build node
authenticate

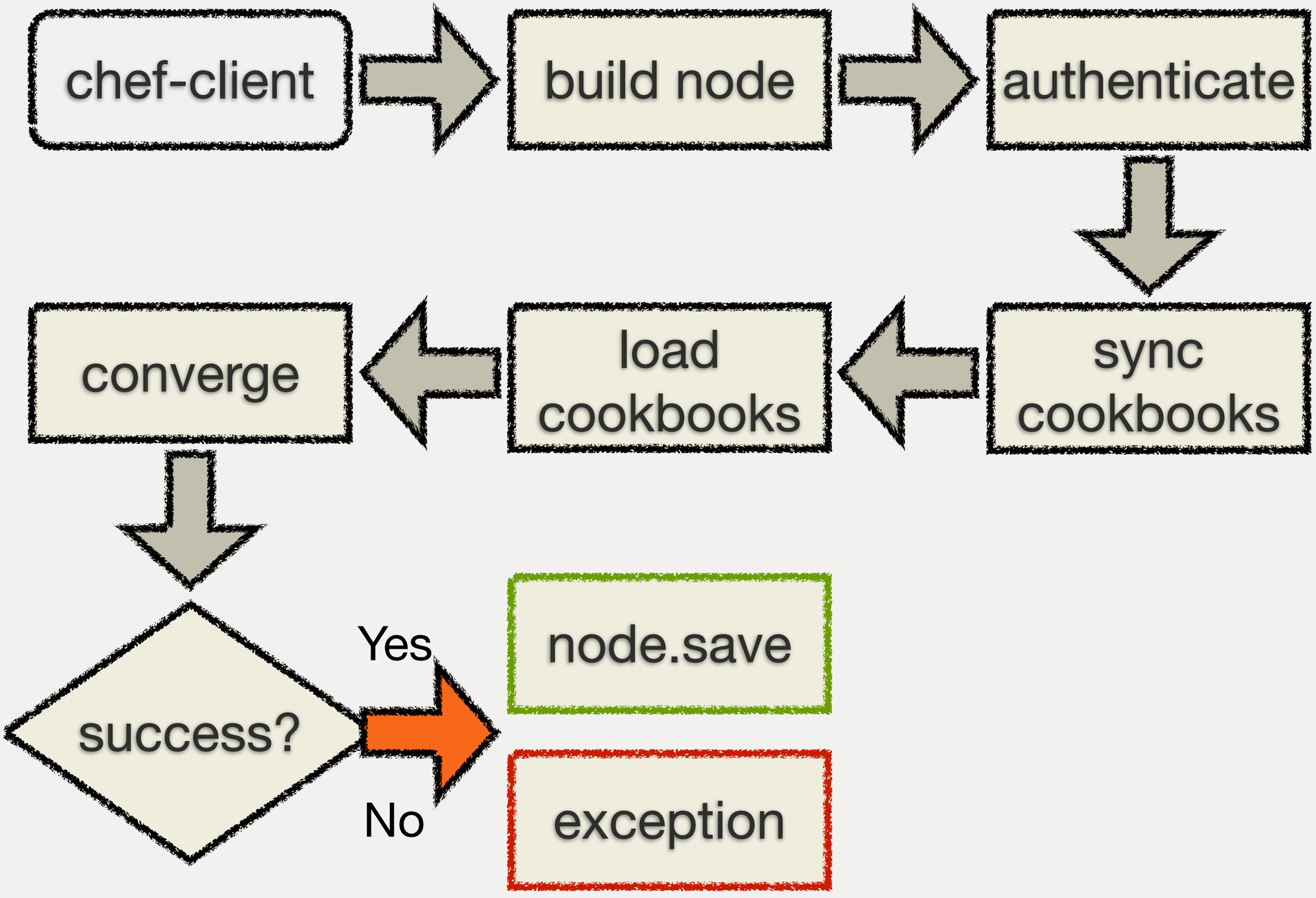

chef-client
build node
authenticate
sync cookbooks
expanded run list (recipes)

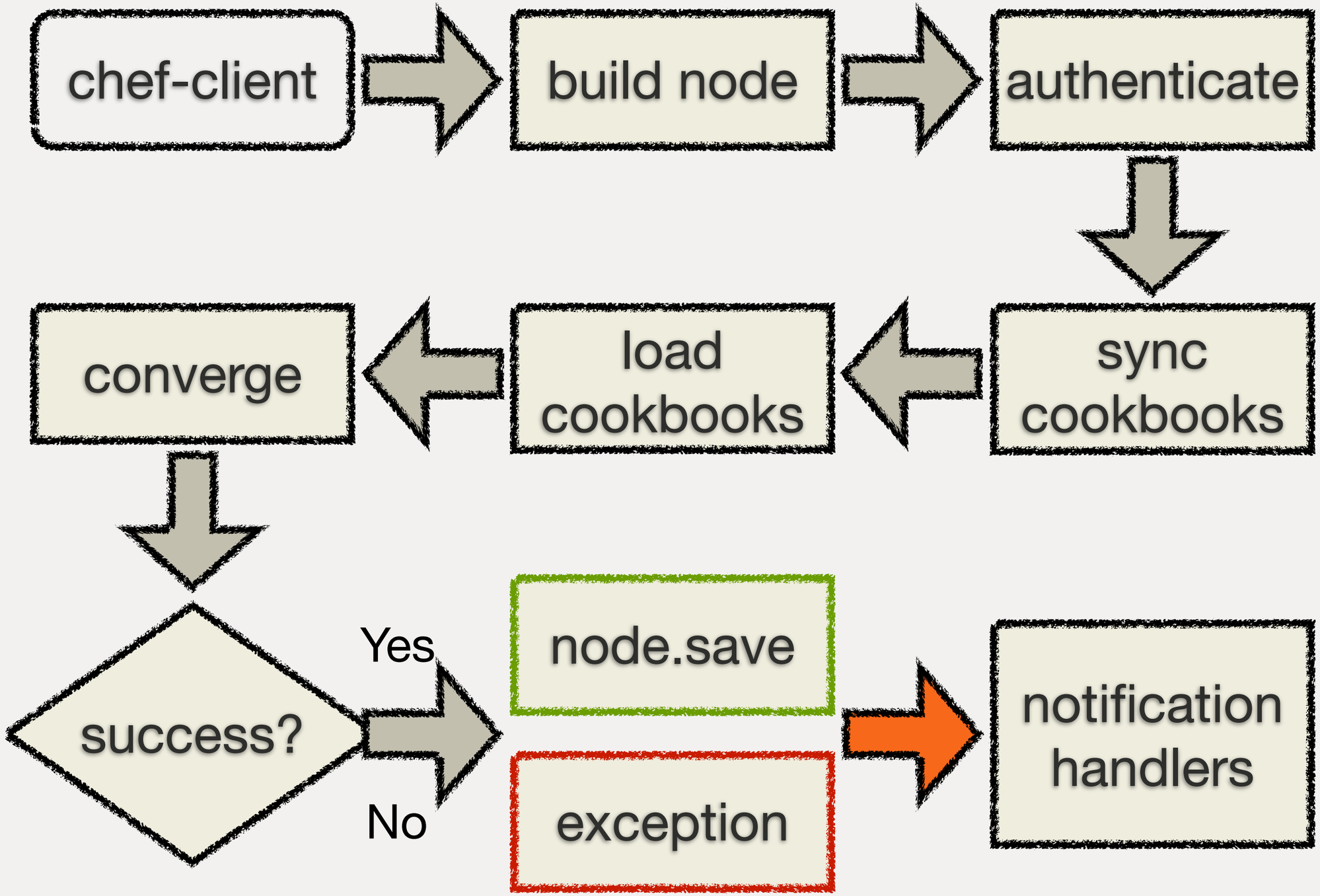

chef-client
build node
authenticate
sync cookbooks
load cookbooks

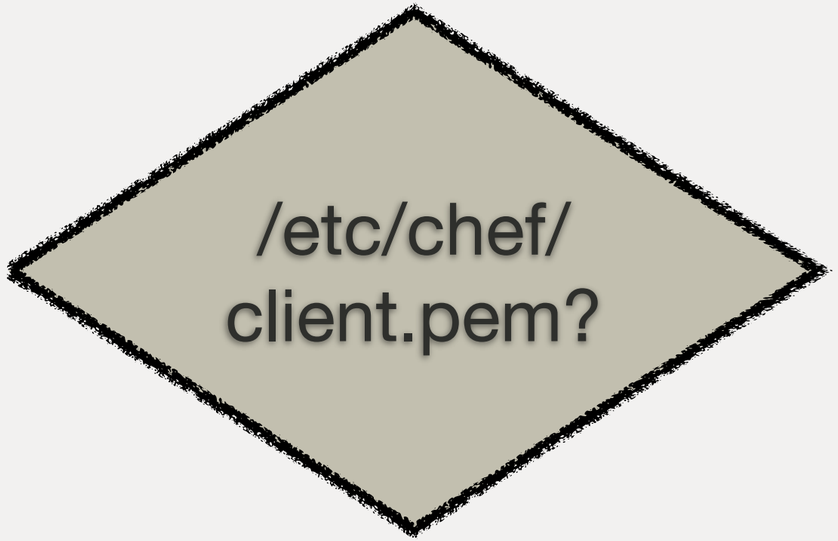

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge

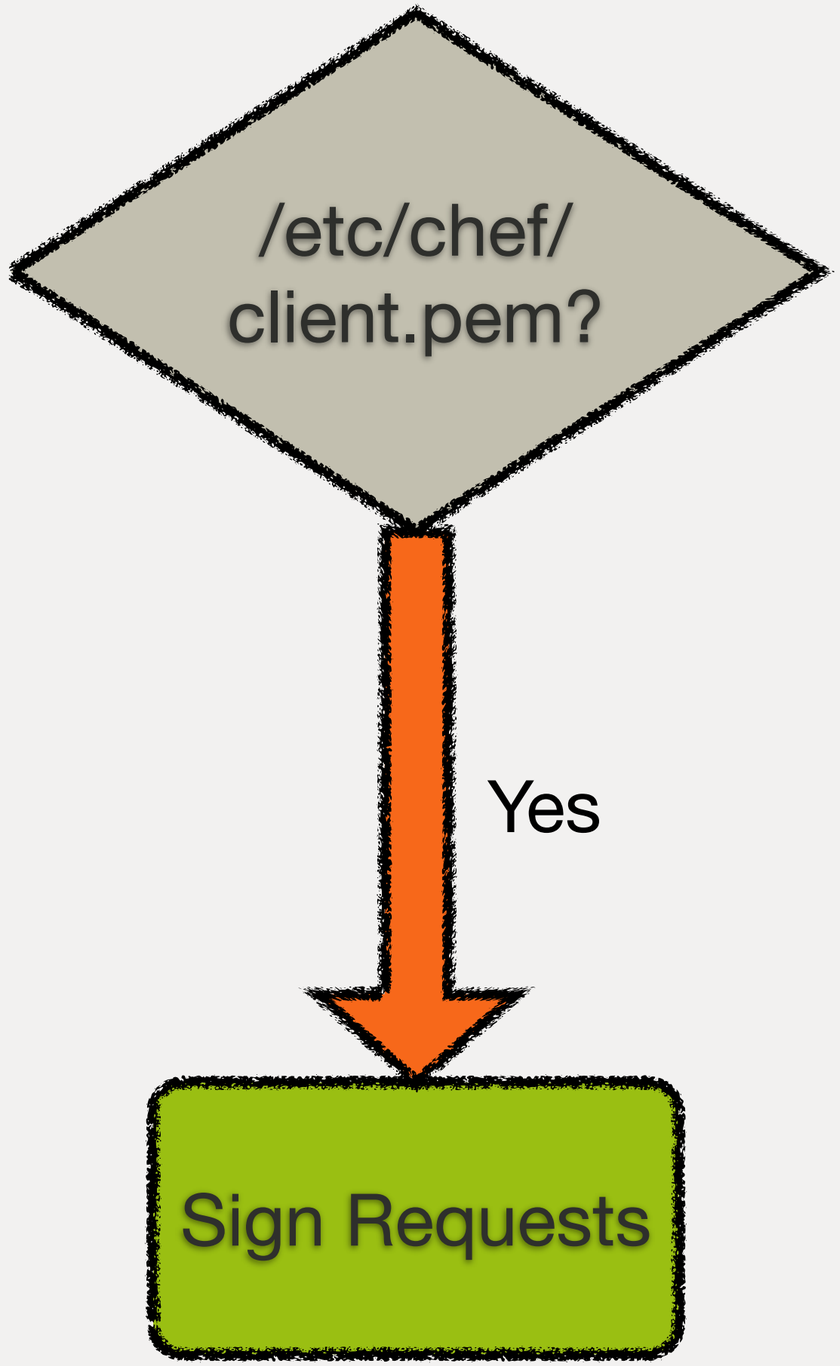

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?

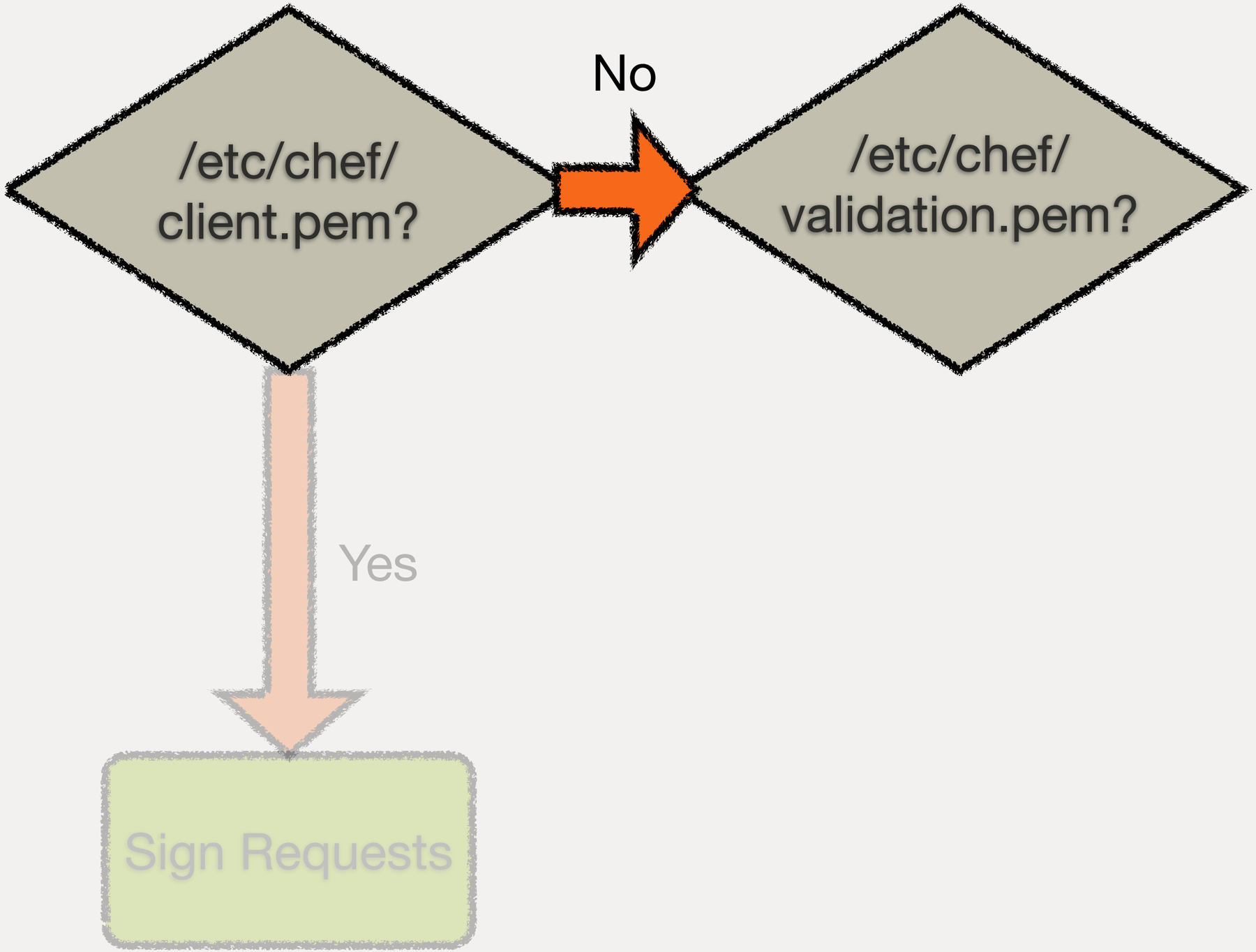

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
node.save

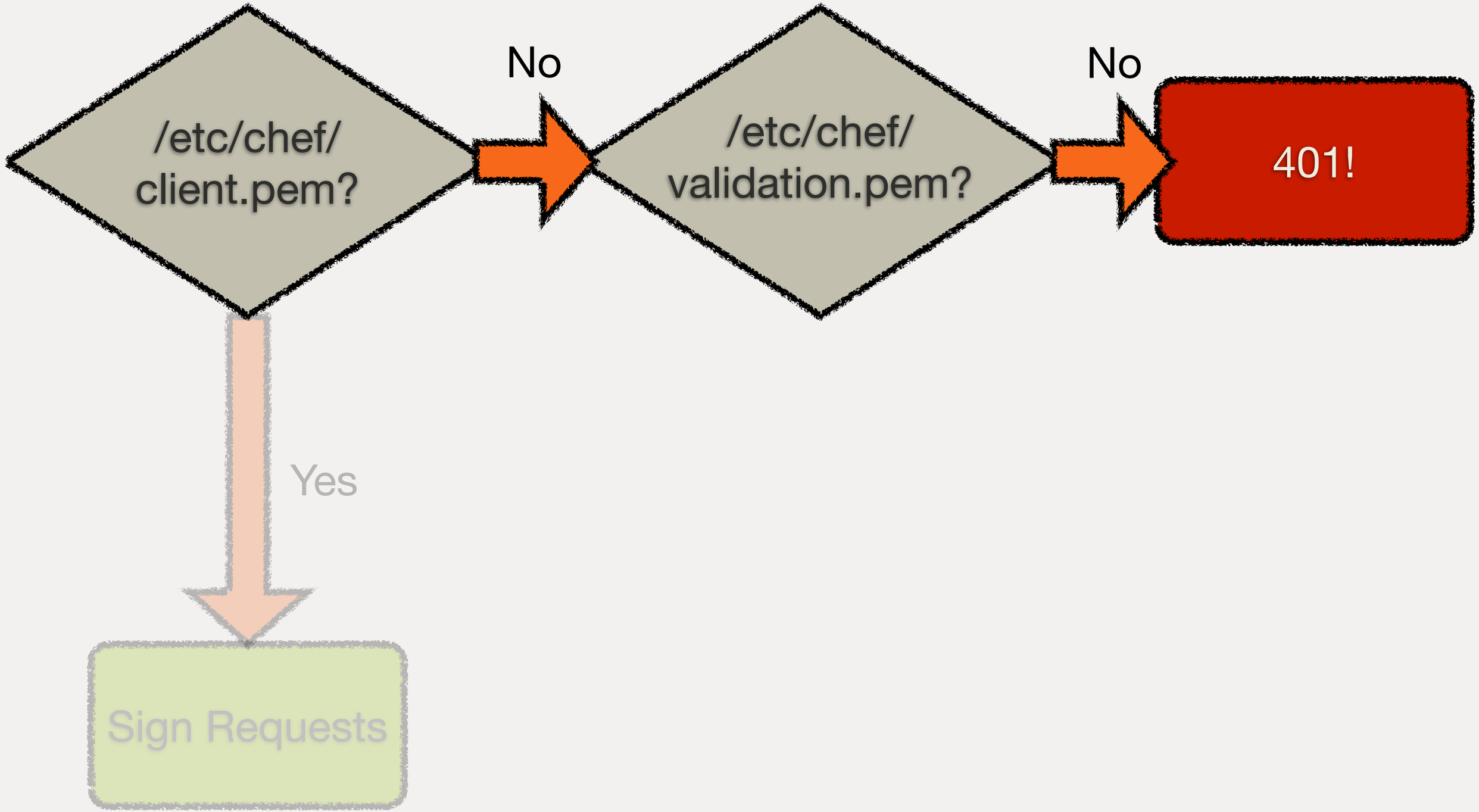

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
No
node.save
exception

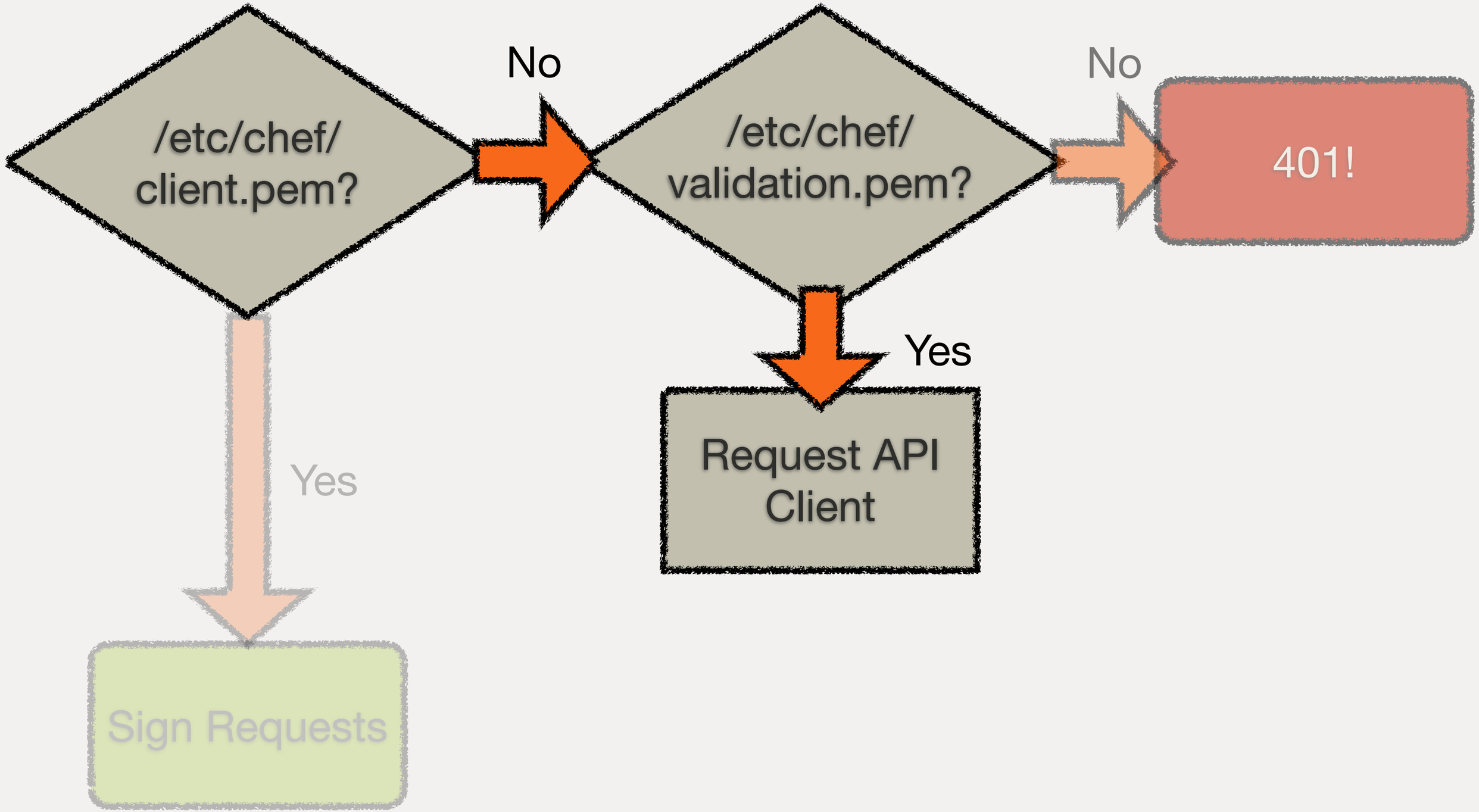

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
No
node.save
exception
notification handlers

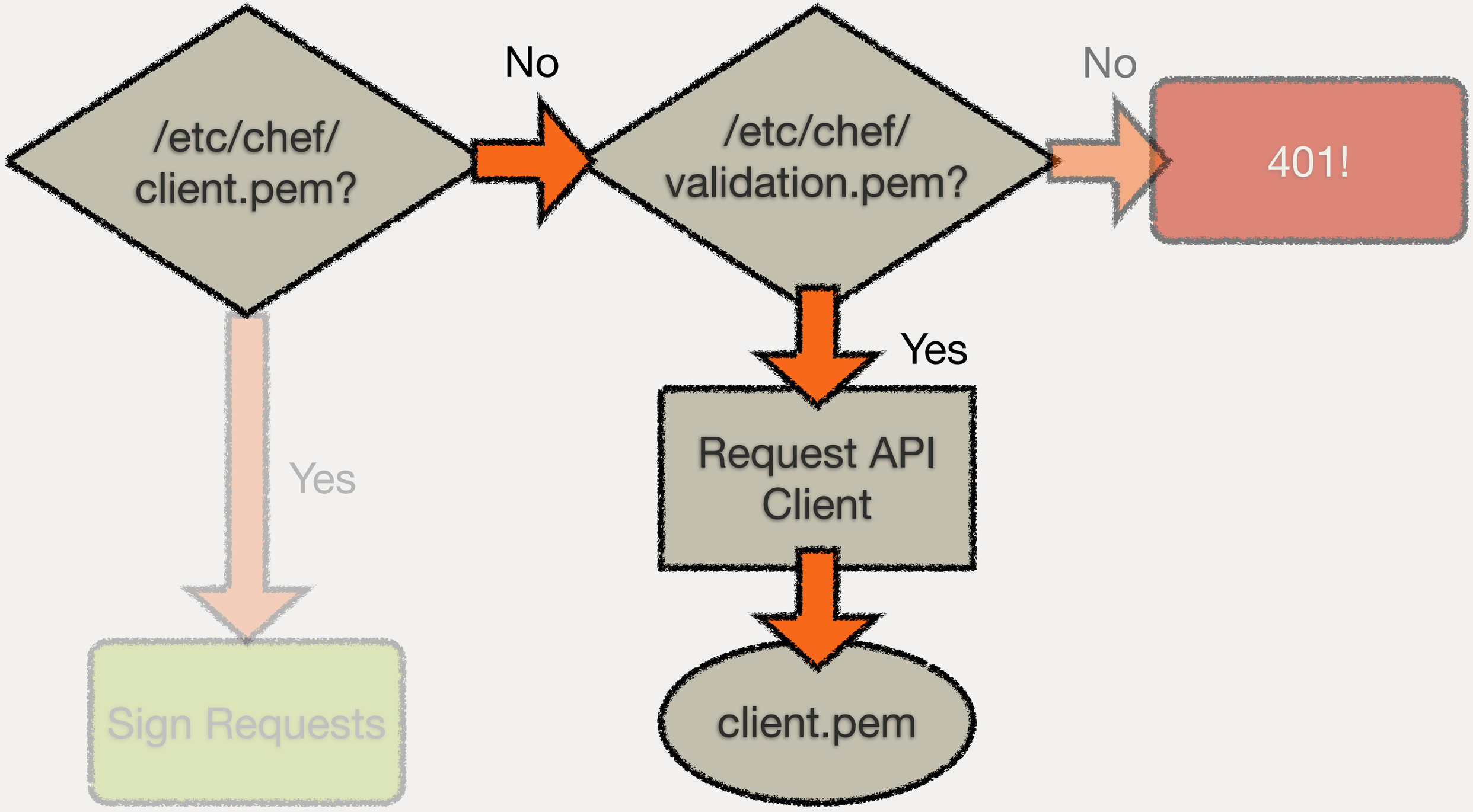

/etc/chef/
client.pem?

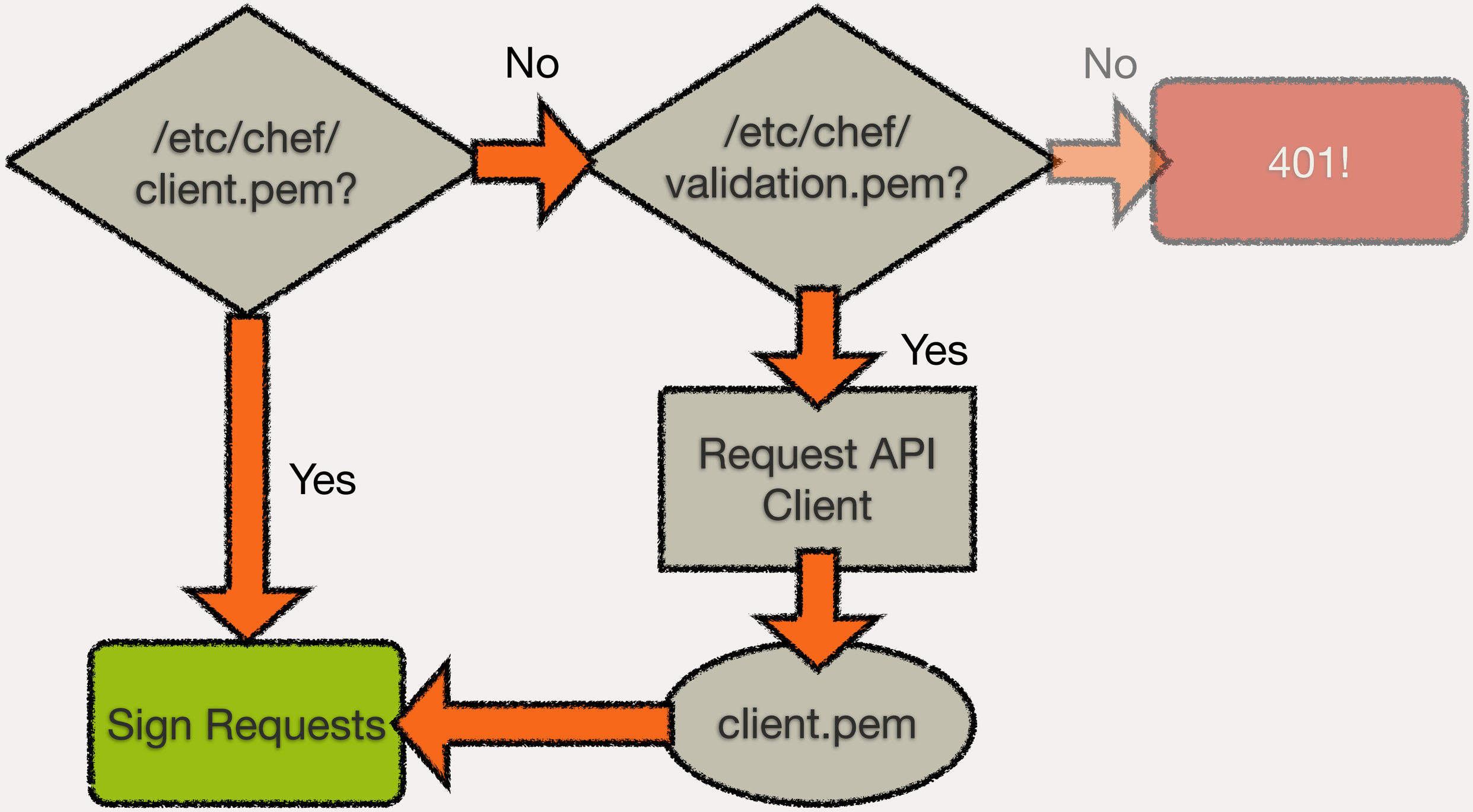

/etc/chef/
client.pem?
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Request API
Client
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Request API
Client
client.pem
Yes
Sign Requests


/etc/chef/
client.pem?
No
/etc/chef/
validation.pem?
No
401!
Yes
Request API
Client
Yes
client.pem
Sign Requests

Questions

- What are the steps in a chef client run?
- How does a new server get a private key with which to authenticate requests?

Introducing the Node object

Attributes & Search

Objectives

- Understand what the Node object represents in Chef
- Learn how to list the Nodes in an organization
- Know how to show details about a Node
- Understand what Node Attributes are
- Know how to add an attribute to a Node
- Know how to retrieve a node attribute directly, and via search

What is the Node object

- A Node object in Chef represents anything that needs to be configured.
- For now, just think of all the servers you need to configure as Node
- When you are writing Cookbooks, the Node object is almost always available to you.

Exercise: Listing nodes

```
$ knife node list
```

```
target1.local
```

Each node must have a unique name

- Every node must have a unique name
- Chef defaults to the *Fully Qualified Domain Name* of the server.

Exercise: Showing details about a node

```
$ knife node show target1.local
```

```
Node Name:   target1.local
Environment: _default
FQDN:        target1.local
IP:          50.19.162.97
Run List:
Roles:
Recipes:
Platform:    ubuntu 12.04
Tags:
```

What is the Node object

- Nodes are made up of Attributes
  - Many are discovered **automatically** (platform, ip address, number of CPUs)
  - Attributes can be added directly to a node
  - Many other objects in Chef can also add Node attributes (Cookbooks, Roles and Environments)

Exercise: Showing all the attributes of a node

```
$ knife node show target1.local -l
```

```
Node Name:   target1.local
Environment: _default
FQDN:        target1.local
IP:          50.19.162.97
Run List:
Roles:
Recipes:
Platform:    ubuntu 12.04
Attributes:
tags:  []

Default Attributes:

Override Attributes:

Automatic Attributes (Ohai Data):
block_device:
  loop0:
    removable:  0
    size:       0
  loop1:
    removable:  0
    size:       0
```

Exercise: Showing the raw format of the node

```
$ knife node show target1.local -Fj
```

```
{
  "name": "target1.local",
  "chef_environment": "_default",
  "run_list": [

  ],
  "normal": {
    "tags": [

    ]
  }
}
```

Exercise: Add an attribute to a node

```
$ knife node edit target1.local
```

knife-edit-542374099379860.js

```
{
  "name": "target1.local",
  "chef_environment": "_default",
  "normal": {
    "tags": [

    ]
  },
  "run_list": [

  ]
}
```

Exercise: Add an attribute to a node

Add a “company” attribute, **save** and **close** your editor

knife-edit-301406175342915.js

```
{
  "name": "target1.local",
  "chef_environment": "_default",
  "normal": {
    "company": "opscode",
    "tags": [

    ]
  },
  "run_list": [

  ]
}
```

What happens now?

- The changed Node data is sent to the Chef Server
- It is stored in the database
- It is indexed for search

Exercise: Show only the new company attribute

```
$ knife node show target1.local -a company
```

```
company:  opscode
```

Exercise: Use search to find the same data

```
$ knife search node “*:*” -a company
```

```
1 items found

company:  opscode
id:       target1.local
```

Questions

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

Writing an Apache cookbook

Packages, Cookbook Files, and Services

Objectives

- Understand what a **cookbook** is
- Know how to create a new cookbook
- Understand what a **recipe** is
- Understand how to use the **package**, **service**, and **cookbook_file** resources
- Know how to **upload a cookbook** to the Chef Server
- Understand what a **run list** is, and how to set it for a node via knife
- How to read 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 Apache
- Start the service, and make sure it will start when the machine boots
- Write out the home page

Exercise: Create a new cookbook

```
$ knife cookbook create apache
```

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

Exercise: Check out what just got created

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

```
total 23
drwxr-xr-x  12 opscode  pandas  442 Oct 22 18:57 .
drwxr-xr-x   4 opscode  pandas  136 Oct 22 18:57 ..
-rw-r--r--   1 opscode  pandas  413 Oct 22 18:57 CHANGELOG.md
-rw-r--r--   1 opscode  pandas   88 Oct 22 18:57 README.md
drwxr-xr-x   2 opscode  pandas   68 Oct 22 18:57 attributes
drwxr-xr-x   2 opscode  pandas   68 Oct 22 18:57 definitions
drwxr-xr-x   3 opscode  pandas  102 Oct 22 18:57 files
drwxr-xr-x   2 opscode  pandas   68 Oct 22 18:57 libraries
-rw-r--r--   1 opscode  pandas  250 Oct 22 18:57 metadata.rb
drwxr-xr-x   2 opscode  pandas   68 Oct 22 18:57 providers
drwxr-xr-x   3 opscode  pandas  102 Oct 22 18:57 recipes
drwxr-xr-x   2 opscode  pandas   68 Oct 22 18:57 resources
drwxr-xr-x   3 opscode  pandas  102 Oct 22 18:57 templates
```

Exercise: Open the default apache recipe in your editor

```
$ vi cookbooks/apache/recipes/default.rb
```

```
#
# Cookbook Name:: apache
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

```

Recipe Naming

- The "default.rb" recipe for a given cookbook is referred to by the name of the cookbook (*apache*)
- If we added another recipe to this cookbook named "mod_ssl.rb", we would refer to it as *apache::mod_ssl*

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

Add the following to cookbooks/apache/recipes/default.rb

```
# All rights reserved - Do Not Redistribute
#

package "apache2" do
  action :install
end

```

Chef Resources

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

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

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

Chef Resources

- Have a type.

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

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

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

Chef Resources

- Have a type.
- Have a name.

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

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

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

Chef Resources

- Have a type.
- Have a name.
- Have parameters.

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

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

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

Chef Resources

- Have a type.
- Have a name.
- Have parameters.
- Take action to put the resource in the declared 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 in the declared 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
```

OPSCODE

So the resource we just wrote...

- Is a **package** resource
- Whose **name** is *apache2*
- With an **install** action

```
package "apache2" do
  action :install
end
```

Notice we didn't say how to install the package

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

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

Add the following to cookbooks/apache/recipes/default.rb

```
package "apache2" do
  action :install
end

service "apache2" do
  action [ :start, :enable ]
end
```

OPSCODE

So the resource we just wrote...

- Is a **service** resource
- Whose **name** is *apache2*
- With two actions: **start** and **enable**

```
service "apache2" do
  action [ :start, :enable ]
end
```

Order Matters

- The order you write resources in a recipe is the order they will be executed in

Exercise: Add a cookbook_file resource to copy the home page in place

Add the following to cookbooks/apache/recipes/default.rb

```
service "apache2" do
  action [ :start, :enable ]
end

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

```

OPSCODE

So the resource we just wrote...

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

```
18 cookbook_file "/var/www/index.html" do
19   source "index.html"
20   mode "0644"
21 end
```

OPSCODE

Best Practice: Omit the action if it is the default

- Has no **action**!
- If you omit the action in Chef, we default to the most common positive action. In this case, it is the :create action.

```
18 cookbook_file "/var/www/index.html" do
19   source "index.html"
20   mode "0644"
21 end
```

OPSCODE

Full contents of the apache recipe

default.rb (~/src/sandbox/...okbooks/apache/recipes) – VIM

```
#
# Cookbook Name:: apache
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

package "apache2" do
  action :install
end

service "apache2" do
  action [ :start, :enable ]
end

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

RULE THE CLOUD

Question!

- Using what we have learned so far, can we make the recipe shorter?

Best Practice: Omit the default action

- If the only action you need from a resource is the default action - omit it from the recipe

Exercise: Add index.html to your cookbooks files/default directory

Add the following to cookbooks/apache/files/default/index.html

```
1 <html>
2   <body>
3     <h1>Hello world</h1>
4   </body>
5 </html>
~
```

What's with the 'default' subdirectory?

- Chef allows you to select the most appropriate file (or template) within a cookbook according to the platform of the node it is being executed on
  - node name (foo.bar.com)
  - platform-version (redhat-6.2)
  - platform-major (redhat-6)
  - platform
  - default
- 98% of the time, you will just use **default**

Exercise: Upload the cookbook

```
$ knife cookbook upload apache
```

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

Exercise: Add the apache recipe to your test nodes run list

```
$ knife node edit target1.local
```

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "tags": [
7
8      ]
9    },
10   "run_list": [
11
12   ]
13 }
14
```

Exercise: Add the apache recipe to your test nodes run list

Add recipe[apache] to the run_list, **save** and **close.**

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "tags": [
7
8      ]
9    },
10   "run_list": [
11     "recipe[apache]"
12   ]
13 }
14
```

The Run List

- Run lists specify what recipes or roles the node should run, along with the order they should be run in
- Run lists are represented by an array
- Recipes are specified by "**recipe[*name*]**"
- Roles are specified by "**role[*name*]**"

Exercise: Run the chef-client on your test node

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache]]
INFO: Run List expands to [apache]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache]
INFO: Storing updated cookbooks/apache/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/apache/CHANGELOG.md in the cache.
INFO: Storing updated cookbooks/apache/metadata.rb in the cache.
INFO: Storing updated cookbooks/apache/README.md in the cache.
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: package[apache2] installed version 2.2.22-1ubuntu1
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: cookbook_file[/var/www/index.html] backed up to /var/chef/backup/var/www/index.html.chef-20121023020345
INFO: cookbook_file[/var/www/index.html] owner changed to 0
INFO: cookbook_file[/var/www/index.html] group changed to 0
INFO: cookbook_file[/var/www/index.html] mode changed to 644
INFO: cookbook_file[/var/www/index.html] created file /var/www/index.html
INFO: Chef Run complete in 13.761588783 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Verify that the home page works

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

Congratulate yourself!

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

Reading the output of a chef-client run

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache]]
INFO: Run List expands to [apache]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
```

- We tell you the node's Run List
- The expanded Run List is the complete list, after nested roles are expanded

Reading the output of a chef-client run

```
INFO: Loading cookbooks [apache]
INFO: Storing updated cookbooks/apache/recipes/default.rb in the
cache.
INFO: Storing updated cookbooks/apache/CHANGELOG.md in the cache.
INFO: Storing updated cookbooks/apache/metadata.rb in the cache.
INFO: Storing updated cookbooks/apache/README.md in the cache.
```

- We start loading the cookbooks in the order specified by the run list
- We download any files we are missing from the server

Reading the output of a chef-client run

```
INFO: Processing package[apache2] action install
(apache::default line 10)
INFO: package[apache2] installed version
2.2.22-1ubuntu1
```

- We are checking to see if the package apache2 is installed
- It was not, and so we installed version 2.2.22-1ubuntu1 (yours may be different)

Reading the output of a chef-client run

```
INFO: Processing service[apache2] action start
(apache::default line 14)
INFO: Processing service[apache2] action enable
(apache::default line 14)
```

- We check to see if apache2 is already started - and it is, so we do nothing
- We check to see if apache2 is already enabled to run at boot - and it is, so we do nothing

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

Reading the output of a chef-client run

```
INFO: Processing cookbook_file[/var/www/index.html] action
create (apache::default line 18)
INFO: cookbook_file[/var/www/index.html] backed up to /var/
chef/backup/var/www/index.html.chef-20121023020345
INFO: cookbook_file[/var/www/index.html] mode changed to 644
INFO: cookbook_file[/var/www/index.html] created file /var/
www/index.html
```

- We check to see if we need to create the index.html file
- There is already one in place, whose contents are different than ours, so we back it up
- We also set the permissions appropriately

Reading the output of a chef-client run

```
INFO: Chef Run complete in 13.761588783 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

- We see Chef Run complete, with the time it took for the run to finish
- Report and exception handlers are now run

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache]]
INFO: Run List expands to [apache]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Chef Run complete in 1.216262701 seconds
INFO: Removing cookbooks/apache/files/default/index.html from the cache; it is no longer
needed by chef-client.
INFO: Running report handlers
INFO: Report handlers complete
```

Questions

- What is 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?
- What do the "Processing" lines in the chef-client output mean?

Writing an MOTD Cookbook

Cookbook Attributes, Cookbook Dependencies, Attribute Precedence, and ERb Templates

OPSCODE

RULE THE CLOUD

Chef

Objectives

- Understand Cookbook Attribute files
- Learn how to use ERB Templates in Chef
- Understand Attribute Precedence
- Learn about Cookbook Metadata
- Know how to specify cookbook dependencies
- Practice the cookbook creation, upload, and test loop

The Problem and the Success Criteria

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

We have a small problem...

- We added a 'company' attribute earlier
- But we don't have an attribute that reflects whether we are in or out of scope for PCI Compliance

OPSCODE

Well factored Cookbooks only contain the information relevant to their domain

- We could add a node attribute for PCI Compliance, but that will become very tiresome at scale
- We know we will likely have other things related to PCI (security settings, for example)
- The best thing to do is create a PCI cookbook, and add our attribute there

RULE THE CLOUD

Exercise: Create a cookbook named 'pci'

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

Exercise: Create a cookbook named 'pci'

```
$ knife cookbook create pci
```

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

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

```
$ vi cookbooks/pci/attributes/default.rb
```

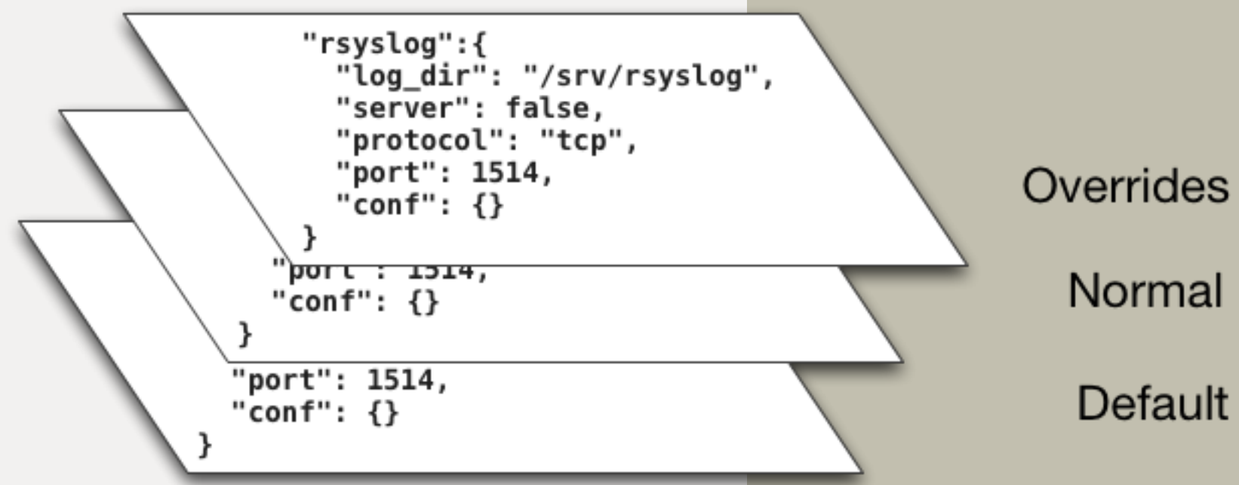
```
default['pci']['in_scope'] = false
~
~
```

- Creates a new Node attribute: node[‘pci’][‘in_scope’]
- Sets the value to the Ruby *false* literal

OPSCODE | Chef

Node Attributes have four levels of precedence

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



Best Practice: Always use 'default' attributes in your cookbooks

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

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

Exercise: Upload the PCI cookbook

```
$ knife cookbook upload pci
```

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

Exercise: Create a cookbook named 'motd'

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

Exercise: Create a cookbook named 'motd'

```
$ knife cookbook create motd
```

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

Exercise: Open the default recipe in your editor

```
#
# Cookbook Name:: motd
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

Exercise: Open the default recipe in your editor

```
$ vi cookbooks/motd/recipes/default.rb
```

```
#
# Cookbook Name:: motd
# Recipe:: default
#
# Copyright 2012, 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 /etc/motd.tail

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

OPSCODE | Chef

The template[/etc/motd.tail] resource

default.rb (~/src/sandbox/...cookbooks/motd/recipes) – VIM

```
#
# Cookbook Name:: motd
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

template "/etc/motd.tail" do
  source "motd.tail.erb"
  mode "0644"
end
```

Exercise: Open motd.tail.erb in your Editor

```
$ vi cookbooks/motd/templates/default/motd.tail.erb
```

```
1 This server is property of <%= node['company'] %>
2 <% if node['pci']['in_scope'] -%>
3 This server is in-scope for PCI compliance
4 <% end -%>
~
```

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

Exercise: Open motd.tail.erb in your Editor

```
1 This server is property of <%= node['company'] %>
2 <% if node['pci']['in_scope'] -%>
3 This server is in-scope for PCI compliance
4 <% end -%>
~
```

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

Exercise: Open motd.tail.erb in your Editor

```
1 This server is property of <%= node['company'] %>
2 <% if node['pci']['in_scope'] -%>
3 This server is in-scope for PCI compliance
4 <% 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

OPSCODE

Templates are what you will use for almost all your 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

RULE THE CLOUD

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

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

Exercise: Upload the motd cookbook

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

Exercise: Upload the motd cookbook

```
$ knife cookbook upload motd
```

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

Exercise: Add the motd recipe to your test nodes run list

Add recipe[motd] to the run_list, **save** and **close.**

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "tags": [
7
8      ]
9    },
10   "run_list": [
11     "recipe[apache]",
12     "recipe[motd]"
13   ]
14 }
15
```

- **Do** add a comma after recipe[apache]
- **Don't** add a comma after recipe[motd]

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd]
INFO: Storing updated cookbooks/motd/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/motd/CHANGELOG.md in the cache.
INFO: Storing updated cookbooks/motd/metadata.rb in the cache.
INFO: Storing updated cookbooks/motd/README.md in the cache.
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)

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

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

Resource Declaration:
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd]
INFO: Storing updat[illegible]d/recipes/[illegible]rb in the cac[illegible]
INFO: Storing updat[illegible]d/CHANGEL[illegible]he cache.
INFO: Storing updat[illegible]books/motd/metadat[illegible] cache.
INFO: Storing updat[illegible]books/motd/READMF[illegible]ache.
INFO: Processing pa[illegible]pache2] action i[illegible] ([illegible]::default [illegible]0)
INFO: Processing se[illegible]ction [illegible](apa[illegible]efault li[illegible]
INFO: Processing se[illegible]ction [illegible]e (apa[illegible]efault l[illegible])
INFO: Processing co[illegible]File[/var/www[illegible]html] [illegible]create [illegible]e::def[illegible]ne 18)
INFO: Processing te[illegible]/etc/motd.ta[illegible]d::def[illegible]ine 10[illegible]

================[illegible]==========[illegible]==========[illegible]==========[illegible]==========
Error executing act[illegible]eate` on r[illegible] 'template[/[illegible]d.ta[illegible]
================================================================================

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

Resource Declaration:
```

FAIL!

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

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

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

Compiled Resource:
------------------
# Declared in /var/chef/cache/cookbooks/motd/recipes/default.rb:10:in `from_file'

template("/etc/motd.tail") do
  provider Chef::Provider::Template
  action "create"
  retries 0
  retry_delay 2
  path "/etc/motd.tail"
  backup 5
  source "motd.tail.erb"
  cookbook_name "motd"
  recipe_name "default"
  mode "0644"
end

ERROR: Running exception handlers
FATAL: Saving node information to /var/chef/cache/failed-run-data.json
ERROR: Exception handlers complete
FATAL: Stacktrace dumped to /var/chef/cache/chef-stacktrace.out
FATAL: Chef::Mixin::Template::TemplateError: undefined method `[]' for nil:NilClass
```

Stacktraces

- 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.

Scroll up

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

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

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

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

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

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

```
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd]
INFO: Storing updated cookbooks/motd/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/motd/metadata.rb in the cache.
INFO: Storing updated cookbooks/motd/README.md
```

- Can anyone guess why?

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

```
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd]
INFO: Storing updated cookbooks/motd/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/motd/metadata.rb in the cache.
INFO: Storing updated cookbooks/motd/README.md
```

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

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

```
$ vi cookbooks/motd/metadata.rb
```

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

Cookbook Metadata

```
maintainer          "YOUR_COMPANY_NAME"
maintainer_email    "YOUR_EMAIL"
license             "All rights reserved"
description         "Installs/Configures motd"
long_description IO.read(File.join(File.dirname(_
version             "0.1.0"
depends "pci"
```

- The cookbook metadata file contains information about the cookbook
- Some of it is mainly decorative - maintainer data, description, licensing

Cookbook Metadata

```
maintainer        "YOUR_COMPANY_NAME"
maintainer_email  "YOUR_EMAIL"
license           "All rights reserved"
description       "Installs/Configures motd"
long_description  IO.read(File.join(File.dirname(_
version           "0.1.0"
depends "pci"
```

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

Cookbook Attributes are applied for all downloaded cookbooks!

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

Exercise: Upload the motd cookbook

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

Exercise: Upload the motd cookbook

```
$ knife cookbook upload motd
```

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

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci]
INFO: Storing updated cookbooks/pci/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/pci/attributes/default.rb in the cache.
INFO: Storing updated cookbooks/pci/CHANGELOG.md in the cache.
INFO: Storing updated cookbooks/pci/metadata.rb in the cache.
INFO: Storing updated cookbooks/pci/README.md in the cache.
INFO: Storing updated cookbooks/motd/metadata.rb in the cache.
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: template[/etc/motd.tail] updated content
INFO: template[/etc/motd.tail] mode changed to 644
INFO: Chef Run complete in 1.760498119 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci]
INFO: Storing updated cookbooks/pci/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/pci/attributes/default.rb in the cache.
INFO: Storing updated cookbooks/pci/CHANGELOG.md in the cache.
INFO: Storing updated cookbooks/pci/metadata.rb in the cache.
INFO: Storing updated cookbooks/pci/README.md in the cache.
INFO: Storing updated cookbooks/motd/metadata.rb in the cache.
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: template[/etc/motd.tail] updated content
INFO: template[/etc/motd.tail] mode changed to 644
INFO: Chef Run complete in 1.760498119 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

OPSCODE | Chef

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbook[ache, motd]
INFO: Storing updated ks/pci/ /defaul n the .
INFO: Storing updated ks/pci tes/def b in che.
INFO: Storing updated s/pc G.md cache
INFO: Storing updated /pc rb cache.
INFO: Storing updated /p D i cache.
INFO: Storing updated eta the cache
INFO: Processing package[ap ion i pache::def line 1
INFO: Processing service[apa tion st ache::defau e 14)
INFO: Processing service[apac ction ena pache::defa ne 14
INFO: Processing cookbook_file r/www/index l] action c apach ault line
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: template[/etc/motd.tail] updated content
INFO: template[/etc/motd.tail] mode changed to 644
INFO: Chef Run complete in 1.760498119 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

WIN!

Exercise: Check your work

```
$ cat /etc/motd.tail
```

```
This server is property of opscode
```

Exercise: Show your test nodes pci attribute

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

```
1 items found

id:   target1.local
pci:
  in_scope:  false
```

Exercise: Set your test node to be in scope for PCI compliance

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "pci": {
7        "in_scope": true
8      },
9      "tags": [
10
11     ]
12   },
13   "run_list": [
14     "recipe[apache]",
15     "recipe[motd]"
16   ]
17 }
18
```

- The node['pci']['in_scope'] attribute is represented as a JSON Hash
- Don't forget the **trailing comma**
- We are setting a **normal** attribute - so it will take precedence

Exercise: Set your test node to be in scope for PCI compliance

```
$ knife node edit target1.local
```

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "pci": {
7        "in_scope": true
8      },
9      "tags": [
10
11     ]
12   },
13   "run_list": [
14     "recipe[apache]",
15     "recipe[motd]"
16   ]
17 }
18
```

- The node['pci']['in_scope'] attribute is represented as a JSON Hash
- Don't forget the **trailing comma**
- We are setting a **normal** attribute - so it will take precedence

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: template[/etc/motd.tail] backed up to /var/chef/backup/etc/motd.tail.chef-20121023023657
INFO: template[/etc/motd.tail] updated content
INFO: template[/etc/motd.tail] mode changed to 644
INFO: Chef Run complete in 1.169403524 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: template[/etc/motd.tail] backed up to /var/chef/backup/etc/motd.tail.chef-20121023023657
INFO: template[/etc/motd.tail] updated content
INFO: template[/etc/motd.tail] mode changed to 644
INFO: Chef Run complete in 1.169403524 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Check your work

```
$ cat /etc/motd.tail
```

```
This server is property of opscode
This server is in-scope for PCI compliance
```

Exercise: Show your test nodes pci attribute

```
$ knife node show target1.local -a pci
```

```
1 items found

id:  target1.local
pci:
  in_scope:  true
```

Congratulations!

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

Objectives

- What goes in a cookbooks 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 most important resources?

Refactoring the Apache Cookbook

Execute, Not If/Only If, Directories, Notifications, Template Variables, and the Chef Docs Site

Objectives

- Know how to use the Execute resource
- Understand how to control idempotence manually with not_if and only_if
- Learn about the Resources page on docs.opscode.com
- Familiarize yourself with the Directory resource
- Understand how to use resource notifications
- Understand what Template Variables are, and how to use them
- Use Ruby variables, loops, and string expansion

The Problem and the Success Criteria

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

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

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

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

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

```
$ vi cookbooks/apache/attributes/default.rb
```

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

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

Exercise: Open the default apache recipe in your editor

```
#
# Cookbook Name:: apache
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

package "apache2" do
  action :install
end

service "apache2" do
  action [ :start, :enable ]
end

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

Exercise: Open the default apache recipe in your editor

```
$ vi cookbooks/apache/recipes/default.rb
```

```
#
# Cookbook Name:: apache
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

package "apache2" do
  action :install
end

service "apache2" do
  action [ :start, :enable ]
end

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

Exercise: Disable the default Apache virtual host with an Execute resouce

```
14 service "apache2" do
15   action [ :start, :enable ]
16 end
17
18 execute "a2dissite default" do
19   only_if do
20     File.symlink?("/etc/apache2/sites-enabled/000-default")
21   end
22   notifies :restart, "service[apache2]"
23 end
24
25 cookbook_file "/var/www/index.html" do
```

- Runs the command “a2dissite default”, but only if the symlink exists
- If the action succeeds, restart Apache

Execute resources are generally not idempotent

- Chef will stop your run if a resource fails
- Most command line utilities are not idempotent - 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

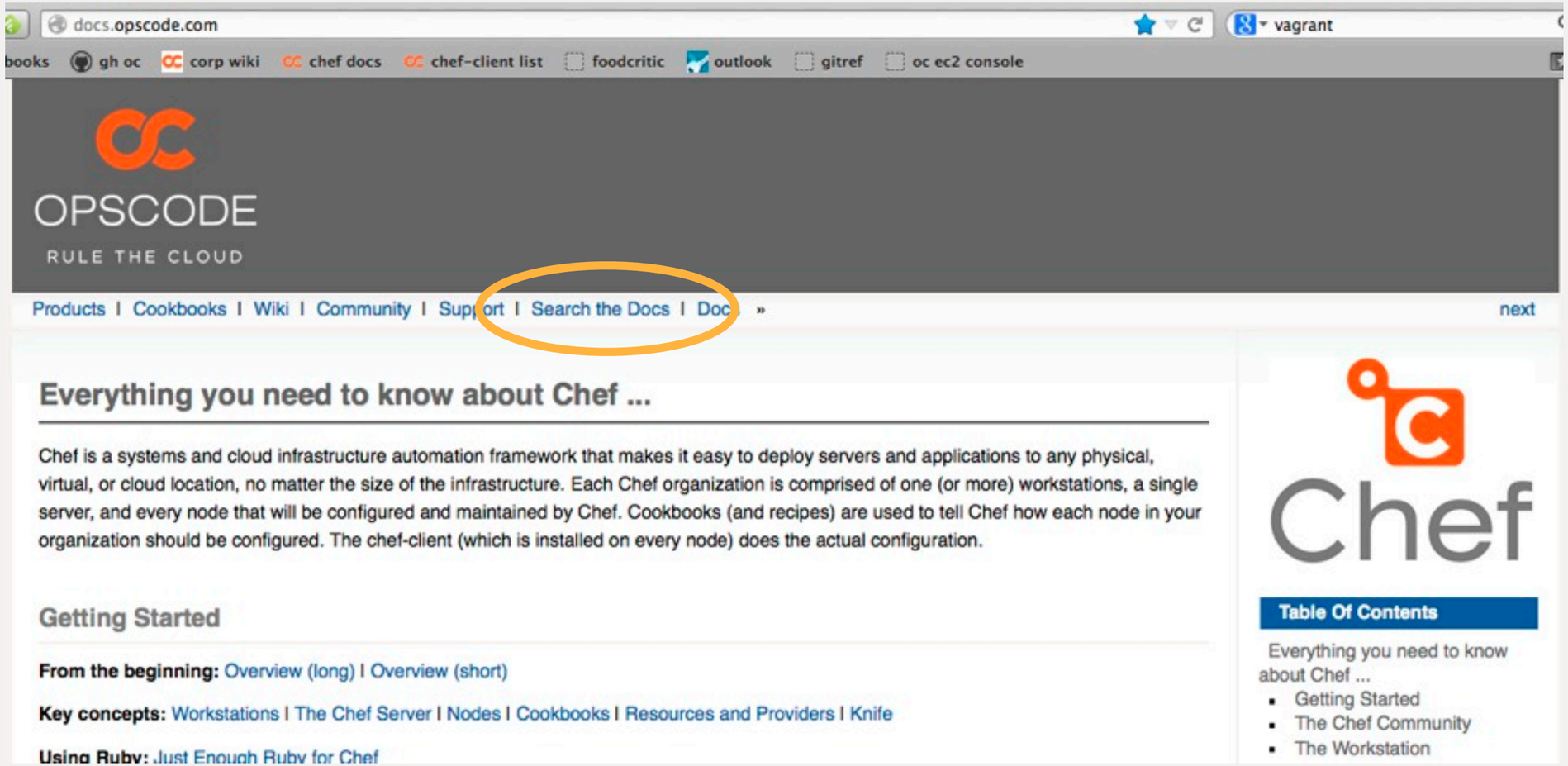
```
only_if do
  File.symlink?("/etc/apache2/sites-enabled/000-default")
end
```

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

Best Practice: The Chef Docs Site

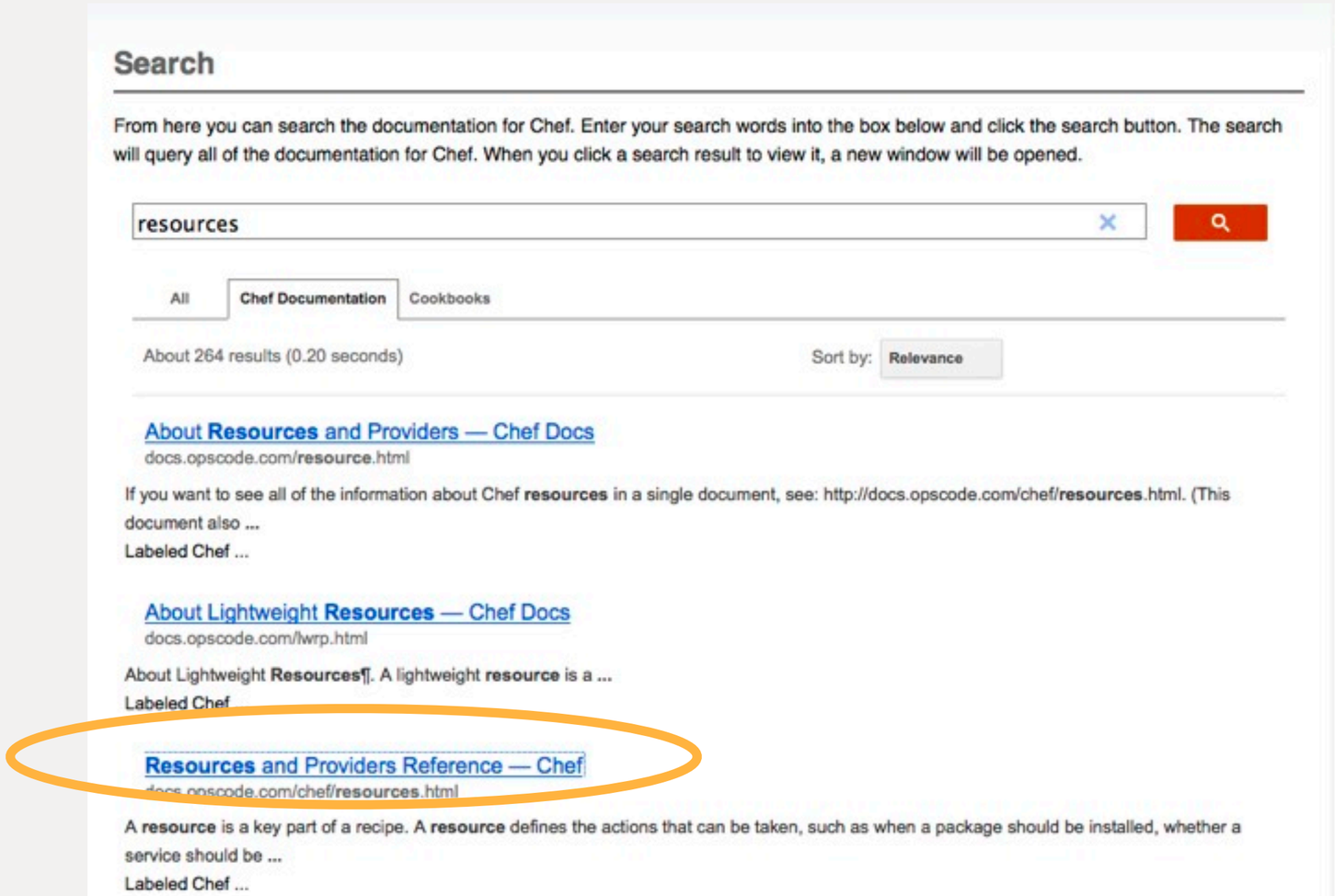
- The Chef Docs Site is the home for all of the documentation about Chef.
  - It is very comprehensive
  - It has a page on every topic
- http://docs.opscode.com
- Let's use the docs to learn more about **not_if** and **only_if**

Exercise: Search for more information about Resources



- Search for “Resources”

OPSCODE | Chef



- Find “Resources and Providers Reference”

The Resources Page

Resources and Providers Reference

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

A resource represents a piece of the system, its current state, and the action that is needed to move it to a desired state. A provider defines the steps that are required to bring that piece of the system from its current to the desired state. An action is de-coupled from the steps required to complete that action, which means that a provider exists for each of the paths that are required to complete the action. This is important because a single action may require different steps, depending on the platform of the system on which the action is being taken.

For example, "install a package" is a single action. To install a package onto various platforms, the steps required for each of those platforms may be different and may require different providers. On a Red Hat or CentOS machine a provider will use the Yum package provider to get the package installed and on a Debian or an Ubuntu machine, a provider will use the APT package installer. One action, two providers, each taking only the steps required to complete that action.

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

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

Chef will look up the provider for the `directory` resource, which happens to be `Chef::Provider::Directory`, call `load_current_resource` to create a new resource called `directory["/tmp/folder"]`, and then, based on the current state of the directory, do the specified action, which in this case is to create a directory called `/tmp/folder`. If the directory already exists, nothing will happen. If the directory was changed in any way, the resource is marked as updated.

This reference describes each of the resources available in the Chef library, including the list of actions available for the resource, the attributes that can be used, the providers that will do the work (and the provider's shortcut resource name), and examples of using each

Chef

Table Of Contents

- Resources and Providers Reference
  - Common Functionality for all Resources
    - Actions
      - Examples
    - Attributes
      - Examples
    - Conditional Execution
      - Attributes
      - Arguments
      - not_if Examples
      - only_if Examples
    - Notifications
      - Notifications Timers
      - Notifies Syntax
        - Examples
      - Subscribes Syntax
        - Examples
    - Relative Paths
      - Examples
    - Windows File Security
      - **Access Control Lists (ACLs)**
      - **Inheritance**
  - Resources
    - cookbook_file
      - Actions

Notifications

```
22   notifies :restart, "service[apache2]"
```

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

Exercise: Iterate over each apache site

```
execute "a2dissite default" do
  only_if do
    File.symlink?("/etc/apache2/sites-enabled/000-default")
  end
  notifies :restart, "service[apache2]"
end

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

- **Delete the cookbook_file resource**
- node['apache']['sites'] is a ruby hash, with keys and values

Exercise: Iterate over each apache site

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

- Calling .each loops over each site

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

- First pass
  - site_name = ‘clowns’
  - site_data = { “port” => 80 }
- Second pass
  - site_name = ‘bears’
  - site_data = { “port” => 81 }

Exercise: Iterate over each apache site

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

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

Exercise: Add a template for the apache sites virtual host configuration

```
25 node['apache']['sites'].each do |site_name, site_data|
26   document_root = "/srv/apache/#{site_name}"
27
28   template "/etc/apache2/sites-available/#{site_name}" do
29     source "custom.erb"
30     mode "0644"
31     variables(
32       :document_root => document_root,
33       :port => site_data['port']
34     )
35     notifies :restart, "service[apache2]"
36   end
```

Template Variables

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

OPSCODE | Chef

Exercise: Add an execute resource to enable the new virtual host

```
28  template "/etc/apache2/sites-available/#{site_name}" do
29    source "custom.erb"
30    mode "0644"
31    variables(
32      :document_root => document_root,
33      :port => site_data['port']
34    )
35    notifies :restart, "service[apache2]"
36  end
37
38  execute "a2ensite #{site_name}" do
39    not_if do
40      File.symlink?("/etc/apache2/sites-enabled/#{site_name}")
41    end
42    notifies :restart, "service[apache2]"
43 end
```

Exercise: Add a directory resource that creates the document_root

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

The directory resource

```
38  execute "a2ensite #{site_name}" do
39    not_if do
40      File.symlink?("/etc/apache2/sites-enabled/#{site_name}")
41    end
42    notifies :restart, "service[apache2]"
43  end
44
45  directory document_root do
46    mode "0755"
47    recursive true
48  end
```

OPSCODE | Chef

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

```
45   directory document_root do
46     mode "0755"
47     recursive true
48   end
49
50   template "#{document_root}/index.html" do
51     source "index.html.erb"
52     variables(
53       :site_name => site_name,
54       :port => site_data['port']
55     )
56     mode "0644"
57   end
58 end
```

OPSCODE | Chef

Don't forget the last "end"

```
50    template "#{document_root}/index.html" do
51      source "index.html.erb"
52      variables(
53        :site_name => site_name,
54        :port => site_data['port']
55      )
56      mode "0644"
57    end
58 end
```

Don't forget the last "end"

```
50    template "#{document_root}/index.html" do
51      source "index.html.erb"
52      variables(
53        :site_name => site_name,
54        :port => site_data['port']
55      )
56      mode "0644"
57    end
58 end
```

See the correct, whole file at https://gist.github.com/2866378

Exercise: Add custom.erb to your templates directory

```
1 <% if @port != 80 -%>
2   Listen <%= @port %>
3 <% end -%>
4
5 <VirtualHost *:<%= @port %>>
6   ServerAdmin webmaster@localhost
7
8   DocumentRoot <%= @document_root %>
9   <Directory />
10    Options FollowSymLinks
11    AllowOverride None
12  </Directory>
13  <Directory <%= @document_root %>>
14    Options Indexes FollowSymLinks MultiViews
15    AllowOverride None
16    Order allow,deny
17    allow from all
18  </Directory>
19 </VirtualHost>
~
```

- Note the two **template variables** are prefixed with an @ symbol
- Our first conditional if!

Exercise: Add custom.erb to your templates directory

```
$ vi cookbooks/apache/templates/default/custom.erb
```

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

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

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

- Note the two **template variables** are prefixed with an @ symbol
- Our first conditional if!

Exercise: Add custom.erb to your templates directory

```
$ vi cookbooks/apache/templates/default/custom.erb
```

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

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

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

- Note the two **template variables** are prefixed with an @ symbol
- Our first conditional if!
- If you are feeling hardcore, type it.
- https://gist.github.com/2866454

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

```
1 <html>
2   <body>
3     <h1>Welcome to <%= node['company'] %></h1>
4     <h2>We love <%= @site_name %></h2>
5     <%= node['ipaddress'] %>:<%= @port %>
6   </body>
7 </html>
~
```

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

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

```
$ vi cookbooks/apache/templates/default/index.html.erb
```

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

- Note the two **template variables** are prefixed with an @ symbol
- https://gist.github.com/2866421

Exercise: Change the cookbooks version number in the metadata

```
4 description        "Installs/
5 long_description IO.read(Fi
6 version            "0.2.0"
```

- Major, Minor, Patch

Exercise: Change the cookbooks version number in the metadata

```
$ vi cookbooks/apache/metadata.rb
```

```
4 description        "Installs/
5 long_description IO.read(Fi
6 version            "0.2.0"
```

- Major, Minor, Patch

Exercise: Upload the apache cookbook

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

Exercise: Upload the apache cookbook

```
$ knife cookbook upload apache
```

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

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing execute[a2dissite default] action run (apache::default line
18)
Site default disabled.
To activate the new configuration, you need to run:
  service apache2 reload
INFO: execute[a2dissite default] ran successfully
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd]]
INFO: Run List expands to [apache, motd]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing execute[a2dissite default] action run (apache::default line
18)
Site default disabled.
To activate the new configuration, you need to run:
  service apache2 reload
INFO: execute[a2dissite default] ran successfully
```

Exercise: Re-run the Chef Client

```
INFO: Processing template[/etc/motd.tail] action
create (motd::default line 10)
INFO: template[/etc/apache2/sites-available/clowns]
sending restart action to service[apache2] (delayed)
INFO: Processing service[apache2] action restart
(apache::default line 14)
INFO: service[apache2] restarted
INFO: Chef Run complete in 5.352761142 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: Processing template[/etc/motd.tail] action
create (motd::default line 10)
INFO: template[/etc/apache2/sites-available/clowns]
sending restart action to service[apache2] (delayed)
INFO: Processing service[apache2] action restart
(apache::default line 14)
INFO: service[apache2] restarted
INFO: Chef Run complete in 5.352761142 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

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 apache virtual hosts")
- Attributes contain the details. ("What virtual hosts should we deploy?")

Questions!

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

Writing a Users cookbook

Users, Groups, Data Bags, Recipe Inclusion and Search

Objectives

- Know what Data Bags are, and how they are used
- Know how to use the User and Group resources
- Learn how to use include_recipe
- Understand the role Search plays in recipes
- Learn how to use Encrypted Data Bags
- Use nested hashes to iterate through data
- Use knife ssh to issue remote commands

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

Best Practice: Data Bags are commonly managed in two ways

- Via files in the data_bag directory of your chef-repo - **do this for anything you are managing by hand**
  - We are using this model in this section
- Created and managed directly via the Chef API by a custom script - **do this for anything you are integrating with**

Exercise: Create a data bag named users

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

```
Created data_bag[users]
```

Exercise: Create a user item in the users data bag

```
$ vi ./data_bags/users/bobo.json
```

```
1 {
2   "id": "bobo",
3   "comment": "Bobo T. Clown",
4   "uid": 2000,
5   "gid": 0,
6   "home": "/home/bobo",
7   "shell": "/bin/bash"
8 }
```

Exercise: Create the data bag item

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

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

Exercise: Create a user item in the users data bag

```
$ vi ./data_bags/users/frank.json
```

```
1 {
2   "id": "frank",
3   "comment": "Frank Belson",
4   "uid": 2001,
5   "gid": 0,
6   "home": "/home/frank",
7   "shell": "/bin/bash"
8 }
```

Exercise: Create the data bag item

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

Exercise: Create the data bag item

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

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

Exercise: Show all the items in users data bag

```
2 items found

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

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

Exercise: Show all the items in users data bag

```
$ knife search users '*:*'
```

```
2 items found

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

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

Exercise: Find Bobo's shell in Chef

```
1 items found

id:      bobo
shell:   /bin/bash
```

Exercise: Find Bobo's shell in Chef

```
$ knife search users 'comment:"Bobo T. Clown"' -a shell
```

```
1 items found

id:      bobo
shell:   /bin/bash
```

Exercise: Create a data bag named groups

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

```
Created data_bag[groups]
```

Exercise: Create a group item in the group data bag

```
$ vi ./data_bags/groups/clowns.json
```

```
1 {
2   "id": "clowns",
3   "gid": 3000,
4   "members": [
5     "bobo",
6     "frank"
7   ]
8 }
```

Exercise: Create the data bag item

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

Exercise: Create the data bag item

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

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

Exercise: Show all the groups in Chef

```
1 items found

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

Exercise: Show all the groups in Chef

```
$ knife search groups '*:*'
```

```
1 items found

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

Exercise: Create a cookbook named 'users'

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

Exercise: Create a cookbook named 'users'

```
$ knife cookbook create users
```

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

Exercise: Open the default recipe in your editor

```
#
# Cookbook Name:: users
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

Exercise: Open the default recipe in your editor

```
$ vi cookbooks/users/recipes/default.rb
```

```
#
# Cookbook Name:: users
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

Exercise: Open the default recipe in your editor

```
# All rights reserved - Do Not Redistribute
#

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

include_recipe "users::groups"
```

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

Exercise: Open the default recipe in your editor

```
# All rights reserved - Do Not Redistribute
#

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

include_recipe "users::groups"
```

- **include_recipe** ensures another recipes resources are complete before we continue
  - Chef will only include each recipe once
- **include_attribute** does the same, but for attribute files

Best Practice: Use include_recipe and include_attribute liberally

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

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

```
$ vi cookbooks/users/recipes/groups.rb
```

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

- 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

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

Exercise: Upload the users cookbook

```
$ knife cookbook upload users
```

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

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

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "pci": {
7        "in_scope": true
8      },
9      "tags": [
10
11     ],
12     "apache": {
13       "sites": {
14       }
15     }
16   },
17   "run_list": [
18     "recipe[apache]",
19     "recipe[motd]",
20     "recipe[users]"
21   ]
22 }
```

Add recipe[users] to the run_list, **save** and **close.**

- **Do** add a comma after recipe[motd]
- **Don't** add a comma after recipe[users]

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Storing updated cookbooks/users/recipes/default.rb in the cache.
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing execute[a2dissite default] action run (apache::default line 18)
INFO: Processing template[/etc/apache2/sites-available/clowns] action create (apache::default line 28)
INFO: Processing execute[a2ensite clowns] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/clowns] action create (apache::default line 45)
INFO: Processing template[/srv/apache/clowns/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/apache2/sites-available/bears] action create (apache::default line 28)
INFO: Processing execute[a2ensite bears] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/bears] action create (apache::default line 45)
INFO: Processing template[/srv/apache/bears/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: Processing user[bobo] action create (users::default line 11)
INFO: user[bobo] created
INFO: Processing user[frank] action create (users::default line 11)
INFO: user[frank] created
INFO: Processing group[clowns] action create (users::groups line 2)
INFO: group[clowns] created
INFO: Chef Run complete in 4.204686792 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Storing updated cookbooks/users/recipes/default.rb in the cache.
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing execute[a2dissite default] action run (apache::default line 18)
INFO: Processing template[/etc/apache2/sites-available/clowns] action create (apache::default line 28)
INFO: Processing execute[a2ensite clowns] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/clowns] action create (apache::default line 45)
INFO: Processing template[/srv/apache/clowns/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/apache2/sites-available/bears] action create (apache::default line 28)
INFO: Processing execute[a2ensite bears] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/bears] action create (apache::default line 45)
INFO: Processing template[/srv/apache/bears/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: Processing user[bobo] action create (users::default line 11)
INFO: user[bobo] created
INFO: Processing user[frank] action create (users::default line 11)
INFO: user[frank] created
INFO: Processing group[clowns] action create (users::groups line 2)
INFO: group[clowns] created
INFO: Chef Run complete in 4.204686792 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Verify the users and groups exist

```
$ sudo cat /etc/passwd
```

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

```
$ sudo cat /etc/group
```

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

Let's review real quick..

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

But wait... what about passwords?

- Sometimes you may need to store sensitive information inside a data bag
- Encrypted Data Bag items allow sensitive information, such as user passwords, to be stored in an encrypted form
- This protects sensitive data from exposure if your Chef Server is compromised

Exercise: Create a data bag named passwords

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

```
Created data_bag[passwords]
```

Exercise: Create a new data bag secret key

```
$ openssl rand -base64 512 | tr -d '\r\n' > secret_key
```

```
(Has no output)
```

Exercise: Create password hashes for your users

```
$ openssl passwd -1 "bobosplaintextpassword"
$ openssl passwd -1 "franksplaintextpassword"
```

```
$1$yEYRpAp8$CgiRRoaX1Y/1eKB5/emss0
$1$bEfGlCWD$yvqNbdLnEj6A36ZTCTvlN0
```

Exercise: Create a userlist item in the password data bag

```
$ vi ./data_bags/passwords/userlist.json
```

```
1 {
2   "id": "userlist",
3   "bobo": "$1$yEYRpAp8$CgiRRoaX1Y/1eKB5/emss0",
4   "frank": "$1$bEfGlCWD$yvqNbdLnEj6A36ZTCTvlN0"
5 }
```

Exercise: Create the data bag item

```
Updated data_bag_item[passwords::userlist]
```

Exercise: Create the data bag item

```
$ knife data_bag from file passwords userlist.json \
  --secret-file secret_key
```

```
Updated data_bag_item[passwords::userlist]
```

Exercise: Show all entries in the userlist item of the passwords data bag

```
id:      userlist
bobo:    ppYTOlWdCzXnZ0G5tuyIwNWTTeJR3FrsMB9X4jPPq{...}
frank:   eiFzAOQdxI7ADkdBD+11SMvYmfwhnDQPSXiBBvwee{...}
```

Exercise: Show all entries in the userlist item of the passwords data bag

```
$ knife data_bag show passwords userlist
```

```
id:      userlist
bobo:    ppYTOlWdCzXnZ0G5tuyIwNWTTeJR3FrsMB9X4jPPq{...}
frank:   eiFzAOQdxI7ADkdBD+11SMvYmfwhnDQPSXiBBvwee{...}
```

Exercise: Show all decrypted entries in the userlist item

```
id:      userlist
bobo:    $1$yEYRpAp8$CgiRRoaX1Y/1eKB5/emss0
frank:   $1$bEfGlCWD$yvqNbdLnEj6A36ZTCTvlN0
```

Exercise: Show all decrypted entries in the userlist item

```
$ knife data_bag show passwords userlist --secret-file secret_key
```

```
id:      userlist
bobo:    $1$yEYRpAp8$CgiRRoaX1Y/1eKB5/emss0
frank:   $1$bEfGlCWD$yvqNbdLnEj6A36ZTCTvlN0
```

Exercise: Open the default recipe in your editor

```
# All rights reserved - Do Not Redistribute
#

decrypted = Chef::EncryptedDataBagItem.load("passwords", "userlist")

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

- We use the same show we just tried with Knife to load the encrypted data_bag item
- All hashes in the item are bound to local variable decrypted

Exercise: Open the default recipe in your editor

```
$ vi cookbooks/users/recipes/default.rb
```

```
# All rights reserved - Do Not Redistribute
#

decrypted = Chef::EncryptedDataBagItem.load("passwords", "userlist")

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

- We use the same show we just tried with Knife to load the encrypted data_bag item
- All hashes in the item are bound to local variable decrypted

Exercise: Open the default recipe in your editor

```
10 decrypted = Chef::EncryptedDataBagItem.load("
11
12 search(:users, "*:*").each do |user_data|
13   user user_data['id'] do
14     comment user_data['comment']
15     uid user_data['uid']
16     gid user_data['gid']
17     home user_data['home']
18     shell user_data['shell']
19     password decrypted[user_data['id']]
20   end
21 end
22
23 include_recipe "users::groups"
```

- The "password" attribute supports adding shadow hash passwords

Iterating over data_bag items

```
# All rights reserved - Do Not Redistribute
#

decrypted = Chef::EncryptedDataBagItem.load("passwords", "userlist")

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

```
"id": "userlist",
"bobo": "$1$yEYRpAp8$CgiRRoaX1Y/1eKB5/emss0",
"frank": "$1$bEfGlCWD$yvqNbdLnEj6A36ZTCTvlN0"
```

- Bobo's password is stored as decrypted['bobo']
- Frank's password is stored as decrypted['frank']

Iterating over data_bag items

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

```
2 items found

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

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

- First pass
  - user_data['id'] = 'bobo'
  - password = decrypted['bobo']
- Second pass
  - user_data['id'] = 'frank'
  - password = decrypted['frank']

Exercise: Upload the users cookbook

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

Exercise: Upload the users cookbook

```
$ knife cookbook upload users
```

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

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Storing updated cookbooks/users/recipes/default.rb in the cache.

================================================================================
Recipe Compile Error in /var/chef/cache/cookbooks/users/recipes/default.rb
================================================================================

Errno::ENOENT
-------------
No such file or directory - file not found '/etc/chef/encrypted_data_bag_secret'

Cookbook Trace:
---------------
  /var/chef/cache/cookbooks/users/recipes/default.rb:10:in `from_file'
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete
INFO: Loading cook          e, motd          users]
INFO: Storing upda    ookbooks/user        /default.      the      e.

================================================================================
Recipe Compile Err          hef/    /coo    s/user    ipes/    ult.rb
================================================================================

Errno::ENOENT
-------------
No such file or directory - file not found '/etc/chef/encrypted_data_bag_secret'

Cookbook Trace:
---------------
  /var/chef/cache/cookbooks/users/recipes/default.rb:10:in `from_file'
```

FAIL!

If you encrypt, you have to decrypt

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

If you encrypt, you have to decrypt

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

```
================================================================
Recipe Compile Error in /var/chef/cache/cookbooks/users/
recipes/default.rb
================================================================

Errno::ENOENT
-------------
No such file or directory - file not found '/etc/chef/
encrypted_data_bag_secret'
```

Exercise: Copy the data bag secret key to the target node

```
$ cat secret_key
$ knife ssh "name:target1.local" -a ipaddress -x ubuntu "echo
  <secret_key> | sudo tee /etc/chef/encrypted_data_bag_secret"
```

```
50.19.162.97
Quq7b95KjPOCDoVlcNpkKpu9oH0OzcdkqE3Kvlq1FuUSFYBJFnjnfHgt9t0Ze5NSoOXSL
f7c4T1qgAptFsNcowAdGhaFa2mLWVJZkmeQj9cadsq7ToiJdC5XGQ3r4dl9bdmvuNcCnd
5ltb2NXZiUDwPwVnlEhXZBJ1nIjbewBCkO5j1hPxMxFhWBXiuNUJxiPwlFdiUjMqr3XSu
FSAUlaqm3fb9woOz2WkhtFR8azLBcmGIGdgQPGkL0XwvRX14nBMVZNHdVcrRAguoz3Ilw
MMHiG0EozdOxeiadXk8cVcQ8M2d96nNGAU3bVZR2CpYxcSjocj5RyXGFxM2dJOXzE4TJu
NXExXEzRNmKrvahHepKnh4dhLCtcUhZYmyNr8UqB987yaAFv61JyZnNS8RNWVb7ZNCZdQ
Bpdl5WIAotPmreEPkwbksjmP5RvLI7MHA0f7JMKWYvAVVbSb4JjvK8mNSPtKIwmPvYaNx
33qnNBRJmA7uwgoJ1gY6hOHnnBP1kOOt8kmUTtcfCpBCvbg47sTQmvk9oHxZEDhOkpxp4
qMPIPd6bdfMqWr178CinRBDNpTv19Z5XBngWA5oFn1Jcu2vXz8YLaMcDechj39yB4kbwU
rDZD9HpZrS0hdmBdHLGqi03wE8Df5oTCi57eGXHoHLHV1Xl1pLrZNYHRdJhIV4=
```

A few ways to specify your decryption key

- If no key is specified, Chef will look in Chef::Config[:encypted_data_bag_secret]
- By default, that is set to /etc/chef/encrypted_data_bag_secret

```
10 decrypted = Chef::EncryptedDataBagItem.load("passwords", "userlist", contents_of_secret_key)
11
```

- Specify the key in your recipe

```
10 my_secret = Chef::EncryptedDataBagItem.load_secret("/path/to/other/secret_key")
11 decrypted = Chef::EncryptedDataBagItem.load("passwords", "userlist", my_secret)
```

- Load a different secret key and use its contents

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing execute[a2dissite default] action run (apache::default line 18)
INFO: Processing template[/etc/apache2/sites-available/clowns] action create (apache::default line 28)
INFO: Processing execute[a2ensite clowns] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/clowns] action create (apache::default line 45)
INFO: Processing template[/srv/apache/clowns/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/apache2/sites-available/bears] action create (apache::default line 28)
INFO: Processing execute[a2ensite bears] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/bears] action create (apache::default line 45)
INFO: Processing template[/srv/apache/bears/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: Processing user[bobo] action create (users::default line 13)
INFO: user[bobo] altered
INFO: Processing user[frank] action create (users::default line 13)
INFO: user[frank] altered
INFO: Processing group[clowns] action create (users::groups line 2)
INFO: Chef Run complete in 1.915067412 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [recipe[apache], recipe[motd], recipe[users]]
INFO: Run List expands to [apache, motd, users]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing execute[a2dissite default] action run (apache::default line 18)
INFO: Processing template[/etc/apache2/sites-available/clowns] action create (apache::default line 28)
INFO: Processing execute[a2ensite clowns] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/clowns] action create (apache::default line 45)
INFO: Processing template[/srv/apache/clowns/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/apache2/sites-available/bears] action create (apache::default line 28)
INFO: Processing execute[a2ensite bears] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/bears] action create (apache::default line 45)
INFO: Processing template[/srv/apache/bears/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: Processing user[bobo] action create (users::default line 13)
INFO: user[bobo] altered
INFO: Processing user[frank] action create (users::default line 13)
INFO: user[frank] altered
INFO: Processing group[clowns] action create (users::groups line 2)
INFO: Chef Run complete in 1.915067412 seconds
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Verify the user passwords are correct

```
$ sudo cat /etc/shadow
```

```
frank:$1$buWwEiTZ$Fmk1E2/utBCRj9l5uSufx1:15634:0:99999:7:::
bobo:$1$B1wv0KDX$t3WXdeILrP7W7Zkz5QeiG.:15634:0:99999:7:::
```

```
$ su - bobo
```

```
Password:
No directory, logging in with HOME=/
bobo@target1:/$
```

Questions

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

Roles

Role-based Attributes and Merge Order Precedence

Objectives

- Know what Roles are, and how they are used to provide clarity
- Be familiar with the Role Ruby DSL
- Know how to show a Role with Knife
- Understand how merge order effects the precedence hierarchy
- Understand nested Roles

What is a Role?

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

What is a Role?

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

Best Practice: Roles live in your chef-repo

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

Exercise: Create the webserver role

```
$ vi ./roles/webserver.rb
```

```
name "webserver"
description "Web Server"
run_list "recipe[apache]"
default_attributes({
  "apache" => {
    "sites" => {
      "admin" => {
        "port" => 82
      }
    }
  }
})
```

- Roles have **names**
- Roles have **descriptions**
- Roles have a **run_list**, just like a node

Exercise: Create the webserver role

```
$ vi ./roles/webserver.rb
```

```
name "webserver"
description "Web Server"
run_list "recipe[apache]"
default_attributes({
  "apache" => {
    "sites" => {
      "admin" => {
        "port" => 82
      }
    }
  }
})
```

- You can set **default** node attributes within roles!
- Save and Quit your editor

Exercise: Create the role

```
Updated Role webserver!
```

Exercise: Create the role

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

```
Updated Role webserver!
```

Exercise: Show the the role with knife

```
chef_type:              role
default_attributes:
  apache:
    sites:
      admin:
        port:  82
description:            Web Server
env_run_lists:          {}
json_class:             Chef::Role
name:                   webserver
override_attributes:    {}
run_list:               ["recipe[apache]"]
```

Exercise: Show the the role with knife

```
$ knife role show webserver
```

```
chef_type:              role
default_attributes:
  apache:
    sites:
      admin:
        port:  82
description:            Web Server
env_run_lists:          {}
json_class:             Chef::Role
name:                   webserver
override_attributes:    {}
run_list:               ["recipe[apache]"]
```

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

```
1 items found

chef_type:              role
default_attributes:
  apache:
    sites:
      admin:
        port:  82
description:            Web Server
env_run_lists:          {}
json_class:             Chef::Role
name:                   webserver
override_attributes:    {}
run_list:               ["recipe[apache]"]
```

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

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

```
1 items found

chef_type:             role
default_attributes:
  apache:
    sites:
      admin:
        port:  82
description:           Web Server
env_run_lists:         {}
json_class:            Chef::Role
name:                  webserver
override_attributes:   {}
run_list:              ["recipe[apache]"]
```

Exercise: Replace recipe[apache] with role[webserver] in your test nodes run list

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "pci": {
7        "in_scope": true
8      },
9      "tags": [
10
11     ],
12     "apache": {
13       "sites": {
14       }
15     }
16   },
17   "run_list": [
18     "role[webserver]",
19     "recipe[motd]",
20     "recipe[users]"
21   ]
22 }
```

Replace `recipe[apache]` with `role[webserver]`, **save** and **close.**

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver], recipe[motd],
recipe[users]]
INFO: Run List expands to [apache, motd, users]
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver], recipe[motd],
recipe[users]]
INFO: Run List expands to [apache, motd, users]
```

Exercise: Re-run the Chef Client

```
INFO: Processing template[/etc/apache2/sites-available/clowns] action create (apache::default line 28)
INFO: Processing execute[a2ensite clowns] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/clowns] action create (apache::default line 45)
INFO: Processing template[/srv/apache/clowns/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/apache2/sites-available/bears] action create (apache::default line 28)
INFO: Processing execute[a2ensite bears] action run (apache::default line 38)
INFO: Processing directory[/srv/apache/bears] action create (apache::default line 45)
INFO: Processing template[/srv/apache/bears/index.html] action create (apache::default line 50)
INFO: Processing template[/etc/apache2/sites-available/admin] action create (apache::default line 28)
INFO: template[/etc/apache2/sites-available/admin] updated content
INFO: template[/etc/apache2/sites-available/admin] mode changed to 644
INFO: Processing execute[a2ensite admin] action run (apache::default line 38)
Enabling site admin.
To activate the new configuration, you need to run:
  service apache2 reload
INFO: execute[a2ensite admin] ran successfully
INFO: execute[a2ensite admin] not queuing delayed action restart on service[apache2] (delayed), as it's already
been queued
INFO: Processing directory[/srv/apache/admin] action create (apache::default line 45)
INFO: directory[/srv/apache/admin] created directory /srv/apache/admin
INFO: directory[/srv/apache/admin] mode changed to 755
INFO: Processing template[/srv/apache/admin/index.html] action create (apache::default line 50)
INFO: template[/srv/apache/admin/index.html] updated content
INFO: template[/srv/apache/admin/index.html] mode changed to 644
```

OPSCODE

Node Attributes that are hashes are merged

- The apache cookbooks attribute file contains:

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

- While our role has...

```
4 default_attributes({
5   "apache" => {
6     "sites" => {
7       "admin" => {
8         "port" => 82
9       },
10    }
11  }
12 })
```

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

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

```
1 items found

apache.sites:
  admin:
    port:  82
  bears:
    port:  81
  clowns:
    port:  80
id:             target1.local
```

Exercise: Edit the webserver role

```
4  default_attributes({
5    "apache" => {
6      "sites" => {
7        "admin" => {
8          "port" => 82
9        },
10       "bears" => {
11         "port" => 8081
12       }
13     }
14   }
15 })
```

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

Exercise: Edit the webserver role

```
$ vi ./roles/webserver.rb
```

```
4  default_attributes({
5    "apache" => {
6      "sites" => {
7        "admin" => {
8          "port" => 82
9        },
10       "bears" => {
11         "port" => 8081
12       }
13     }
14   }
15 })
```

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

Exercise: Create the role

```
Updated Role webserver!
```

Exercise: Create the role

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

```
Updated Role webserver!
```

Exercise: Re-run the Chef Client

```
[2012-10-23T03:28:34+00:00] INFO: Processing template[/etc/apache2/sites-available/bears] action
create (apache::default line 28)
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] backed up to /var/
chef/backup/etc/apache2/sites-available/bears.chef-20121023032834
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] updated content
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] owner changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] group changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] mode changed to 644
[2012-10-23T03:28:34+00:00] INFO: Processing execute[a2ensite bears] action run (apache::default
line 38)
[2012-10-23T03:28:34+00:00] INFO: Processing directory[/srv/apache/bears] action create
(apache::default line 45)
[2012-10-23T03:28:34+00:00] INFO: Processing template[/srv/apache/bears/index.html] action create
(apache::default line 50)
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] backed up to /var/chef/
backup/srv/apache/bears/index.html.chef-20121023032834
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] updated content
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] owner changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] group changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] mode changed to 644
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
[2012-10-23T03:28:34+00:00] INFO: Processing template[/etc/apache2/sites-available/bears] action
create (apache::default line 28)
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] backed up to /var/
chef/backup/etc/apache2/sites-available/bears.chef-20121023032834
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] updated content
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] owner changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] group changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/etc/apache2/sites-available/bears] mode changed to 644
[2012-10-23T03:28:34+00:00] INFO: Processing execute[a2ensite bears] action run (apache::default
line 38)
[2012-10-23T03:28:34+00:00] INFO: Processing directory[/srv/apache/bears] action create
(apache::default line 45)
[2012-10-23T03:28:34+00:00] INFO: Processing template[/srv/apache/bears/index.html] action create
(apache::default line 50)
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] backed up to /var/chef/
backup/srv/apache/bears/index.html.chef-20121023032834
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] updated content
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] owner changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] group changed to 0
[2012-10-23T03:28:34+00:00] INFO: template[/srv/apache/bears/index.html] mode changed to 644
```

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

```
1 items found

apache.sites:
  admin:
    port:  82
  bears:
    port:  8081
  clowns:
    port:  80
id:             target1.local
```

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

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

```
1 items found

apache.sites:
  admin:
    port:  82
  bears:
    port:  8081
  clowns:
    port:  80
id:             target1.local
```

When you combine precedence and merge order, you get the complete picture of node attribute setting

	Attribute Files	Environment	Role	Node/Recipe
Default	1	2	3	4
Normal	5			6
Override	7	**9**	8	10
Automatic	11			

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

OPSCODE

Best Practice: Be careful when computing a value in an attribute file based on non-automatic data

- Because we merge data as we evaluate the Chef run, if you have something like the following in an attribute file:

```
1 default[:sneaky_port] = node['apache']['sites']['bears']['port'] * 2
```

- You will compute the value based on only what was available when the attribute file was parsed - so in our example, port 81 rather than the 8081 we just set it to.
- If you need to do this kind of computation, do it **late** in the merge order - usually in a recipe

RULE THE CLOUD

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

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

- **Save** and **Quit** your editor

Exercise: Create the base role

```
$ vi ./roles/base.rb
```

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

- **Save** and **Quit** your editor

Exercise: Create the role

```
Updated Role base!
```

Exercise: Create the role

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

```
Updated Role base!
```

Exercise: Add the base role to the webserver roles run list

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

- Put role[base] at the front of the run_list
- **Save** and **Quit** your editor

Exercise: Add the base role to the webserver roles run list

```
$ vi ./roles/webserver.rb
```

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

- Put role[base] at the front of the run_list
- **Save** and **Quit** your editor

Exercise: Update the role

```
Updated Role webserver!
```

Exercise: Update the role

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

```
Updated Role webserver!
```

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver], recipe[motd],
recipe[users]]
INFO: Run List expands to [motd, users, apache]
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver], recipe[motd],
recipe[users]]
INFO: Run List expands to [motd, users, apache]
```

Best Practice: Be explicit about what you need or expect

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

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

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

Objectives

- Understand what an Environment is, and how it is different from an Organization
- Know how to set cookbook version constraints
- Understand when to set attributes in an environment
- Know how to leverage per-environment run lists in a role

What is an Environment?

- Environments allow you to:
  - Control what cookbook versions are used by a group of nodes
  - Set attributes that are relevant
- Typical environments would be
  - Dev
  - Test
  - Production

Exercise: Use knife to show the available cookbook versions

```
apache   0.2.0  0.1.0
```

Exercise: Use knife to show the available cookbook versions

```
$ knife cookbook show apache
```

```
apache   0.2.0  0.1.0
```

Best Practice: If you need to share Cookbooks or Roles, use an Environment

- Environments allow for segregation within a single organization.
- If you need to share cookbooks or roles, you likely want an Environment rather than an organization.

Best Practice: Environments live in your chef-repo, until they don't

- Like Data Bags and Roles, you have options with how to create an Environment
- If you are managing them **manually**
  - The best practice is that all of your Environments live in the environments directory of your chef-repo
- If you are managing them **automatically**
  - Create all of your Environments with the automation, or with Knife directly

Exercise: List the current environments

```
_default
```

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

Exercise: List the current environments

```
$ knife environment list
```

```
_default
```

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

Exercise: Create a dev environment

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

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

Exercise: Create a dev environment

```
$ vi ./environments/dev.rb
```

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

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

Cookbook Version Constraints

- = Equal to
- > Greater than
- < Less than
- >= Greater than or equal to
- <= Less than or equal to
- ~> Approximately greater than
- Learn more at http://docs.opscode.com/chef/essentials_cookbook_versions.html

Best Practice: Use only the = operator

- When you need more complex version constraints, you'll be glad they are there
- But do not be tempted to build complicated schemes tying cookbook development branches to complex version constraints
  - That way leads madness :)

Exercise: Create the dev environment

```
Updated Environment dev
```

Exercise: Create the dev environment

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

```
Updated Environment dev
```

Exercise: Show your Chef dev environment

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

Exercise: Show your Chef dev environment

```
$ knife environment show dev
```

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

Exercise: Change your nodes chef_environment to "dev"

```
1  {
2    "name": "target1.local",
3    "chef_environment": "dev",
4    "normal": {
5      "company": "opscode",
6      "pci": {
7        "in_scope": true
8      },
9      "tags": [
10
11     ],
12     "apache": {
13       "sites": {
14       }
15     }
16   },
17   "run_list": [
18     "role[webserver]"
19   ]
20 }
```

Replace `_default` with `dev`

Delete the pci section (in *red*)

Exercise: Re-run the Chef Client

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

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

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

Exercise: Create a production environment

```
1 name "production"
2 description "For Prods!"
3 cookbook "apache", "= 0.1.0"
4 override_attributes({
5   "pci" => {
6     "in_scope" => true
7   }
8 })
```

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

Exercise: Create a production environment

```
$ vi ./environments/production.rb
```

```
1 name "production"
2 description "For Prods!"
3 cookbook "apache", "= 0.1.0"
4 override_attributes({
5   "pci" => {
6     "in_scope" => true
7   }
8 })
```

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

Exercise: Create the production environment

```
Updated Environment production
```

Exercise: Create the production environment

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

```
Updated Environment production
```

Exercise: Change your nodes chef_environment to "production"

```
1  {
2    "name": "target1.local",
3    "chef_environment": "production",
4    "normal": {
5      "company": "opscode",
6      "tags": [
7
8      ],
9      "apache": {
10       "sites": {
11       }
12     }
13   },
14   "run_list": [
15     "role[webserver]"
16   ]
17 }
```

Replace dev with production

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver], recipe[motd], recipe[users]]
INFO: Run List expands to [motd, users, apache]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Storing updated cookbooks/apache/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/apache/metadata.rb in the cache.
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: Processing user[bobo] action create (users::default line 13)
INFO: Processing user[frank] action create (users::default line 13)
INFO: Processing group[clowns] action create (users::groups line 2)
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Chef Run complete in 2.187502202 seconds
INFO: Removing cookbooks/apache/CHANGELOG.md from the cache; it is no longer needed by chef-client.
INFO: Removing cookbooks/apache/templates/default/custom.erb from the cache; it is no longer needed by chef-client.
INFO: Removing cookbooks/apache/templates/default/index.html.erb from the cache; it is no longer needed by chef-client.
INFO: Running report handlers
INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver], recipe[motd], recipe[users]]
INFO: Run List expands to [motd, users, apache]
INFO: Starting Chef Run for target1.local
INFO: Running start handlers
INFO: Start handlers complete.
INFO: Loading cookbooks [apache, motd, pci, users]
INFO: Storing updated cookbooks/apache/recipes/default.rb in the cache.
INFO: Storing updated cookbooks/apache/metadata.rb in the cache.
INFO: Processing template[/etc/motd.tail] action create (motd::default line 10)
INFO: Processing user[bobo] action create (users::default line 13)
INFO: Processing user[frank] action create (users::default line 13)
INFO: Processing group[clowns] action create (users::groups line 2)
INFO: Processing package[apache2] action install (apache::default line 10)
INFO: Processing service[apache2] action start (apache::default line 14)
INFO: Processing service[apache2] action enable (apache::default line 14)
INFO: Processing cookbook_file[/var/www/index.html] action create (apache::default line 18)
INFO: Chef Run complete in 2.187502202 seconds
INFO: Removing cookbooks/apache/CHANGELOG.md from the cache; it is no longer needed by chef-client.
INFO: Removing cookbooks/apache/templates/default/custom.erb from the cache; it is no longer needed by chef-client.
INFO: Removing cookbooks/apache/templates/default/index.html.erb from the cache; it is no longer needed by chef-client.
INFO: Running report handlers
INFO: Report handlers complete
```

OPSCODE

Best Practice: Rollbacks are real, but desired state is real-er

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

RULE THE CLOUD

Exercise: Update production to use version 0.2.0 of the apache cookbook

- Make sure the apache cookbook is set to version 0.2.0

Exercise: Update production to use version 0.2.0 of the apache cookbook

```
$ vi ./environment/production.rb
```

- Make sure the apache cookbook is set to version 0.2.0

Exercise: Upload the updated production environment

```
Updated Environment production
```

Exercise: Upload the updated production environment

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

```
Updated Environment production
```

Security Policy Change!

- Nodes in the production environment should not have users created
- How can we do that without removing the users recipe from our base role?

Exercise: Add an environment specific run list for production to the base role

```
1  name "base"
2  description "Base Server Role"
3  def_run_list = [ "recipe[motd]", "recipe[users]" ]
4  env_run_lists({
5    "_default" => def_run_list,
6    "dev" => def_run_list,
7    "production" => [
8      "recipe[motd]"
9    ]
10 })
```

- **Delete** the run_list

- **Save** and **Quit** your editor

Exercise: Add an environment specific run list for production to the base role

```
$ vi ./roles/base.rb
```

```
name "base"
description "Base Server Role"
def_run_list = [ "recipe[motd]", "recipe[users]" ]
env_run_lists({
  "_default" => def_run_list,
  "dev" => def_run_list,
  "production" => [
    "recipe[motd]"
  ]
})
```

- **Delete** the run_list

- **Save** and **Quit** your editor

Exercise: Create the role

```
Updated Role base!
```

Exercise: Create the role

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

```
Updated Role base!
```

Exercise: Re-run the Chef Client

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver]]
INFO: Run List expands to [motd, apache]
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: *** Chef 10.16.0 ***
INFO: Run List is [role[webserver]]
INFO: Run List expands to [motd, apache]
```

Questions?

- What is an Environment?
- How is it different from an Organization?
- What is a cookbook version constraint?
- Which cookbook constraint should we probably not be using?
- What kind of node attributes do you typically set from an Environment?
- What is a per-environment run list?

Using Community Cookbooks

Objectives

- Use the Opscode Chef Community site to find, preview and download cookbooks.
- Use knife to work with the Community Site API.
- Download, extract, examine and implement cookbooks from the Community site.

The easy way...

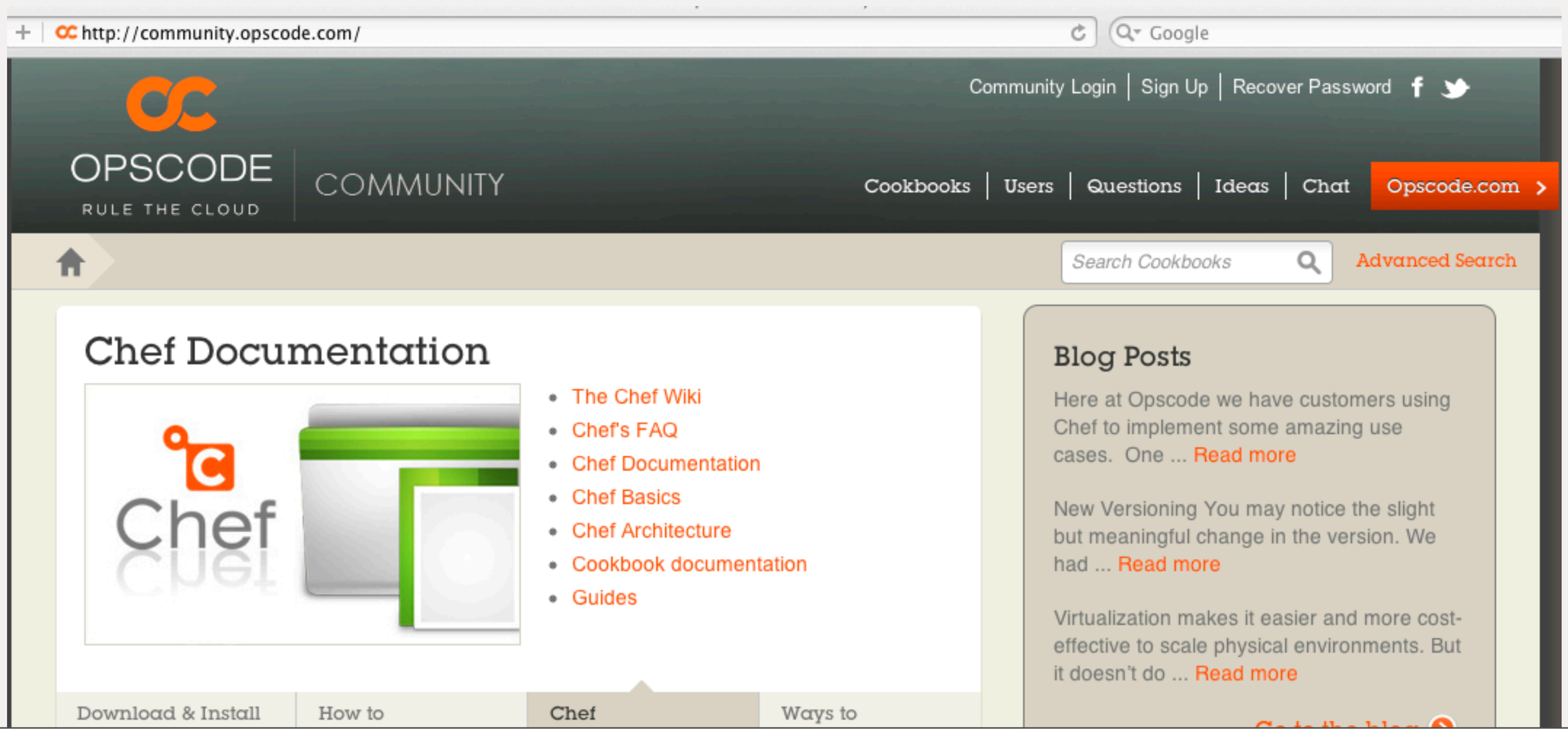
We've been writing some cookbooks so far...

Hundreds already exist for a large number of use cases and purposes. Many (but only a fraction) are maintained by Opscode.

Think of it like RubyGems.org, CPAN.org, or other focused plugin-style distribution sites.

Exercise: Find and preview cookbooks on the site

http://community.opscode.com/



Exercise: Find and preview cookbooks on the site

http://community.opscode.com/

Exercise: Search for a chef-client cookbook

http://community.opscode.com/cookbooks

Community Login | Sign Up | Recover Password

OPSCODE
RULE THE CLOUD
COMMUNITY

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Cookbooks

Search Cookbooks

Advanced Search

All Categories

mysql
★★★★★ 5075 downloads

apache2
★★★★★ 3057 downloads

nginx
★★★★☆ 2903 downloads

See all Cookbooks

Databases

mysql
★★★★★ 5075 downloads

postgresql
★★★★☆ 1332 downloads

riak
★★★★★ 777 downloads

See all

Web Servers

apache2
★★★★★ 3057 downloads

nginx
★★★★☆ 2903 downloads

passenger_apache2
★★★☆☆ 475 downloads

See all

Process Management

runit
★★★★★ 865 downloads

bluepill
★★★★★ 420 downloads

god
★★★☆☆ 211 downloads

See all

Monitoring & Trending

nagios
★★★★☆ 1268 downloads

graylog2
★★★★☆ 882 downloads

revealcloud
★★★★★ 878 downloads

See all

Add a New Cookbook

Need Help?

If you have questions, or you're stuck, we're here to help.

Get Help

REST API

Access all cookbooks on this site programatically.

Learn More

Exercise: Search for a chef-client cookbook

http://community.opscode.com/cookbooks

Google

Community Login | Sign Up | Recover Password

OPSCODE
RULE THE CLOUD
COMMUNITY

Search for: chef-client

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Cookbooks

Search Cookbooks

Advanced Search

All Categories

See all Cookbooks

mysql
★★★★★ 5075 downloads

apache2
★★★★★ 3057 downloads

nginx
★★★★☆ 2903 downloads

Add a New Cookbook

Databases

mysql
★★★★★ 5075 downloads

postgresql
★★★★☆ 1332 downloads

riak
★★★★★ 777 downloads

See all

Web Servers

apache2
★★★★★ 3057 downloads

nginx
★★★★☆ 2903 downloads

passenger_apache2
★★★☆☆ 475 downloads

See all

Process Management

runit
★★★★★ 865 downloads

bluepill
★★★★★ 420 downloads

god
★★★☆☆ 211 downloads

See all

Monitoring & Trending

nagios
★★★★☆ 1268 downloads

graylog2
★★★★☆ 882 downloads

revealcloud
★★★★★ 878 downloads

See all

Need Help?

If you have questions, or you're stuck, we're here to help.

Get Help

REST API

Access all cookbooks on this site programatically.

Learn More

OPSCODE | Chef

Search Results...

http://community.opscode.com/search?query=chef-client&scope=cookbook

Google

Community Login | Sign Up | Recover Password

OPSCODE
RULE THE CLOUD
COMMUNITY

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Search

chef-client

Advanced Search

Search results for 'chef-client'

432 downloads 36 followers

chef-client

Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), mac_os_x_server (>= 0.0.0), openbsd (>= 0.0.0), redhat (>= 0.0.0), ubuntu (>= 0.0.0), windows (>= 0.0.0)

2 ratings Category: Utilities

Manages aspects of only chef-client

27 downloads 0 followers

chef-client-cron

Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), openbsd (>= 0.0.0), redhat (>= 0.0.0), ubuntu (>= 0.0.0)

0 ratings Category: Utilities

Manages aspects of only chef-client

4 downloads 2 followers

OPSCODE | Chef

Search Results...

http://community.opscode.com/search?query=chef-client&scope=cookbook

Google

Community Login | Sign Up | Recover Password

OPSCODE
RULE THE CLOUD
COMMUNITY

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Search

chef-client

Advanced Search

Search results for 'che

We're probably looking for this one

chef-client

432 downloads 36 followers

Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), mac_os_x_server (>= 0.0.0), openbsd (>= 0.0.0), redhat (>= 0.0.0), ubuntu (>= 0.0.0), windows (>= 0.0.0)

2 ratings Category: Utilities

Manages aspects of only chef-client

chef-client-cron

27 downloads 0 followers

Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), openbsd (>= 0.0.0), redhat (>= 0.0.0), ubuntu (>= 0.0.0)

0 ratings Category: Utilities

Manages aspects of only chef-client

4 downloads 2 followers

OPSCODE | Chef

Viewing a cookbook

http://community.opscode.com/cookbooks/chef-client

Google

Community Login | Sign Up | Recover Password

OPSCODE
RULE THE CLOUD
COMMUNITY

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Cookbooks > chef-client | Details | Comments (0) | Source Code

Search Cookbooks | Advanced Search

chef-client

Manages aspects of only chef-client

2 ratings | Downloaded 432 times | Last updated Jun 10, 2012

Download

Follow

Category: Utilities
Homepage: http://github.com/opscode-cookbooks/chef-client
License: Apache 2.0
Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), mac_os_x_server (>= 0.0.0), openbsd (>= 0.0.0), redhat (>= 0.0.0), ubuntu (>= 0.0.0), windows (>= 0.0.0)
Created: Dec 16, 2010
Last updated: Jun 10, 2012
Versions: 1.1.4, 1.1.2, 1.1.0, 1.0.4, 1.0.2, 1.0.0, 0.99.5, 0.99.4, 0.99.3, 0.99.2, 0.99.1, 0.99.0

Maintainer

opscode

Collaborators

Description

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

Requirements

OPSCODE | Chef

Viewing a cookbook

http://community.opscode.com/cookbooks/chef-client

Community Login | Sign Up | Recover Password

OPSCODE RULE THE CLOUD | COMMUNITY

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Cookbooks > chef-client | Details | Comments (0) | Source Code | Search Cookbooks | Advanced Search

chef-client

Manages aspects of only chef-client

2 ratings | Downloaded 432 times | Last updated

Browse source code

Download

Follow

Category: Utilities
Homepage: http://github.com/opscode-cookbooks/chef-client
License: Apache 2.0
Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), mac_os_x_server (>= 0.0.0), openbsd (>= 0.0.0), redhat (>= 0.0.0), ubuntu (>= 0.0.0), windows (>= 0.0.0)
Created: Dec 16, 2010
Last updated: Jun 10, 2012
Versions: 1.1.4, 1.1.2, 1.1.0, 1.0.4, 1.0.2, 1.0.0, 0.99.5, 0.99.4, 0.99.3, 0.99.2, 0.99.1, 0.99.0

Maintainer

opscode

Collaborators

Description

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

Requirements

OPSCODE | Chef

Viewing a cookbook

http://community.opscode.com/cookbooks/chef-client

Community Login | Sign Up | Recover Password

OPSCODE RULE THE CLOUD | COMMUNITY

Cookbooks | Users | Questions | Ideas | Chat | Opscode.com

Cookbooks > chef-client | Details | Comments (0) | Source Code | Search Cookbooks | Advanced Search

chef-client

Manages aspects of only chef-client

2 ratings | Downloaded 432 times | Last updated

Category: Utilities
Homepage: http://github.com/opscode-cookbooks/chef-client
License: Apache 2.0
Platforms: centos (>= 0.0.0), debian (>= 0.0.0), fedora (>= 0.0.0), freebsd (>= 0.0.0), mac_os_x (>= 0.0.0), mac_os_x_server (>= 0.0.
Created: Dec 16, 2
Last updated: Jun 10, 2
Versions: 1.1.4, 1.1

Download

Follow

Maintainer

opscode

Collaborators

Description

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

Requirements

Browse source code

README displayed on the page (if it has one)

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

Exercise: knife cookbook site search

```
$ 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:    opscode
    cookbook_name:          chef
  chef-client:
    cookbook:               http://cookbooks.opscode.com/api/v1/cookbooks/chef-client
    cookbook_description:   Manages aspects of only chef-client
    cookbook_maintainer:    opscode
    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
  forkable_client:
    cookbook:               http://cookbooks.opscode.com/api/v1/cookbooks/forkable_client
    cookbook_description:   Provides forked chef-client runs
    cookbook_maintainer:    chrisroberts
    cookbook_name:          forkable_client
```

Exercise: knife cookbook site show

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

```
  average_rating:  4.66667
  category:        Utilities
  created_at:      2010-12-16T23:00:45Z
  description:     Manages aspects of only chef-client
  external_url:    github.com/opscode-cookbooks/chef-client
  latest_version:  http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/
  1_2_0
  maintainer:      opscode
  name:            chef-client
  updated_at:      2012-08-30T20:55:46Z
  versions:
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_2_0
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_1_4
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_1_2
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_1_0
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_0_4
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_0_2
      http://cookbooks.opscode.com/api/v1/cookbooks/chef-client/versions/1_0_0
```

Exercise: Download chef-client cookbook

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

```
Downloading chef-client from the cookbooks site at
version 2.2.1 to /Users/YOU/chef-repo/chef-
client-2.2.1.tar.gz
Cookbook saved: /Users/YOU/chef-repo/chef-
client-2.2.1.tar.gz
```

Exercise: Extract chef-client cookbook tarball

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

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

So, you just downloaded source code from the internet.

As root.

To load in the magic machine that:

- **Makes your computers run code**

Read the *entire* cookbook first!

Examining the 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 delete_validation recipe

```
$ vi cookbooks/chef-client/recipes/delete_validation.rb
```

```
20 unless node[:recipes].include?("chef-server")
21   file Chef::Config[:validation_key] do
22     action :delete
23     backup false
24     only_if { ::File.exists?(Chef::Config[:client_key]) }
25   end
26 end
```

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

```
$ vi ./roles/base.rb
```

```
1  name "base"
2  description "Base Server Role"
3  run_list "recipe[chef-client::delete_validation]", "recipe[motd]",
   "recipe[users]"
4  #def_run_list = "recipe[motd]", "recipe[users]"
5  #env_run_lists({
6  #   "_default" => def_run_list,
7  #   "dev" => def_run_list,
8  #   "production" => [
9  #     "recipe[motd]"
10 #   ]
11 #})
```

- Add the **delete_validation** recipe
- Comment the environment specifics (*in red*)

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 default recipe

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

```
12 #
13 # Unless required by applicable law or agreed to in writi
14 # distributed under the License is distributed on an "AS
15 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either ex
16 # See the License for the specific language governing per
17 # limitations under the License.
18 #
19
20 include_recipe "chef-client::service"
~
```

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 service recipe

```
$ vi ./cookbooks/chef-client/recipes/service.rb
```

```
71 case node["chef_client"]["init_style"]
72 when "init"
73
74   dist_dir, conf_dir = value_for_platform(
75     ["ubuntu", "debian"] => { "default" => ["debian", "default"] },
76     ["redhat", "centos", "fedora", "scientific", "amazon"] => { "default" =>
77     ["suse"] => { "default" => ["suse", "sysconfig"] }
78   )
79
80   template "/etc/init.d/chef-client" do
81     source "#{dist_dir}/init.d/chef-client.erb"
82     mode 0755
83     variables(
84       :client_bin => client_bin
85     )
86     notifies :restart, "service[chef-client]", :delayed
87   end
88
89   template "/etc/#{conf_dir}/chef-client" do
90     source "#{dist_dir}/#{conf_dir}/chef-client.erb"
91     mode 0644
92     notifies :restart, "service[chef-client]", :delayed
93   end
94
95   service "chef-client" do
96     supports :status => true, :restart => true
97     action :enable
98   end
```

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

Best Practice: Well written cookbooks change behavior based on attributes

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

Exercise: Upload the chef-client cookbook

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

```
Uploading chef-client              [2.2.1]
ERROR: Cookbook chef-client depends on cookbook 'cron'
version '>= 0.0.0',
ERROR: which is not currently being uploaded and
cannot be found on the server.
```

OPSCODE | Chef

Exercise: Add chef-client recipe to base role

```
> vi ./roles/base.rb
```

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

RULE THE CLOUD

Exercise: upload the base role

```
$ knife role from file base.rb

Updated Role base!
```

Exercise: Run chef-client on your node

```
$ sudo chef-client
```

```
INFO: Processing file[/etc/chef/validation.pem] action delete (chef-client::delete_validation line 21)
INFO: file[/etc/chef/validation.pem] deleted file at /etc/chef/validation.pem
INFO: Processing directory[/var/run/chef] action create (chef-client::service line 42)
INFO: directory[/var/run/chef] created directory /var/run/chef
INFO: Processing directory[/var/cache/chef] action create (chef-client::service line 42)
INFO: directory[/var/cache/chef] created directory /var/cache/chef
INFO: Processing directory[/var/lib/chef] action create (chef-client::service line 42)
INFO: directory[/var/lib/chef] created directory /var/lib/chef
INFO: Processing directory[/var/log/chef] action create (chef-client::service line 42)
INFO: directory[/var/log/chef] created directory /var/log/chef
INFO: Processing template[/etc/init.d/chef-client] action create (chef-client::service line 61)
INFO: template[/etc/init.d/chef-client] mode changed to 755
INFO: template[/etc/init.d/chef-client] updated content
INFO: Processing template[/etc/default/chef-client] action create (chef-client::service line 70)
INFO: template[/etc/default/chef-client] mode changed to 644
INFO: template[/etc/default/chef-client] updated content
INFO: template[/etc/default/chef-client] not queuing delayed action restart on service[chef-client]
(delayed), as it's already been queued
INFO: Processing service[chef-client] action enable (chef-client::service line 76)
INFO: service[chef-client] enabled
INFO: template[/etc/init.d/chef-client] sending restart action to service[chef-client] (delayed)
INFO: Processing service[chef-client] action restart (chef-client::service line 76)
INFO: service[chef-client] restarted
```

Exercise: Verify chef-client is running

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

```
root      3227  1.1  1.7 113928 30040 ?        Sl
20:23   0:01 /opt/chef/embedded/bin/ruby /usr/bin/
chef-client -d -P /var/run/chef/client.pid -L /var/
log/chef/client.log -c /etc/chef/client.rb -i 1800 -s
20
```

Convergent infrastructure

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

Download and use ntp cookbook

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

Exercise: Download the ntp cookbook

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

```
Downloading ntp from the cookbooks site at version
1.1.8 to /Users/YOU/chef-repo/ntp-1.2.0.tar.gz
Cookbook saved: /Users/YOU/chef-repo/ntp-1.2.0.tar.gz
```

Exercise: Extract the ntp cookbook

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

```
x ntp/
x ntp/attributes/
x ntp/CHANGELOG.md
x ntp/CONTRIBUTING
x ntp/LICENSE
x ntp/metadata.json
x ntp/metadata.rb
x ntp/README.md
x ntp/recipes/
x ntp/templates/
x ntp/templates/default/
x ntp/templates/default/ntp.conf.erb
x ntp/recipes/default.rb
x ntp/attributes/default.rb
```

Examining the ntp cookbook

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

- default recipe

Exercise: view the ntp default recipe

```
> vi ./cookbooks/ntp/recipes/default.rb
```

```
22 case node[:platform]
23 when "ubuntu","debian"
24   package "ntpdate" do
25     action :install
26   end
27   package "ntp" do
28     action :install
29   end
30 when "redhat","centos","fedora"
31   package "ntp" do
32     action :install
33   end
34   package "ntpdate" do
```

```
56 service node[:ntp][:service] do
57   supports :status => true, :restart => true
58   action [ :enable, :start ]
59 end
60
61 template "/etc/ntp.conf" do
62   source "ntp.conf.erb"
63   owner "root"
64   group root_group
65   mode "0644"
66   notifies :restart, resources(:service => node[:n
67 end
```

- The template notifies the service

Exercise: Upload the ntp cookbook

```
$ knife cookbook upload ntp

Uploading ntp                    [1.2.0]
Uploaded 1 cookbook.
```

OPSCODE | Chef

Exercise: Add the ntp recipe to the base role

```
$ vi ./roles/base.rb
```

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

RULE THE CLOUD

Exercise: Upload the base role

```
$ knife role from file base.rb

Updated Role base!
```

Exercise: Run chef-client on the node

```
$ sudo chef-client
```

```
INFO: Processing package[ntp] action install (ntp::default line 21)
INFO: Processing package[ntpdate] action install (ntp::default line 21)
INFO: Processing directory[/var/lib/ntp] action create (ntp::default
line 26)
INFO: Processing directory[/var/log/ntpstats/] action create
(ntp::default line 26)
INFO: Processing service[ntp] action enable (ntp::default line 33)
INFO: Processing service[ntp] action start (ntp::default line 33)
INFO: Processing template[/etc/ntp.conf] action create (ntp::default
line 38)
INFO: template[/etc/ntp.conf] backed up to /var/chef/backup/etc/
ntp.conf.chef-20121023040506
INFO: template[/etc/ntp.conf] updated content
INFO: template[/etc/ntp.conf] owner changed to 0
INFO: template[/etc/ntp.conf] group changed to 0
INFO: template[/etc/ntp.conf] mode changed to 644
```

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?

Just enough Ruby for Chef

OPSCODE

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

RULE THE CLOUD

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 test node

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

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

Exercise: Variable assignment

```
$ /opt/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}
```

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

chef-shell - The Chef Console

OPSCODE

RULE THE CLOUD

Chef

Objectives

- Learn about chef-shell, the interactive Chef console (REPL)
- Understand Chef Shell's execution contexts
- Use Chef Shell to inspect and debug recipes
- Interact with a Chef Server API
- Learn about the magic that is knife exec

Chef Shell

- Interactive Ruby console (REPL) for Chef.
- Three startup modes (standalone, solo, client).
- Three run-time contexts (main, attribute, recipe).
- Configured with a specific or implied config file.

Shef Startup

Three modes for startup

- Standalone: no cookbooks and empty run list (default)
- Solo: attempts to act as chef-solo, using solo config and JSON attributes. (-s or --solo)
- Client: contacts chef server for node object, retrieving cookbooks and using specified run list (-z or --client).

Exercise: Start the chef-shell REPL

```
target1:~$ sudo chef-shell -z
```

```
loading configuration: /etc/chef/client.rb
Session type: client
Loading...[2012-10-23T04:08:35+00:00] INFO: Run List is [role[webserver]]
[2012-10-23T04:08:35+00:00] INFO: Run List expands to [chef-client::delete_validation, chef-client,
ntp, motd, users, apache]
..[2012-10-23T04:08:36+00:00] INFO: Loading cookbooks [apache, chef-client, motd, ntp, pci, users]
.done.

This is chef-shell, the Chef shell.
 Chef Version: 10.16.0
 http://www.opscode.com/chef
 http://wiki.opscode.com/display/chef/Home

run `help' for help, `exit' or ^D to quit.

Ohai2u ubuntu@target1.local!
chef >
```

Exercise: Get help

```
chef > help
```

```
Chef Shell Help
================================================================================
| Command                    | Description
================================================================================
| help                       | prints this help message
| version                    | prints information about chef
| recipe                     | switch to recipe mode
| attributes                 | switch to attributes mode
| run_chef                   | run chef using the current recipe
| chef_run                   | returns an object to control a paused chef run
| chef_run.resume            | resume the chef run
| chef_run.step              | run only the next resource
| chef_run.skip_back         | move back in the run list
| chef_run.skip_forward      | move forward in the run list
| reset                      | resets the current recipe
| become_node                | assume the identity of another node.
| echo                       | turns printout of return values on or off
| echo?                      | says if echo is on or off
| tracing                    | turns on or off tracing of execution. *verbose*
| tracing?                   | says if tracing is on or off
| ls                         | simple ls style command
| node                       | returns the current node (i.e., this host)
| ohai                       | pretty print the node's attributes
| edit                       | edit an object in your EDITOR
| clients                    | Find and edit API clients
```

Exercise: Get help

```
chef > help(:nodes)
```

```
Command: nodes
==============================================================================
## SUMMARY ##
  +nodes+ Allows you to query your chef server for information about your nodes.

## LIST ALL NODES ##
  You can list all nodes using +all+ or +list+

      nodes.all #=> [<Chef::Node...>, <Chef::Node...>, ...]

  To limit the information returned for each node, pass a code block to the +all+
  subcommand:

      nodes.all { |node| node.name } #=> [NODE1_NAME, NODE2_NAME, ...]

## SHOW ONE NODE ##
  You can show the data for a single node using the +show+ subcommand:

      nodes.show("NODE_NAME") => <Chef::Node @name="NODE_NAME" ...>

## SEARCH FOR NODES ##
  You can search for nodes using the +search+ or +find+ subcommands:

      nodes.find(:name => "app*") #=> [<Chef::Node @name="app1.example.com" ...>, ...]
```

Exercise: Look at the company attribute for the node object

```
chef > node['company']
```

```
=> "opscode"
```

Exercise: Change the context you are in to that of a cookbook attribute file

```
chef > attributes_mode
```

```
chef:attributes >
```

Exercise: Add a default attribute

```
chef:attributes > node.normal["interactive"] = "is cool"
```

```
=> "is cool"
```

Exercise: Add a default attribute

```
chef:attributes > node["interactive"]
```

```
=> "is cool"
```

Exercise: Search for new default attribute

```
$ knife node show target1.local -a interactive
```

```
interactive:
```

Exercise: Save node state

```
chef:attributes > node.save
```

```
=> <Chef::Node:0x11671f4 @name="target1.local">
```

Exercise: Search for new default attribute

```
$ knife node show target1.local -a interactive
```

```
interactive:  is cool
```

Exercise: Display all node attributes

```
$ knife node show target1.local -Fj -l | less
```

```
  "name": "target1.local",
  "chef_environment": "dev",
  "run_list": [
    "role[webserver]"
  ],
  "normal": {
    "company": "opscode",
    "tags": [
    ],
    "interactive": "is cool"
  },
  "default": {
    "apache": {
      "sites": {
        "clowns": {
          "port": 80
        },
        "bears": {
          "port": 81
        }
      }
    }
  },
  "override": {
  },
  "automatic": {
    "languages": {
      "ruby": {
```

Exercise: Switch to recipe context

```
chef:attributes > recipe_mode
```

```
chef:recipe >
```

Exercise: Use a package resource

```
chef:recipe > package “jove”
```

```
 => <package[jove] @name: "jove" @noop: nil @before: nil
@params: {} @provider: nil @allowed_actions:
[:nothing, :install, :upgrade, :remove, :purge, :reconfig]
@action: :install @updated: false @updated_by_last_action:
false @supports: {} @ignore_failure: false @retries: 0
@retry_delay: 2 @immediate_notifications: []
@delayed_notifications: [] @source_line: "(irb#1):1:in
`irb_binding'" @resource_name: :package @package_name: "jove"
@version: nil @candidate_version: nil @response_file: nil
@source: nil @options: nil @cookbook_name: nil @recipe_name:
nil @enclosing_provider: nil>
```

Exercise: Run the recipe

```
chef:recipe > run_chef
```

```
DEBUG: Processing package[jove] on george-training.novalocal
INFO: Processing package[jove] action install ((irb#1) line 1)
DEBUG: package[jove] checking package status for jove
jove:
  Installed: (none)
  Candidate: 4.16.0.72-2.2
  Version table:
     4.16.0.72-2.2 0
        500 http://archive.ubuntu.com/ubuntu/ precise/universe i386
Packages
DEBUG: package[jove] current version is nil
DEBUG: package[jove] candidate version is 4.16.0.72-2.2
DEBUG: Executing apt-get -q -y install jove=4.16.0.72-2.2
```

Exercise: reset

```
chef:recipe > reset
```

```
Loading.....INFO: Run List is [role[webserver]]
INFO: Run List expands to [motd, users, ntp, apache]
...INFO: Loading cookbooks [apache, motd, ntp, pci,
users]
.done.

 => nil
```

Exercise: Debug an existing recipe

```
chef:recipe > include_recipe “users”
```

```
 => [#<Chef::Recipe:0x9e8cb38 @cookbook_name="users", @recipe_name="default",
@run_context=#<Chef::RunContext:0x9f23dd0 @node=<Chef::Node:0x5054a4c @name="george-training.novalocal">,
@cookbook_collection={"pci"=>#<Chef::CookbookVersion:0xa01add8 @name="pci", @frozen=false,
@attribute_filenames=["/var/chef/cache/cookbooks/pci/attributes/default.rb"], @definition_filenames=[],
@template_filenames=[], @file_filenames=[], @recipe_filenames=["/var/chef/cache/cookbooks/pci/recipes/
default.rb"], @recipe_filenames_by_name={"default"=>"/var/chef/cache/cookbooks/pci/recipes/default.rb"},
@library_filenames=[], @resource_filenames=[], @provider_filenames=[], @metadata_filenames=[],
@root_dir=nil, @root_filenames=["/var/chef/cache/cookbooks/pci/metadata.rb", "/var/chef/cache/cookbooks/
pci/README.md"], @couchdb_id=nil, @couchdb=#<Chef::CouchDB:0xa01ace8 @db="chef", @rest=#<Chef::REST:
0xa01acd4 @url="http://localhost:5984", @cookies={}, @default_headers={},
@auth_credentials=#<Chef::REST::AuthCredentials:0xa01ac98 @key_file=nil, @client_name=nil>,
@sign_request=true, @sign_on_redirect=true, @redirects_followed=0, @redirect_limit=10>>,
@couchdb_rev=nil, @status=:ready, @manifest={"name"=>"pci-0.0.1", "definitions"=>[],
"json_class"=>"Chef::CookbookVersion", "files"=>[], "attributes"=>[{"name"=>"default.rb",
"url"=>"https://s3.amazonaws.com/opscode-platform-production-data/organization-
f976ba971caf40918cea53a9de6076cc/checksum-0d35ace084267f0438afac50907c2163?
AWSAccessKeyId=AKIAJOZTD2N26S7W6APA&Expires=1340264374&Signature=YVJMj7Upfjfkhrbz7WFPLLKq7kg%3D",
"checksum"=>"0d35ace084267f0438afac50907c2163", "path"=>"attributes/default.rb",
"specificity"=>"default"}], "metadat
```

Exercise: Examine the resources it created

```
chef:recipe > resources
```

```
["user[frank]", "user[bobo]", "group[clowns]"]
 => ["user[frank]", "user[bobo]", "group[clowns]"]
```

Exercise: Grab a single resource

```
chef:recipe > resources("user[frank]")
```

```
<user[frank] @name: "frank" @noop: nil @before: nil @params: {} @provider: nil @allowed_actions:
[:nothing, :create, :remove, :modify, :manage, :lock, :unlock] @action: :create @updated: false
@updated_by_last_action: false @supports: {:manage_home=>false, :non_unique=>false}
@ignore_failure: false @retries: 0 @retry_delay: 2 @immediate_notifications: []
@delayed_notifications: [] @source_line: "/var/chef/cache/cookbooks/users/recipes/default.rb:11:in
`block in from_file'" @resource_name: :user @username: "frank" @comment: "Frank Belson" @uid: 2001
@gid: 0 @home: "/home/frank" @shell: "/bin/bash" @password: nil @system: false @manage_home: false
@non_unique: false @cookbook_name: "users" @recipe_name: "default" @enclosing_provider: nil>
 => <user[frank] @name: "frank" @noop: nil @before: nil @params: {} @provider: nil
@allowed_actions: [:nothing, :create, :remove, :modify, :manage, :lock, :unlock] @action: :create
@updated: false @updated_by_last_action: false @supports: {:manage_home=>false, :non_unique=>false}
@ignore_failure: false @retries: 0 @retry_delay: 2 @immediate_notifications: []
@delayed_notifications: [] @source_line: "/var/chef/cache/cookbooks/users/recipes/default.rb:11:in
`block in from_file'" @resource_name: :user @username: "frank" @comment: "Frank Belson" @uid: 2001
@gid: 0 @home: "/home/frank" @shell: "/bin/bash" @password: nil @system: false @manage_home: false
@non_unique: false @cookbook_name: "users" @recipe_name: "default" @enclosing_provider: nil>
```

Exercise: Show the resource *as you would write it if you were very, very verbose*

```
chef:recipe > puts resources("user[frank]").to_text
```

```
# Declared in /var/chef/cache/cookbooks/users/recipes/default.rb:11:in `block in
from_file'
user("frank") do
  action(:create)
  supports({:manage_home=>false, :non_unique=>false})
  retries(0)
  retry_delay(2)
  username("frank")
  comment("Frank Belson")
  uid(2001)
  gid(0)
  home("/home/frank")
  shell("/bin/bash")
  cookbook_name("users")
  recipe_name("default")
end
```

Exercise: Exit to the main context

```
chef:recipe > exit
```

```
chef >
```

Exercise: List all the nodes

```
chef > nodes.list
```

```
=> [node[target1.local]]
```

Exercise: Print all the node names and their environment

```
nodes.all { |n| "#{n.name}: #{n.chef_environment}" }
```

```
target1.local: production
 => [nil]
```

Exercise: Change the environment on every node to dev

```
nodes.transform(:all) { |n| n.chef_environment("dev") }
```

```
=> [node[target1.local]]
```

Exercise: From your workstation, use knife to do the same thing

```
$ knife exec -E 'nodes.transform(:all) { |n|
n.chef_environment("dev") }'
```

Has.. no output >:)

Exercise: Check it from search ...

```
$ knife search node '*:*' -a chef_environment
```

```
1 items found

chef_environment:  dev
id:                target1.local
```

Best Practice: Be Careful

- We will not save you from yourself
- You can do the same tricks on roles, data bags (and items), clients.. all kinds of stuff
- You can (and will) blow off your foot if you aren't careful

Questions

- What is Chef Shell?
- What are the execution contexts of Chef Shell? How do you know which one you are in?
- How do you run a recipe that is in your nodes run list from Chef Shell?
- What does “transform” do?

Building LWRPs

Course Objectives

- Understand the components of LWRPs
- Be familiar with the Resource DSL
- Be familiar with the Provider DSL
- Build a LWRP from scratch
- Look at examples of other LWRPs

OPSCODE

When to use Lightweight Resources and Providers

- LWRPs are used when you want to abstract a repeated pattern of behavior with a declarative interface that doesn't already exist in Chef
- Examples
  - Managing a service with a new supervision system (e.g. bluepill or daemontools)
  - Subsystem user management (e.g samba)
  - Sending a message to a RESTful API for a monitoring service with `http_request`

RULE THE CLOUD

Components of an LWRP

- LWRPs have two components: the resource and the provider
  - Resources are used in Recipes to declare the state to configure on our system
  - Providers configure that state on the system during convergence
- They are defined in the "resources" and "providers" directory of cookbooks

The Problem and the Success Criteria

- The Problem: We don't have anything cute and awesome on our system yet
- Success Criteria: We will create a mouse that tells us cute and awesome things about itself

Exercise: Create a cookbook named 'mouse'

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

Exercise: Create a cookbook named 'mouse'

```
$ knife cookbook create mouse
```

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

Exercise: Open the default recipe in your editor

```
#
# Cookbook Name:: users
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

Exercise: Open the default recipe in your editor

```
$ vi cookbooks/mouse/recipes/default.rb
```

```
#
# Cookbook Name:: users
# Recipe:: default
#
# Copyright 2012, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

Exercise: Open the default recipe in your editor

```
#
# Cookbook Name:: mou
# Recipe:: default
#
# Copyright 2012, YOU
#
# All rights reserved
#

mouse "Itchy"
```

- We call a new resource: “mouse”
- We create a new instance of that resource named “Itchy”

Exercise: Upload the mouse cookbook

```
$ knife cookbook upload mouse
```

```
Uploading mouse                    [0.0.1]
Uploaded 1 cookbook.
```

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

```
$ knife node run list add target1.local "recipe[mouse]"
```

```
run_list:
    role[webserver]
    recipe[mouse]
```

Example: How we would have done that before

```
$ knife node edit target1.local
```

```
1  {
2    "name": "target1.local",
3    "chef_environment": "_default",
4    "normal": {
5      "company": "opscode",
6      "tags": [
7
8      ],
9      "apache": {
10       "sites": {
11       }
12     }
13   },
14   "run_list": [
15     "role[webserver]",
16     "recipe[mouse]"
17   ]
18 }
19
```

Exercise: Run the Chef Client

```
================================================================================
Recipe Compile Error in /var/chef/cache/cookbooks/mouse/recipes/default.rb
================================================================================

NameError
---------
Cannot find a resource for mouse on ubuntu version 12.04

Cookbook Trace:
---------------
  /var/chef/cache/cookbooks/mouse/recipes/default.rb:10:in `from_file'

Relevant File Content:
----------------------
/var/chef/cache/cookbooks/mouse/recipes/default.rb:

  3:  # Recipe:: default
  4:  #
  5:  # Copyright 2012, YOUR_COMPANY_NAME
  6:  #
  7:  # All rights reserved - Do Not Redistribute
  8:  #
  9:
 10>> mouse "Itchy"
 11:

ERROR: Running exception handlers
FATAL: Saving node information to /var/chef/cache/failed-run-data.json
ERROR: Exception handlers complete
FATAL: Stacktrace dumped to /var/chef/cache/chef-stacktrace.out
```

Exercise: Run the Chef Client

```
$ sudo chef-client
```

```
================================================================================
Recipe Compile Error in /var/chef/cache/cookbooks/mouse/recipes/default.rb
================================================================================

NameError
---------
Cannot find a resource for mouse on ubuntu version 12.04

Cookbook Trace:
---------------
  /var/chef/cache/cookbooks/mouse/recipes/default.rb:10:in `from_file'

Relevant File Content:
----------------------
/var/chef/cache/cookbooks/mouse/recipes/default.rb:

  3:  # Recipe:: default
  4:  #
  5:  # Copyright 2012, YOUR_COMPANY_NAME
  6:  #
  7:  # All rights reserved - Do Not Redistribute
  8:  #
  9:
 10>> mouse "Itchy"
 11:

ERROR: Running exception handlers
FATAL: Saving node information to /var/chef/cache/failed-run-data.json
ERROR: Exception handlers complete
FATAL: Stacktrace dumped to /var/chef/cache/chef-stacktrace.out
```

Exercise: Create resources for the mouse cookbook

```
$ touch ./cookbooks/mouse/resources/default.rb
$ knife cookbook upload mouse
```

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

Exercise: Re-run the Chef Client

```
INFO: Processing mouse[Itchy] action nothing (mouse::default line 10)

================================================================================
Error executing action `nothing` on resource 'mouse[Itchy]'
================================================================================

ArgumentError
-------------
Cannot find a provider for mouse[Itchy] on ubuntu version 12.04

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

 10: mouse "Itchy"

Compiled Resource:
------------------
# Declared in /var/chef/cache/cookbooks/mouse/recipes/default.rb:10:in `from_file'

mouse("Itchy") do
  action :nothing
  retries 0
  retry_delay 2
  cookbook_name "mouse"
  recipe_name "default"
end

ERROR: Running exception handlers
FATAL: Saving node information to /var/chef/cache/failed-run-data.json
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
INFO: Processing mouse[Itchy] action nothing (mouse::default line 10)

================================================================================
Error executing action `nothing` on resource 'mouse[Itchy]'
================================================================================

ArgumentError
-------------
Cannot find a provider for mouse[Itchy] on ubuntu version 12.04

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

 10: mouse "Itchy"

Compiled Resource:
------------------
# Declared in /var/chef/cache/cookbooks/mouse/recipes/default.rb:10:in `from_file'

mouse("Itchy") do
  action :nothing
  retries 0
  retry_delay 2
  cookbook_name "mouse"
  recipe_name "default"
end

ERROR: Running exception handlers
FATAL: Saving node information to /var/chef/cache/failed-run-data.json
```

Exercise: Create providers for the mouse cookbook

```
$ touch ./cookbooks/mouse/providers/default.rb
$ knife cookbook upload mouse
```

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

Exercise: Re-run the Chef Client

```
 INFO: Processing mouse[Itchy] action nothing
(mouse::default line 10)
 INFO: Chef Run complete in 2.50350876 seconds
 INFO: Running report handlers
 INFO: Report handlers complete
```

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
 INFO: Processing mouse[Itchy] action nothing
(mouse::default line 10)
 INFO: Chef Run complete in 2.50350876 seconds
 INFO: Running report handlers
 INFO: Report handlers complete
```

The Resource DSL

- Two methods used: `actions` and `attribute`
  - `actions` specifies allowed actions that can be used in a recipe
  - `attribute` specifies a new parameter attribute for the resource

The Provider DSL

- Only one method used: `action`
  - Specify the `action` name with a Ruby symbol
  - Name maps to allowed actions defined in the resource `actions`
- You can also re-use any Chef Resources inside your providers

Exercise: Set an action in our mouse default recipe

```
10 mouse "Itchy" do
11   action :say
12 end
~
```

- We call resource: “mouse”
- With name “Itchy”
- With action “:say” in the parameter block

Exercise: Set an action in our mouse default recipe

```
$ vi ./cookbooks/mouse/recipes/default.rb
```

```
10 mouse "Itchy" do
11   action :say
12 end
~
```

- We call resource: “mouse”
- With name “Itchy”
- With action “:say” in the parameter block

Exercise: Create allowed actions for the mouse resource

```
1 actions :say
~
```

- We have one allowed resource action

Exercise: Create allowed actions for the mouse resource

```
$ vi ./cookbooks/mouse/resources/default.rb
```

```
1 actions :say
~
```

- We have one allowed resource action

Exercise: Create provider for the ":say" action

```
1 action :say do
2   puts "My name is #{new_resource.name}"
3 end
~
```

- When our mouse resource calls "`action :say`", execute this block

Exercise: Create provider for the ":say" action

```
$ vi ./cookbooks/mouse/providers/default.rb
```

```
action :say do
  puts "My name is #{new_resource.name}"
end
~
```

- When our mouse resource calls "`action :say`", execute this block

Exercise: Upload the mouse cookbook

```
$ knife cookbook upload mouse
```

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

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
 INFO: Processing mouse[Itchy] action say
(mouse::default line 10)
My name is Itchy
 INFO: Chef Run complete in 3.676864182 seconds
 INFO: Running report handlers
 INFO: Report handlers complete
```

Exercise: Use a Chef Resource within your provider

```
1 action :say do
2   log "My name is #{new_resource.name}"
3 end
~
```

- The `log` resource uses Chef's logger object to print messages at Chef::Config[log_level]

Exercise: Use a Chef Resource within your provider

```
$ vi ./cookbooks/mouse/providers/default.rb
```

```
1 action :say do
2   log "My name is #{new_resource.name}"
3 end
~
```

- The `log` resource uses Chef's logger object to print messages at Chef::Config[log_level]

Exercise: Upload the mouse cookbook

```
$ knife cookbook upload mouse
```

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

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
 INFO: Processing mouse[Itchy] action say
(mouse::default line 10)
 INFO: Processing log[My name is Itchy] action write
(/var/chef/cache/cookbooks/mouse/providers/default.rb
line 2)
 INFO: My name is Itchy
 INFO: Chef Run complete in 3.825978405 seconds
 INFO: Running report handlers
 INFO: Report handlers complete
```

Exercise: Create attribute parameters for the mouse resource

```
actions :say

attribute :given_name, :name_attribute => true
attribute :phrase, :default => "squeak"
attribute :tail, :default => true, :kind_of => [ TrueClass, FalseClass ]
```

- `attribute` takes the name of the attribute and an (optional) hash of validation parameters
- These validation parameters are specific to the LWRP Resource DSL

Exercise: Create attribute parameters for the mouse resource

```
$ vi ./cookbooks/mouse/resources/default.rb
```

```
1 actions :say
2
3 attribute :given_name, :name_attribute => true
4 attribute :phrase, :default => "squeak"
5 attribute :tail, :default => true, :kind_of => [ TrueClass, FalseClass ]
```

- `attribute` takes the name of the attribute and an (optional) hash of validation parameters
- These validation parameters are specific to the LWRP Resource DSL

Resource Validation Parameters

```
actions :say

attribute :given_name, :name_attribute => true
attribute :phrase, :default => "squeak"
attribute :tail, :default => true, :kind_of => [ TrueClass, FalseClass ]
```

- **:name_attribute** sets given_name to the resource name
- **:default** sets a default value for this parameter
- **:kind_of** ensures the value is a particular class
- More validation parameters listed on the docs!

Exercise: Extend provider actions for new resource parameters

```
1 action :say do
2   log "My name is #{new_resource.given_name}"
3   unless new_resource.phrase =~ /^squeak$/
4     log "#{new_resource.phrase}"
5   end
6   log "I #{new_resource.tail ? 'do' : 'do not' } have a tail"
7 end
```

- The unless statement checks for anything except a squeak
- A Ruby ternary operation checks for the existence of a tail

Exercise: Extend provider actions for new resource parameters

```
$ vi ./cookbooks/mouse/providers/default.rb
```

```ruby
action :say do
  log "My name is #{new_resource.given_name}"
  unless new_resource.phrase =~ /^squeak$/
    log "#{new_resource.phrase}"
  end
  log "I #{new_resource.tail ? 'do' : 'do not' } have a tail"
end
```

- The unless statement checks for anything except a squeak
- A Ruby ternary operation checks for the existence of a tail

Exercise: Modify the recipe to use our new parameters

```
10 mouse "Itchy" do
11   action :say
12   phrase "SQUEAK!"
13   tail false
14 end
```

Exercise: Modify the recipe to use our new parameters

```
$ vi ./cookbooks/mouse/recipes/default.rb
```

```
10 mouse "Itchy" do
11   action :say
12   phrase "SQUEAK!"
13   tail false
14 end
```

Exercise: Upload the mouse cookbook

```
$ knife cookbook upload mouse
```

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

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
 INFO: Processing mouse[Itchy] action say (mouse::default line 10)
 INFO: Processing log[My name is Itchy] action write (/var/chef/
cache/cookbooks/mouse/providers/default.rb line 2)
 INFO: My name is Itchy
 INFO: Processing log[SQUEAK!] action write (/var/chef/cache/
cookbooks/mouse/providers/default.rb line 4)
 INFO: SQUEAK!
 INFO: Processing log[I do not have a tail] action write (/var/
chef/cache/cookbooks/mouse/providers/default.rb line 6)
 INFO: I do not have a tail
 INFO: Chef Run complete in 2.748432652 seconds
 INFO: Running report handlers
 INFO: Report handlers complete
```

Exercise: Set the default action to use for this resource

```
5 attribute :tail, :default =
6
7 def initialize(*args)
8   super
9   @action = :say
10 end
~
```

- `action` is an instance variable
- its value must be a Ruby symbol

Exercise: Set the default action to use for this resource

```
$ vi ./cookbooks/mouse/resources/default.rb
```

```
5 attribute :tail, :default =
6
7 def initialize(*args)
8   super
9   @action = :say
10 end
~
```

- `action` is an instance variable
- its value must be a Ruby symbol

Exercise: Add another mouse to our recipe

```
mouse "Itchy" do
  action :say
  phrase "squeak"
  tail false
end

mouse "Mighty Mouse" do
  phrase "Here I come to save the day!"
  tail true
end
```

Exercise: Add another mouse to our recipe

```
$ vi ./cookbooks/mouse/recipes/default.rb
```

```
10 mouse "Itchy" do
11   action :say
12   phrase "squeak"
13   tail false
14 end
15
16 mouse "Mighty Mouse" do
17   phrase "Here I come to save the day!"
18   tail true
19 end
```

Exercise: Upload the mouse cookbook

```
$ knife cookbook upload mouse
```

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

Exercise: Re-run the Chef Client

```
$ sudo chef-client
```

```
 INFO: Processing mouse[Itchy] action say (mouse::default line 10)
 INFO: Processing log[My name is Itchy] action write (/var/chef/cache/cookbooks/mouse/
providers/default.rb line 2)
 INFO: My name is Itchy
 INFO: Processing log[I do not have a tail] action write (/var/chef/cache/cookbooks/mouse/
providers/default.rb line 6)
 INFO: I do not have a tail
 INFO: Processing mouse[Mighty Mouse] action say (mouse::default line 16)
 INFO: Processing log[My name is Mighty Mouse] action write (/var/chef/cache/cookbooks/mouse/
providers/default.rb line 2)
 INFO: My name is Mighty Mouse
 INFO: Processing log[Here I come to save the day!] action write (/var/chef/cache/cookbooks/
mouse/providers/default.rb line 4)
 INFO: Here I come to save the day!
 INFO: Processing log[I do have a tail] action write (/var/chef/cache/cookbooks/mouse/
providers/default.rb line 6)
 INFO: I do have a tail
 INFO: Chef Run complete in 4.138446206 seconds
 INFO: Running report handlers
 INFO: Report handlers complete
```

LWRP examples in Opscode Cookbooks

- Opscode uses LWRPs in its open-source cookbooks
  - aws_elastic_ip
  - aws_ebs_volume
  - bluepill_service
  - mysql_database
  - php_pear
  - riak_cluster
  - samba_user
  - windows_package
- http://docs.opscode.com/lwrp.html

Questions

- What are the components of an LWRP?
- What are the resource DSL's two methods?
- What method is used in providers to create actions?
- What are validation parameters? What are two examples?
- What does ":name_attribute" mean? How is it used?

Chef Report Handlers

OPSCODE

RULE THE CLOUD

Objectives

- Quick reintroduction to Chef report handlers
- Writing custom report handlers
- Distributing handlers to nodes
- Configuring Chef to use custom handlers

Handlers

- Report handlers run at the end of a chef-client run
- They are Ruby programs that can read the status information of your chef-client run
- Report handlers are always executed when the chef-client run finishes
- Error handlers are executed when the chef-client run does not finish successfully

Finding New Handlers

- The Chef community shares handlers, some are listed on the Chef documentation
  - http://docs.opscode.com/essentials_handlers_available_handlers.html

Writing Custom Handlers

- Your custom handlers can make use of any ruby gem or library you need
- You distribute them to your nodes with the chef_handler cookbook

OPSCODE

Install the chef_handler cookbook

```
Downloading chef_handler from the cookbooks site at version 1.1.4 to /Users/me/
work/chef-repo/chef_handler-1.1.4.tar.gz
Cookbook saved: /Users/me/work/chef-repo/chef_handler-1.1.4.tar.gz
```

OPSCODE

Install the chef_handler cookbook

```
$ knife cookbook site download chef_handler
```

```
Downloading chef_handler from the cookbooks site at version 1.1.4 to /Users/me/
work/chef-repo/chef_handler-1.1.4.tar.gz
Cookbook saved: /Users/me/work/chef-repo/chef_handler-1.1.4.tar.gz
```

Install the chef_handler cookbook

```
x chef_handler/
x chef_handler/attributes/
x chef_handler/CHANGELOG.md
x chef_handler/CONTRIBUTING
x chef_handler/files/
x chef_handler/LICENSE
x chef_handler/metadata.json
x chef_handler/metadata.rb
x chef_handler/providers/
x chef_handler/README.md
x chef_handler/recipes/
x chef_handler/resources/
x chef_handler/resources/default.rb
x chef_handler/recipes/default.rb
x chef_handler/recipes/json_file.rb
x chef_handler/providers/default.rb
x chef_handler/files/default/
x chef_handler/files/default/handlers/
x chef_handler/files/default/handlers/README
x chef_handler/attributes/default.rb
```

OPSCODE

Install the chef_handler cookbook

```
$ tar -xvzf chef_handler*.tar.gz -C cookbooks/
```

```
x chef_handler/
x chef_handler/attributes/
x chef_handler/CHANGELOG.md
x chef_handler/CONTRIBUTING
x chef_handler/files/
x chef_handler/LICENSE
x chef_handler/metadata.json
x chef_handler/metadata.rb
x chef_handler/providers/
x chef_handler/README.md
x chef_handler/recipes/
x chef_handler/resources/
x chef_handler/resources/default.rb
x chef_handler/recipes/default.rb
x chef_handler/recipes/json_file.rb
x chef_handler/providers/default.rb
x chef_handler/files/default/
x chef_handler/files/default/handlers/
x chef_handler/files/default/handlers/README
x chef_handler/attributes/default.rb
```

Upload chef_handler cookbook

```
Uploading chef_handler                [1.1.4]
upload complete
```

OPSCODE

Upload chef_handler cookbook

```
$ knife cookbook upload chef_handler
```

```
Uploading chef_handler              [1.1.4]
upload complete
```

Write a new email handler

- We're going to write a new handler that will email us when chef-client runs
- It will send us all of the resources that changed during the chef-client run
- This example uses the pony rubygem
  - http://rubydoc.info/gems/pony/1.4/frames
- http://docs.opscode.com/essentials_handlers_write.html

Introducing Library versus Application Cookbooks

- When using the chef_handler cookbook, all of your custom handlers **could** be delivered from the cookbooks/chef_handler/files/default/handlers directory
- There is a chef_handler/recipes/default.rb recipe to deliver all the files to /var/chef/handlers directory on your nodes
- **But…**

Introducing Library versus Application Cookbooks

- We could modify the chef_handler cookbook to add a recipe for the email handler
- However, we'd basically be "forking" it by doing so
- Instead, best to create our own cookbook and use chef_handler as a "library"
- Code reusability!

Exercise: Create a cookbook named 'email_handler'

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

Exercise: Create a cookbook named 'email_handler'

```
$ knife cookbook create email_handler
```

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

Write a new email handler

```
$ vi cookbooks/email_handler/recipes/default.rb
```

```
chef_gem “pony” do
  action :install
end
```

The chef_handler Resource

- **chef_handler** is a resource packaged with the chef_handler cookbook
- It has two **actions**, :enable and :disable
- It has three arguments
  - **source** - the file containing the handler code
  - **arguments** - any pieces of information needed to initialize the handler
  - **supports** - :report, :exception
- Defaults:
  - :enable
  - :report => true, :exception => true

OPSCODE

Add the handler object

```
$ vi cookbooks/email_handler/recipes/default.rb
```

```
chef_gem “pony” do
  action :install
end

include_recipe “chef_handler”

cookbook_file “#{node['chef_handler']['handler_path']}/email_handler.rb” do
  source “handlers/email_handler.rb”
  owner “root”
  group “root”
  mode “0755”
end

chef_handler “MyCompany::EmailMe” do
  source "#{node['chef_handler']['handler_path']}/email_handler.rb”
  arguments [node‘email_handler’][‘from_address’],
             node‘email_handler’][‘to_address’]]
  action :enable
end
```

http://tinyurl.com/camtrrw

RULE THE CLOUD

Set the attributes

```
$ vi cookbooks/email_handler/attributes/default.rb
```

```
default['email_handler']['from_address'] = "root@localhost"
default['email_handler']['to_address'] = "root@localhost"
```

Write the Handler

```
$ vi cookbooks/email_handler/files/default/handlers/email_handler.rb
```

```
1 require 'rubygems'
2 require 'pony'
3
4 module MyCompany
5   class EmailMe << Chef::Handler
6
```

- The module and class match what we defined as the name of the chef_handler in the recipe
- Gist: https://gist.github.com/4121693

The initialize Method

```
5   class EmailMe << Chef::Handler
6
7     def initialize(from_address, to_address)
8       @from_address = from_address
9       @to_address = to_address
10    end
11
```

- Initialize the handler with the arguments we passed in the definition
- You can create the method with any args you need to meet your requirements

The report Method

```
9       @to_address = to_address
10    end
11
12    def report
13      status = success? ? "Successful" : "Failed"
14      subject = "#{status} Chef run report from #{node.name}"
15      report_string = ""
16
```

- note we're starting a new method here

The updated_resources Hash

```
17      # report on changed resources
18      if ! run_status.updated_resources.empty?
19        # get some info about all the changed resources!
20        run_status.updated_resources.each do |r|
21          report_string += "The resource #{r.name} was changed in cookbook #{r.
   cookbook_name} at #{r.source_line}\n"
22        end
23      else
24        report_string += "No resources changed by chef-client\n"
25      end
```

- updated_resources records information about all the resources changed during a chef-client run
- read through this hash with .each, pull interesting information out about each resource

Finish email_handler.rb

```
24          report_string += "No resources changed by chef-client\n"
25        end
26
27        Pony.mail(:to => @to_address,
28                  :from => @from_address,
29                  :subject => subject,
30                  :body => report_string)
31      end
32    end
33 end
```

- Use Pony.mail to send a message containing info on changed resources

Update metadata!

```
$ vi cookbooks/email_handler/metadata.rb
```

```
name              “email_handler”
maintainer        "You"
maintainer_email "you@somewhere.com"
license           "Apache 2.0"
description       "Email me on every Chef run"
long_description IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version           “0.1.0”

depends “chef_handler”
```

Exercise: Upload the email_handler cookbook

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

Exercise: Upload the email_handler cookbook

```
$ knife cookbook upload email_handler
```

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

Add your new handler to your base role

```
name “base”
description “Base Server Role”
run_list “recipe[email_handler]”, “recipe[chef-
client::delete_validation]”, “recipe[chef-client]”,
“recipe[ntp]”, “recipe[motd]”, “recipe[users]”,
```

OPSCODE

Add your new handler to your base role

```
$ vi roles/base.rb
```

```
name “base”
description “Base Server Role”
run_list “recipe[email_handler]”, “recipe[chef-
client::delete_validation]”, “recipe[chef-client]”,
“recipe[ntp]”, “recipe[motd]”, “recipe[users]”,
```

OPSCODE

Best practice: Put exception handlers early

- Put exception handlers early in the run list - probably the first item
- That way, they are registered so that if any subsequent resource fails, they will still fire

RULE THE CLOUD

Upload the updated role

```
Updated Role base!
```

Upload the updated role

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

```
Updated Role base!
```

OPSCODE

Run chef-client

```
[2012-11-20T14:36:47-08:00] INFO: Enabling chef_handler[MyCompany::EmailMe] as a
report handler
[2012-11-20T14:36:47-08:00] INFO: Enabling chef_handler[MyCompany::EmailMe] as a
exception handler
```

OPSCODE

Run chef-client

```
$ sudo chef-client
```

```
[2012-11-20T14:36:47-08:00] INFO: Enabling chef_handler[MyCompany::EmailMe] as a
report handler
[2012-11-20T14:36:47-08:00] INFO: Enabling chef_handler[MyCompany::EmailMe] as a
exception handler
```

Read the Message using "mail"

- We have no mail client on this box, so let's just install one by hand so we can read the mail -- in real life we should do this with Chef
- sudo apt-get install heirloom-mailx

Read the Message Using “mail”

```
opscode:~$ sudo mail
Heirloom Mail version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N  1 mydefault@domain.loc  Tue Nov 20 14:43  21/1096  "Successful Chef run report from localhost"
& 1
Message  1:
From mydefault@domain.local  Tue Nov 20 14:43:07 2012
Return-Path: <mydefault@domain.local>
X-Original-To: root@localhost
Delivered-To: root@localhost
Date: Tue, 20 Nov 2012 14:43:07 -0800
From: mydefault@domain.local
To: root@localhost
Subject: Successful Chef run report from localhost
Content-Type: text/plain;
 charset=UTF-8
Status: R

The resource /var/chef/handlers was changed in cookbook chef_handler at /var/chef/cache/cookbooks/chef_handler/
recipes/default.rb:23:in `from_file'
The resource MyCompany::EmailMe was changed in cookbook email_handler at /var/chef/cache/cookbooks/
email_handler/recipes/email_handler.rb:9:in `from_file'
The resource ntpd was changed in cookbook ntp at /var/chef/cache/cookbooks/ntp/recipes/default.rb:33:in
`from_file'
The resource /etc/ntp.conf was changed in cookbook ntp at /var/chef/cache/cookbooks/ntp/recipes/default.rb:
38:in `from_file'
```

Summary

- Report and Exception handlers need a bit of set up to be helpful
- Use the chef_handler cookbook to deliver your custom site handlers and the chef_handler resource
- Your handlers have access to all of the resources and information available in your chef-client run
- Use Ruby to talk to ticketing systems, email, metrics, monitoring
- Example of one to send run data to Graphite: https://github.com/juliandunn/graphite_handler

Further Resources

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://opscode.com/
- http://community.opscode.com/
- http://docs.opscode.com/
- http://lists.opscode.com
- http://youtube.com/user/Opscode
- irc.freenode.net #chef

Food Fight Show

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

Local User Groups and Resources

- Philadelphia Rubyists http://phillyrb.org/
- Chef-NYC http://www.meetup.com/Chef-NYC/
- Philly DevOps http://phillydevops.org/

A Crash Course in Test-Driven Cookbook Development using Berkshelf, Vagrant and Virtualization

Beta material! Look out!

OPSCODE

RULE THE CLOUD

Chef

Advanced Cookbook Authoring

- These slides are "borrowed" from the "Advanced Cookbook Authoring" workshop at ChefConf 2013 next week
- Think of this as the "lite" and "unpolished" version of that course
- No typing, just me lecturing
  - You need more powerful laptops to do the work than you have at the moment

Dependencies

- Here's what we are asking of attendees:
- Training Requirements
  - Completed the **3-day Chef Fundamentals** course, or **equivalent**
- System Requirements:
  - Full **administrator** access to a development workstation running **Mac OSX 10.7 or 10.8**
  - RAM **4+GB**; CPU **i5+**; **SSD** recommended
  - **No** existing Ruby installed via **MacPorts**

The course is new, the tools are not

- We will eventually offer this as an actual course (1/2 day or 1 day)
  - Probably supporting Windows as well - course was developed for Mac OS X
- Think of this as bonus material you get for free because you finished so fast!

Objectives

- Build a **local Chef development environment** on a computer running **OSX**
- Learn to treat **cookbooks** as **build artifacts**
- Use **Berkshelf** for **dependency management**
- Use **Vagrant** for **rapid iteration**
- Gain familiarity with **source control best practices**
- Understand the **"Application Cookbook"** design pattern

Additional Software

- In addition to Chef, the course will make use of the following software:
  - Apple XCode + XCode CLI
  - Oracle VirtualBox (VMWare Fusion support TBD)
  - Vagrant
  - Berkshelf
  - Git
  - CentOS (RedHat clone) 6.x
  - VeeWee & Bento

Objectives

- Install all **software** needed for **Chef development** on your **local system**
  - ~~Compiler - Xcode 4 + CLI~~
  - ~~Source Control - Git~~
  - Editor - Sublime Text
  - Virtual Machine Supervisor - Virtualbox
  - Virtual Machine Integration - Vagrant
  - Opscode Chef Omnibus Installer
  - Ruby Gems - Berkshelf & Foodcritic

VM Supervisor: Virtualbox

Downloads - Oracle VM VirtualBox

https www.virtualbox.org/wiki/Downloads Reader

VirtualBox

search...

Login Preferences

Download VirtualBox

Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox platform packages**. The binaries are released under the terms of the GPL version 2.
  - **VirtualBox 4.2.6 for Windows hosts** x86/amd64
  - **VirtualBox 4.2.6 for OS X hosts** x86/amd64
  - **VirtualBox 4.2.6 for Linux hosts**
  - **VirtualBox 4.2.6 for Solaris hosts** x86/amd64
- **VirtualBox 4.2.6 Oracle VM VirtualBox Extension Pack** All supported platforms
  Support for USB 2.0 devices, VirtualBox RDP and PXE boot for Intel cards.
  See this chapter from the User Manual for an introduction to this Extension

About
Screenshots
Downloads
Documentation
End-user docs
Technical docs
Contribute
Community

VM Supervisor: Virtualbox

https://www.virtualbox.org/wiki/Downloads

- Creator: Oracle
- License: GPL v2
  - Free for all users on any platform
- We use Virtualbox instead of VMWare Fusion because the Vagrant VMWare Fusion plugin costs money

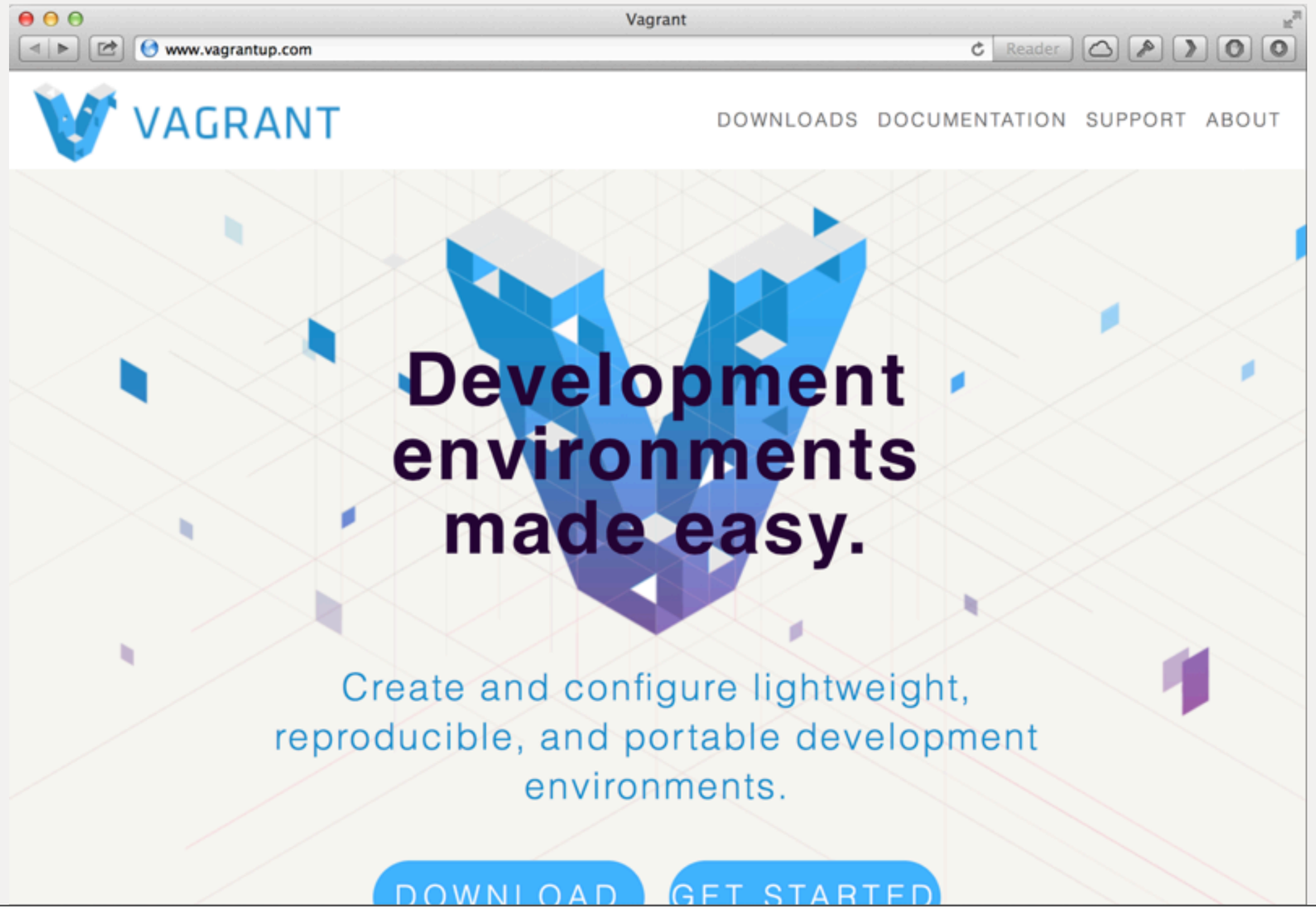
Exercise: Install VirtualBox

https://www.virtualbox.org/wiki/Downloads

- Download & Install VirtualBox

OPSCODE | Chef

VM Integration: Vagrant



VM Integration: Vagrant

http://downloads.vagrantup.com

- Creator: Hashicorp
- License: MIT License
  - Free for all users on any platform
- Written in Ruby
- Can provision to Virtualbox, VMWare Fusion, AWS, LXC, Rackspace Cloud, etc.
- Highly extensible
- Plugin support

VM Integration: Vagrant

http://downloads.vagrantup.com

- Download & Install Vagrant

Install Chef

- You already have this on your system

Install Foodcritic & Berkshelf

```
$ sudo /opt/chef/embedded/bin/gem install berkshelf foodcritic
```

```
$ sudo /opt/chef/embedded/bin/gem install berkshelf foodcritic
Fetching: timers-1.1.0.gem (100%)
Fetching: celluloid-0.13.0.gem (100%)
Fetching: i18n-0.6.1.gem (100%)
Fetching: multi_json-1.7.2.gem (100%)
...
Successfully installed nokogiri-1.5.9
Successfully installed rak-1.4
Successfully installed polyglot-0.3.3
Successfully installed treetop-1.4.12
Successfully installed foodcritic-2.0.1
24 gems installed
$
```

Symlink berks & foodcritic binaries

```
$ ln -s /opt/chef/embedded/bin/berks /usr/local/bin
$ ln -s /opt/chef/embedded/bin/foodcritic /usr/local/bin
```

```
$ ln -s /opt/chef/embedded/bin/berks /usr/local/bin
$ ln -s /opt/chef/embedded/bin/foodcritic /usr/local/bin

$ which foodcritic
/usr/local/bin/foodcritic

$ which berks
/usr/local/bin/berks
$
```

Here's that part that we can do

- foodcritic is the tool that we **can** play with in this survey course
- So let's go ahead and install that

```
C:\opscode\chef\embedded\bin\gem install foodcritic
```

Install Berkshelf Vagrant Plugin

```
$ vagrant plugin install berkshelf-vagrant
```

```
$ vagrant plugin install berkshelf-vagrant
Installing the 'berkshelf-vagrant' plugin. This can
take a few minutes...
Installed the plugin 'berkshelf-vagrant (1.0.6)'!

$ vagrant plugin list
berkshelf-vagrant (1.0.6)
$
```

Software Development Workflows For Systems Administrators: A Crash Course

OPSCODE

Objectives

- Understand the stages of the typical software development workflow
- Understand how to properly manage Chef code
- Understand the following concepts:
  - Source code
  - Compiler
  - Artifacts
  - Testing
- Understand Chef cookbook testing requirements

RULE THE CLOUD

Devops Is A Two-Way Street!

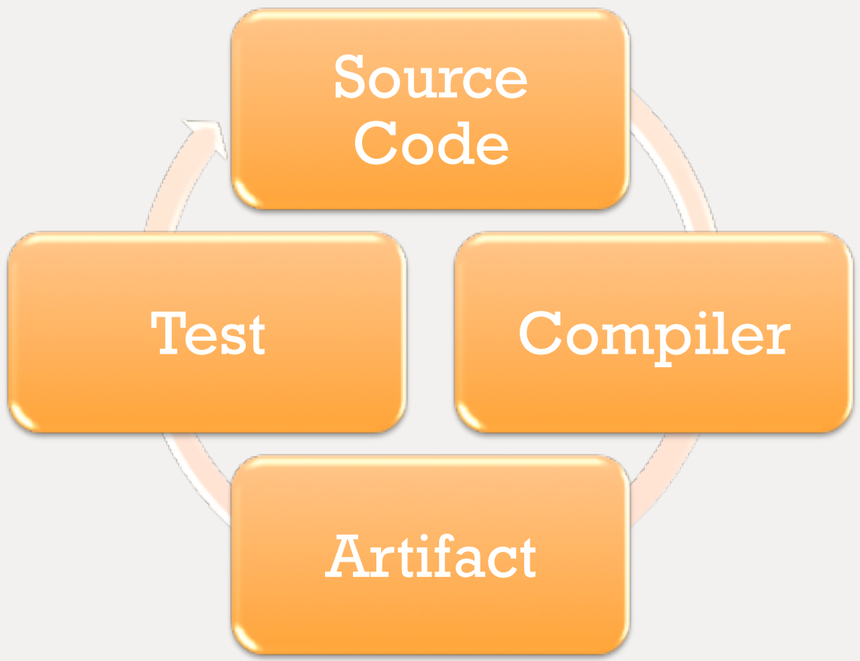
- **Systems Administrators** often talk and think about **DevOps** solely in terms of getting **developers** to act more like **operations**.
  - It's great when **developers** care about **uptime**!
  - It's great when **developers** care about **scaling**!
  - It's great when **developers** care about **deployment**!

Devops Is A Two-Way Street!

- This approach only tells **half the story**. **Systems Administrators** may have lots to teach **developers**, but **also** have as much or **more** to **learn** from **developers** as well!

OPSCODE

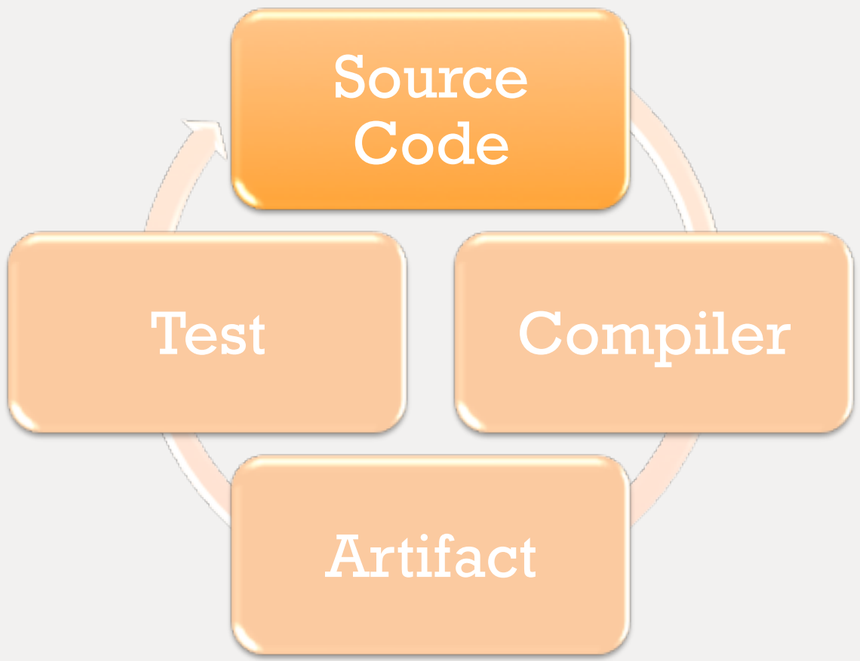
Typical Development Workflow



- Typical Software Development Workflow:
  - **Write** Source
  - **Compile** Source
  - **Deploy** Artifacts
  - Write **Tests**
  - Run **Tests**

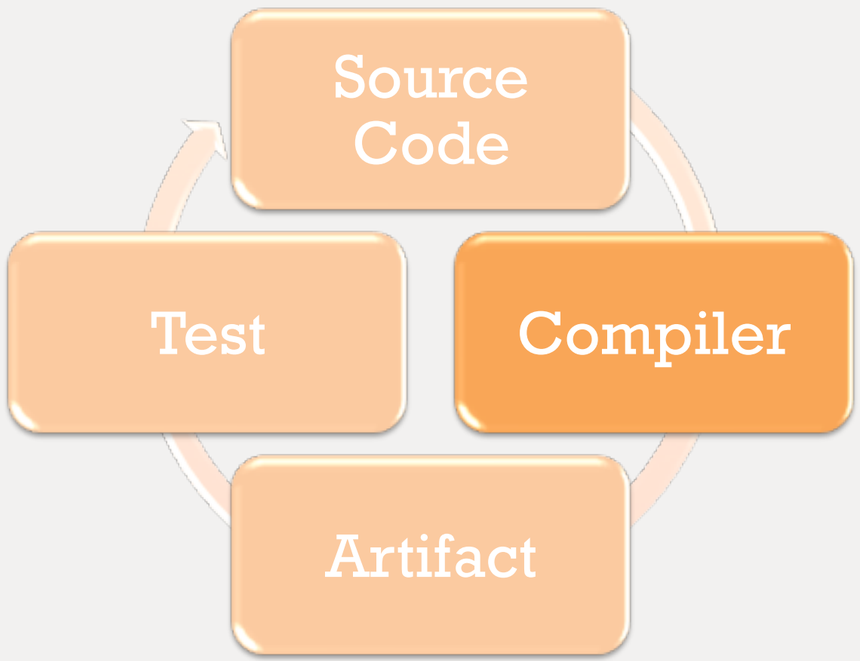
RULE THE CLOUD

Typical Development Workflow



- Source Code
  - The recipe for a computer program
  - Edited directly
  - Managed via Source Control Software

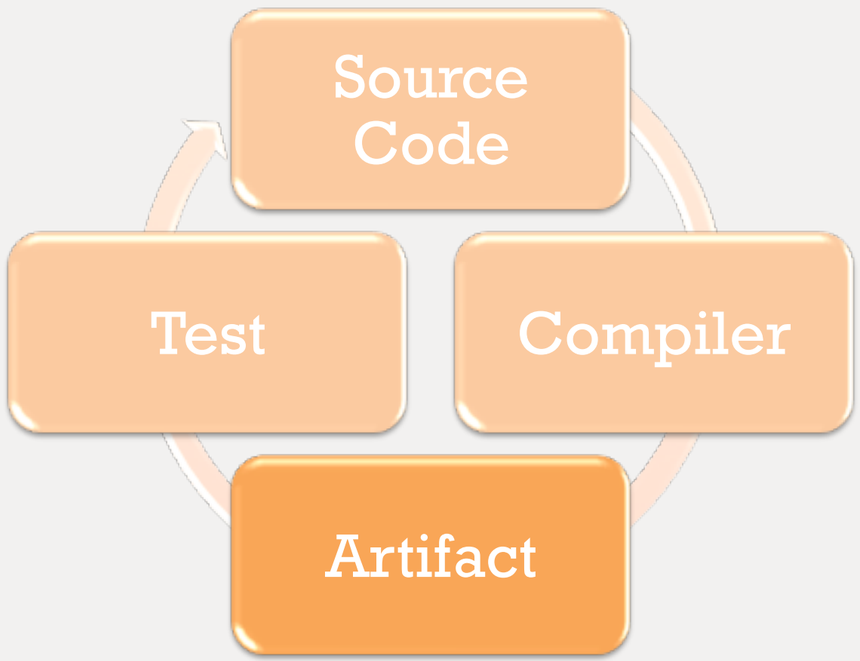
Typical Development Workflow



- Compiler
  - Takes source code and converts it into executable programs

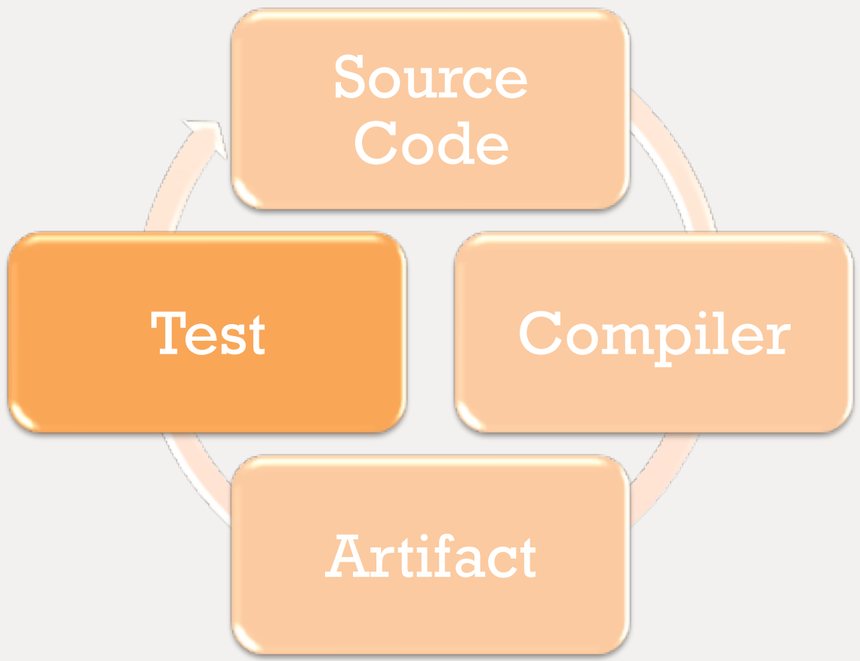
OPSCODE

Typical Development Workflow



- Artifact
  - Artifacts are executable programs created by compilers.
  - Compiled artifacts cannot be edited directly. Source code must be changed and re-compiled to produce a new build artifact.

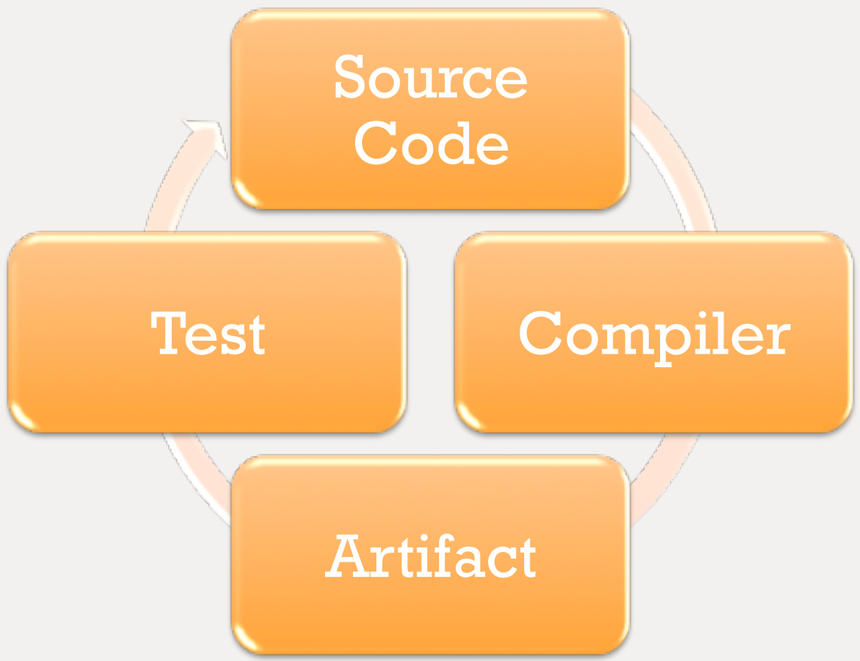
Typical Development Workflow



- Testing
  - Developers write tests to verify that their code works as intended
  - These tests are run at different stages of the code lifecycle to ensure correctness

OPSCODE

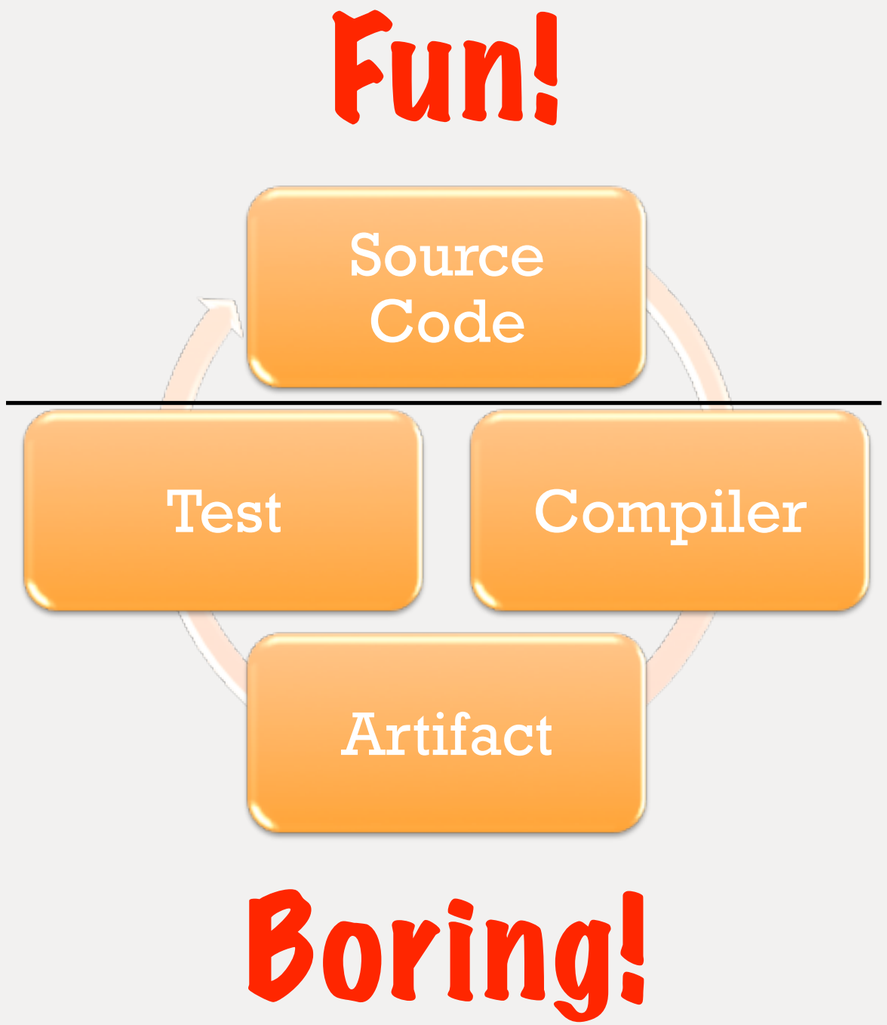
Typical Development Workflow



- When developing software, most time isn't actually spent coding
- 10-second changes to source code can take minutes to vet
  - Compiling code
  - Deploying code
  - Writing & Running tests

RULE THE CLOUD

Typical Development Workflow



- Things that are fun:
  - Designing programs!
  - Writing source code!
- Things that are boring:
  - Compiling code
  - Deploying artifacts
  - Running tests

OPSCODE

Developing for Chef: Cookbooks Are Source Code

- Chef Cookbook source code is just like the source code for other programs.
  - Changes are **tracked and versioned** with source control software such as git
  - Code is **inert** until it has been deployed
  - Code can be used to **re-create the desired system configuration** from scratch

RULE THE CLOUD

OPSCODE

Developing for Chef: Cookbooks Aren't Pre-Compiled

- Because **Ruby** is an **interpreted** language, Chef code is **not compiled** until it is executed by the Chef-Client
- There is **no lengthy compilation step** between **committing** Chef Cookbooks and **deploying** them

RULE THE CLOUD

Developing for Chef: Cookbooks Are Artifacts

- Chef Cookbook artifacts are just like the artifacts for other programs.
  - Cookbooks are **deployed**
  - Cookbooks are **released with versions**
  - Cookbooks are **immutable** once deployed
  - Cookbooks are **packaged for distribution**
  - Cookbooks can **depend** on one another

OPSCODE

Developing for Chef: Cookbook Are Artifacts

"This means something! This is important!"

Close Encounters of the Third Kind, copyright 1977, Columbia Pictures

Developing for Chef: Cookbook Are Artifacts

Depending entirely on context, Chef **Cookbooks** can be either **source code** OR **artifacts**.

"This means something! This is important!"

Close Encounters of the Third Kind, copyright 1977, Columbia Pictures

Developing for Chef: Cookbook Are Artifacts

- This sounds obvious, but it's easy to overlook: Treating cookbooks **only as source code** or **only as artifacts** will lead to **long-term problems** when using Chef cookbooks to manage **infrastructure**.
- Because Chef **cookbooks are source code and artifacts**, depending on the context, they have all the properties of each.

Developing for Chef: Cookbook Are Artifacts

A **good** Chef **workflow** treats cookbooks **appropriately** depending on **context**

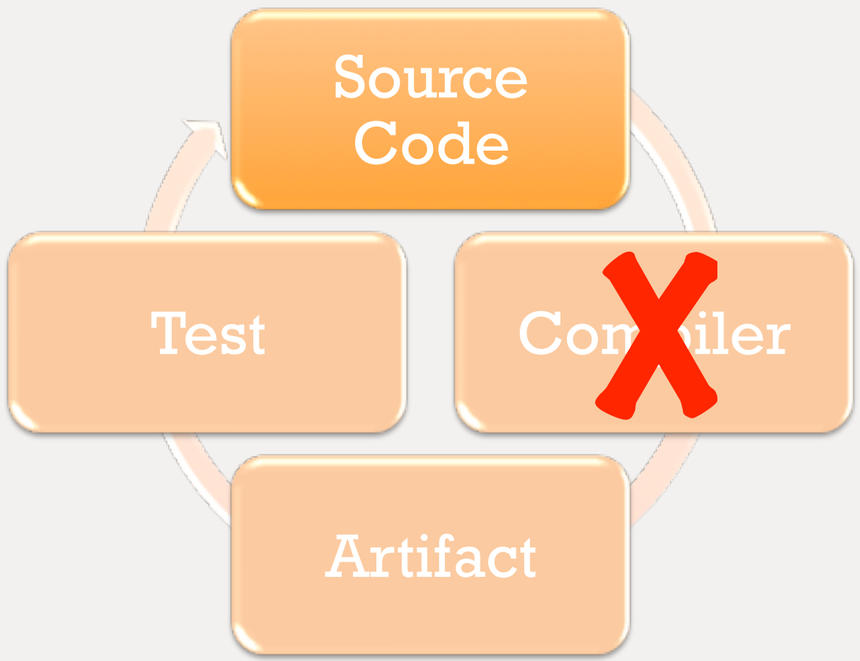
- Cookbooks are source!
  - **Versioned**
  - Stored in a **source repository**
  - **Inert** while in development
- Cookbooks are artifacts!
  - **Versioned**
  - **Deployed** to a Chef Server
  - **Executable** by Chef-client once deployed

A brief aside about **deploying** cookbook **artifacts:**

- Common question: "How do I **automatically** keep my **Chef Code Repository** and my **Chef Server** Artifact Repository in **sync**?"
- Answer: The same way that one would automatically keep a code repo and an artifact repo in sync with any other software released to production. Deploy with care!

OPSCODE

Developing For Chef: Cookbooks as Source Code

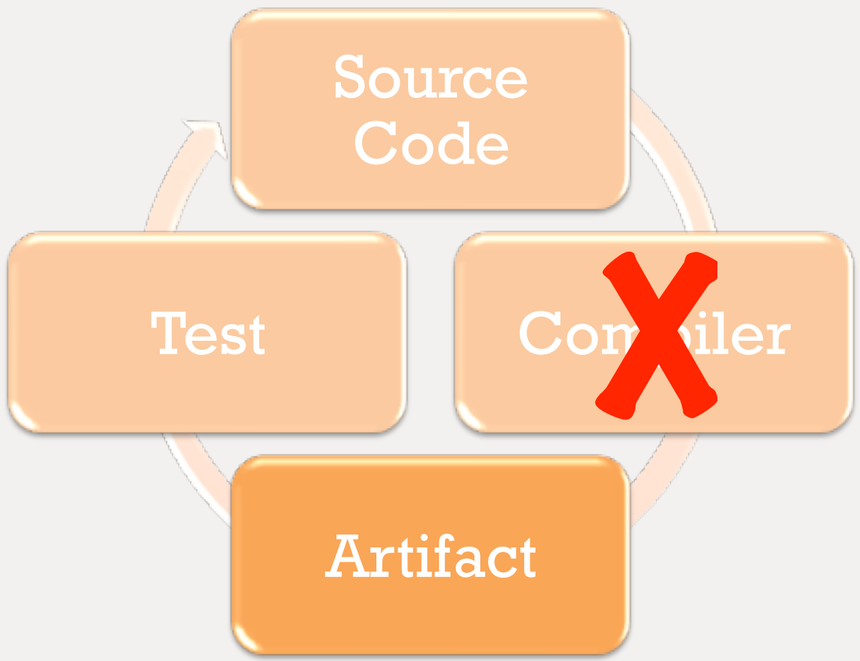


- Cookbooks are **Source!**
  - `knife cookbook create`
  - Code: `recipes/default.rb`
  - Version & dependencies: `metadata.rb`
  - `git commit`
  - `git tag`

RULE THE CLOUD

OPSCODE

Developing For Chef: Cookbooks as Artifacts



- Cookbooks are **Artifacts!**
  - `knife cookbook upload`
  - Immutable once deployed, can only be replaced
  - Version & dependencies: `metadata.rb`
  - `chef-client`

RULE THE CLOUD

Developing for Chef: Testing Cookbooks

Developing for Chef: Testing Cookbooks

- Three things necessary to test a Chef cookbook:

Developing for Chef: Testing Cookbooks

- Three things necessary to test a Chef cookbook:
  1. Need: A place to **store** the cookbook **artifact**.

Developing for Chef: Testing Cookbooks

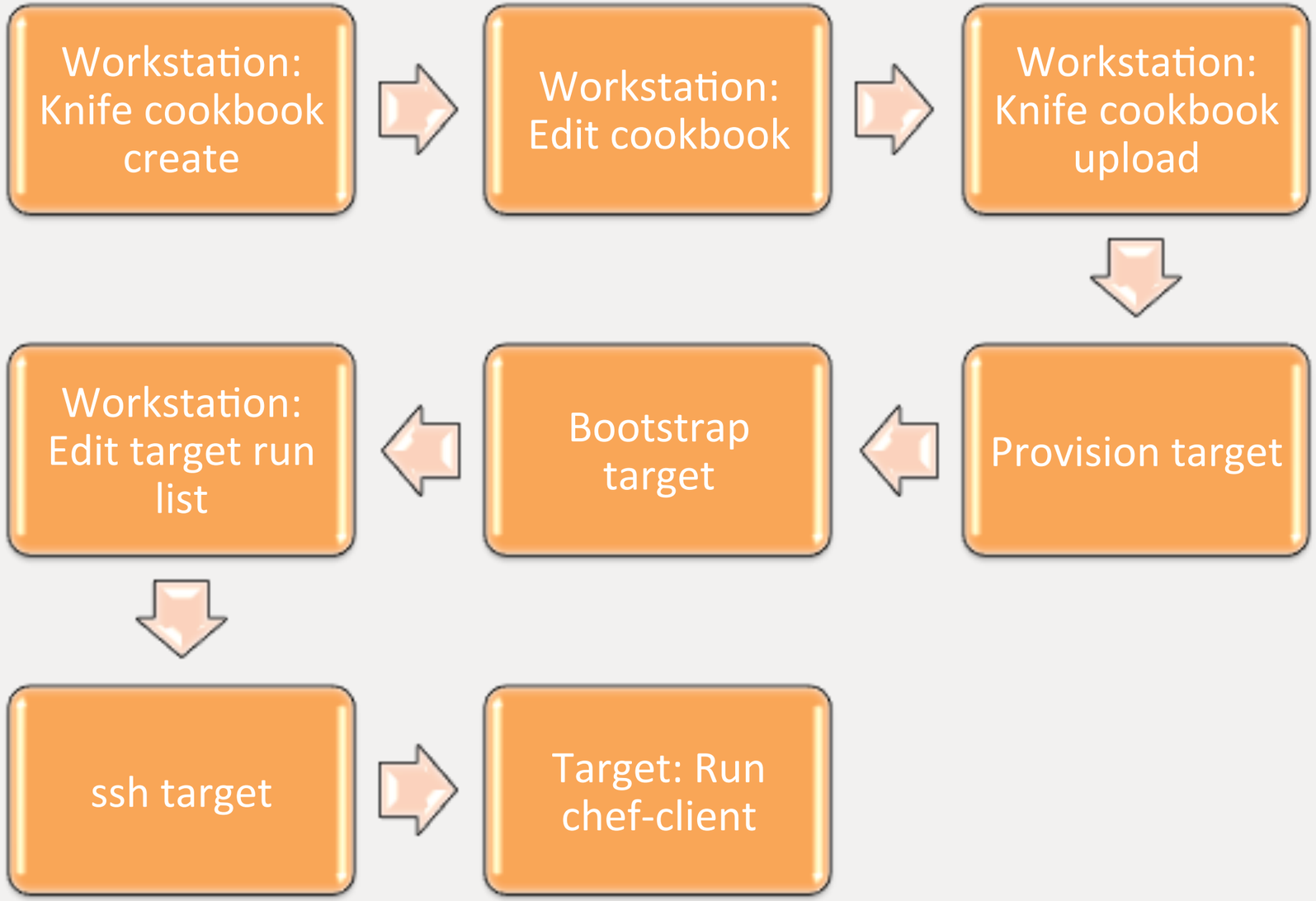
- Three things necessary to test a Chef cookbook:
  1. Need: A place to **store** the cookbook **artifact**.
  2. Need: A **Chef-client** with access to the cookbook.

Developing for Chef: Testing Cookbooks

- Three things necessary to test a Chef cookbook:
  1. Need: A place to **store** the cookbook **artifact**.
  2. Need: A **Chef-client** with access to the cookbook.
  3. Need: A **target server** running the same OS as production.

OPSCODE

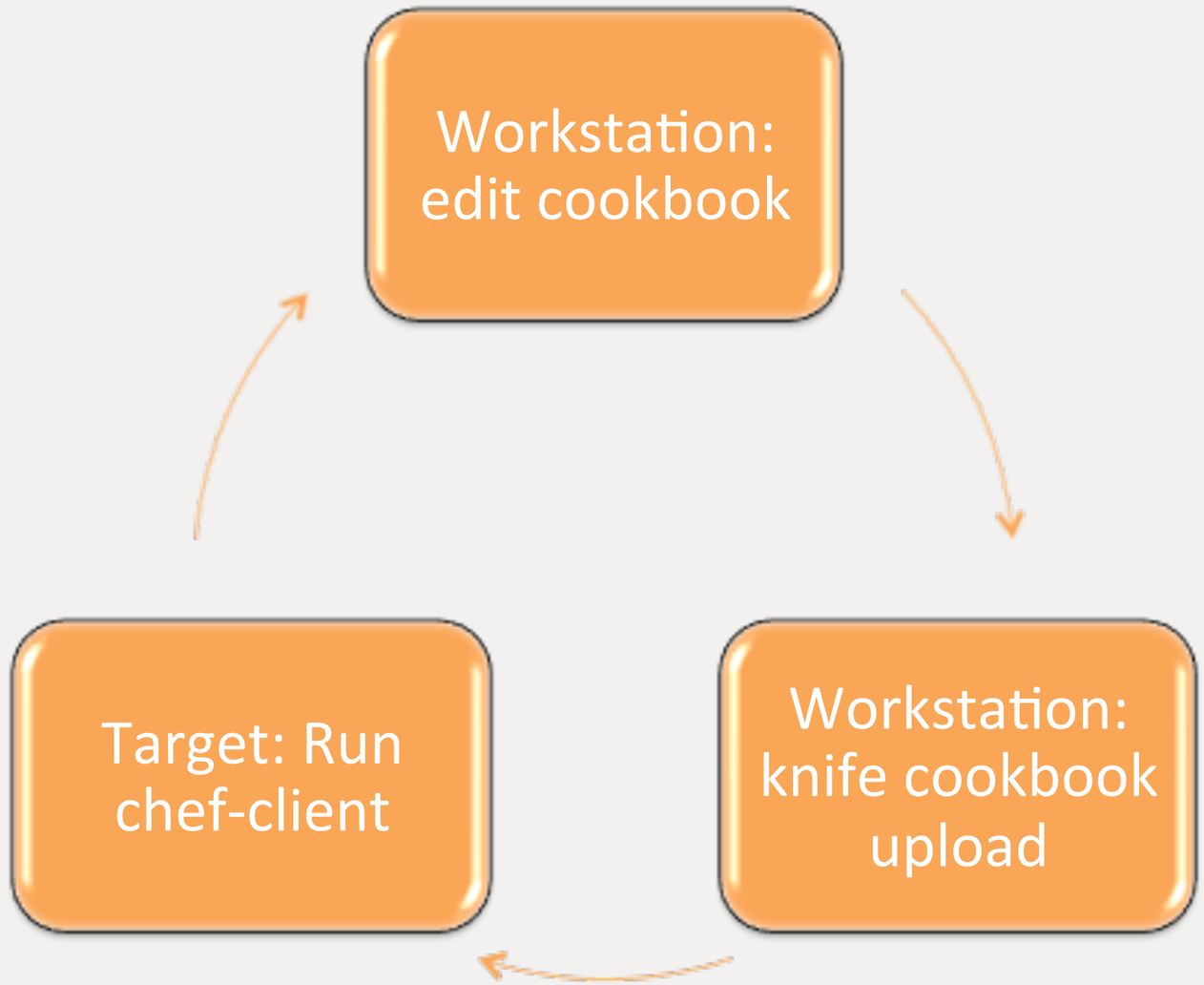
Testing a Chef Cookbook: First Run



RULE THE CLOUD

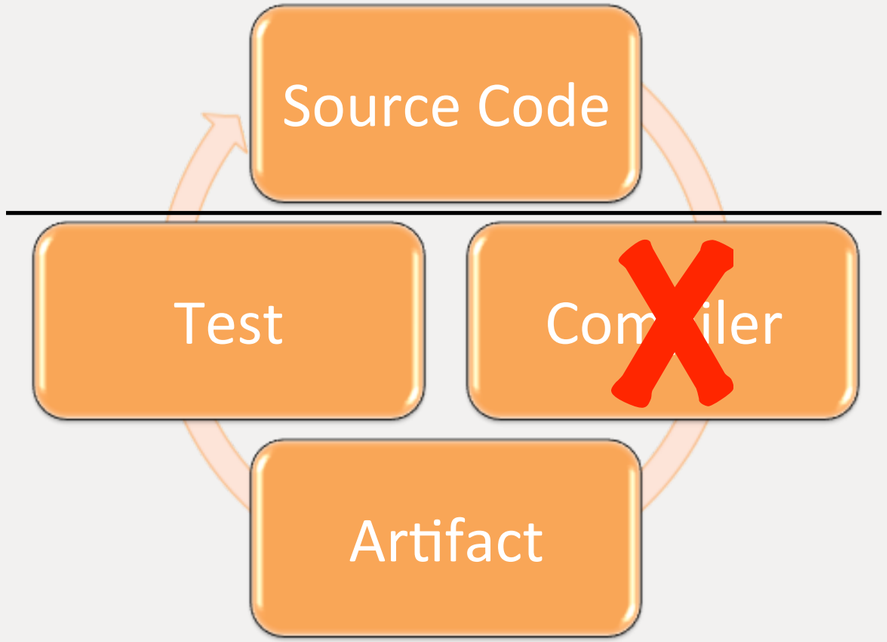
OPSCODE

Testing a Chef Cookbook: Subsequent Runs



RULE THE CLOUD

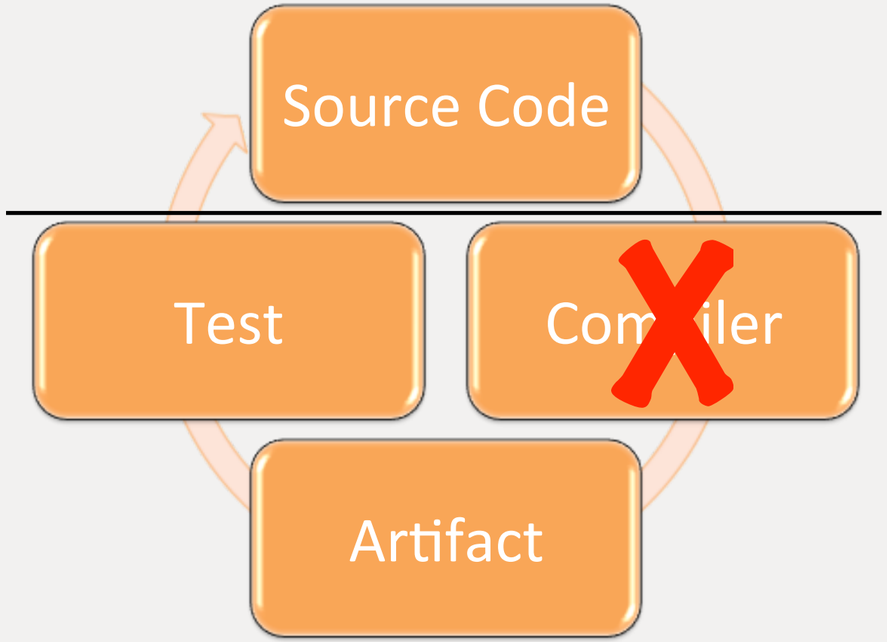
Developing for Chef: Testing Cookbooks



- Our happiness as developers hinges on iterating through the bottom part of this loop as rapidly as possible
- We're spending all our time on the boring side of the process, and almost no time on the good stuff!

Developing for Chef: Testing Cookbooks

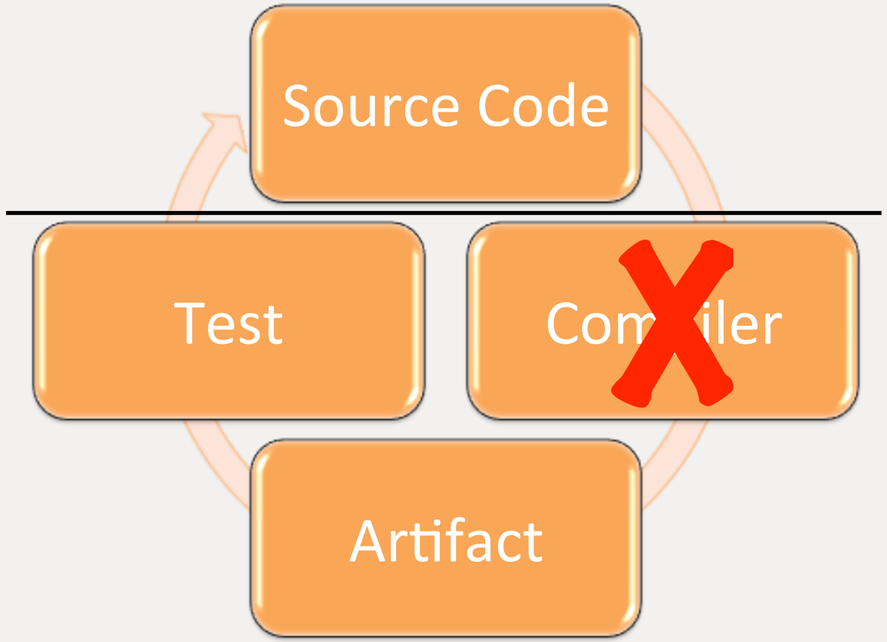
Less Fun!



- Our happiness as developers hinges on iterating through the bottom part of this loop as rapidly as possible
- We're spending all our time on the boring side of the process, and almost no time on the good stuff!

Developing for Chef: Testing Cookbooks

Less Fun!



More Boring!

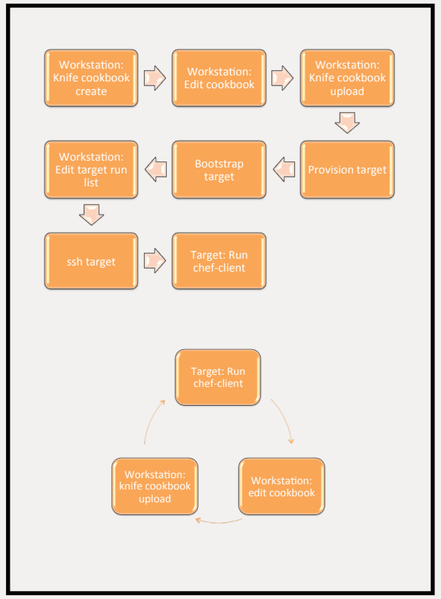
- Our happiness as developers hinges on iterating through the bottom part of this loop as rapidly as possible
- We're spending all our time on the boring side of the process, and almost no time on the good stuff!

Developing for Chef: Testing Cookbooks

- Our happiness as developers hinges on iterating through the bottom part of this loop as rapidly as possible
- We're spending all our time on the boring side of the process, and almost no time on the good stuff!

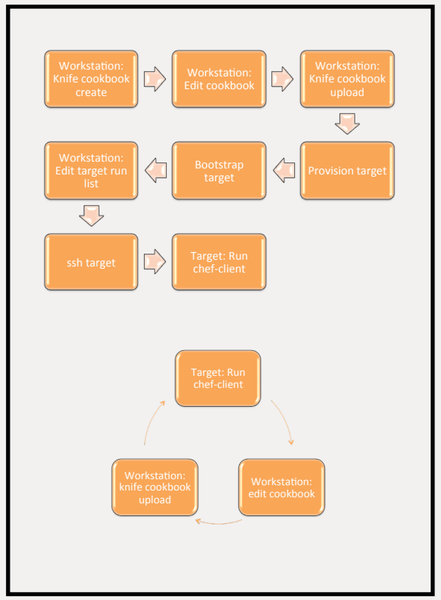
What if…

What if…



…we could automate all of this…

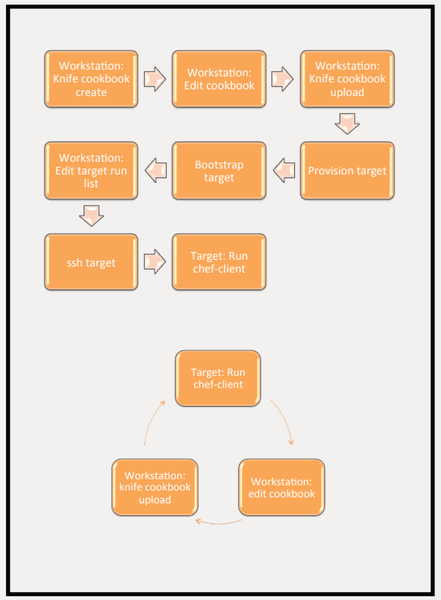
What if...



...to run entirely on this...

...we could automate all of this...

What if...



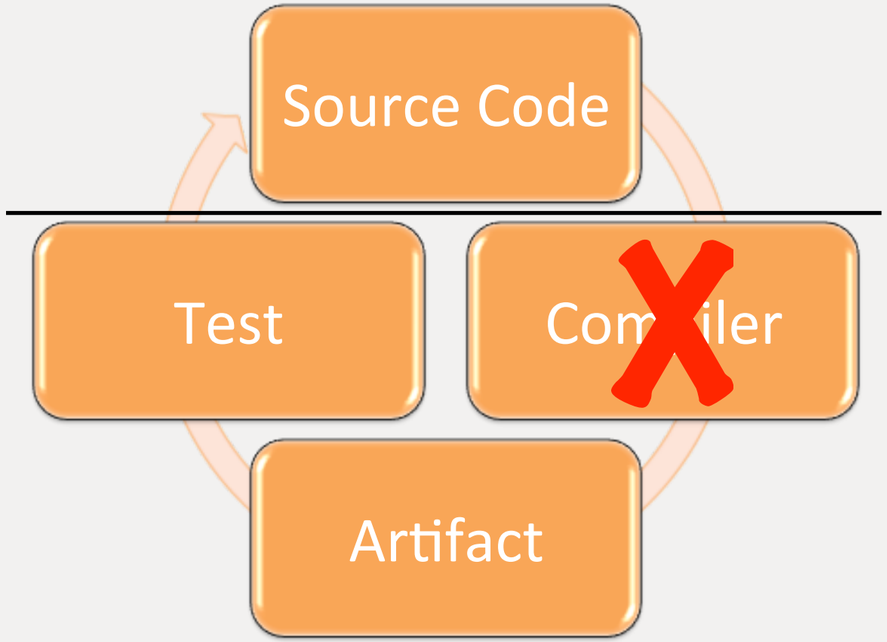
...we could automate all of this...

...to run entirely on this...

...even while aboard this?

Developing for Chef: Rapid Iteration

More Fun!



Less Boring!

- Faster iteration would mean less time waiting around for cookbook deploys and tests
- Easier Maintenance
- More frequent testing for better code
- Business needs met more quickly

OPSCODE

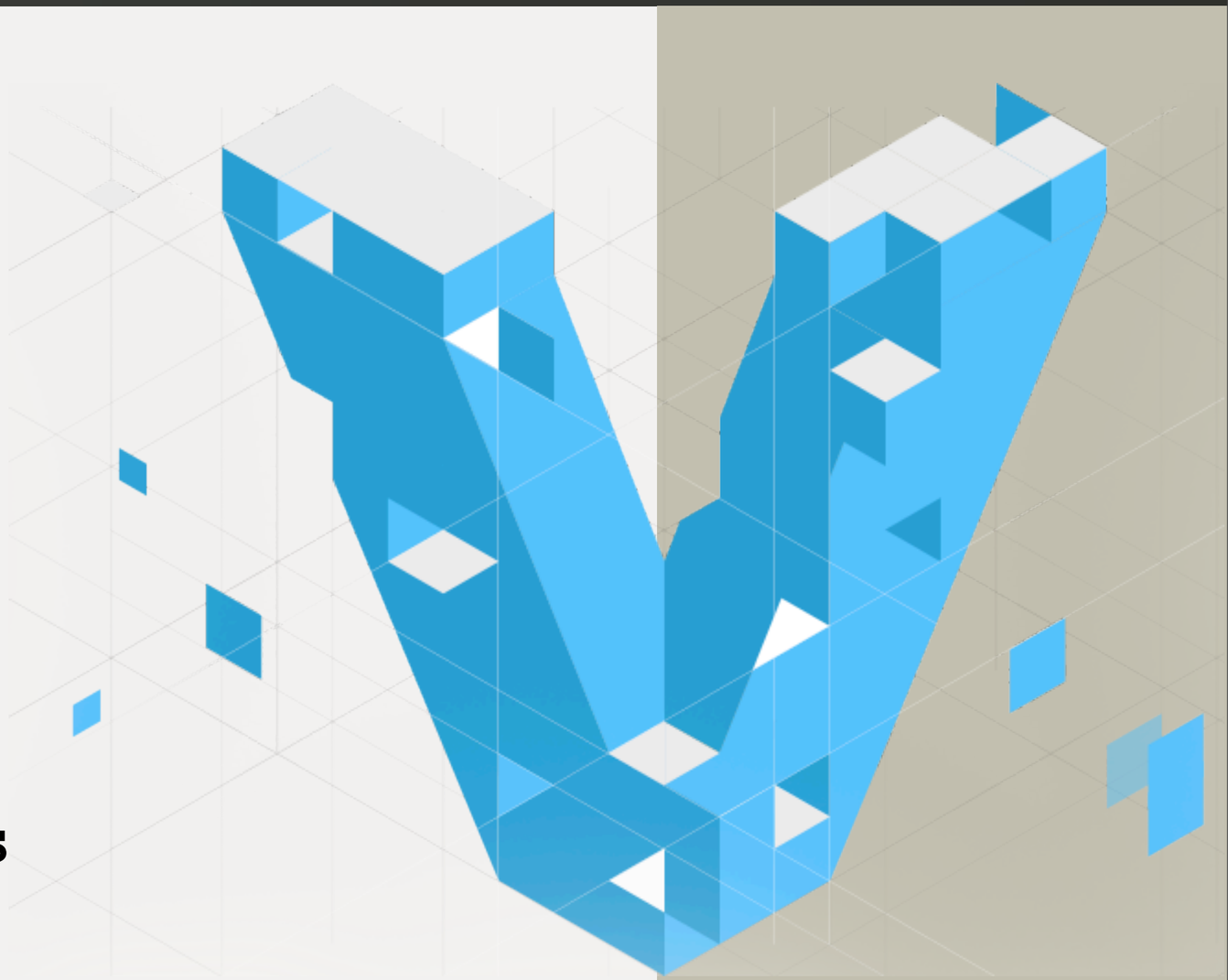
Sneak Preview: Berkshelf

- Cookbook **manager**
  - No more maintaining copies of unmodified community cookbooks in a personal repo!
- **Automatically resolves** all cookbook **dependencies**
- Artifact **installer**
- If you're familiar with **bundler**, berks is **easy**.
- Integrates tightly with...

OPSCODE

Sneak Preview: Vagrant

- Easily create and manipulate VirtualBox **Virtual Machines**
- Create VMs **unique** to each project
- **Rebuild** the whole dev / test environment **from scratch** in seconds
- **Automatically test** Chef recipes
- **Built-in** Chef-solo & Chef-Client **provisioners**



RULE THE CLOUD

Questions

- What are the 4 stages in the software development workflow?
- Which stage is not necessary when working in Chef? Why?
- Cookbooks are both _____ & _____.
- Should cookbooks be stored in a source code repo, or in an artifact repo?
- Which 3 things are necessary to test a cookbook?

Import a Virtual Machine Image

OPSCODE

Objectives

- Understand the Vagrant 'box' file model for re-usable virtual machine images
- Import the Opscode-built CentOS 6.3 image for use in development

RULE THE CLOUD

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a virtual machine image to provision for our local test environment.
- Success Criteria:
  - A CentOS 6.3 image is ready for use.

RULE THE CLOUD

OPSCODE

About Box Files

- Vagrant Boxes are **base system images**, similar to AMIs or VMWare Templates
- Box files include a **customized** .vmdk **disk image**, an .ovf xml document describing the .vmdk image, and a Vagrantfile
- **Customization** is kept to a **minimum**, box files are as close to **out-of-the-box** images as possible.
- Box files are **stored** by Vagrant at **~/.vagrant.d/**
- Vagrant **interacts** with boxes through the **'vagrant box'** command.

RULE THE CLOUD

OPSCODE

Box File Customization

- Vagrant Boxes are **minimal**, and **customized** as **little** as possible - let **Chef** do the work!
  - Box files include a **user** named **vagrant**
    - Has **common** vagrant ssh **public key**
    - **passwordless sudo** (password is 'vagrant')
  - Box files include **Chef** for **provisioning**
  - Box files include **VirtualBox** guest **additions**
  - **Build** your own **easily** with **veewee**!
    - https://github.com/jedi4ever/veewee/

OPSCODE

Opscode Bento

https://github.com/opscode/bento

- **Bento** is a project that encapsulates **Veewee definitions** for **building** Vagrant **baseboxes**. Opscode uses these boxes **internally** for testing Hosted Chef, Private Chef and our open source cookbooks.
- Both the files used to create the box images, and the box images themselves, are available for free

Install A Vagrant Box

https://github.com/opscode/bento

```
$ vagrant box add <name> <location>
```

```
$ vagrant box add opscode-centos-6.3 '~/Downloads/opscode_centos-6.3_chef-11.2.0.box'
Downloading with Vagrant::Downloaders::File...
Copying box to temporary location...
Extracting box...
Cleaning up downloaded box...
Successfully added box 'opscode-centos-6.3' with provider 'virtualbox'!
$
```

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a virtual machine image to provision for our local test environment.
- Success Criteria:
  - ~~A CentOS 6.3 image is ready for use.~~

RULE THE CLOUD

OPSCODE

Some Notes on Box Files

- Vagrant boxes are **artifacts**
  - They can be defined by source in **veewee**, **compiled** to artifacts, and those artifacts can be shared
- Box files **only** need to be **downloaded once**.
  - Other projects **using** the **same box image** will use the **cache** in the ~/vagrant.d folder
- Vagrant boxes are **not suitable** for **production**
  - Because these images include a privileged user with a **shared key** and **no password**, they are **only** suitable for local **development** and **testing**.

RULE THE CLOUD

Questions

- What are box files?
- How are box files customized?
- How can one easily define and build personal Vagrant box images?
- Are these images suitable for use in staging or production?
- Which Github project includes the definitions used to build the box images with which Opscode tests and develops Chef?

Create a Cookbook, Berkshelf-style

OPSCODE

Section Objectives

- Configure Berkshelf
- Use Berkshelf to create a cookbook repo skeleton
- Do an initial test with Vagrant
- Learn about Vagrant up, Vagrant destroy, and Vagrant provision
- Use Foodcritic to test code correctness
- Start documentation-driven development
- Use Git to set source control checkpoints

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a properly configured repository with all supporting files to create the myface application cookbook.
- Success Criteria:
  - Berkshelf is properly configured.
  - The myface repository is initialized.
  - The VM test environment is provisioned.
  - The README file is updated.
  - Foodcritic passes with zero errors

RULE THE CLOUD

Exercise: Configure Berkshelf

- About Berkshelf configuration
  - Berkshelf will run just fine without a config file, and will substitute default values if no config file is present
- Common settings
  - Chef server information, similar to knife.rb
  - Change Berkshelf behavior flags
  - Change defaults used when Berkshelf generates cookbooks

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a properly configured repository with all supporting files to create the myface application cookbook.
- Success Criteria:
  - ~~Berkshelf is properly configured.~~
  - The myface repository is initialized.
  - The VM test environment is provisioned.
  - The README file is updated.
  - Foodcritic passes with zero errors

RULE THE CLOUD

OPSCODE

Berkshelf replaces ‘cookbook create’

- We used to do “knife cookbook create”
- Berkshelf is going to replace that.

RULE THE CLOUD

Exercise: Create the MyFace Cookbook

```
~/Development$ berks cookbook myface --chef-minitest
```

```
~/Development$ berks cookbook myface --chef-minitest
      create  myface/files/default
      ...
      create  myface/Berksfile
      ...
         run  git init from "./myface"
      create  myface/files/default/tests/minitest/support
      create  myface/files/default/tests/minitest/default_test.rb
      create  myface/files/default/tests/minitest/support/helpers.rb
      create  myface/Gemfile
      create  myface/Vagrantfile
Installing minitest-handler (0.1.7) from site: 'http://cookbooks.opscode.com/api/v1/
cookbooks'
Using myface (0.1.0) at path: '/Users/charlesjohnson/Development/myface'
Installing chef_handler (1.1.4) from site: 'http://cookbooks.opscode.com/api/v1/
cookbooks'
$
```

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a properly configured repository with all supporting files to create the myface application cookbook.
- Success Criteria:
  - ~~Berkshelf is properly configured.~~
  - ~~The myface repository is initialized.~~
  - The VM test environment is provisioned.
  - The README file is updated.
  - Foodcritic passes with zero errors

RULE THE CLOUD

Exercise: Test MyFace

```
$ cd myface
myface$ vagrant up
```

```
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
[default] Importing base box 'opscode-ubuntu-12.04'...
[default] Matching MAC address for NAT networking...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[Berkshelf] Updating Vagrant's berkshelf: '/Users/charlesjohnson/.berkshelf/
vagrant/berkshelf-20130331-66007-8tjtk'
[Berkshelf] Using minitest-handler (0.1.7)
[Berkshelf] Using myface (0.1.0) at path: '/Users/charlesjohnson/Development/
berkshelf/myface'
[Berkshelf] Using chef_handler (1.1.4)
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
...
```

Exercise: Test MyFace

```
$ cd myface
myface$ vagrant up
```

```
...
[default] Running provisioner: VagrantPlugins::Chef::Provisioner::ChefSolo...
Generating chef JSON and uploading...
Running chef-solo...
stdin: is not a tty
[2013-03-31T22:48:34+00:00] INFO: *** Chef 11.2.0 ***
[2013-03-31T22:48:34+00:00] INFO: Setting the run_list to ["recipe[minitest-
handler::default]", "recipe[myface::default]"] from JSON
[2013-03-31T22:48:34+00:00] INFO: Run List is [recipe[minitest-handler::default],
recipe[myface::default]]
[2013-03-31T22:48:34+00:00] INFO: Run List expands to [minitest-handler::default,
myface::default]
[2013-03-31T22:48:34+00:00] INFO: Starting Chef Run for myface-berkshelf
[2013-03-31T22:48:34+00:00] INFO: Running start handlers
[2013-03-31T22:48:34+00:00] INFO: Start handlers complete.
...
```

Exercise: Test MyFace

```
$ cd myface
myface$ vagrant up
```

```
...
[2013-03-31T22:48:51+00:00] INFO: Chef Run complete in 16.90671831 seconds
[2013-03-31T22:48:51+00:00] INFO: Running report handlers
Run options: -v --seed 32328
# Running tests:
minitesthandler::default#test_0001_runs_no_tests =
0.00 s =
.
myface::default#test_0001_runs_no_tests_by_default =
0.00 s =
.
Finished tests in 0.003538s, 565.3082 tests/s, 0.0000 assertions/s.
2 tests, 0 assertions, 0 failures, 0 errors, 0 skips

[2013-03-31T22:48:51+00:00] INFO: Report handlers complete
$
```

OPSCODE

What Just Happened?

- We just created a **new** Chef cookbook **skeleton** with a **built-in test suite**
- A local **git repo** was **initialized** to hold our new Chef cookbook
- A **virtual machine** environment was automatically **described** for **testing** our Chef cookbook
- We **provisioned** and started that **virtual machine**
- We **test**-ran our new Chef **cookbook** in this **isolated**, virtualized **test environment**
- And we did it all ***in three commands***

RULE THE CLOUD

New Kids In Town

- "**berkshelf cookbook**" creates the same files as "**knife cookbook create**", but with a couple of new additions:
  - **Vagrantfile**
  - **Berksfile**
  - With the **--chef-minitest** command, also:
    - **files/default/tests/minitest/default_test.rb**
    - **files/default/tests/minitest/support/helpers.rb**

OPSCODE

What is the Vagrantfile

- The **Vagrantfile** is a **makefile** that describes all of the information necessary to deploy a local virtual development environment
  - VM Image **Name**
  - VM Image **Location**
  - Virtualization **Settings** - Network, CPU, RAM, etc.
- The Vagrantfile also contains **provisioning** Information - **Chef recipes**, shell scripts, etc.

RULE THE CLOUD

Exercise: Inspect the Vagrantfile

```
myface$ subl Vagrantfile
```

```
Vagrant.configure("2") do |config|
  # All Vagrant configuration is done here. The most common configuration
  # options are documented and commented below. For a complete reference,
  # please see the online documentation at vagrantup.com.

  config.vm.hostname = "myface-berkshelf"

  # Every Vagrant virtual environment requires a box to build off of.
  config.vm.box = "opscode-centos-6.3"

  # The url from where the 'config.vm.box' box will be fetched if it
  # doesn't already exist on the user's system.
  config.vm.box_url = "https://opscode-vm.s3.amazonaws.com/vagrant/
opscode_centos-6.3_chef-11.2.0.box"
```

Exercise: Inspect the Vagrantfile

```
myface$ subl Vagrantfile
```

```
config.vm.provision :chef_solo do |chef|
    chef.json = {
      :mysql => {
        :server_root_password => 'rootpass',
        :server_debian_password => 'debpass',
        :server_repl_password => 'replpass'
      }
    }

    chef.run_list = [
        "recipe[myface::default]",
        "recipe[minitest-handler::default]"
    ]
  end
end
```

What is the Berksfile

- The Berksfile describes the locations of all cookbooks that are depended on by the current cookbook.
- With this information, Berkshelf can automatically & recursively resolve all dependencies for the cookbook.

Exercise: Inspect the Berksfile

```
myface$ subl Berksfile
```

```
site :opscode

group :integration do
  cookbook "minitest-handler"
end

metadata
```

OPSCODE

The Problem and the Success Criteria

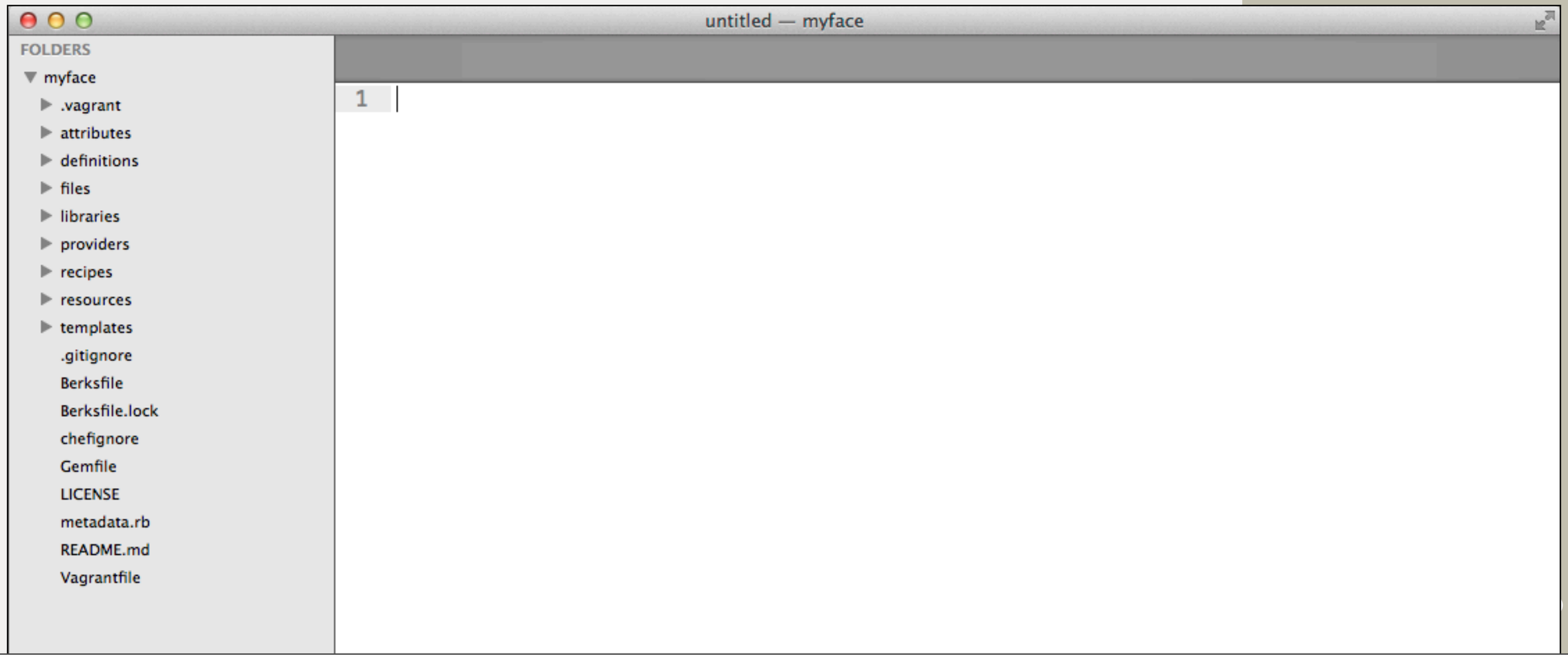
- The Problem:
  - We need a properly configured repository with all supporting files to create the myface application cookbook.
- Success Criteria:
  - ~~Berkshelf is properly configured.~~
  - ~~The myface repository is initialized.~~
  - ~~The VM test environment is provisioned.~~
  - The README file is updated.
  - Foodcritic passes with zero errors

RULE THE CLOUD

OPSCODE

Exercise: Update README.md

```
myface$ subl .
```



OPSCODE

Exercise: Update README.md

```
myface$ subl README.md
```

```
# myface cookbook

# Requirements
Chef 11.04 or higher
CentOS 6.3

# Usage
Example cookbook written to show Application Cookbook pattern

# Attributes
None (yet)

# Recipes
default.rb

# Author
Author:: Charles Johnson (charles@opscode.com)
```

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a properly configured repository with all supporting files to create the myface application cookbook.
- Success Criteria:
  - ~~Berkshelf is properly configured.~~
  - ~~The myface repository is initialized.~~
  - ~~The VM test environment is provisioned.~~
  - ~~The README file is updated.~~
  - Foodcritic passes with zero errors

RULE THE CLOUD

What is the FoodCritic

- Foodcritic is a **code linting** tool for Chef
  - Checks **code correctness**
  - Catches **common mistakes** before they cause **problems**
  - Extensible & Configurable
    - **Create** new **rulesets**
    - **Ignore** existing **rulesets**
  - **Integrates** nicely into **CI pipelines**

Exercise: Run Foodcritic

```
myface$ foodcritic .
```

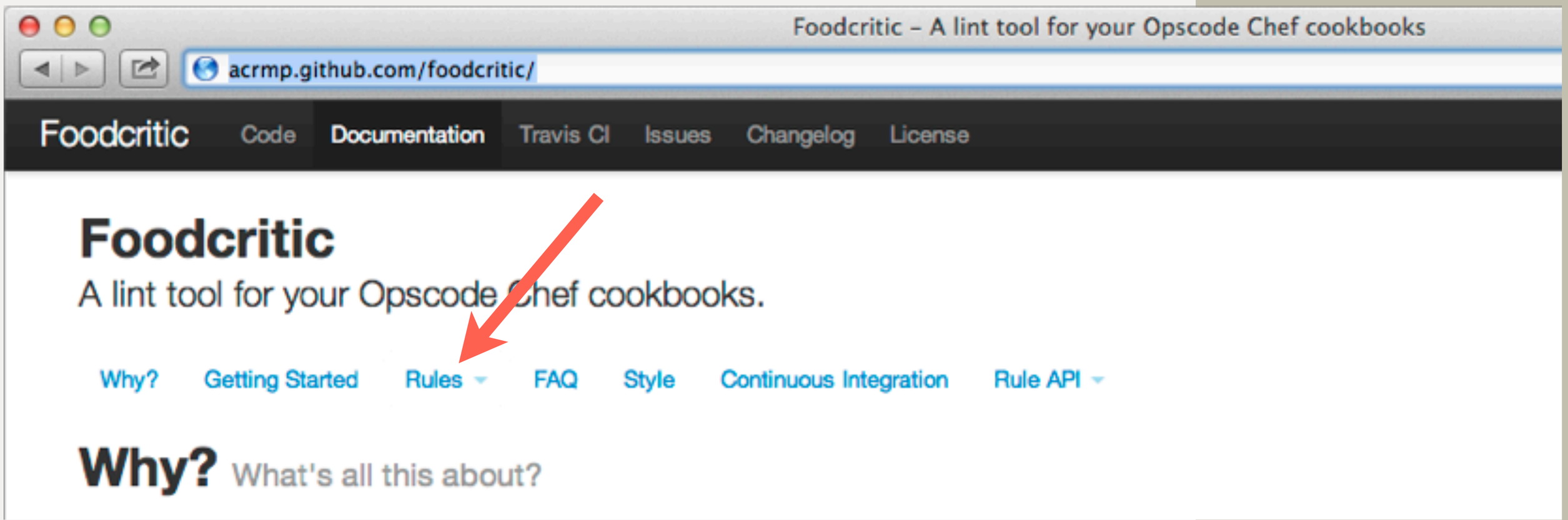
```
$ foodcritic .
FC008: Generated cookbook metadata needs updating:
./metadata.rb:3
$
```

This is the part we can do!

- Go into one of the cookbooks (e.g. “apache”) and type
- c:\opscode\chef\embedded\bin \foodcritic .

Exercise: Run Foodcritic

http://acrmp.github.com/foodcritic/



Exercise: Correct Foodcritic Errors

```
myface$ subl metadata.rb
```

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

Exercise: Correct Foodcritic Errors

```
myface$ subl metadata.rb
```

```
name                "myface"
maintainer          "Charles Johnson"
maintainer_email    "charles@opscode.com"
license             "All rights reserved"
description         "Installs/Configures myface"
long_description
IO.read(File.join(File.dirname(__FILE__), 'README.md'))
version             "0.1.0"
```

Exercise: Run Foodcritic

```
myface$ foodcritic .
```

```
$ foodcritic .

$ echo $?
0
$
```

OPSCODE

Best Practice: Run Foodcritic Before Each Commit

- Always **check** in **corrected** code!
- Make **Foodcritic** a part of your **build pipeline** with commit hooks or other methods
- The **Foodcritic** web page has extensive **documentation** & **examples** showing how to build Foodcritic rules into **Jenkins** or **Travis-CI**

OPSCODE

The Problem and the Success Criteria

- The Problem:
  - We need a properly configured repository with all supporting files to create the myface application cookbook.
- Success Criteria:
  - ~~Berkshelf is properly configured.~~
  - ~~The myface repository is initialized.~~
  - ~~The VM test environment is provisioned.~~
  - ~~The README file is updated.~~
  - ~~Foodcritic passes with zero errors~~

RULE THE CLOUD

Exercise: Tear Down Vagrant

```
myface$ vagrant destroy -f
```

```
$ vagrant destroy -f
[default] Forcing shutdown of VM...
[default] Destroying VM and associated drives...
[Berkshelf] Cleaning Vagrant's berkshelf
$
```

Exercise: Test MyFace

```
myface$ vagrant up
```

```
...
[2013-03-31T22:48:51+00:00] INFO: Chef Run complete in 16.90671831 seconds
[2013-03-31T22:48:51+00:00] INFO: Running report handlers
Run options: -v --seed 32328
# Running tests:
minitesthandler::default#test_0001_runs_no_tests =
0.00 s =
.
myface::default#test_0001_runs_no_tests_by_default =
0.00 s =
.
Finished tests in 0.003538s, 565.3082 tests/s, 0.0000 assertions/s.
2 tests, 0 assertions, 0 failures, 0 errors, 0 skips

[2013-03-31T22:48:51+00:00] INFO: Report handlers complete
$
```

OPSCODE

Best Practice: Rebuild the Test Environment Before Commit

- Development VMs should have a short lifespan. Cycle them frequently!
- Always test how code will work when run from scratch.
- For this class, we're going to destroy and rebuild the test environment completely before every commit.

Exercise: Commit the Code Repo

```
myface$ git add .; git commit -a -m "Initial commit"
```

```
$ git add .; git commit -a -m "Initial commit"
[master (root-commit) 002d253] Initial commit
 11 files changed, 253 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 Berksfile
 create mode 100644 Gemfile
 create mode 100644 LICENSE
 create mode 100644 README.md
 create mode 100644 Vagrantfile
 create mode 100644 chefignore
 create mode 100644 files/default/tests/minitest/default_test.rb
 create mode 100644 files/default/tests/minitest/support/helpers.rb
 create mode 100644 metadata.rb
 create mode 100644 recipes/default.rb
$
```

Questions

- How does one bring up a VM with Vagrant?
- How does Chef know which recipes to run in the Vagrant VM?
- How does one run foodcritic?
- How does one destroy the VM?
- How often should one run Foodcritic?
- How often should one test from scratch?
- When should one commit the repo?

Premature EOF

- That's all the time I had to crib slides from the cookbook authoring class
- Let me show a demo of how I develop cookbooks with this workflow using the Teamforge cookbook I work on.
- Show:
  - rspec tests ("unit tests")
  - provisioning the machine from scratch
  - running minitests