Chef Introductory Workshop - Managing Linux

training@opscode.com
http://opscode.com/training

Introductions

OPSCODE | Chef

Instructor Introduction

OPSCODE | Chef

Introduce yourselves

Objectives and Expectations

OPSCODE | Chef

System Administration with Chef: Agenda

- Setup workstation environment
- Anatomy of a Chef Run
- Hands on exercises
- Lunch
- More hands on exercises

- We will take breaks, of course!

OPSCODE | Chef

Workshop Objectives

- Automate common system administration tasks with Chef.
- Understand Chef's architecture.
- Be familiar with Chef's various tools.
- Know how to get further help.

OPSCODE | Chef

Expectations

- This is a one day workshop, not a comprehensive course.
- We will do several hands on exercises.
- We want to focus on being immediately productive solving common tasks.
- You should be equipped to hit the ground running and know where to go next.

OPSCODE

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
- The exercises cover common problems to system administration use cases
- Ask questions when they come to you.
- Ask for help when you need it.

RULE THE CLOUD

OPSCODE

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 late
- You won't be left behind
- We will troubleshoot and fix bugs on the spot
- The result is you reaching fluency fast

Overview of Chef

What is this thing again?

OPSCODE

Evolving towards Configuration Management

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

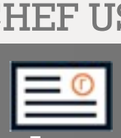
RULE THE CLOUD

OPSCODE

Chef

Chef is an automation platform for developers & systems engineers to continuously define, build, and manage infrastructure.

CHEF USES:


Recipes and 
Cookbooks
that describe Infrastructure as Code.

Chef enables people to easily build & manage complex & dynamic applications at massive scale

- New model for describing infrastructure that promotes reuse
- Programmatically provision and configure
- Reconstruct business from code repository, data backup, and bare metal resources

OPSCODE

Chef

Chef is an automation platform for developers & systems engineers to continuously define, build, and manage infrastructure.

CHEF USES:

Recipes and **Cookbooks** that describe Infrastructure as Code.

Chef enables people to easily build & manage complex & dynamic applications at massive scale

- New model for describing infrastructure that promotes reuse
- Programmatically provision and configure
- Reconstruct business from code repository, data backup, and bare metal resources

OPSCODE

Applications

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

OPSCODE

Infrastructure

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

OPSCODE

Collection of Resources

- Networking
- Files
- Directories
- Symlinks
- Mounts

- Routes
- Users
- Groups
- Tasks
- Packages
- Software
- Services
- Configuration
- Other Stuff

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

OPSCODE

Acting in Concert

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/

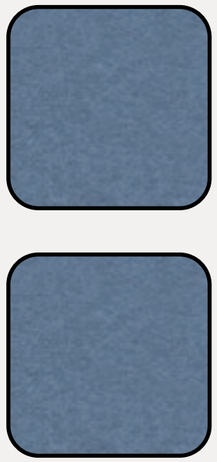
OPSCODE | Chef

See Node

Application Server

OPSCODE | Chef

See Nodes

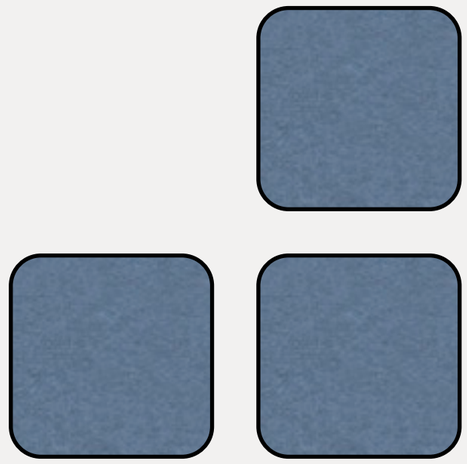


Application Server

Application Database

OPSCODE | Chef

See Nodes Grow

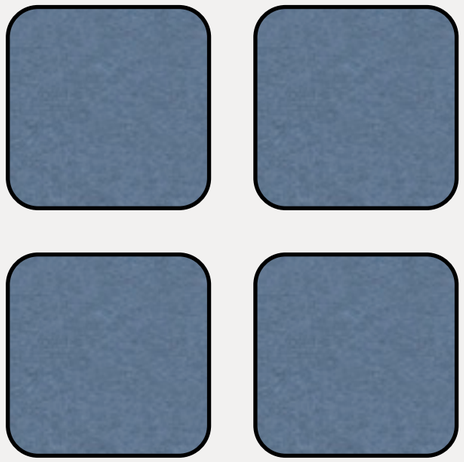


Application Server

Application Databases

OPSCODE | Chef

See Nodes Grow

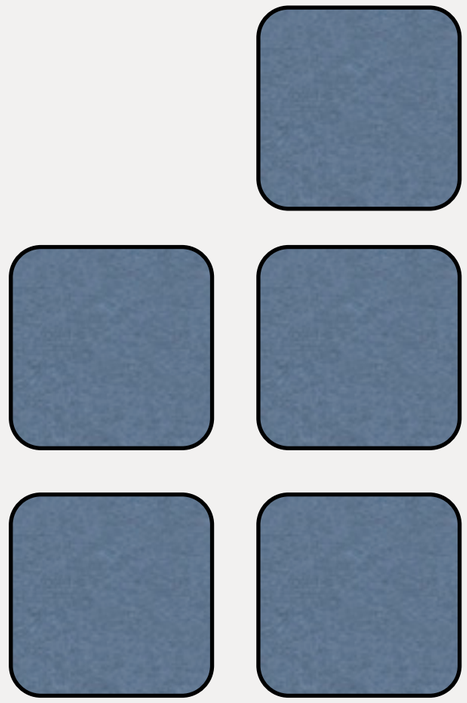


Application Servers

Application Databases

OPSCODE | Chef

See Nodes Grow



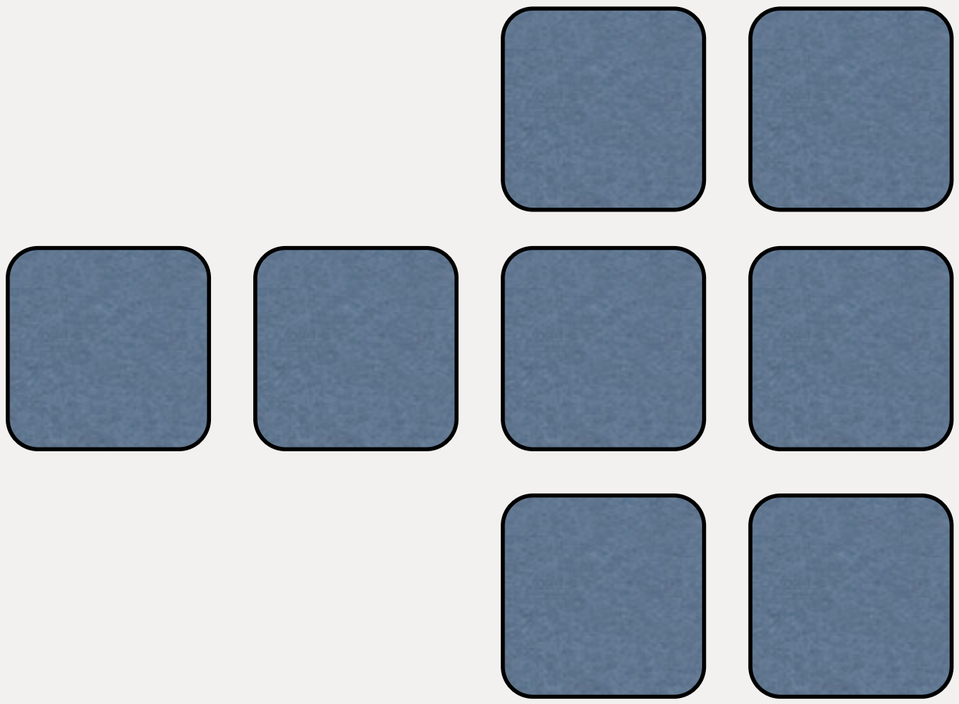
Load Balancer

Application Servers

Application Databases

OPSCODE | Chef

See Nodes Grow



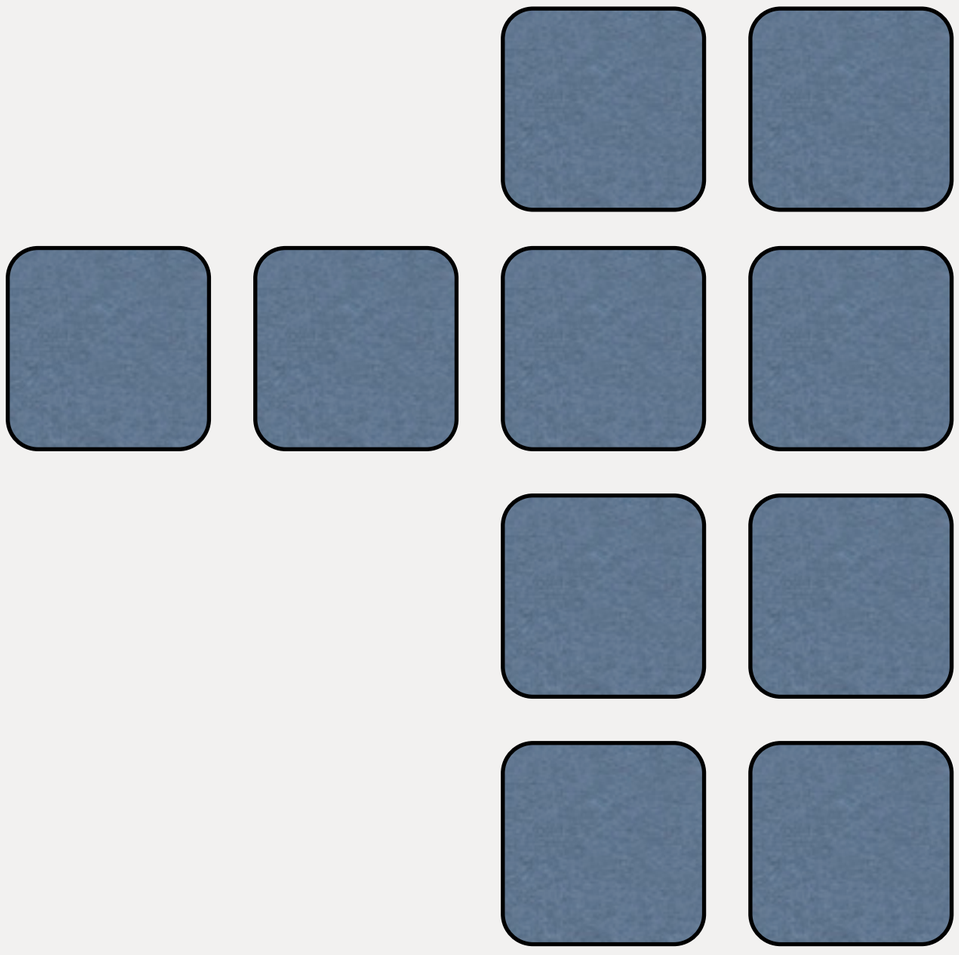
Load Balancers

Application Servers

Application Databases

OPSCODE | Chef

See Nodes Grow



Load Balancers

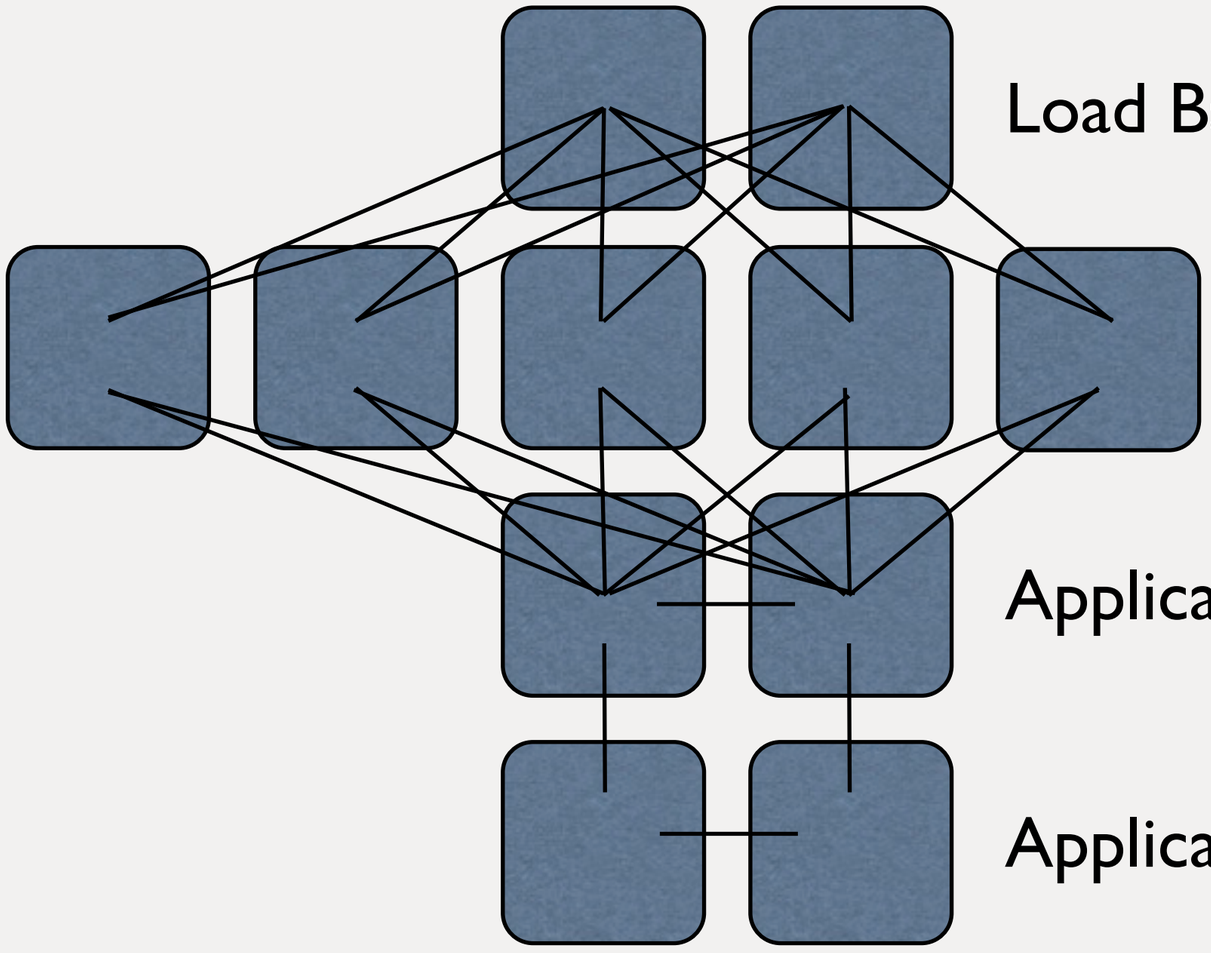
Application Servers

Application Database Cache

Application Databases

OPSCODE | Chef

Tied Together with Configuration



Load Balancers

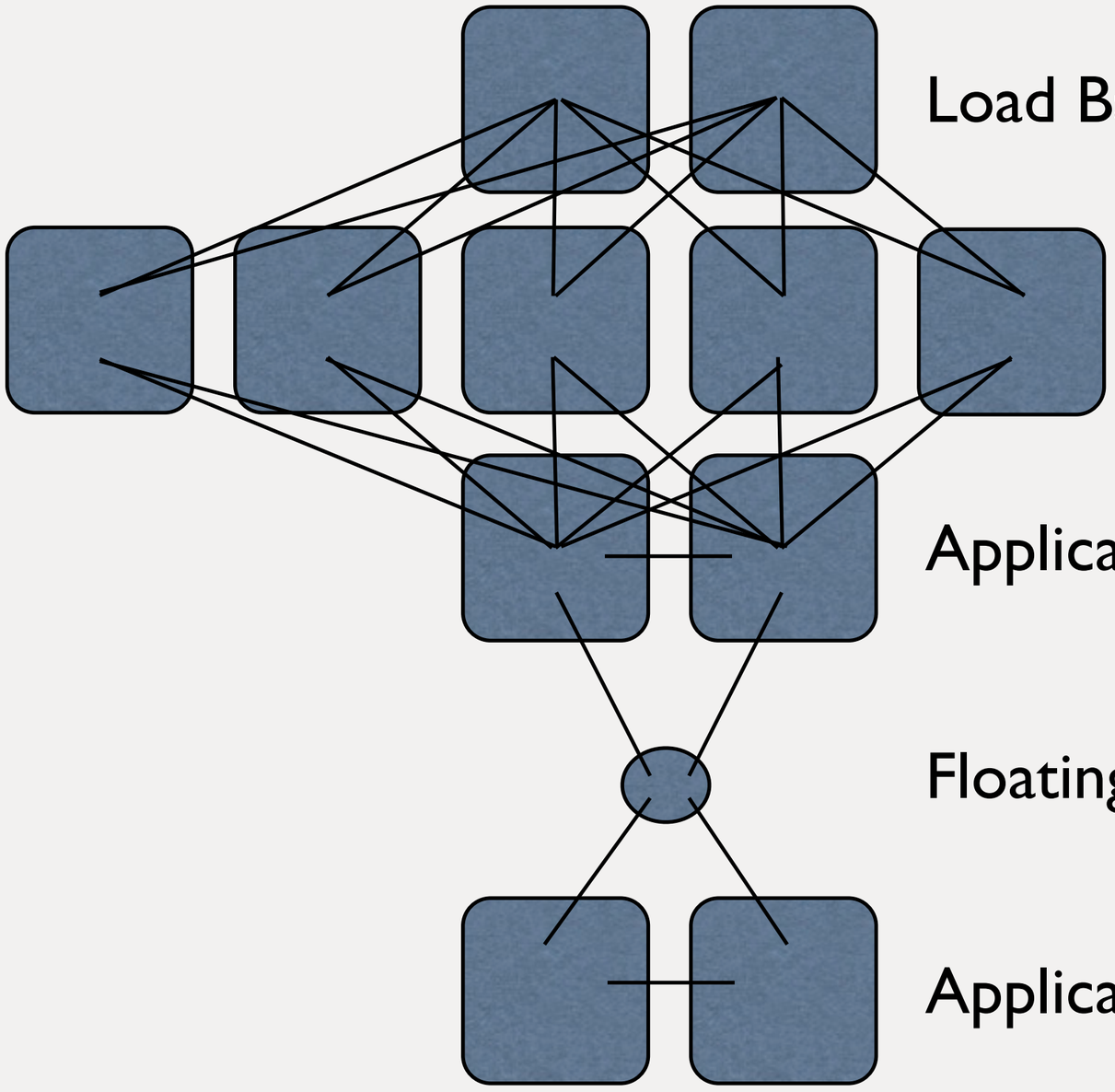
Application Servers

Application Database Cache

Application Databases

OPSCODE | Chef

Infrastructure is a Snowflake



Load Balancers

Application Servers

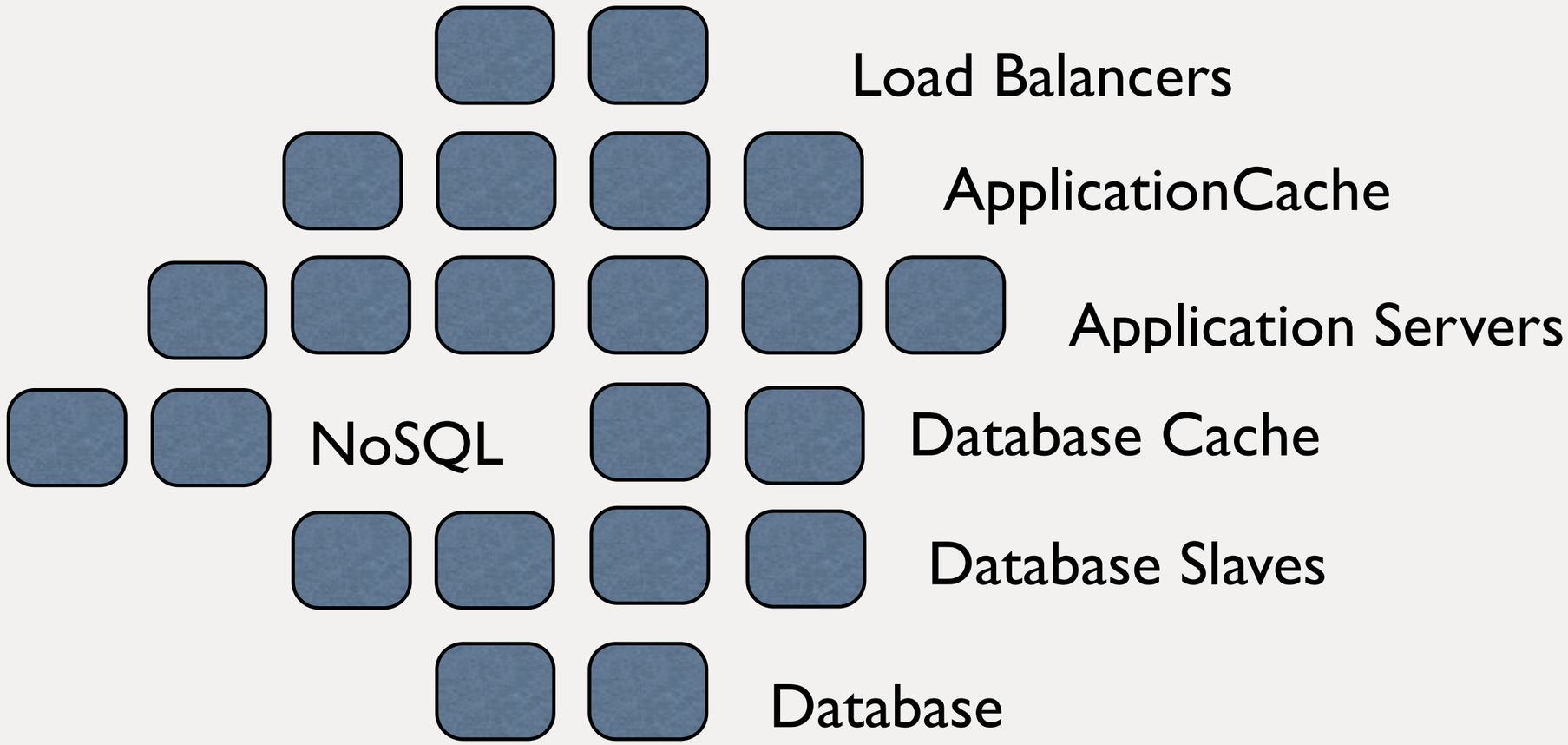
Application Database Cache

Floating IP?

Application Databases

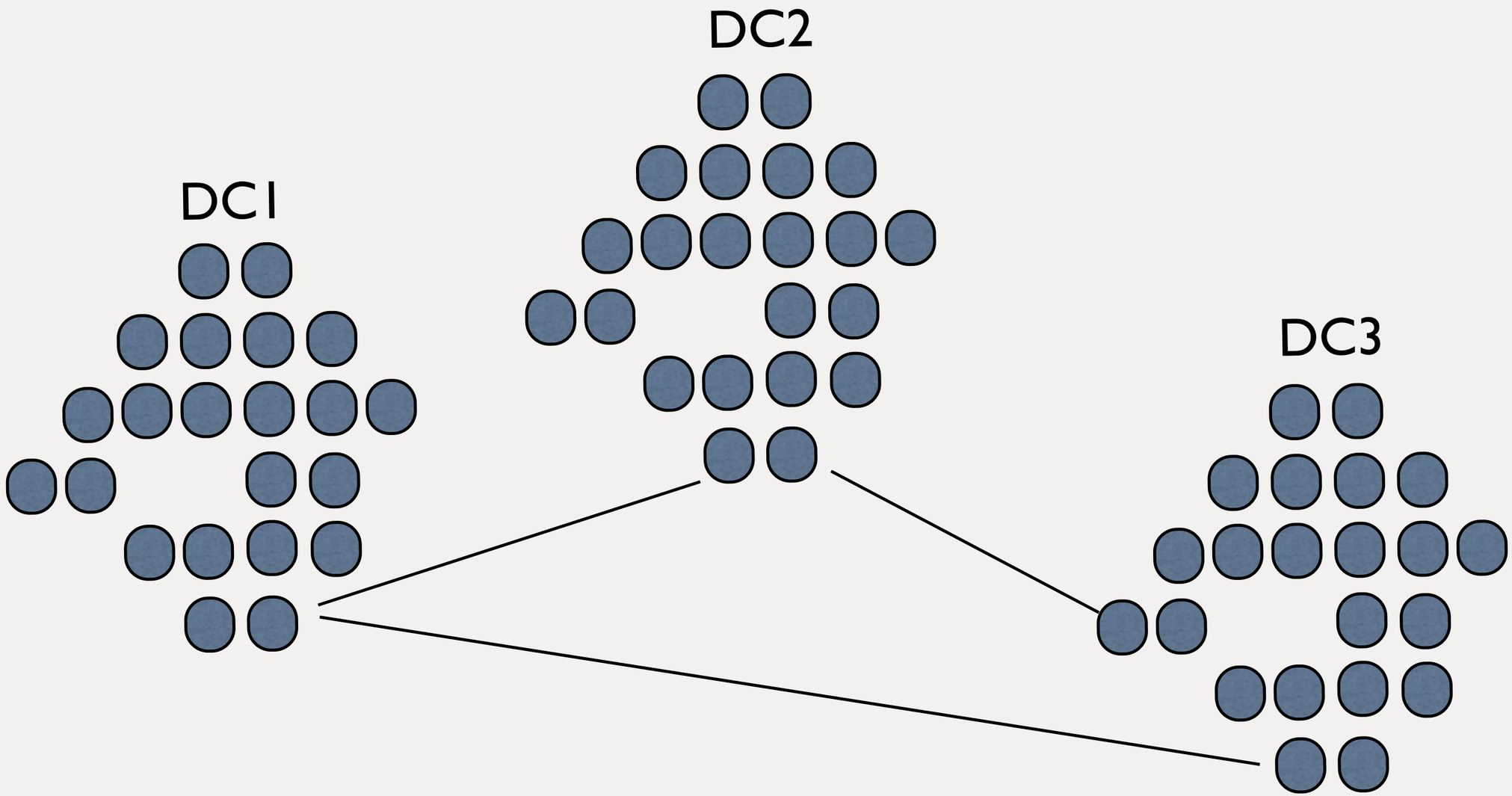
OPSCODE | Chef

Evolving Complexity

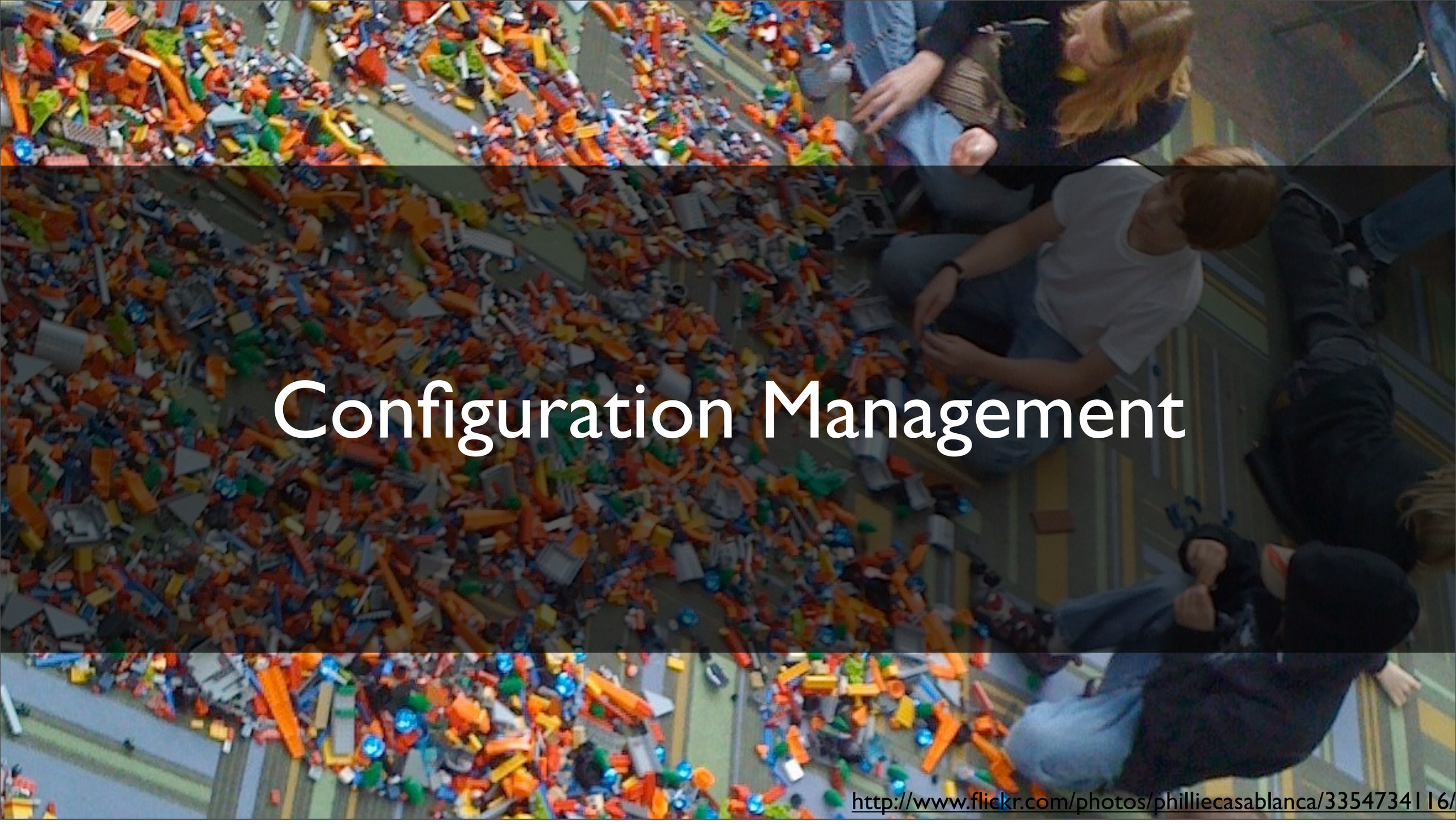


OPSCODE | Chef

Complexity Grows Quickly



Configuration Management



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

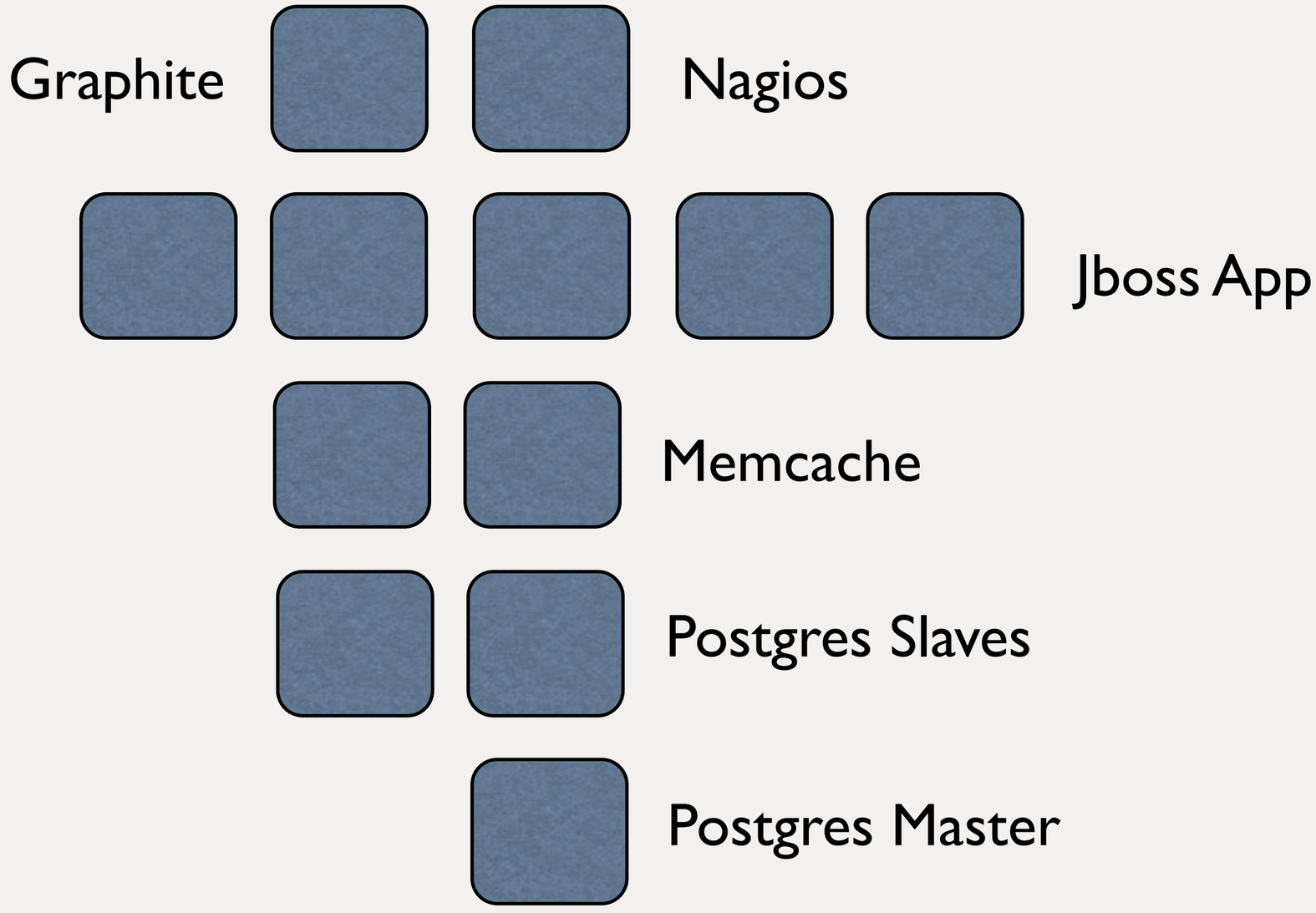
Golden Images are not the answer

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

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

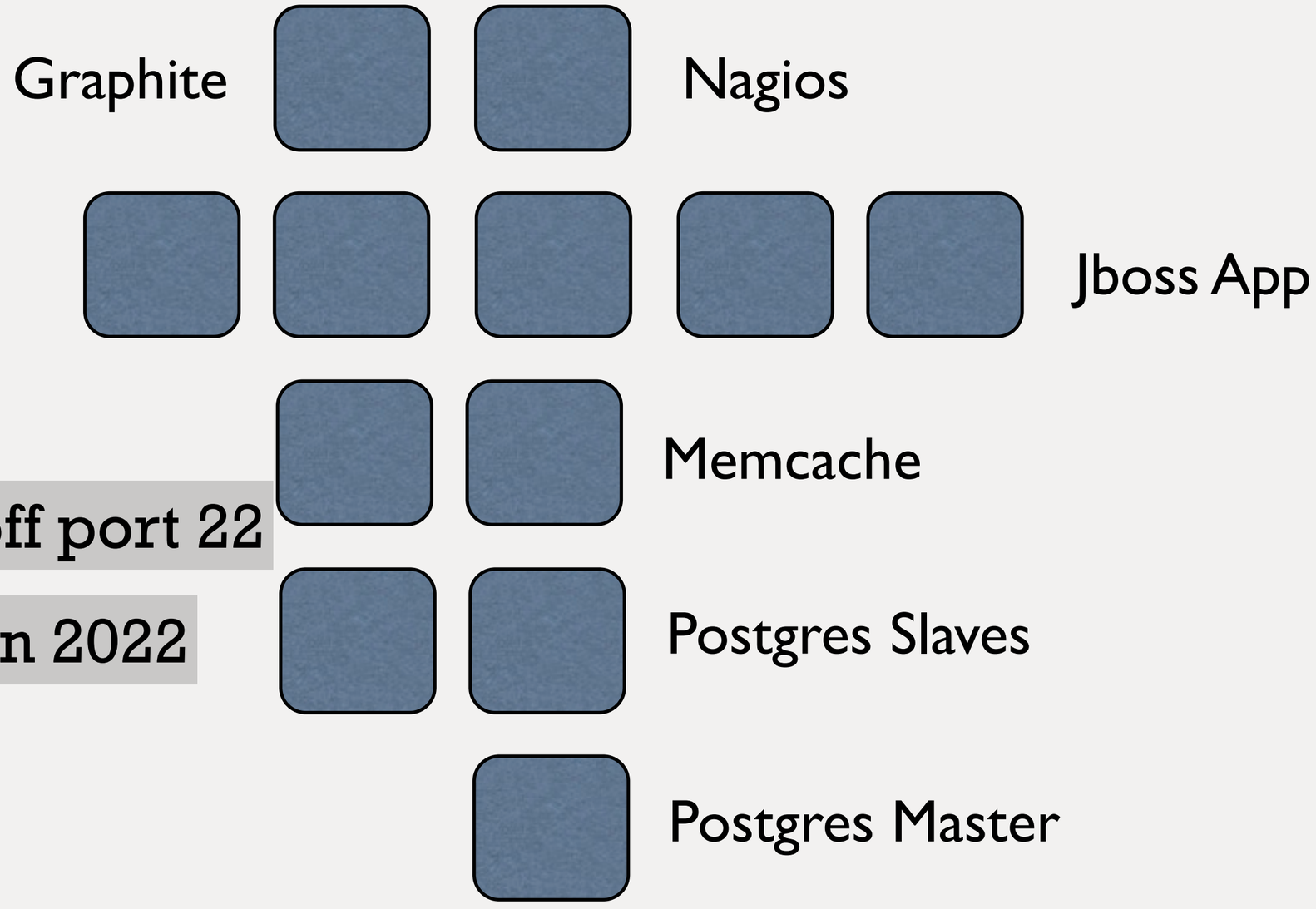
OPSCODE

Typical Infrastructure



OPSCODE

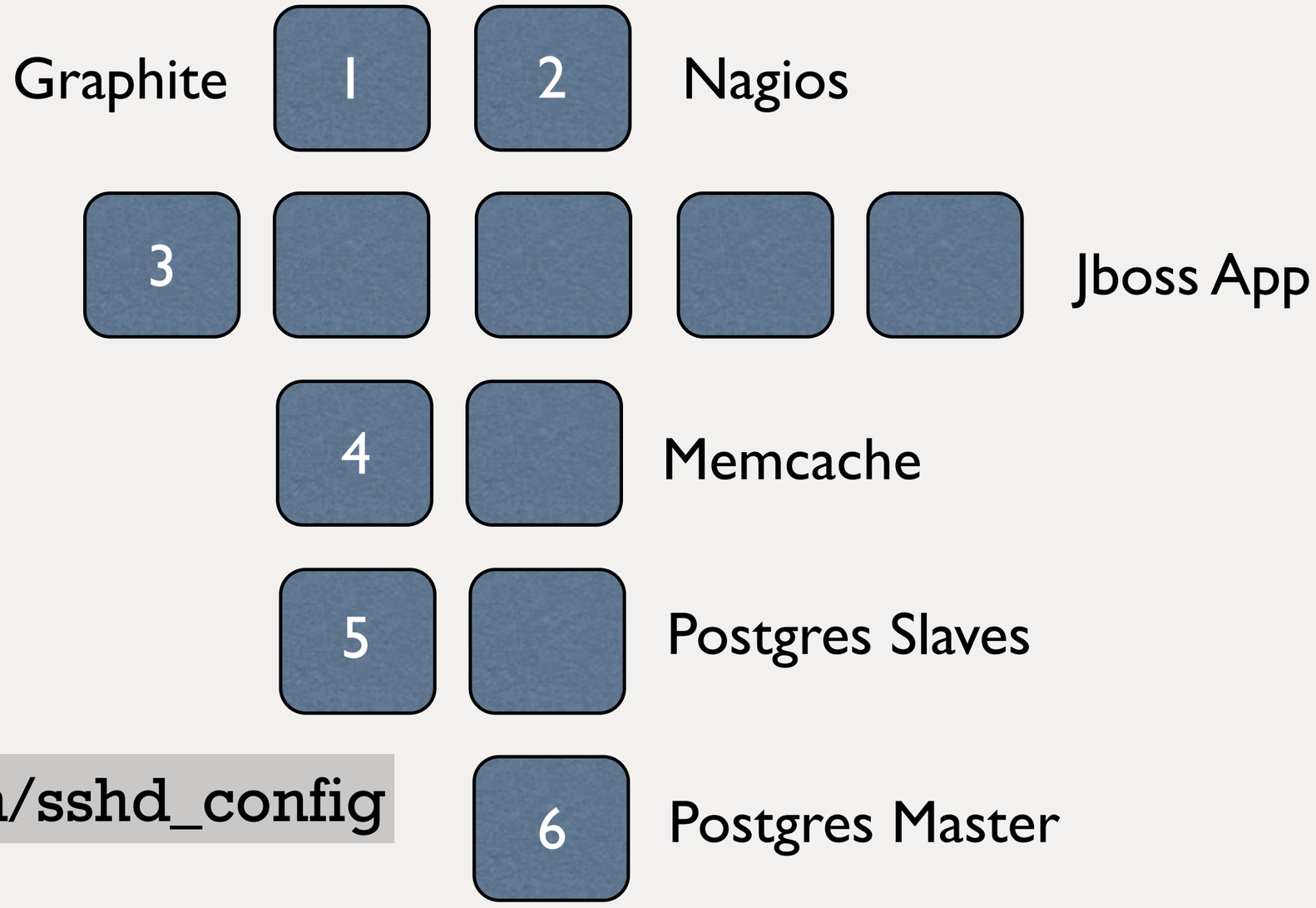
New Compliance Mandate!



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

OPSCODE

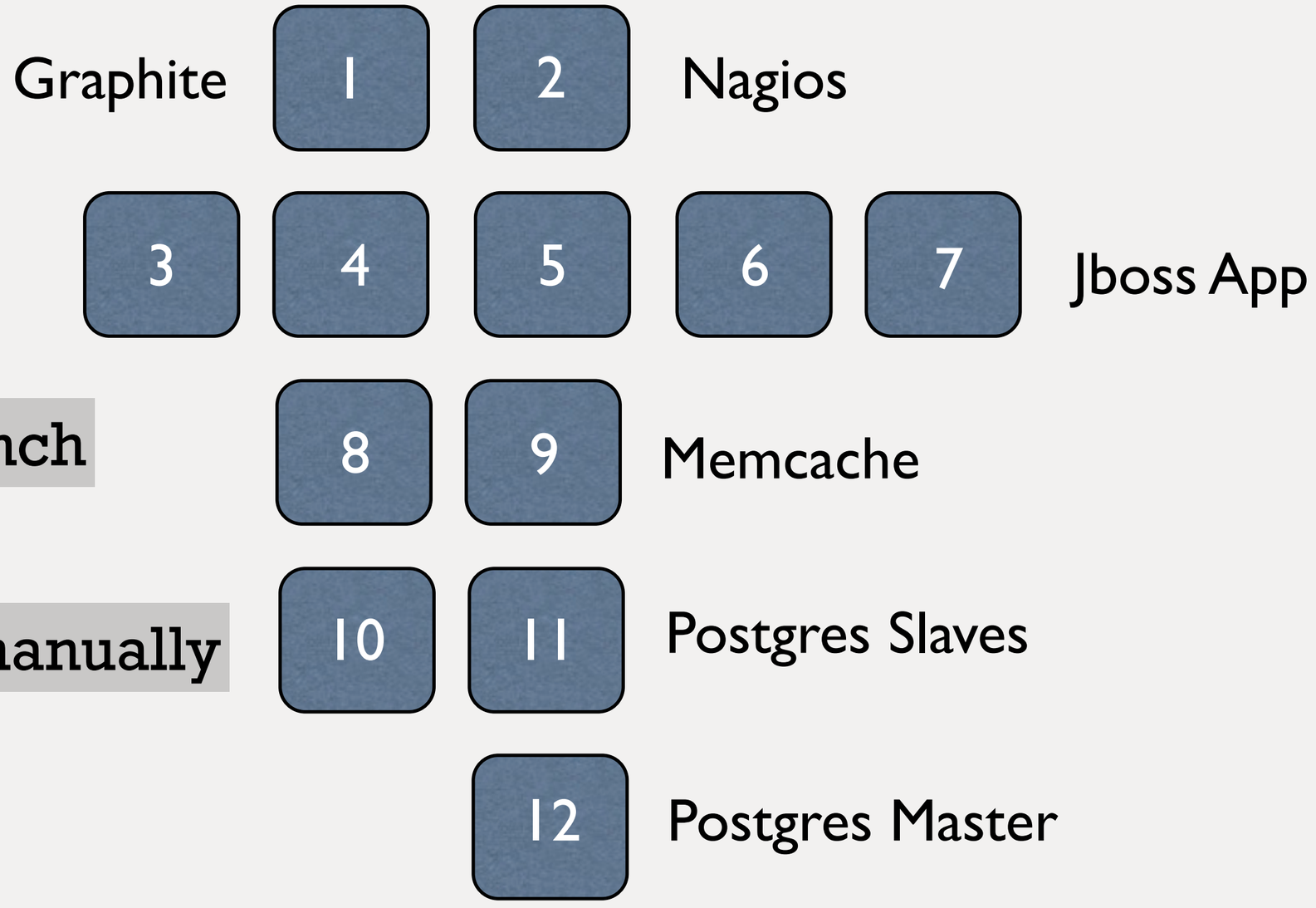
6 Golden Image Updates



- edit /etc/ssh/sshd_config

OPSCODE

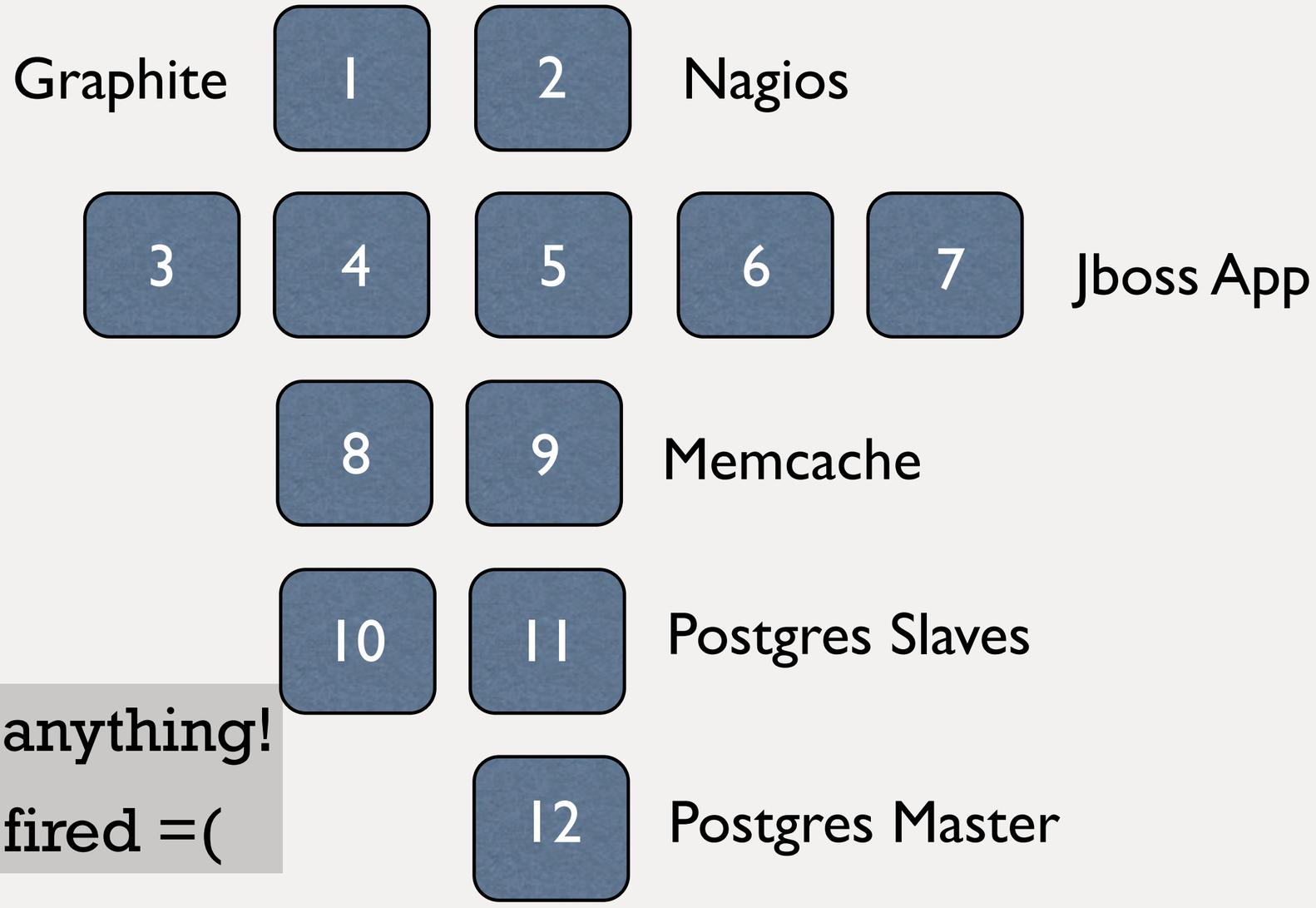
12 Instance Replacements



- Delete, launch
- Repeat
- Typically manually

OPSCODE

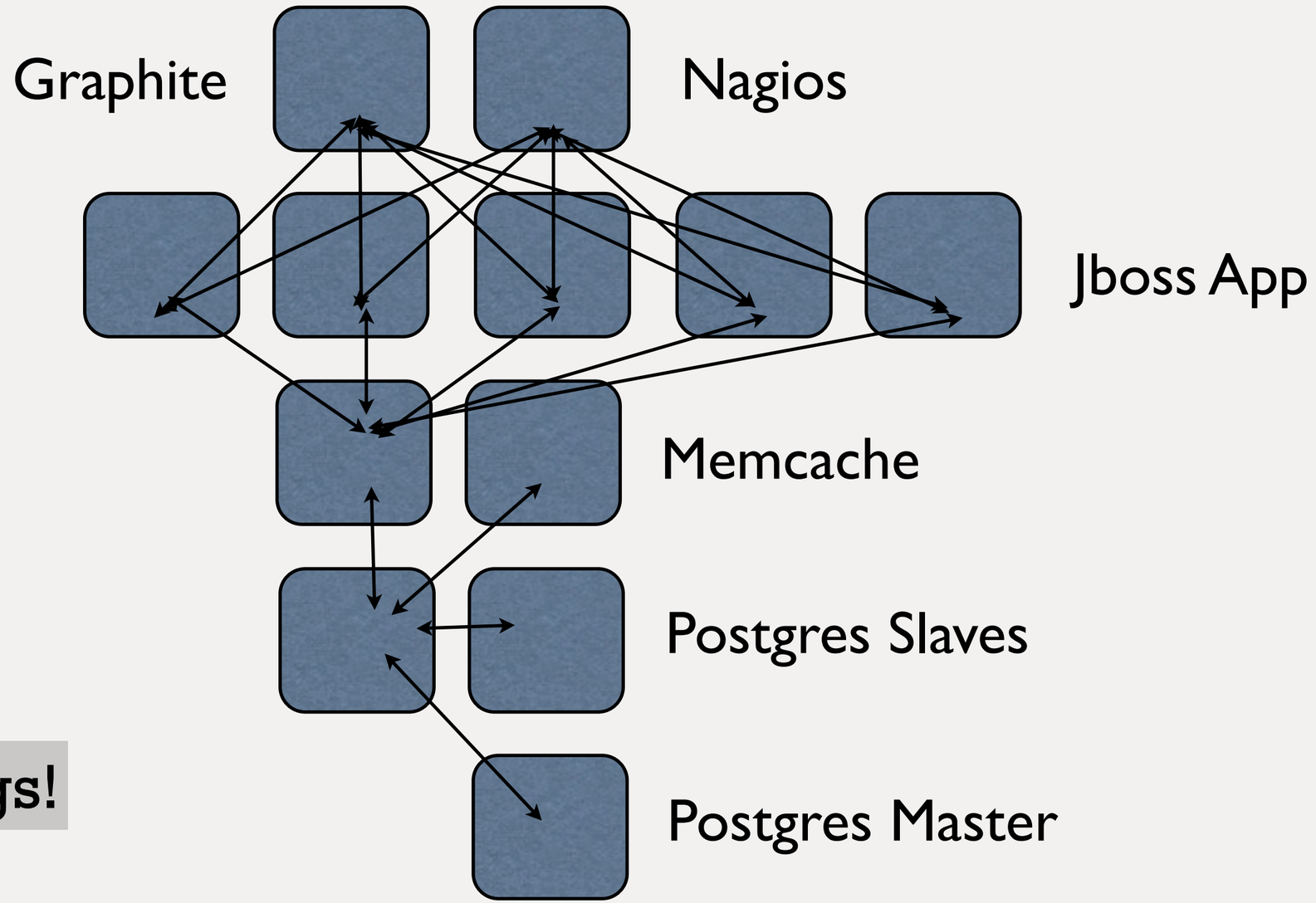
Done in Maintenance Windows



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

OPSCODE

Different IP Addresses?



- Invalid configs!

OPSCODE

Configuration Desperation

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

OPSCODE

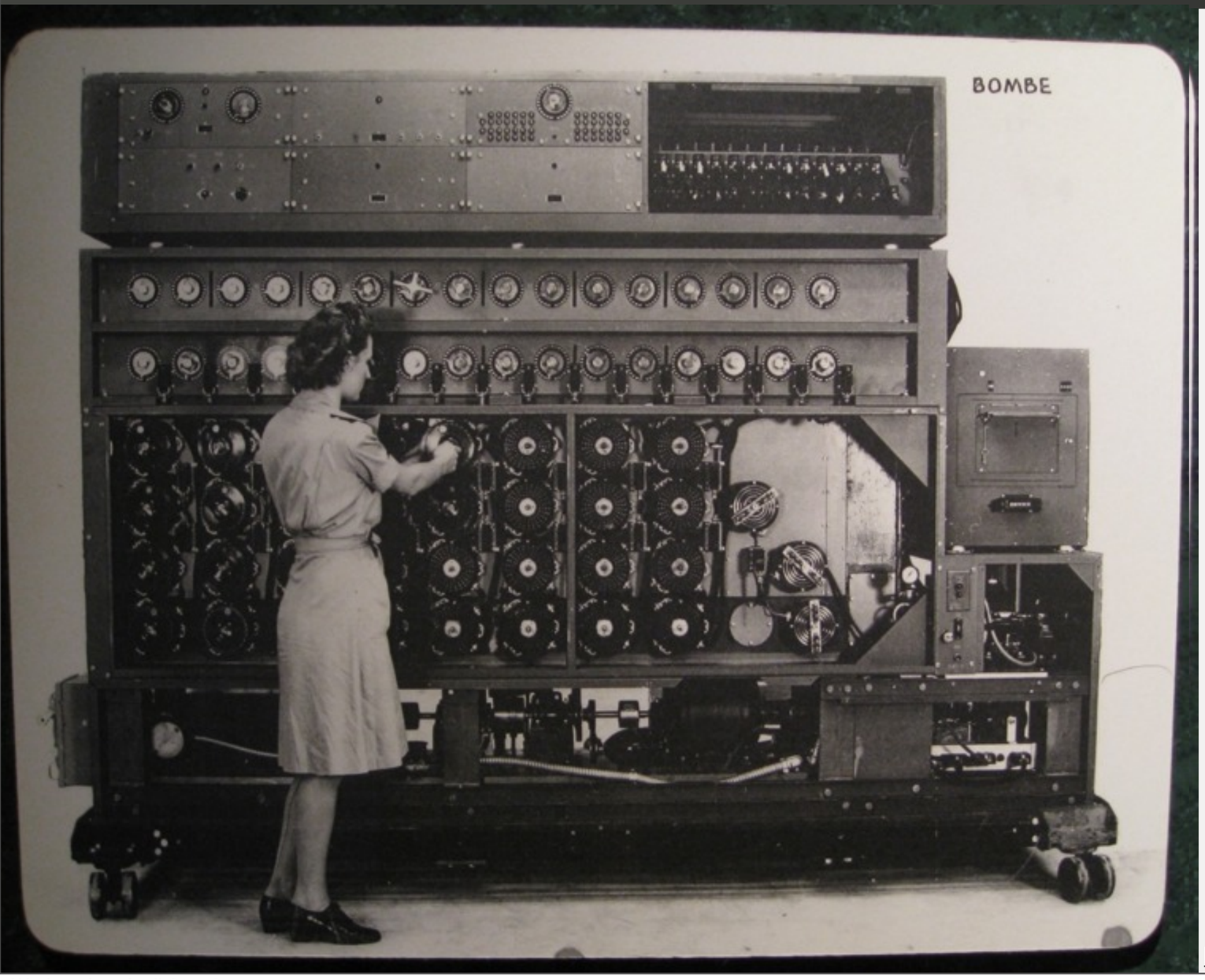
Chef Solves this Problem

Chef

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

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

Programs

- Chef generates configurations directly on nodes from their run list
- Reduce management complexity through abstraction
- Store the configuration of your programs in version control

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

OPSCODE

Declarative Interface to Resources

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

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

OPSCODE

That looks like this

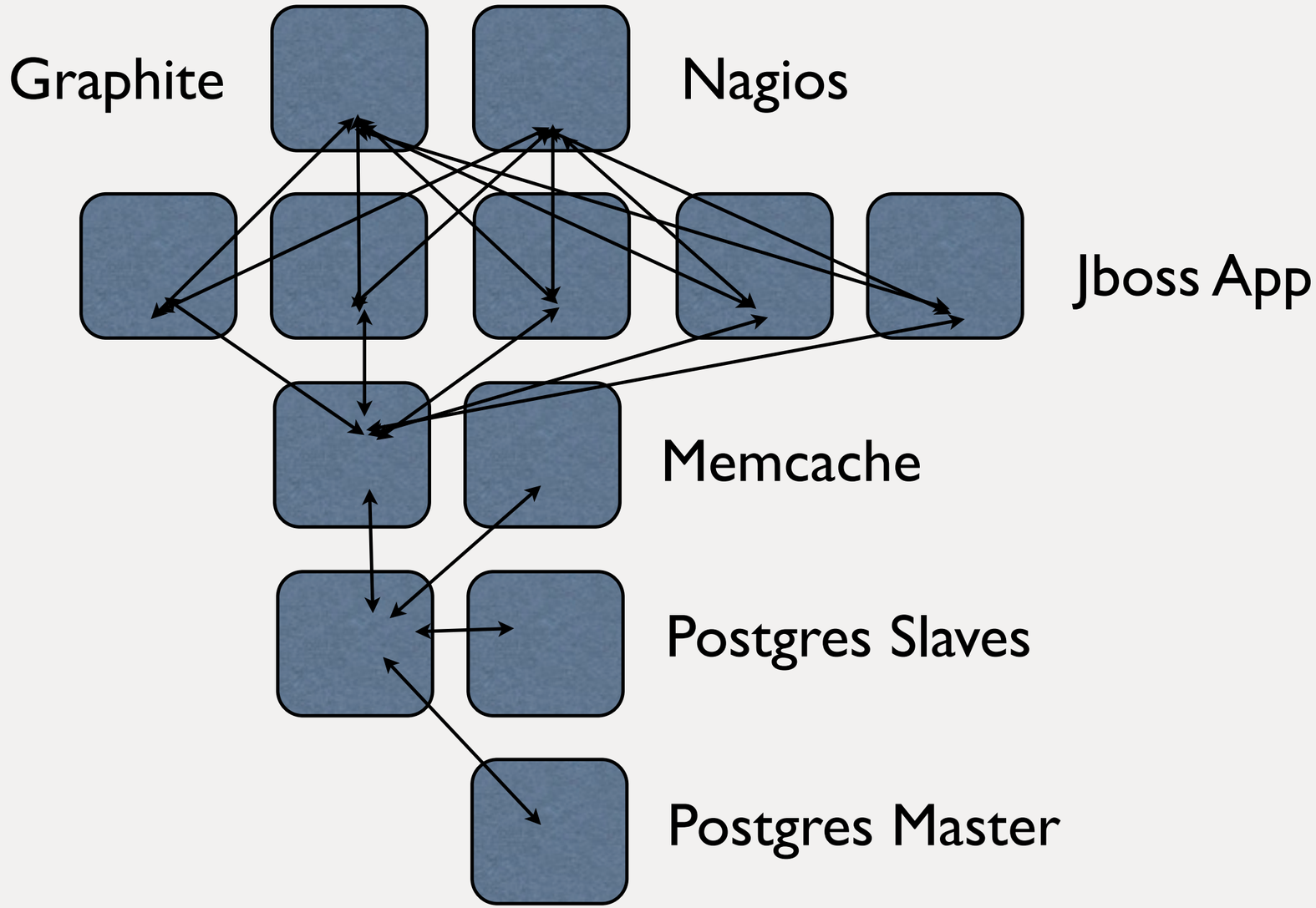
```
package "ntp" do
  action :install
end
```

```
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
```

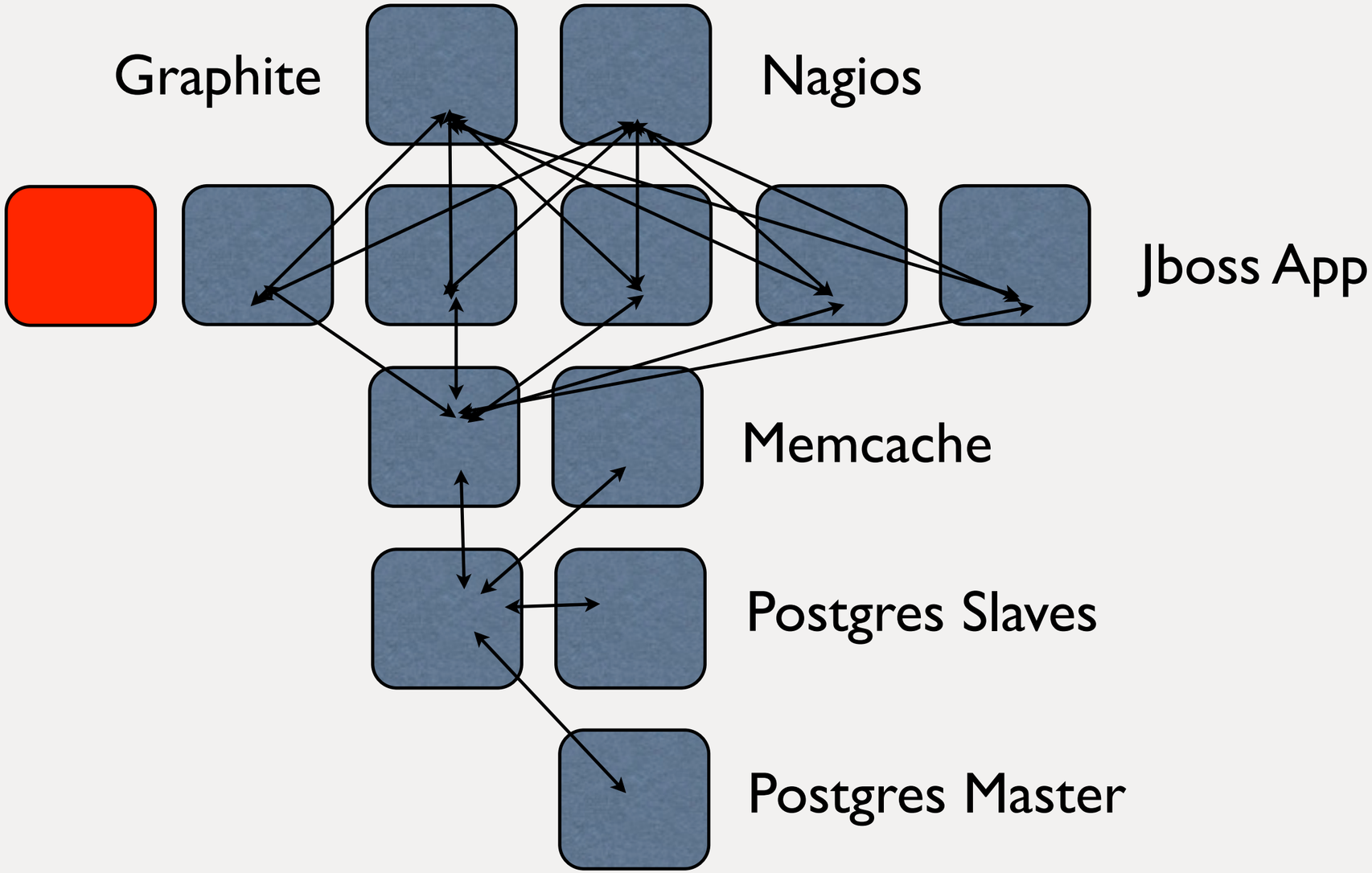
OPSCODE

So when this



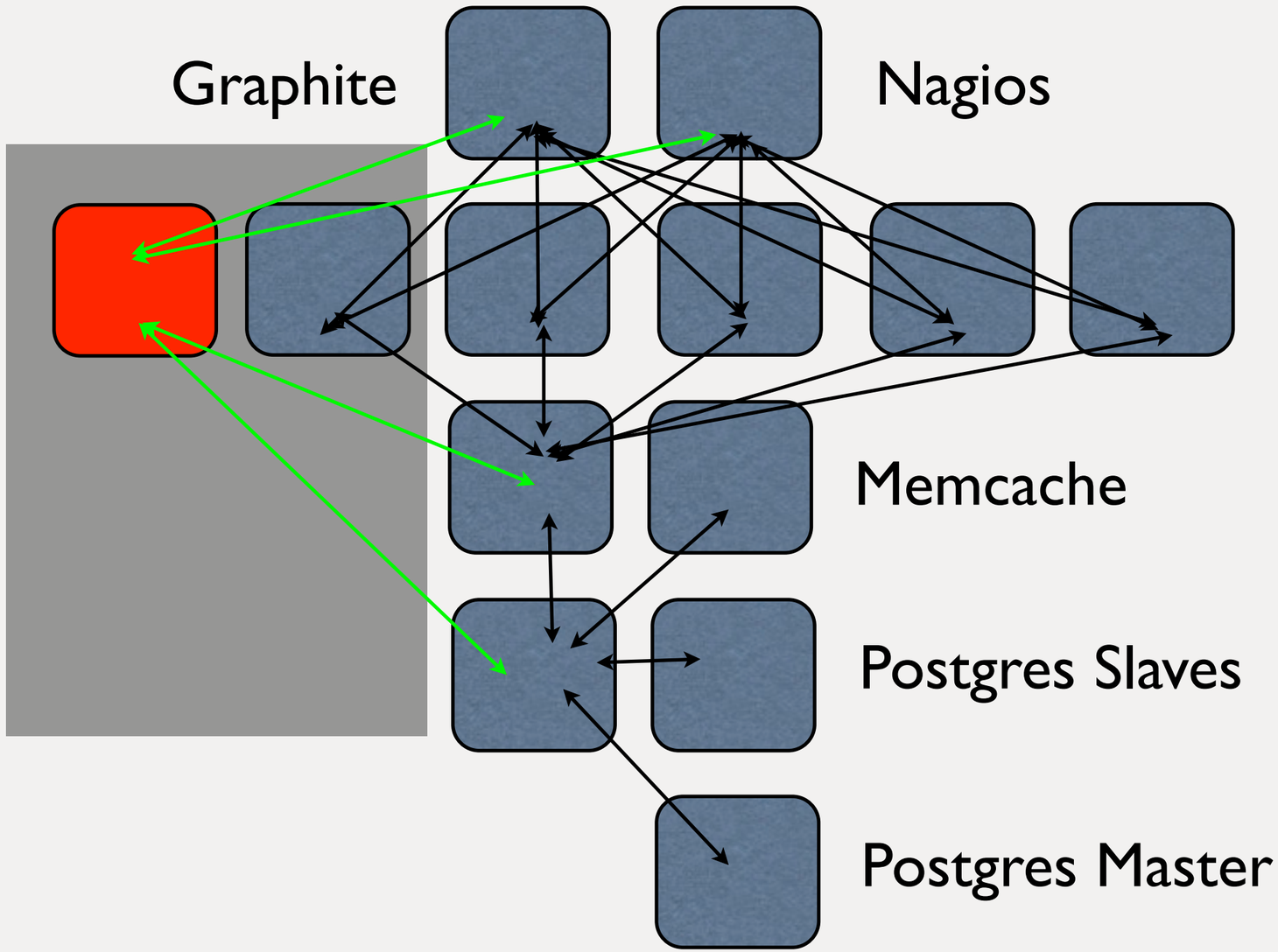
OPSCODE

Becomes this



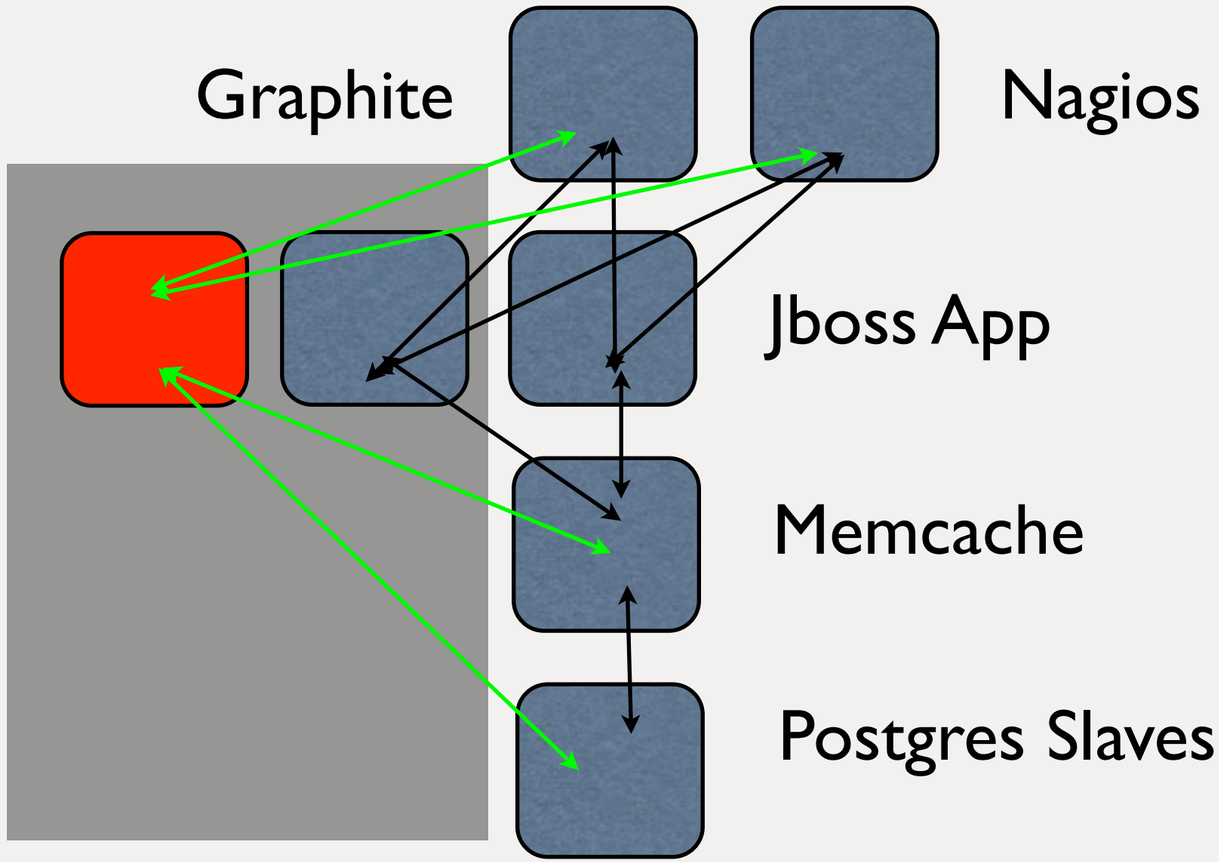
OPSCODE

This can happen automatically



OPSCODE

Count the resources



- 12+ resource changes for 1 node addition

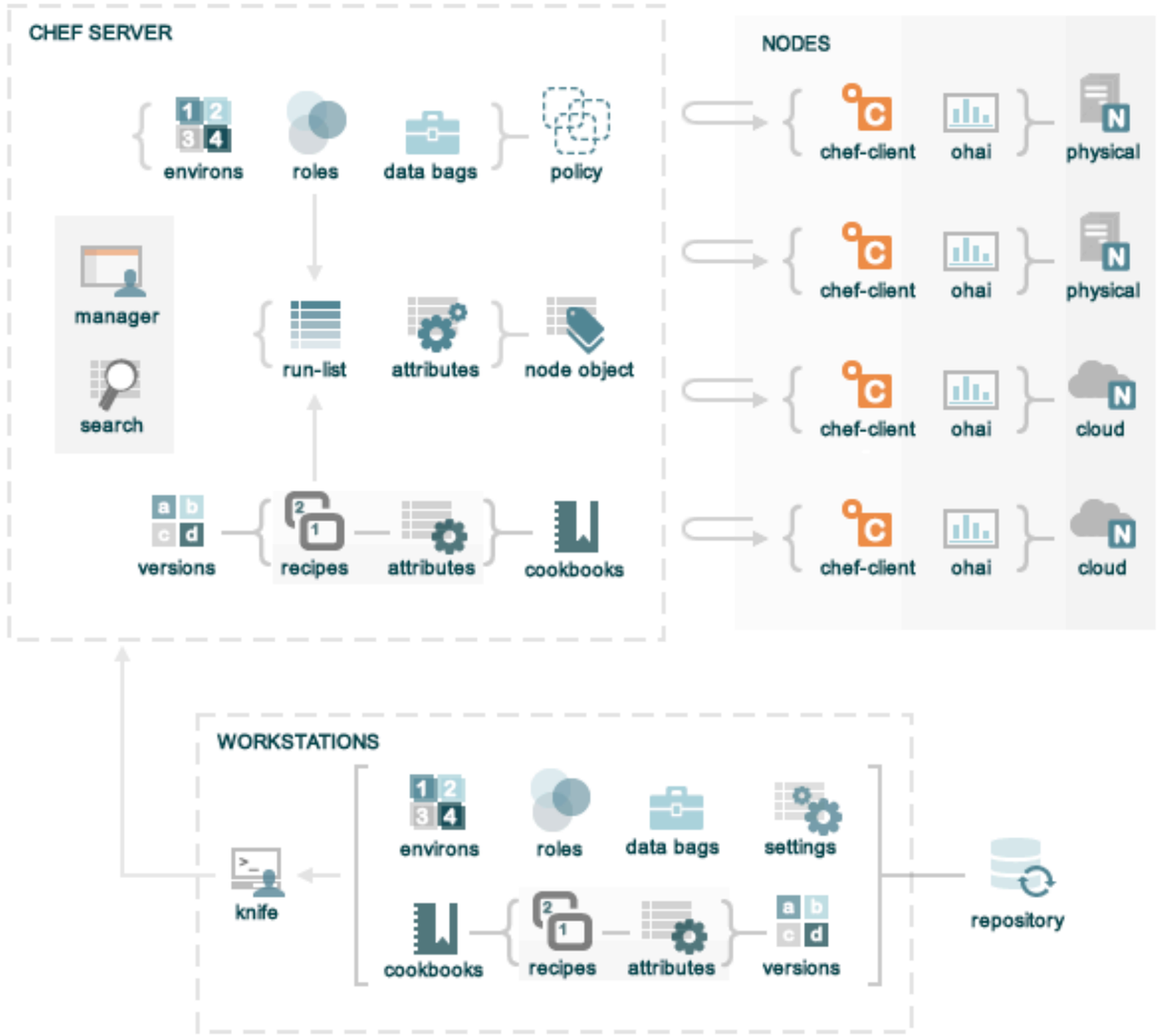
- Load balancer config
- Nagios host ping
- Nagios host ssh
- Nagios host HTTP
- Nagios host app health
- Graphite CPU
- Graphite Memory
- Graphite Disk
- Graphite SNMP
- Memcache firewall
- Postgres firewall
- Postgres authZ config

Getting Started

- Workstation Setup
- Chef Server Account
- Chef Repository
- Remote target managed node

OPSCODE

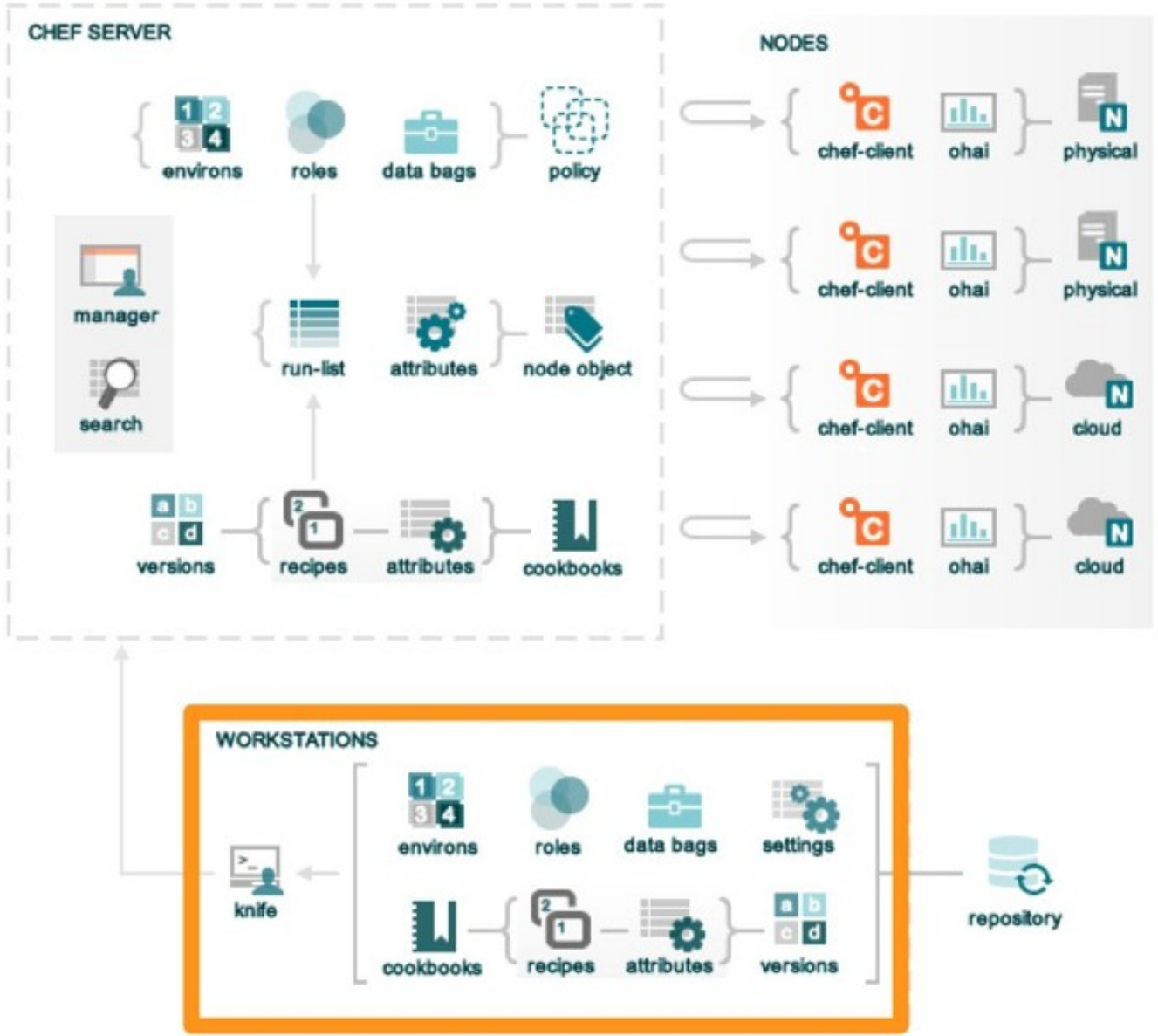
Landscape of Chef-managed Infrastructure



OPSCODE | Chef

Workstation Setup

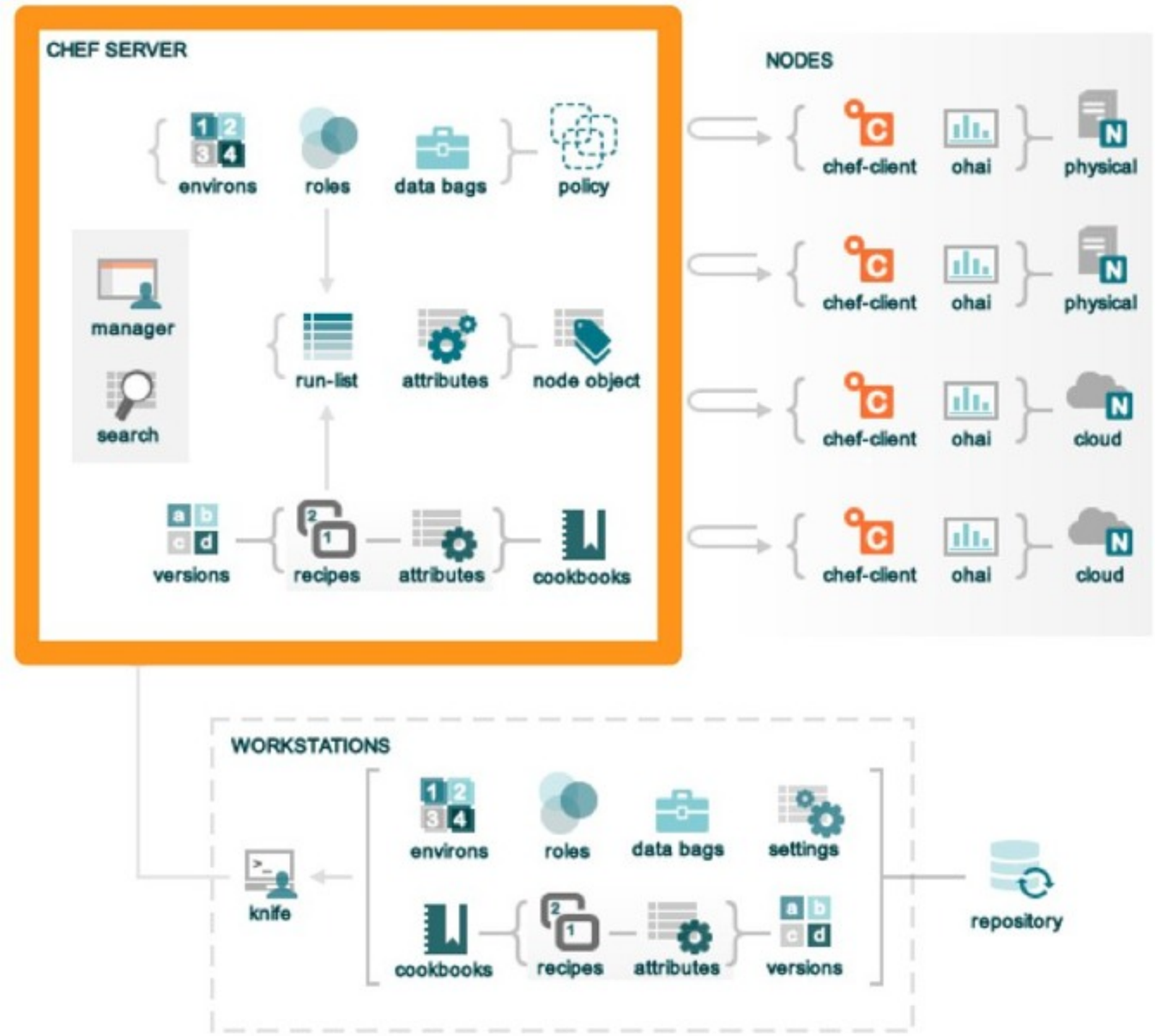
- Install Chef (if not already installed)
- https://www.opscode.com/chef/install/



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 *

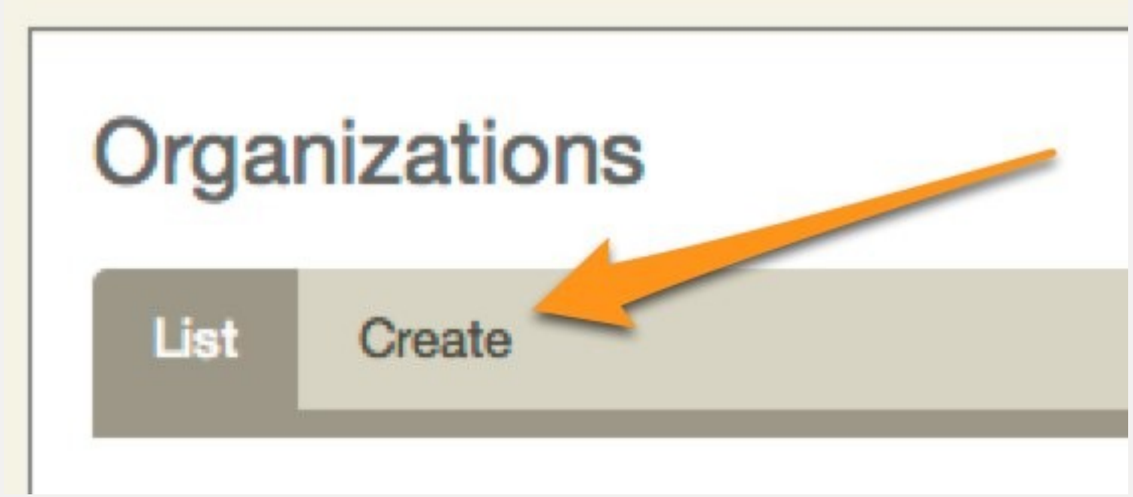
Country * | State *

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

‹ Back | Next ›

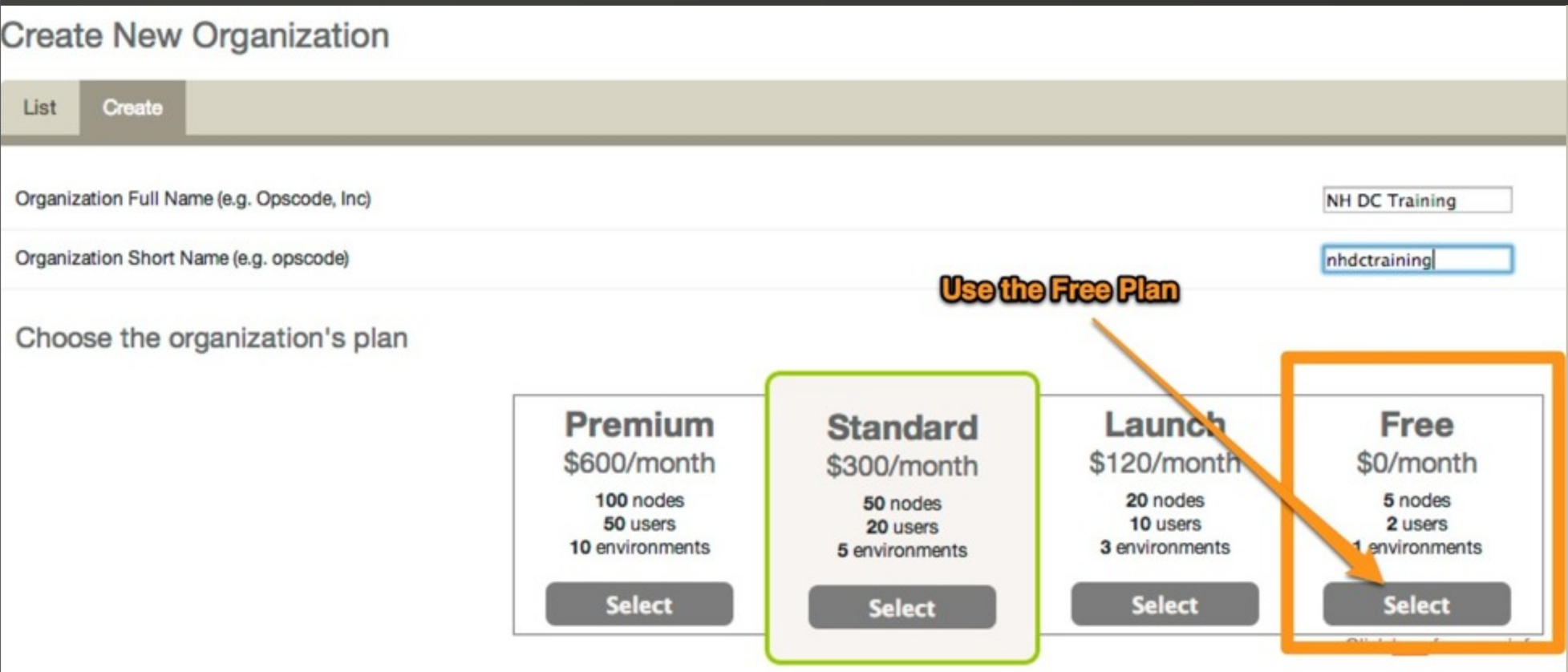
OPSCODE | Chef

Create an Organization



OPSCODE | Chef

Create New Organization



Organization Short Name must be GLOBALLY unique!

OPSCODE | Chef

Download the Validation Key and Knife Config

Organization: training

List Create

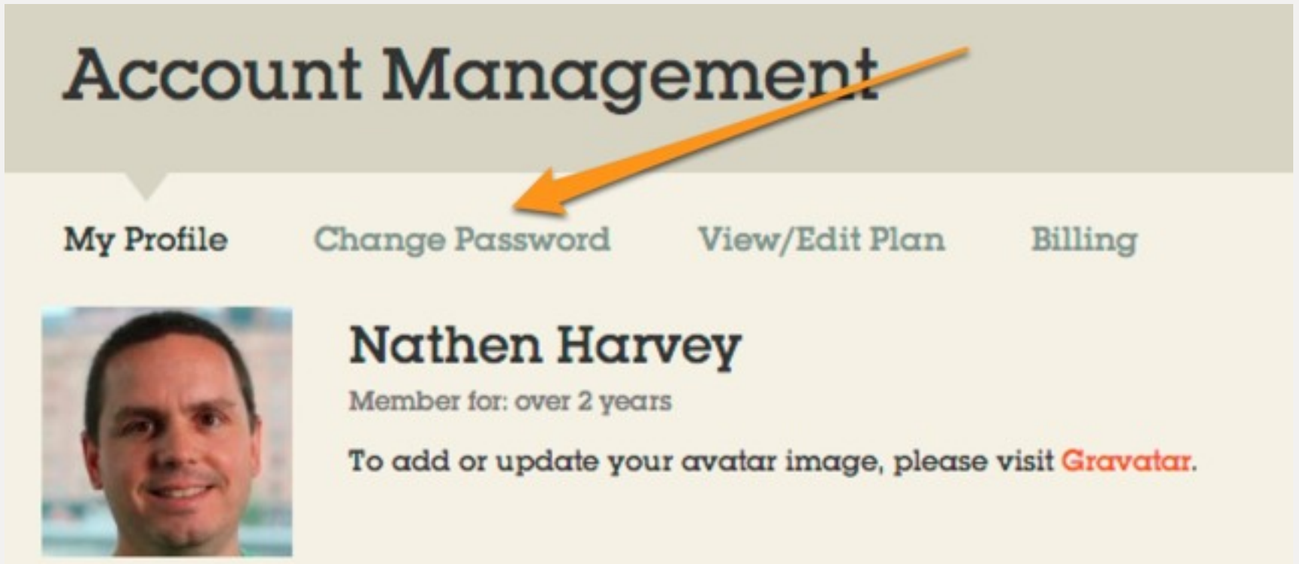
Save the validation key and Knife configuration file

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 | Chef

Target Instances

ec2-based Instance

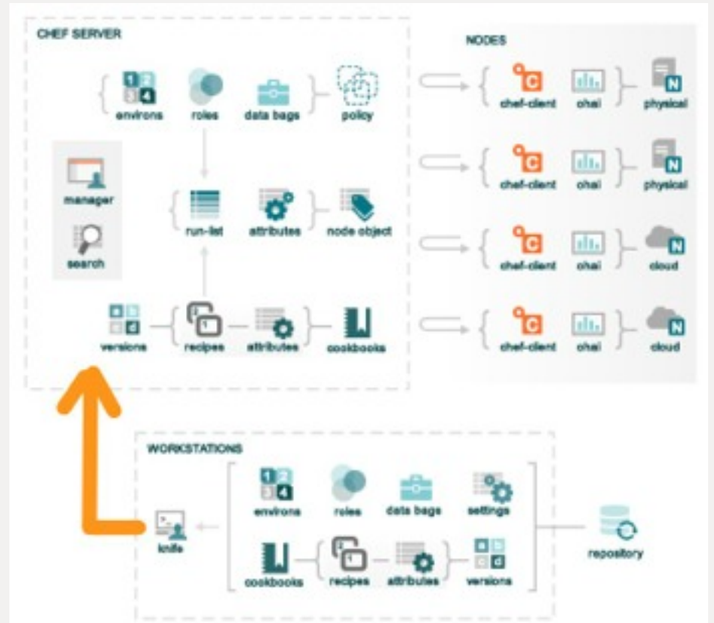
- ec2-STUDENT_ID.compute-1.amazonaws.com
- Ubuntu 12.04
- SSH
  - *Username*: **opscode**
  - *Password*: **opscode**

OPSCODE | Chef

Copy Chef Server Files

```
# copy your user key, validation key and knife config:
> 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
```



OPSCODE | Chef

Verify Knife

```
> knife --version
Chef: 11.4.0
```

Your version may differ, that's okay!

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

Bootstrap the Target Instance

"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)

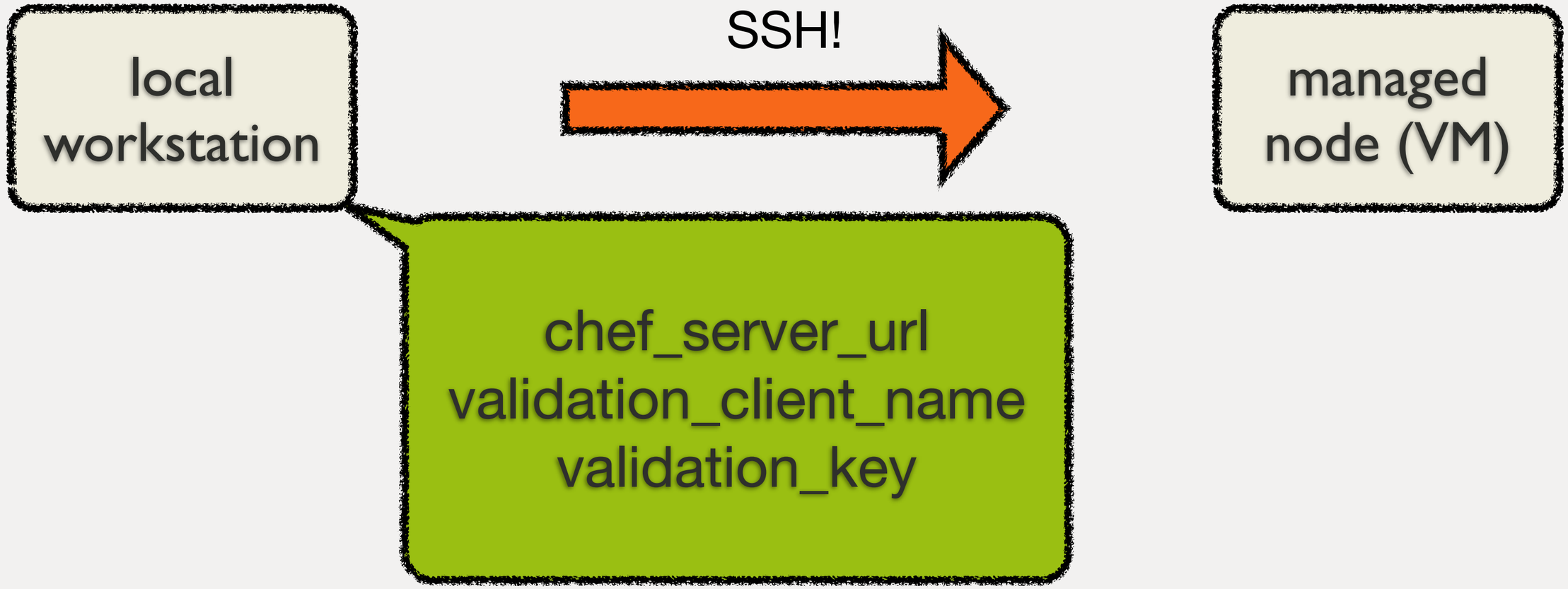
```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```

local workstation

managed node (VM)

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

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



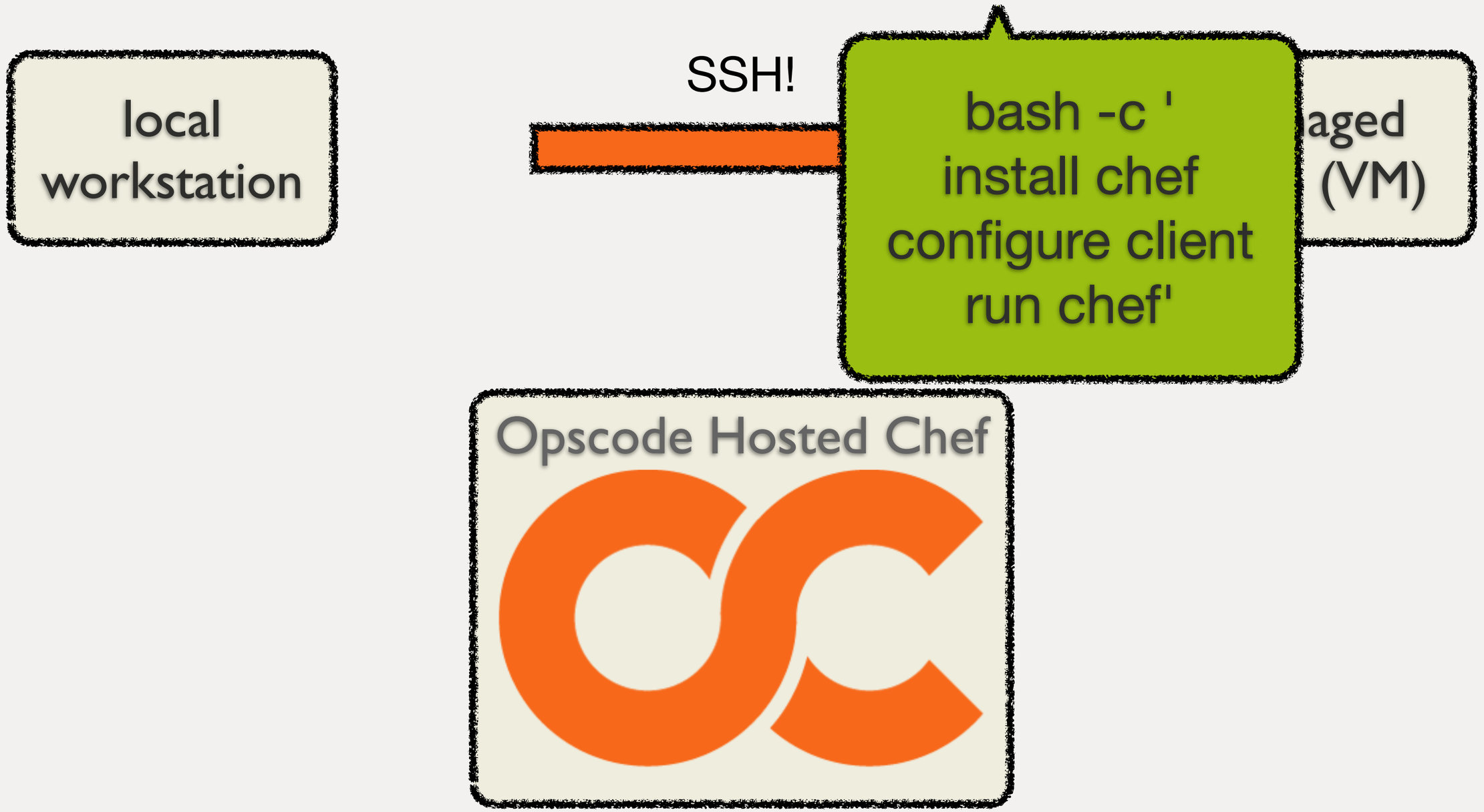
```
knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD
```

local workstation

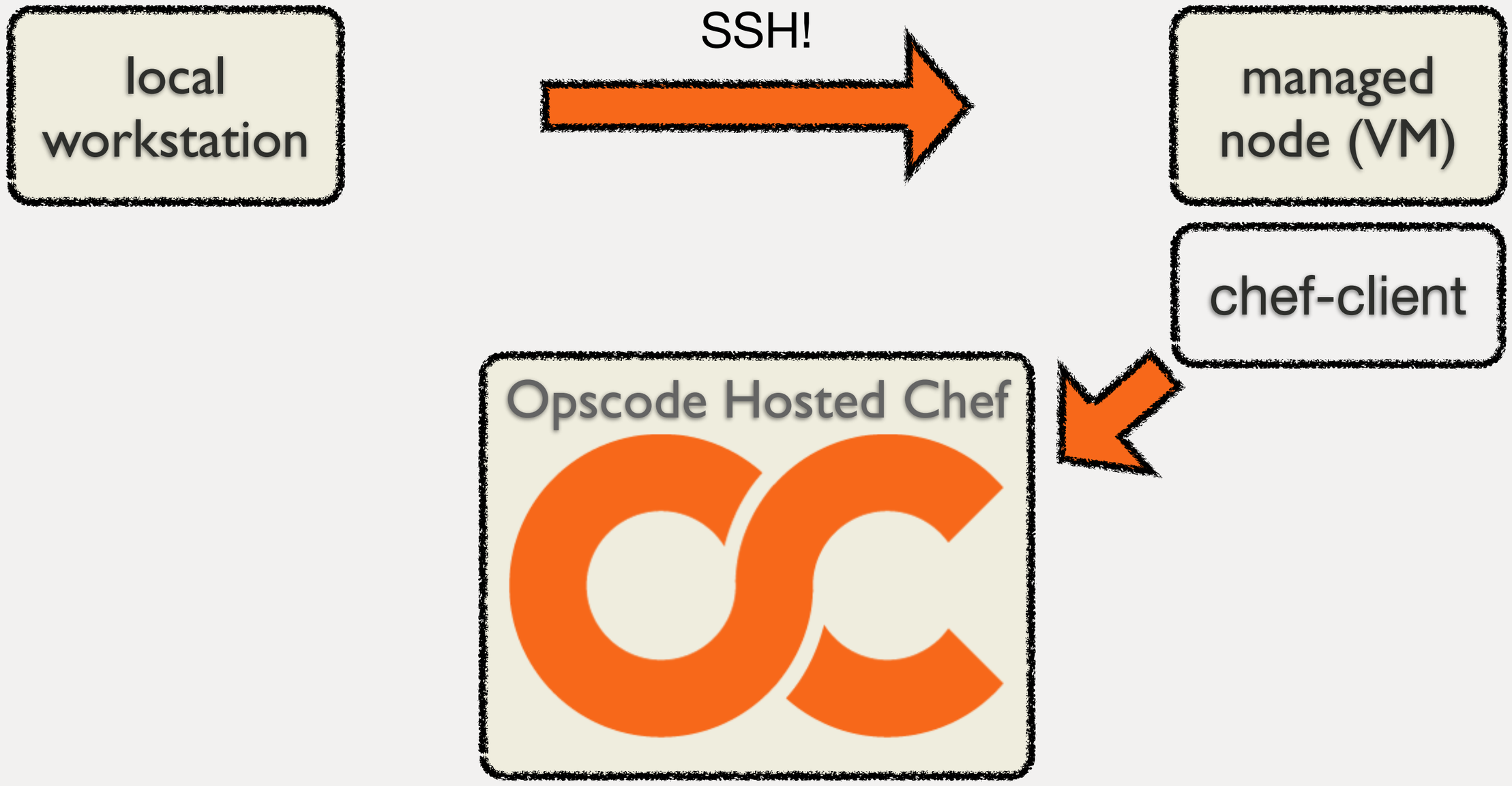
SSH!

managed node (VM)

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



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



Chef 101 Terminology

OPSCODE | Chef

chef-client runs on your systems

OPSCODE | Chef

chef-client talks to a Chef Server

OPSCODE | Chef

API Clients authenticate with RSA keys

The server has the public key

OPSCODE | Chef

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

OPSCODE | Chef

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

Current Status - Managed Node

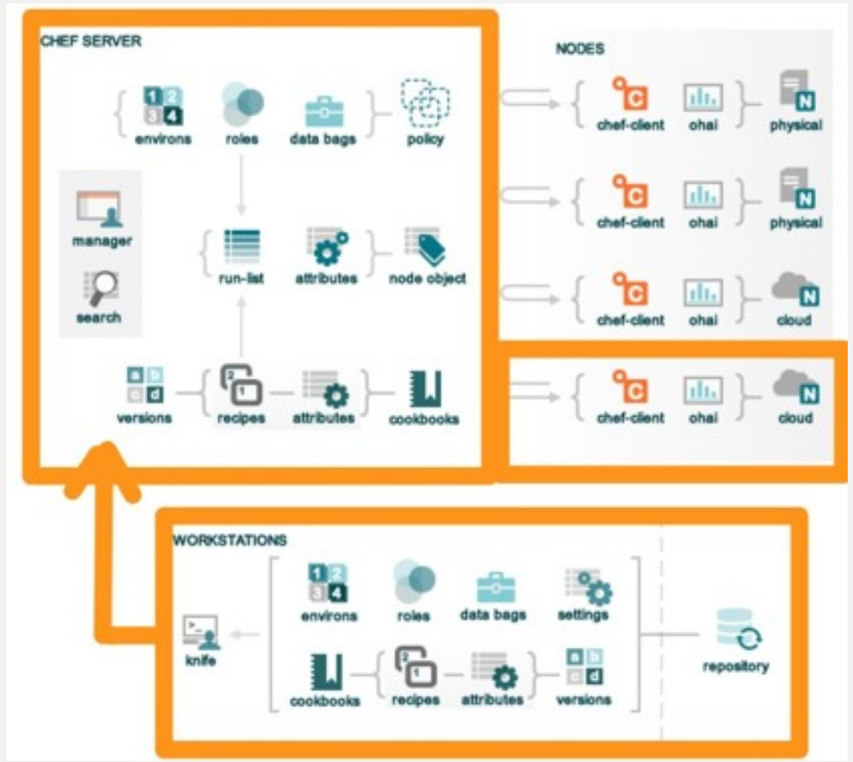
OPSCODE | Chef

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
```



OPSCODE | Chef

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
```

OPSCODE | Chef

/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)
```

OPSCODE | Chef

/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
```

Private Keys

- Remember from the authentication cycle: Chef Server requires keys to authenticate.
- client.pem - private key for API client
- validation.pem - private key for ORGNAME-validator

Writing an Apache cookbook

Packages, Cookbook Files, and Services

OPSCODE

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

OPSCODE

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.

OPSCODE

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.

RULE THE CLOUD

OPSCODE

Required steps

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

RULE THE CLOUD

OPSCODE | Chef

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
```

OPSCODE | Chef

Exercise: Open the default apache recipe in your editor

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

```
1 #
2 # Cookbook Name:: apache
3 # Recipe:: default
4 #
5 # Copyright 2012, YOUR_COMPANY_NAME
6 #
7 # All rights reserved - Do Not Redistribute
8 #
9
```

OPSCODE

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*

OPSCODE | Chef

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

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

```
 7 # All rights reserved - Do Not Redistribute
 8 #
 9
10 package "apache2" do
11   action :install
12 end
13
```

OPSCODE | Chef

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
```

OPSCODE | Chef

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
```

OPSCODE | Chef

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
```

OPSCODE | Chef

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

```
10 package "apache2" do
11     action :install
12 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

OPSCODE | Chef

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

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

```
10 package "apache2" do
11   action :install
12 end
13
14 service "apache2" do
15   action [ :start, :enable ]
16 end
17
```

OPSCODE

So the resource we just wrote...

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

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

OPSCODE

Order Matters

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

RULE THE CLOUD

OPSCODE | Chef

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

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

```
14 service "apache2" do
15   action [ :start, :enable ]
16 end
17
18 cookbook_file "/var/www/index.html" do
19   source "index.html"
20   mode "0644"
21 end
22
```

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

```
1 #
2 # Cookbook Name:: apache
3 # Recipe:: default
4 #
5 # Copyright 2012, YOUR_COMPANY_NAME
6 #
7 # All rights reserved - Do Not Redistribute
8 #
9
10 package "apache2" do
11   action :install
12 end
13
14 service "apache2" do
15   action [ :start, :enable ]
16 end
17
18 cookbook_file "/var/www/index.html" do
19   source "index.html"
20   mode "0644"
21 end
~
```

RULE THE CLOUD

OPSCODE | Chef

Question!

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

OPSCODE | Chef

Best Practice: Omit the default action

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

OPSCODE | Chef

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>
~
```

OPSCODE | Chef

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.
```

OPSCODE | Chef

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

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

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

    ]
  },
  "run_list": [

  ]
}

```

OPSCODE | Chef

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

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

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

    ]
  },
  "run_list": [
    "recipe[apache]"
  ]
}

```

OPSCODE

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*]**"

OPSCODE | Chef

Exercise: Run the chef-client on your test node

```
$ sudo chef-client -Fdoc -lfatal
```

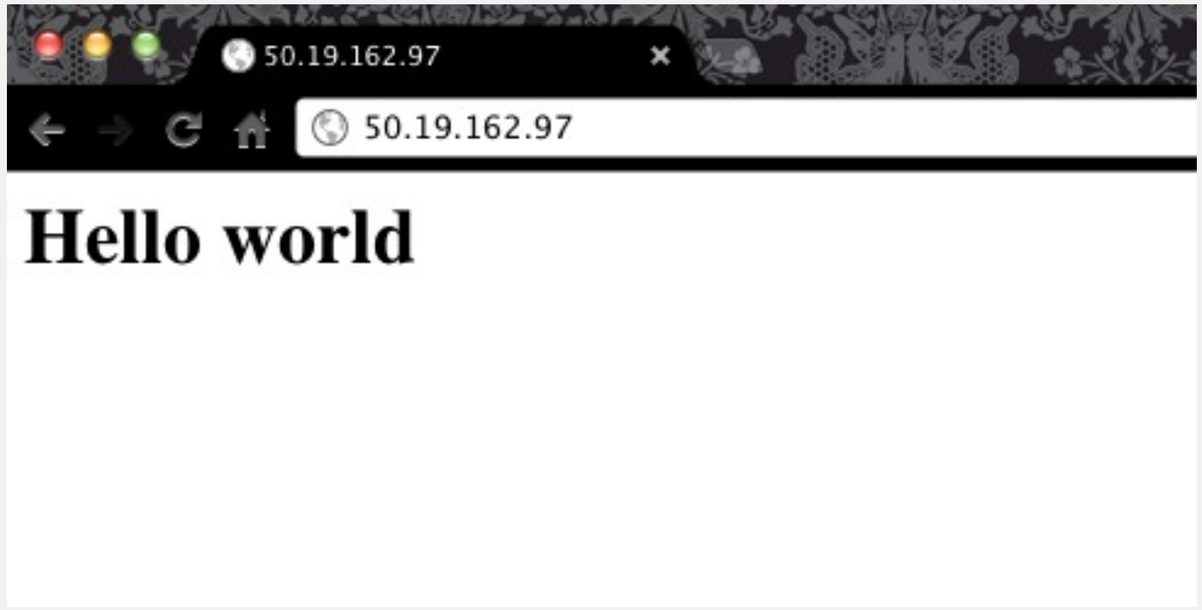
```
Starting Chef Client, version 11.4.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[apache2] action install
    - install version 2.2.22-1ubuntu1 of package apache2

  * service[apache2] action enable (up to date)
  * service[apache2] action start (up to date)
  * cookbook_file[/var/www/index.html] action create
    - create a new cookbook_file /var/www/index.html
        --- /tmp/chef-tempfile20130215-12136-mlqrm6    2013-02-15 13:47:53.136288008 -0800
        +++ /var/chef/cache/cookbooks/apache/files/default/index.html 2013-02-15 13:33:26.685884277 -0800
        @@ -0,0 +1,5 @@
        +<html>
        +  <body>
        +    <h1>Hello world</h1>
        +  </body>
        +</html>

Chef Client finished, 3 resources updated
```

OPSCODE | Chef

Exercise: Verify that the home page works



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

OPSCODE | Chef

Congratulate yourself!

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

OPSCODE | Chef

Reading the output of a chef-client run

```
Starting Chef Client, version 11.4.0
INFO: *** Chef 11.4.0 ***
INFO: [inet6] no default interface, picking the first ipaddress
INFO: Run List is [recipe[apache]]
INFO: Run List expands to [apache]
INFO: Starting Chef Run for target.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

OPSCODE | Chef

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

OPSCODE | Chef

Reading the output of a chef-client run

```
Recipe: apache::default
  * package[apache2] action install
    - install version 2.2.22-1ubuntu1 of package
apache2
```

- 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)

OPSCODE | Chef

Reading the output of a chef-client run

```
* service[apache2] action enable (up to date)
 * service[apache2] action start (up to date)
```

- 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

OPSCODE | Chef

Convergence and Idempotence

- Actions on resources in Chef are designed to be **convergent**
- 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
- Convergent resources are **idempotent**
- Idempotent functions yield the same result with every application

Reading the output of a chef-client run

```
  * service[apache2] action enable (up to date)
  * service[apache2] action start (up to date)
  * cookbook_file[/var/www/index.html] action create
    - create a new cookbook_file /var/www/index.html
        --- /tmp/chef-tempfile20130215-12136-mlqrm6   2013-02-15 13:47:53.136288008 -0800
        +++ /var/chef/cache/cookbooks/apache/files/default/index.html   2013-02-15
13:33:26.685884277 -0800
        @@ -0,0 +1,5 @@
        +<html>
        +  <body>
        +    <h1>Hello world</h1>
        +  </body>
        +</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 -Fdoc -lfatal
```

```
Starting Chef Client, version 11.4.0
resolving cookbooks for run list: ["apache"]
Synchronizing Cookbooks:
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[apache2] action install (up to date)
  * service[apache2] action enable (up to date)
  * service[apache2] action start (up to date)
  * cookbook_file[/var/www/index.html] action create (up to date)
Chef Client finished, 0 resources updated
```

OPSCODE

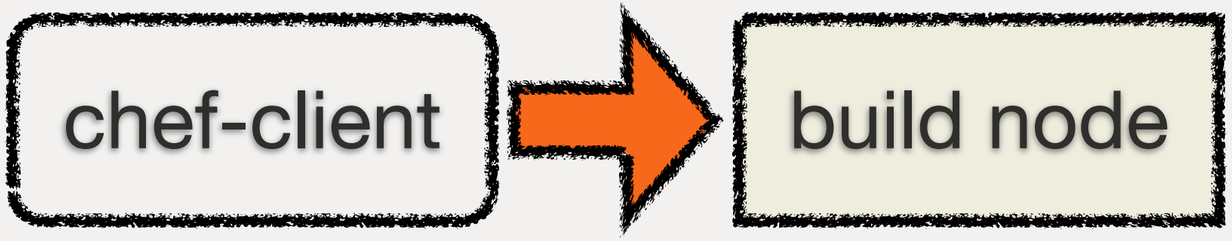
Questions

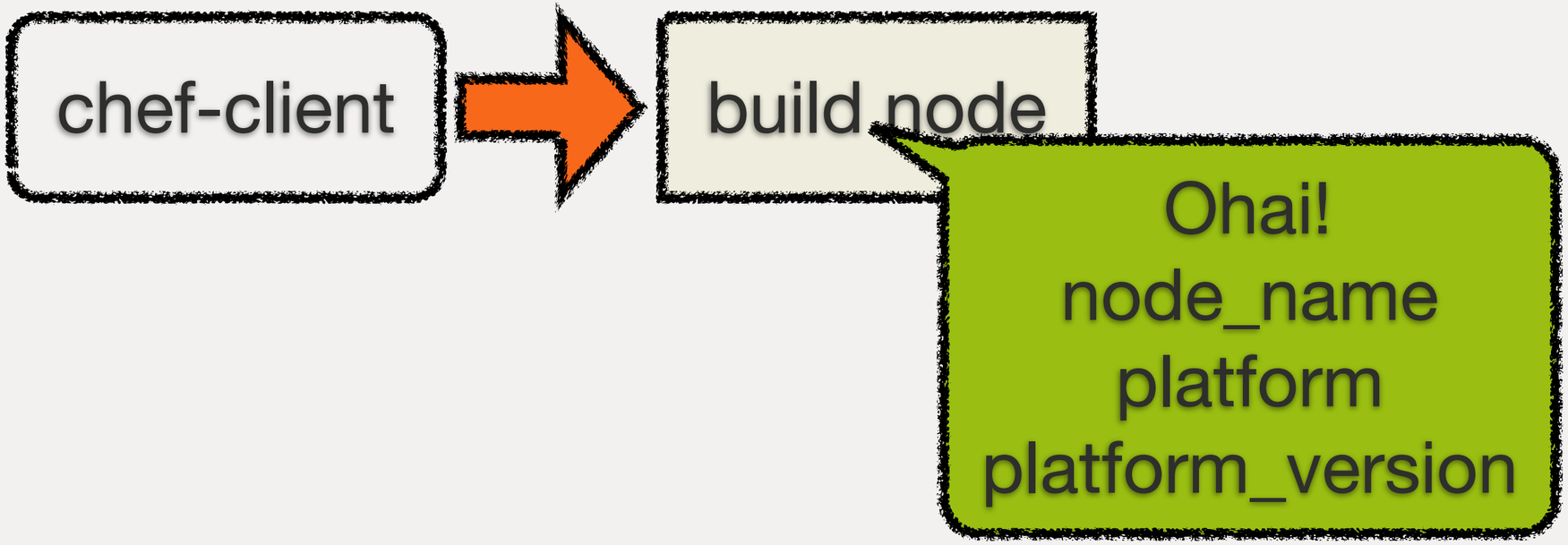
- What is a cookbook?
- How do you create a new cookbook?
- What is a recipe?
- What is a resource?
- How do you upload a cookbook to the Chef Server?
- What is a run list?

RULE THE CLOUD

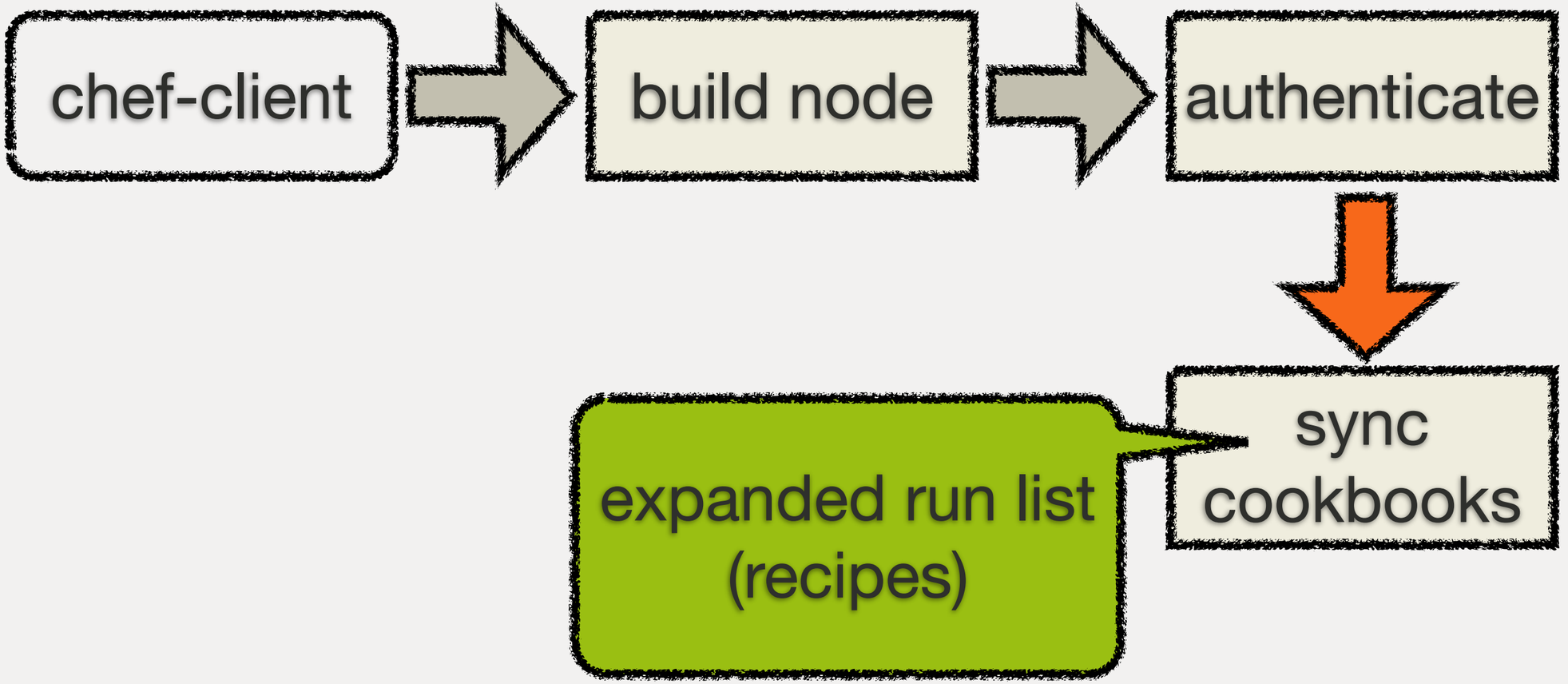
Anatomy of a Chef Run

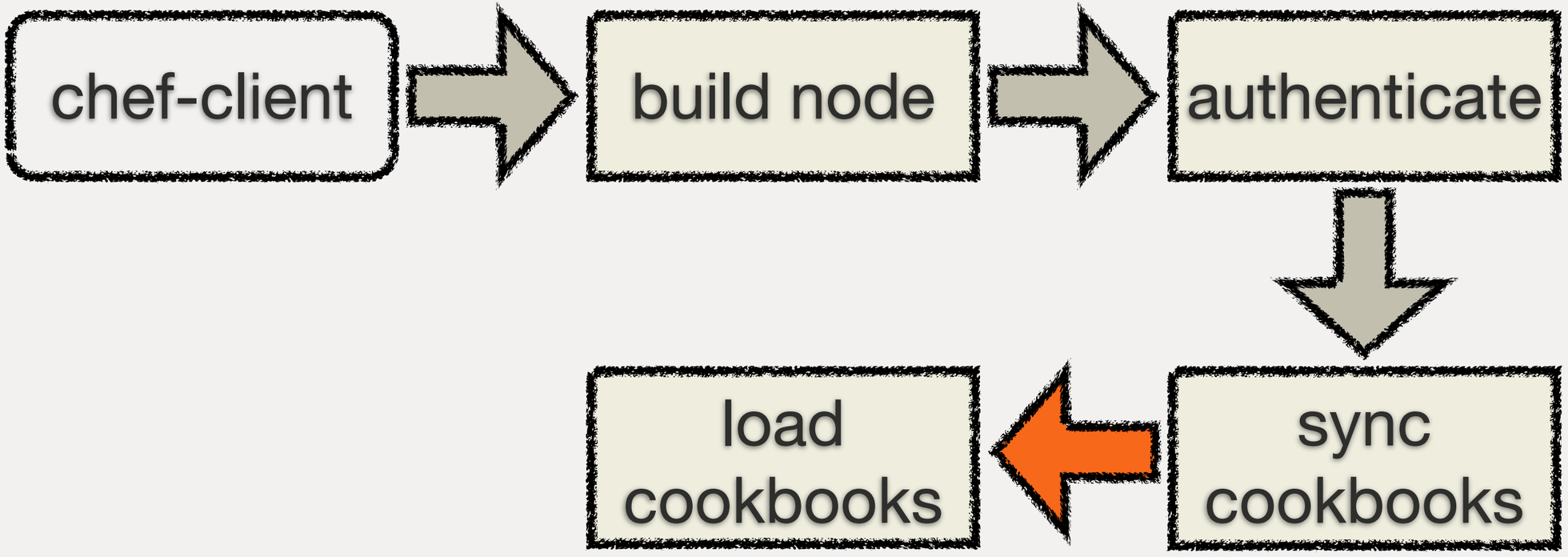
chef-client

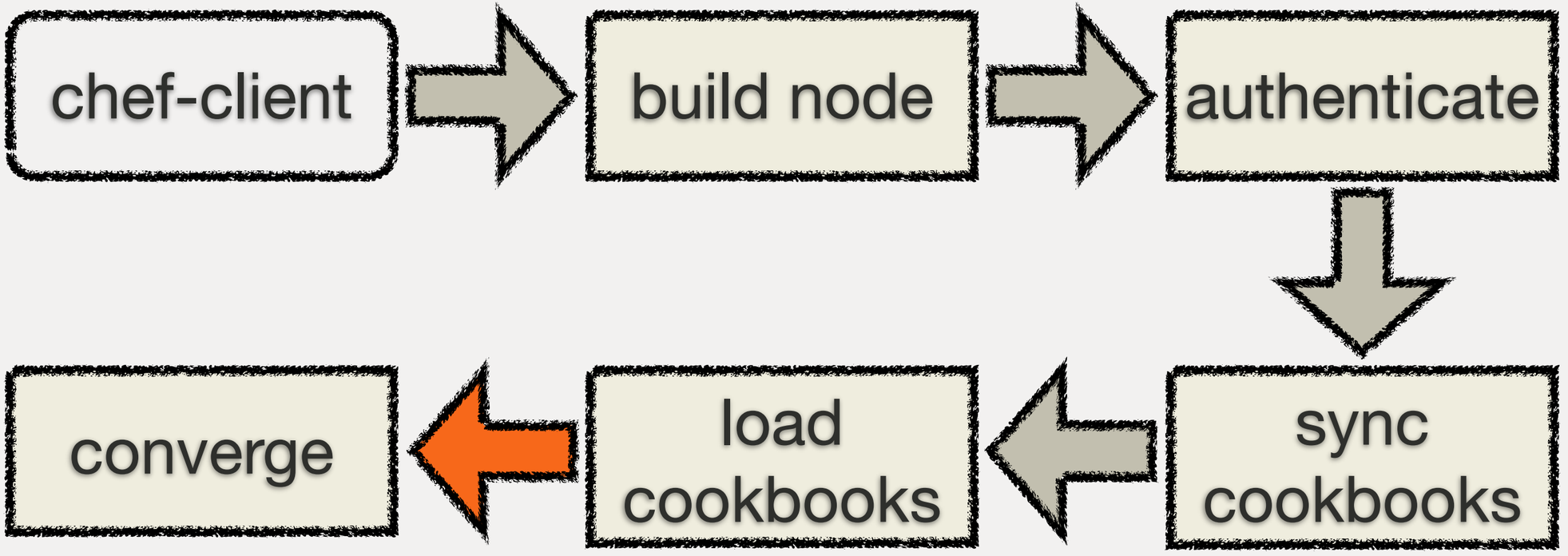

chef-client
build node

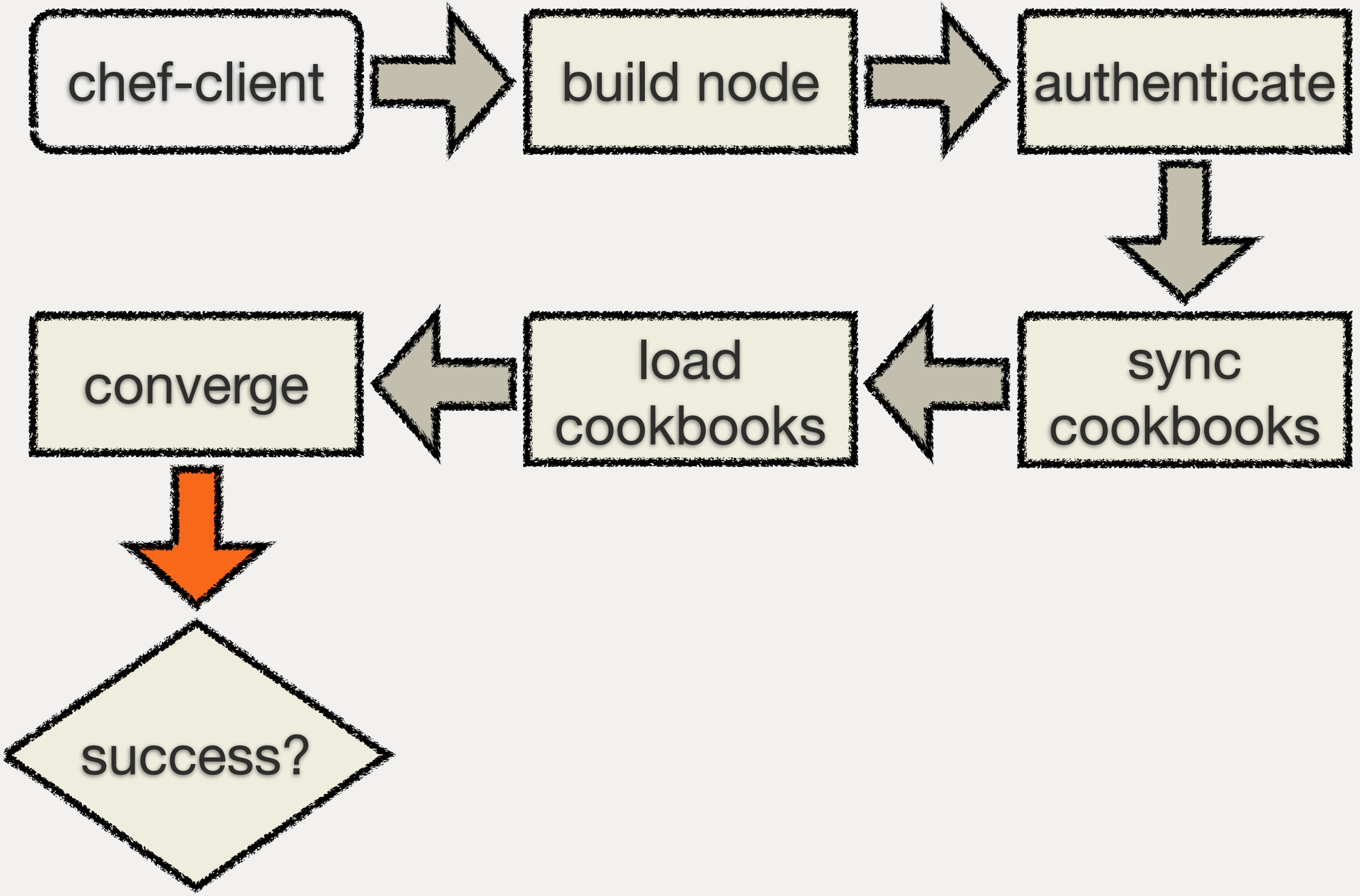

chef-client
build node
Ohai!
node_name
platform
platform_version

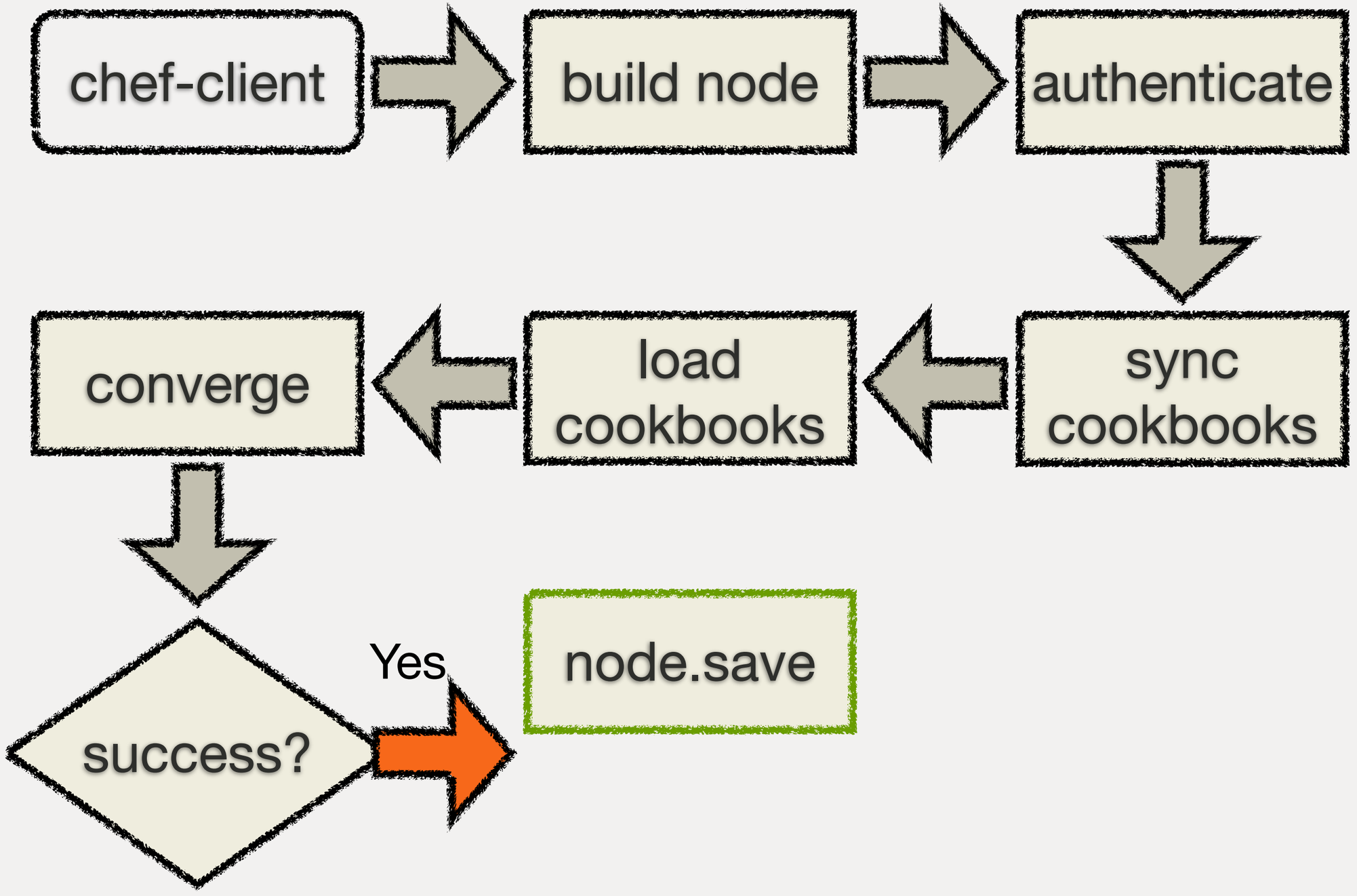

chef-client
build node
authenticate

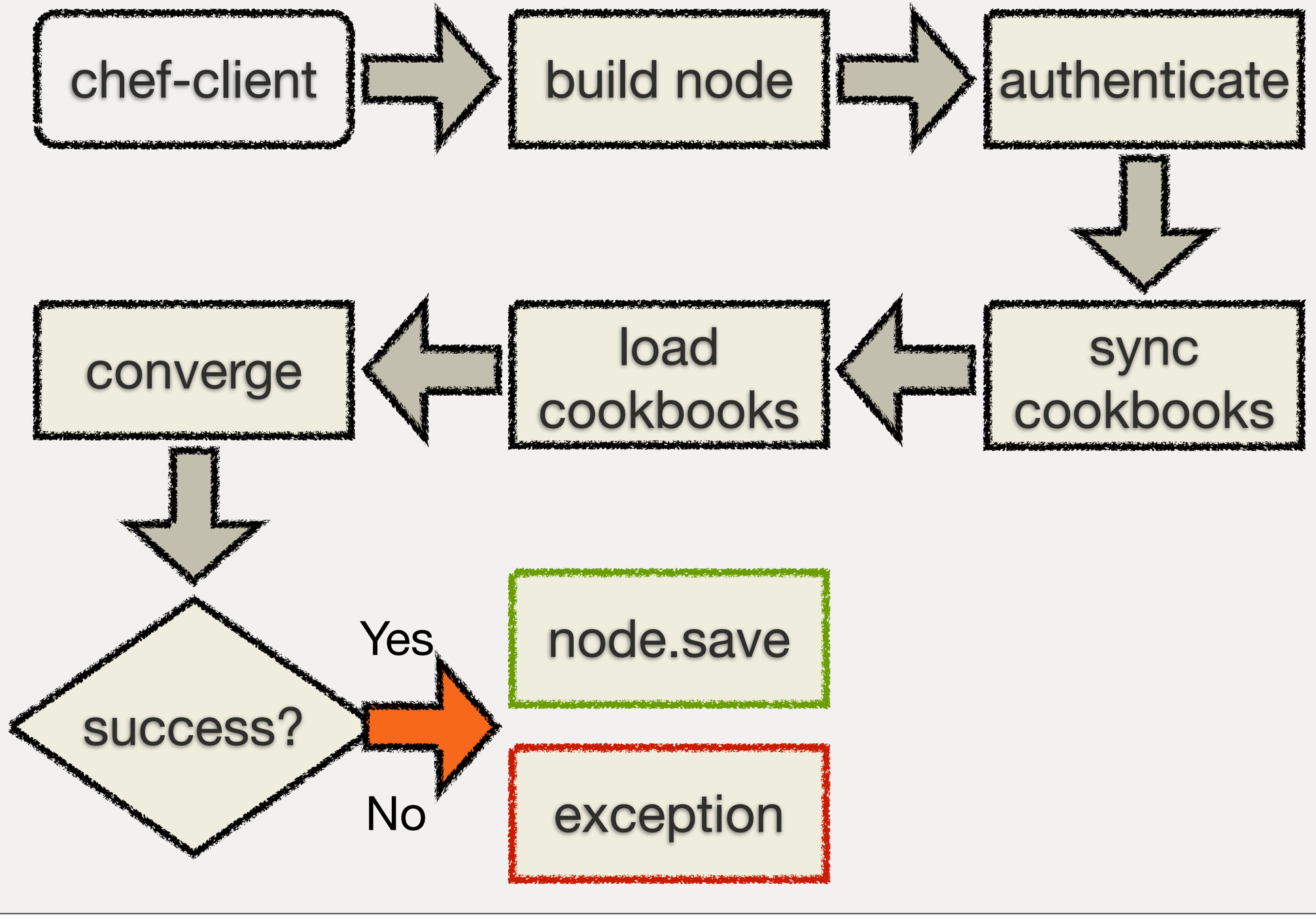

chef-client
build node
authenticate
sync cookbooks
expanded run list (recipes)

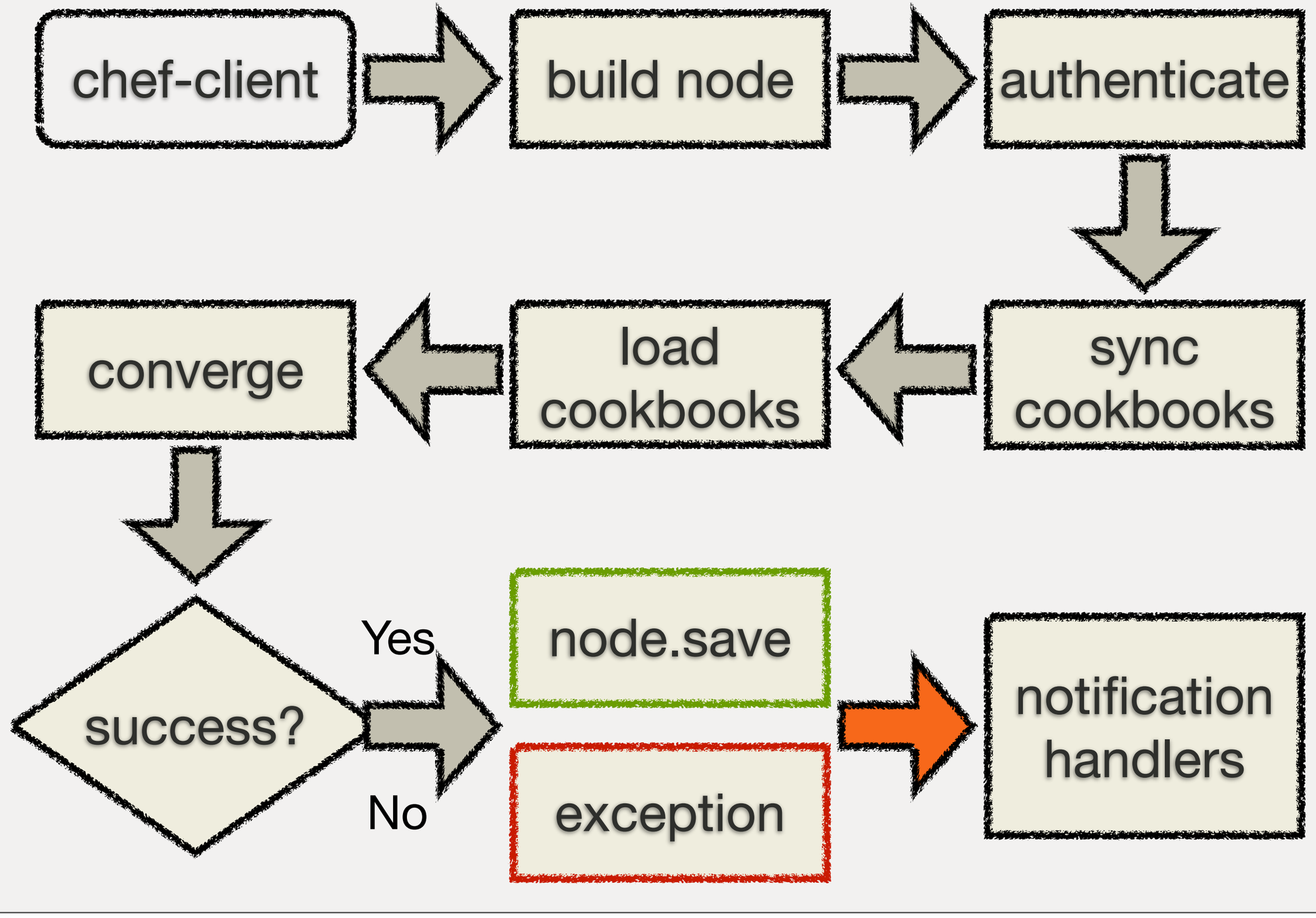

chef-client
build node
authenticate
sync cookbooks
load cookbooks

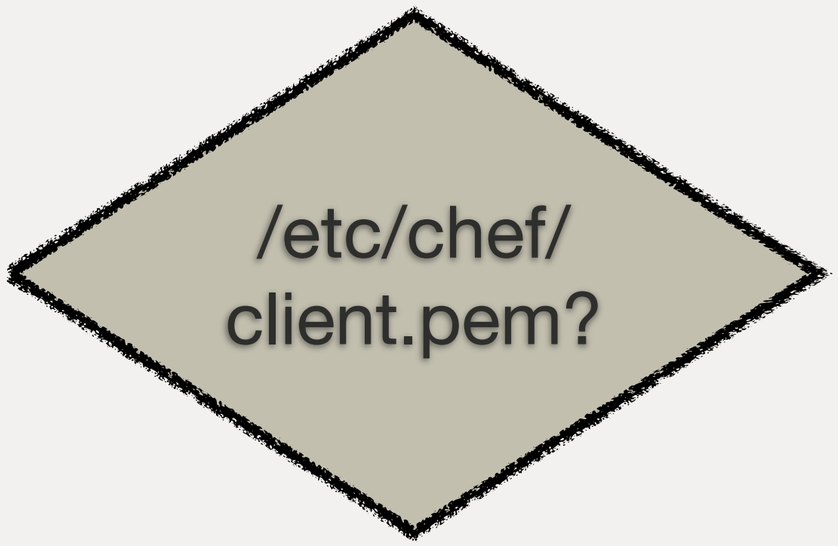

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge

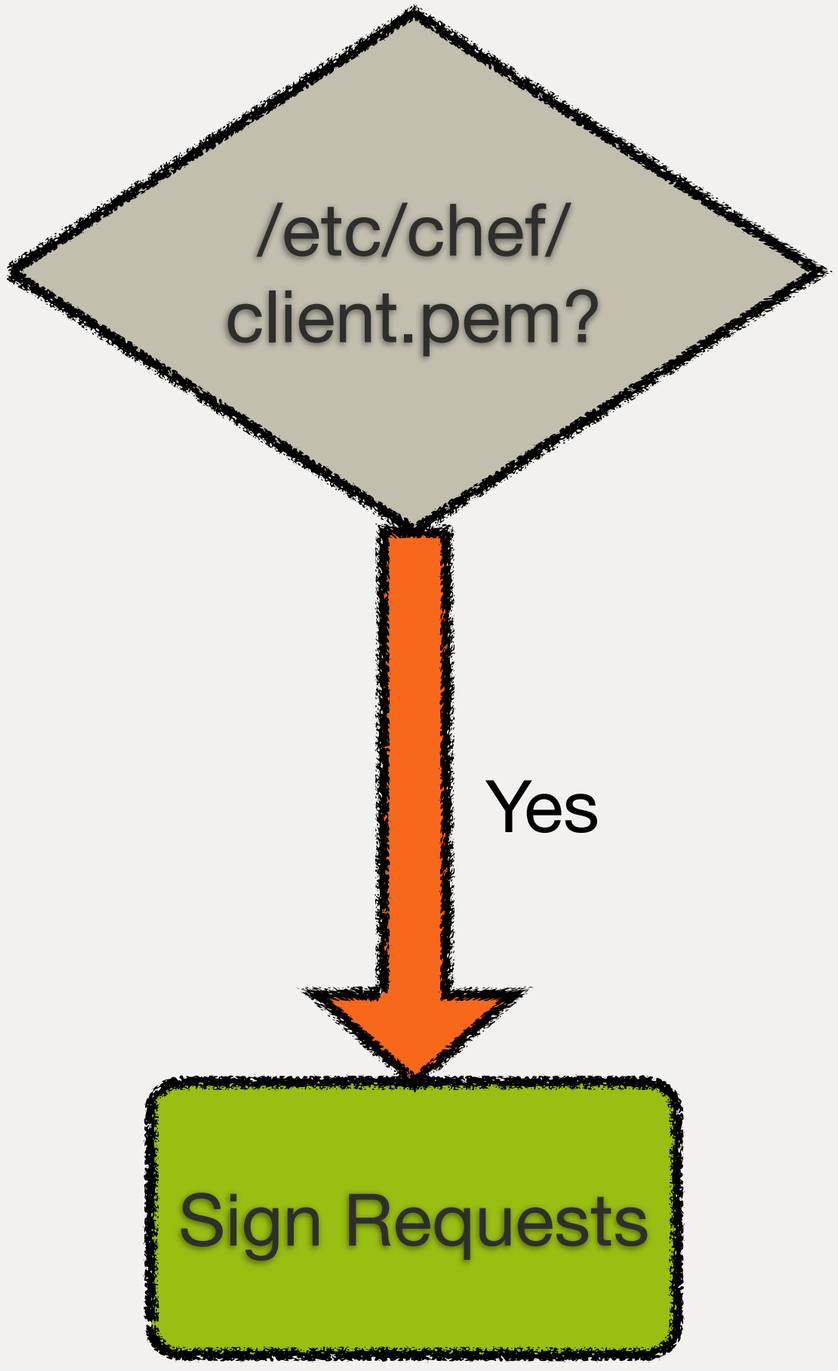

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?

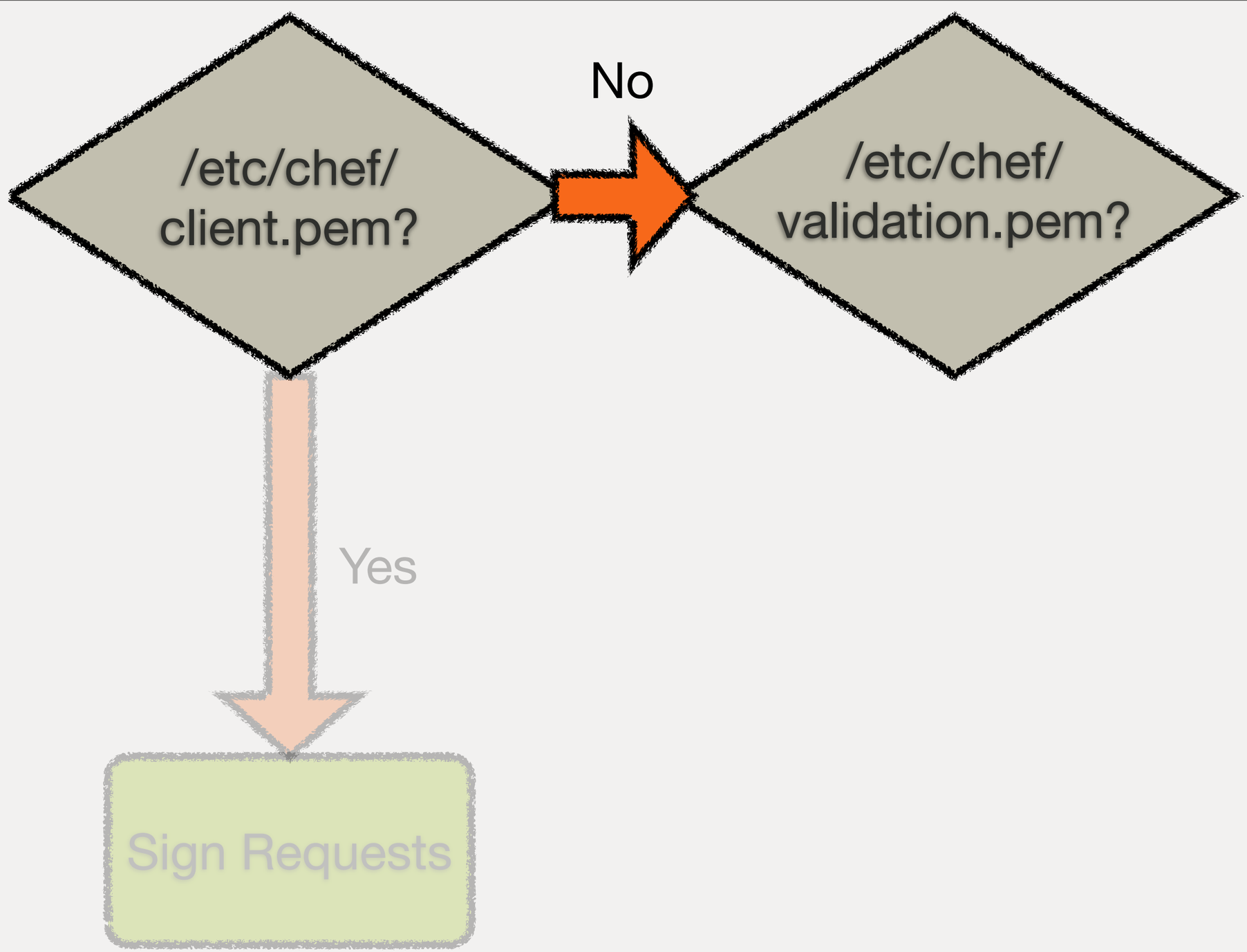

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
node.save

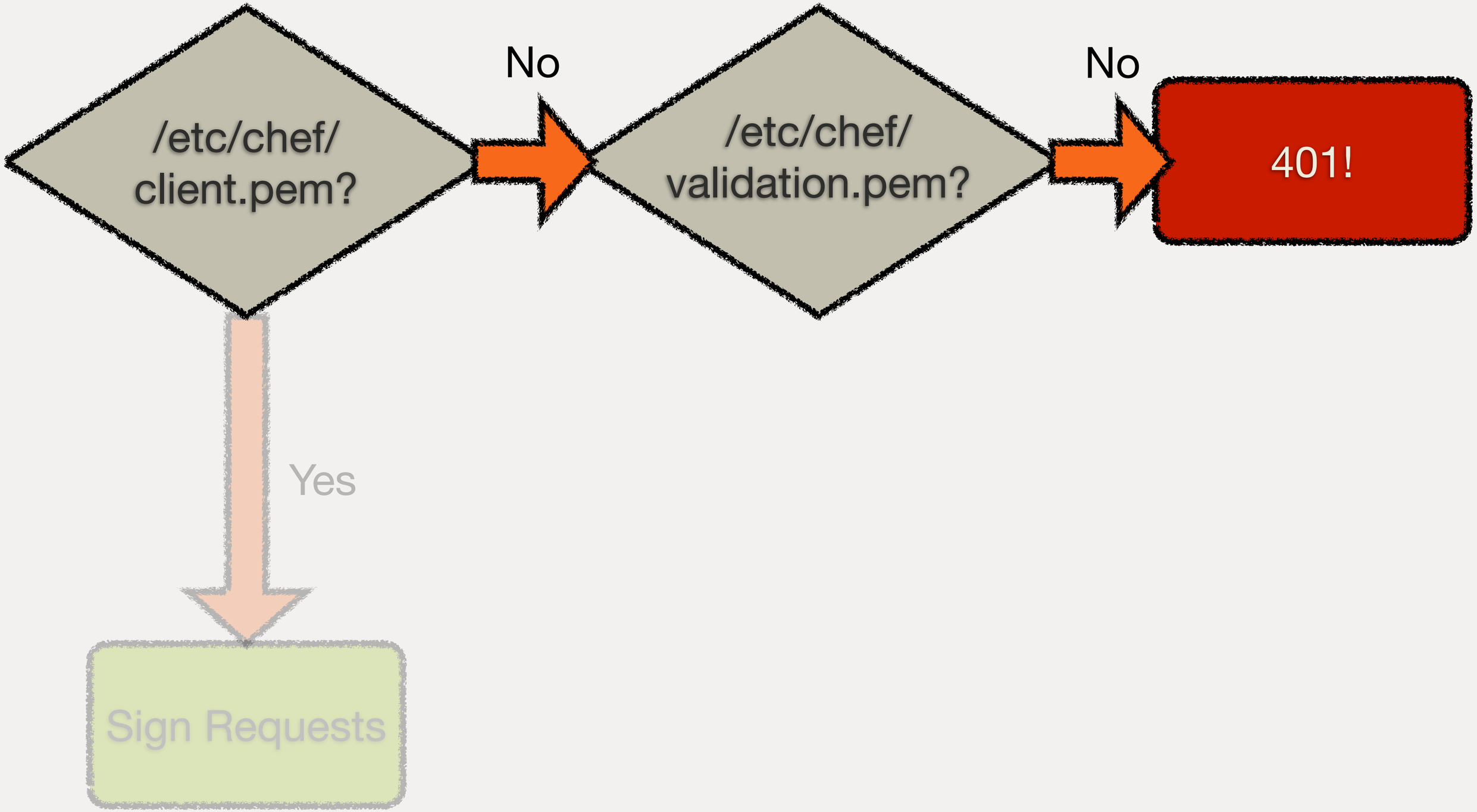

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
No
node.save
exception

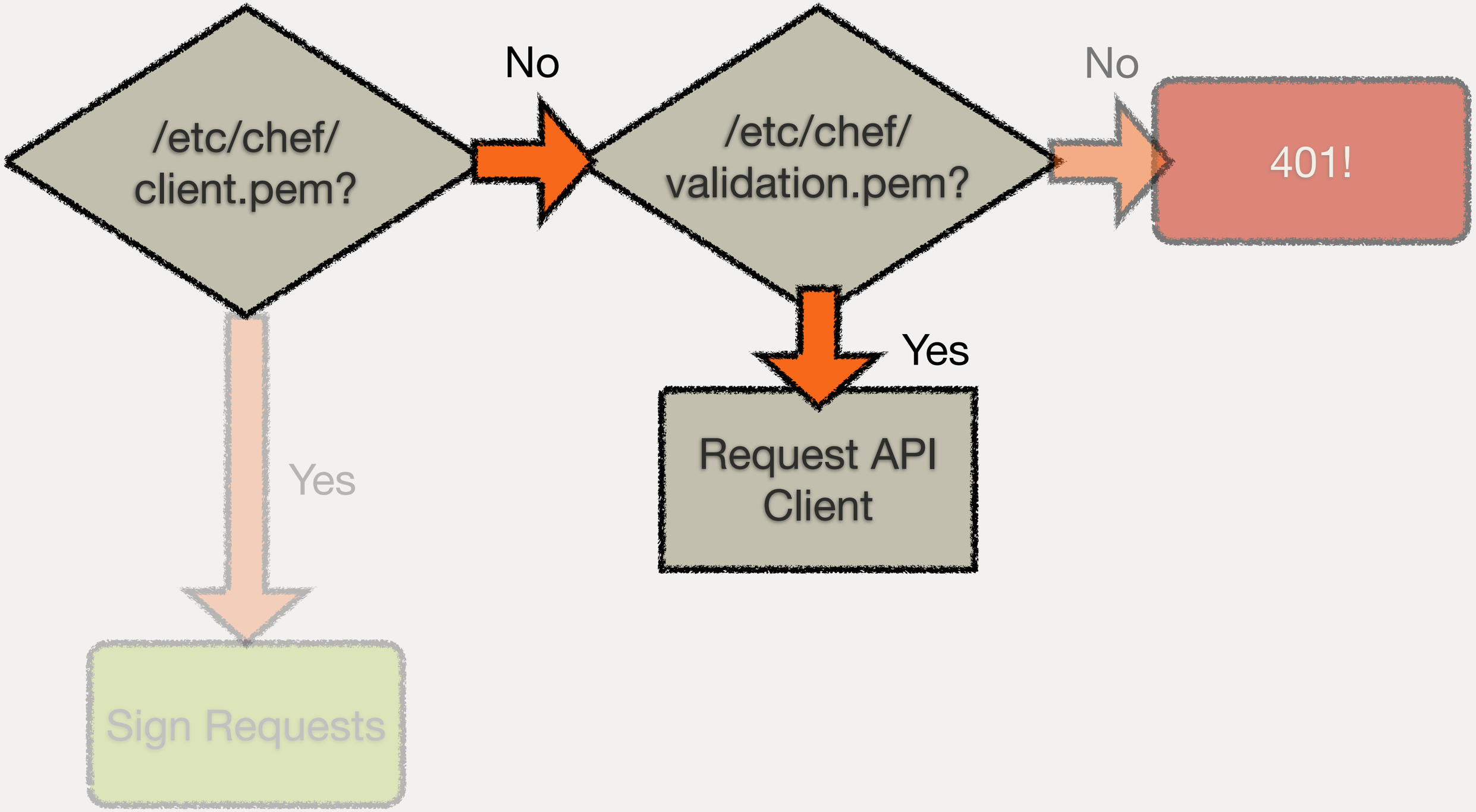

chef-client
build node
authenticate
sync cookbooks
load cookbooks
converge
success?
Yes
No
node.save
exception
notification handlers

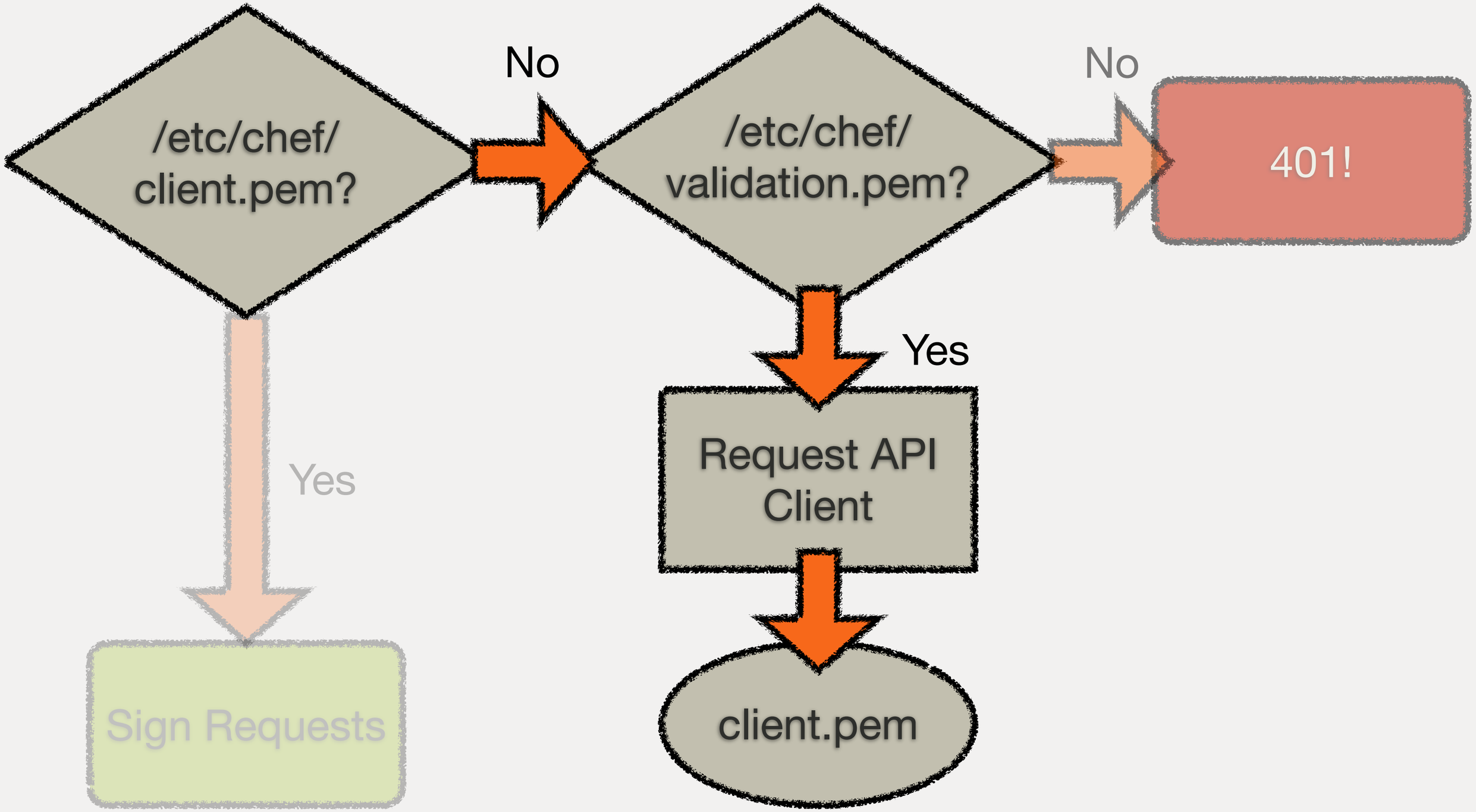

/etc/chef/
client.pem?

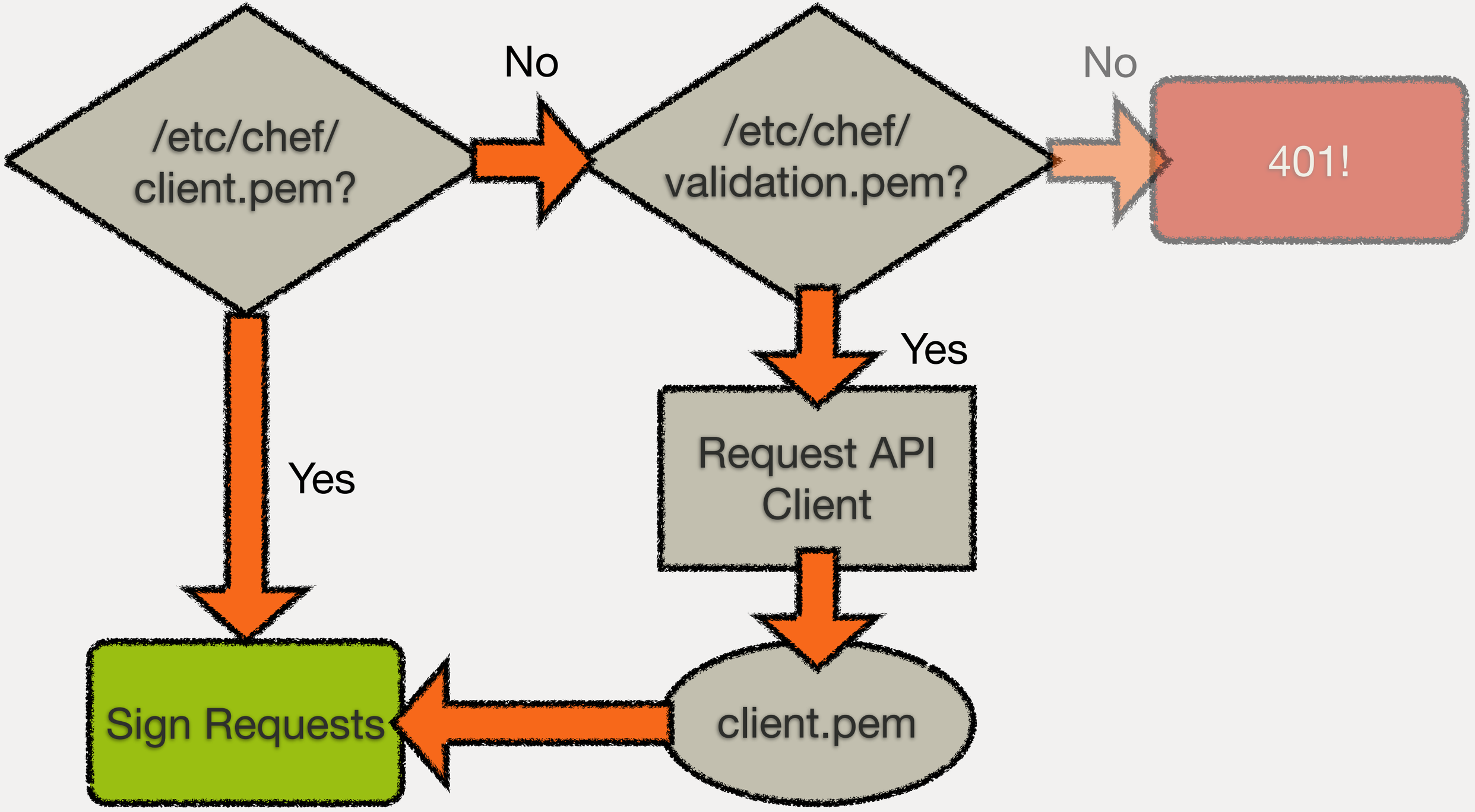

/etc/chef/
client.pem?
Yes
Sign Requests


/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
Yes
Request API
Client
Sign Requests
client.pem

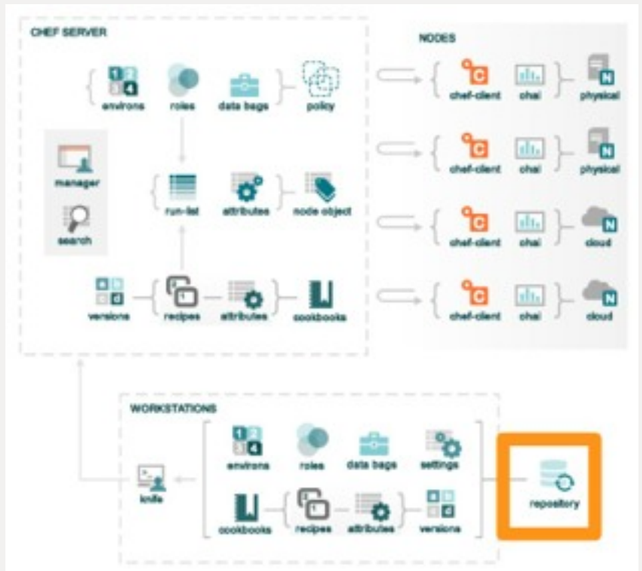
Questions? no? you sure? Break!

Install & Configure Nagios

OPSCODE | Chef

Setup Your Chef Repository

```
> git clone git://github.com/opscode/chef-repo-workshop-
sysadmin
> cd chef-repo-workstop-sysadmin
> rm -rf .git
> mkdir -p .chef
```



Upload to Chef Server

OPSCODE | Chef

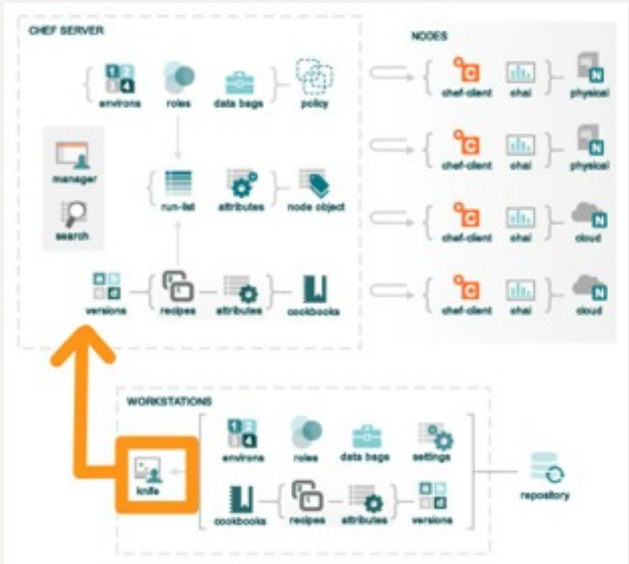
Upload the Chef Repository to Chef Server

```
> knife cookbook upload -a

> knife role from file roles/*.rb

> knife data bag create users

> knife data bag from file users nagiosadmin.json
```



local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

Opscode Hosted Chef

local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

chef_server_url
node_name
client_key

Opscode Hosted Chef

local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

Opscode Hosted Chef

local workstation

knife cookbook upload -a

Opscode Hosted Chef

cookbooks

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

local workstation

knife cookbook upload -a

knife role from file base.rb

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

Opscode Hosted Chef

cookbooks

roles

local workstation

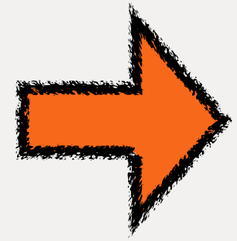
Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

knife cookbook upload -a

knife role from file base.rb

knife data bag from file ...



Opscode Hosted Chef

cookbooks

roles

data bags

OPSCODE | Chef

Exercise: Add the base and monitoring roles to your test node's run list

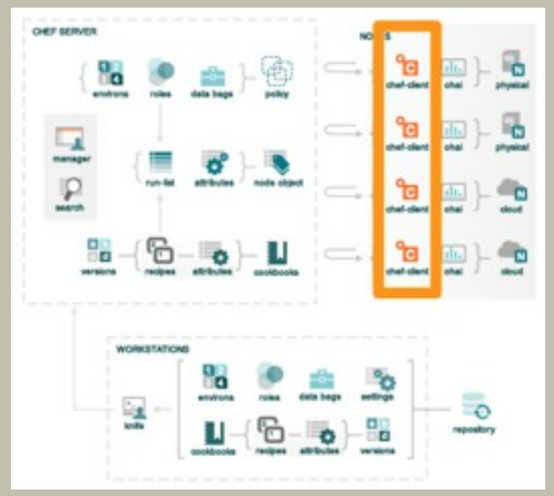
```
$ knife node edit target1.local
```

```
$ knife node run_list add target1.local \
  "role[base],role[monitoring]"
```

```
$ knife node run_list remove target1.local \
  "recipe[apache]"
```

OPSCODE

Running the Chef Client



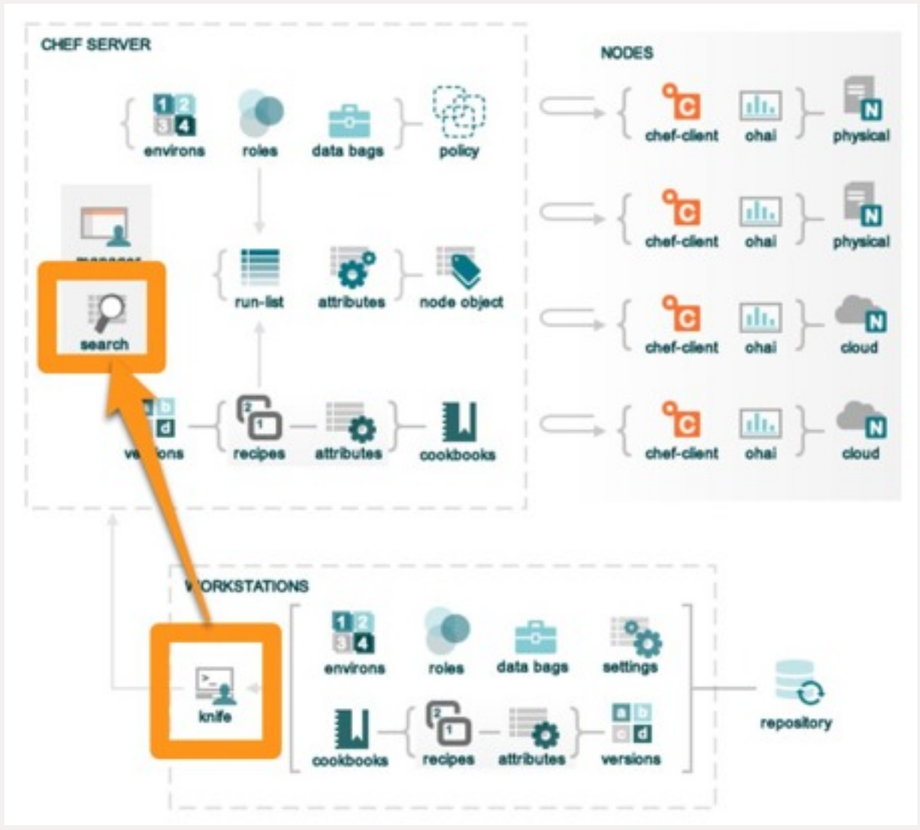
- Automatically
  - cron
  - daemon

RULE THE CLOUD

OPSCODE

knife ssh

```
$ knife ssh "role:base"
```

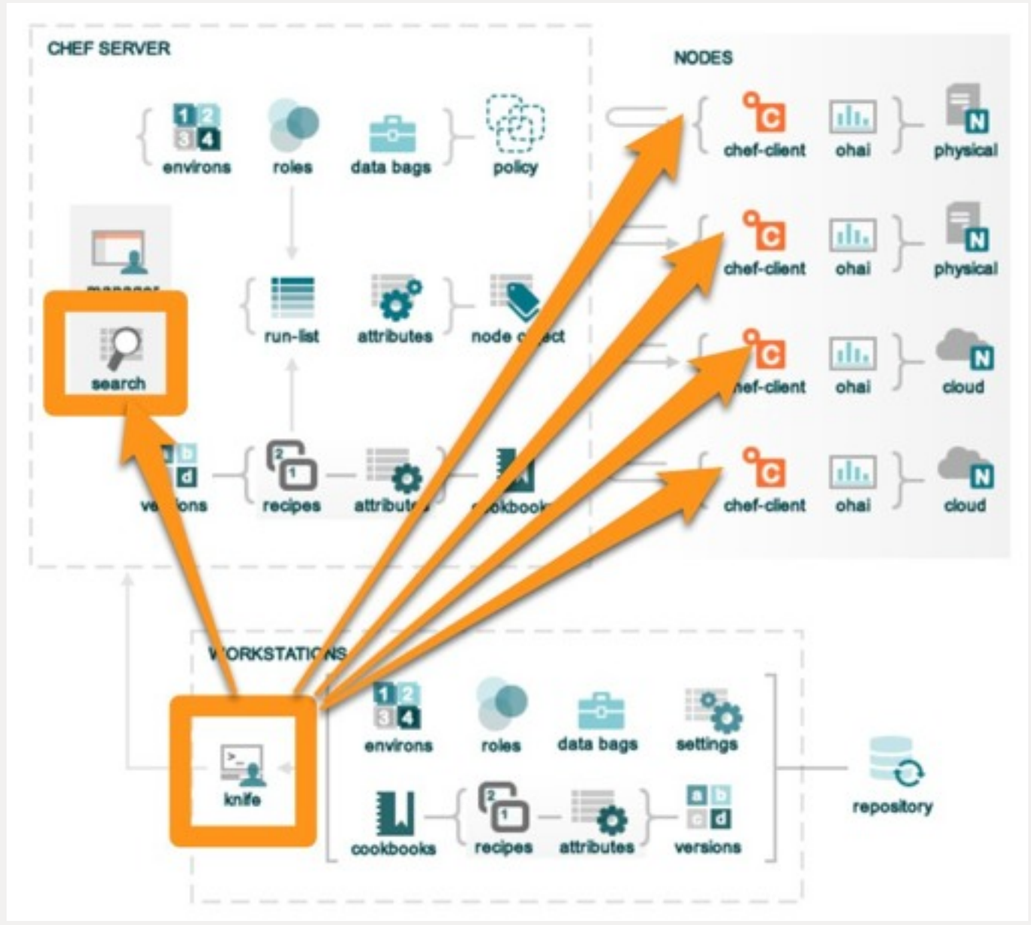


RULE THE CLOUD

OPSCODE

knife ssh

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



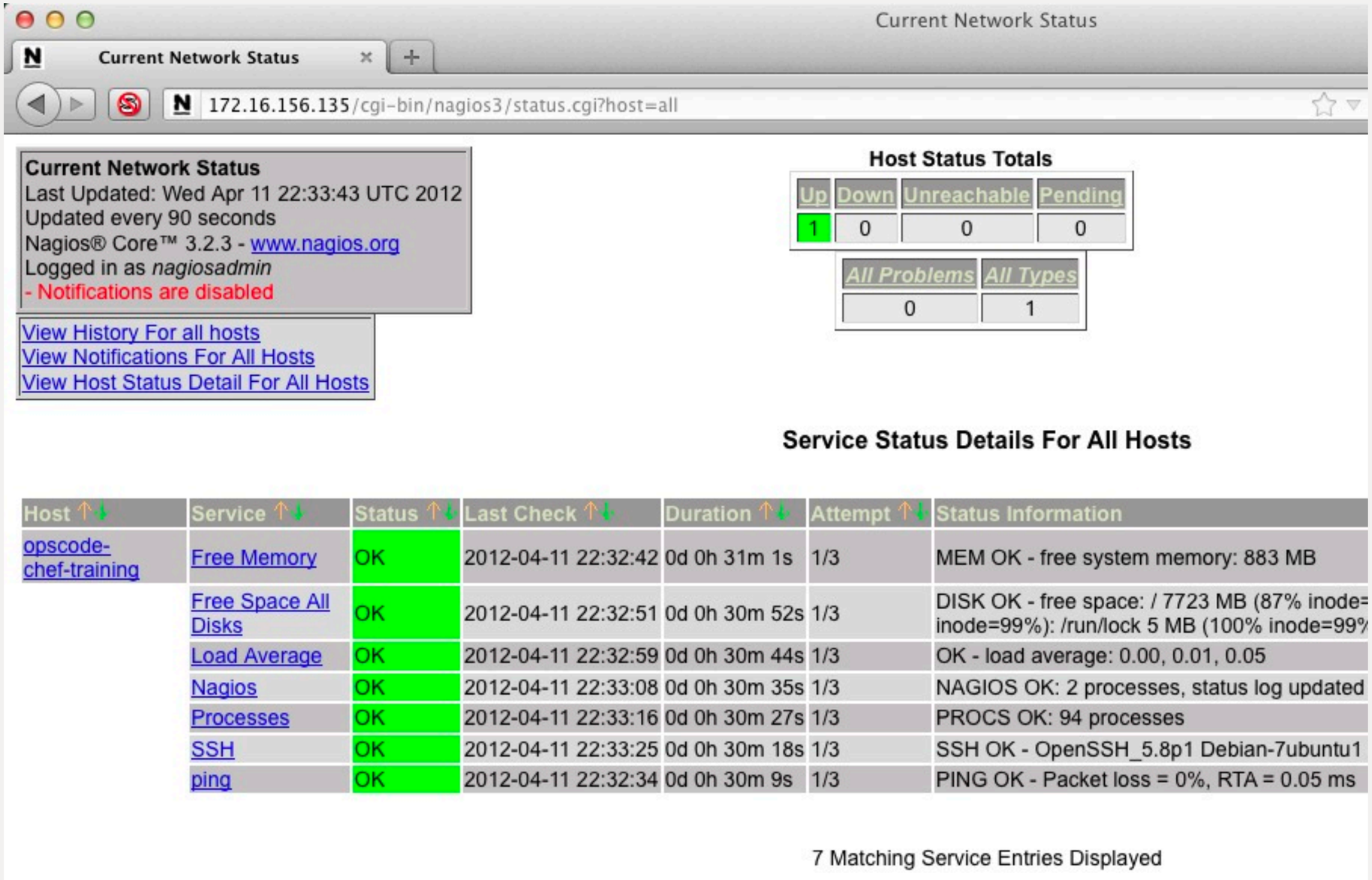
RULE THE CLOUD

OPSCODE | Chef

The Node is a Monitoring (Nagios) Server

- Navigate to http://PUBLIC_HOSTNAME
- *Username*: **nagiosadmin**
- *Password*: **nagios**

- *Username*: **nagiosadmin**
- *Password*: **nagios**



- Nagios installation and management is a common pain point.
- The Nagios cookbook makes heavy use of Chef's search feature.
- Dynamic host groups, service checks, etc.

Detailed discussion about the cookbook is outside the scope of this workshop.

roles/base.rb

```
name "base"
description "Base role applied to all nodes."
run_list(
  "recipe[apt]",
  "recipe[nagios::client]"
)

default_attributes(
  "nagios" => {
    "server_role" => "monitoring"
  }
)
```

roles/monitoring.rb

```
name "monitoring"
description "Monitoring server"
run_list(
  "recipe[nagios::server]"
)

default_attributes(
  "nagios" => {
    "server_auth_method" => "htauth"
  }
)
```

Hands on Exercises

OPSCODE | Chef

The pattern for each exercise is a common Chef workflow

- Download cookbooks from Chef Community Site with Knife.
- Extract the cookbook's .tar.gz into cookbooks directory.
- Review the code you're going to run as root.
- Upload the cookbook to the Chef Server.
- Apply the cookbook to your node(s) with a role.
  - Edit role's run list (base, monitoring)
  - Modify attributes as required

OPSCODE | Chef

Download, extract, upload

```
> knife cookbook site download COOKBOOK

> tar -zxvf COOKBOOK*.tar.gz -C cookbooks

> less cookbooks/COOKBOOK/README.md
> less cookbooks/COOKBOOK/recipes/default.rb

> knife cookbook upload COOKBOOK
```

local workstation

Chef Repository

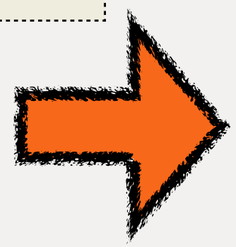
.chef/knife.rb
cookbooks/
data_bags/
roles/

local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

knife cookbook ***site*** download



Chef Community Site

Chef

local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

knife cookbook ***site*** download

Chef Community Site

Chef

local workstation

tar -zxvf cb.tar.gz -C cookbooks

Chef Community Site

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

```
cookbooks/COOKBOOK
├── metadata.rb
├── recipes
│   └── default.rb
└── templates
    └── default
        └── my-tmpl.erb
```

local workstation

tar -zxvf cb.tar.gz -C cookbooks

Chef Community Site

Chef

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

knife cookbook upload

Chef Community Site

Chef

Opscode Hosted Chef

local workstation

Chef Repository

.chef/knife.rb
cookbooks/
data_bags/
roles/

knife cookbook upload

Chef Community Site

Chef

Opscode Hosted Chef

Exercise: chef-client cookbook

Policy statement: Delete the validation key.

New concepts:

- Obtain cookbooks
- Upload cookbooks
- Chef resources
- Chef Configuration in recipes
- Roles Ruby DSL
- Run list items
- "Knife SSH"

- Download the chef-client cookbook.
- Extract it to the cookbooks directory.
- Upload it to the Chef Server.
- Add "recipe[chef-client::delete_validation]" to the base role's run list.
- Run Chef on the target managed node.

OPSCODE | Chef

Exercise: chef-client cookbook

```
> knife cookbook site download chef-client

> tar -zxvf chef-client*.tar.gz -C cookbooks

> knife cookbook upload chef-client
```

FAIL!

OPSCODE | Chef

Exercise: chef-client cookbook

```
> knife cookbook upload chef-client
Uploading chef-client    [2.1.10]
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.
```

cookbooks/chef-client/recipes/delete_validation.rb

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

OPSCODE | Chef

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
```

OPSCODE | Chef

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
```

OPSCODE | Chef

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 | Chef

Best Practice: Chef Resource Documentation

- Chef pros use the Chef Documentation page for resource documentation.
  - http://docs.opscode.com/chef/resources.html
- Bookmark the page and refer to it often!

OPSCODE | Chef

Apply recipe to base role

- Roles are located in the "roles" directory.
- The base role is "roles/base.rb".
- We uploaded this earlier with knife. Now we're going to edit it.

roles/base.rb

```
name "base"
 description "Base role applied to all nodes."
 run_list(
   "recipe[chef-client::delete_validation]"
 )
```

OPSCODE | Chef

Upload the role

```
> knife role from file base.rb

> knife role show base

> knife node list

> knife node run_list add
    acme-target1.training “role[base]”

> knife node show acme-target1.training
```

OPSCODE | Chef

Run Chef on the target managed node

```
> knife ssh role:base "sudo chef-client" -x opscode

...
Recipe: chef-client::delete_validation
  * file[/etc/chef/validation.pem] action delete
    - delete file /etc/chef/validation.pem
```

What happened?

```
knife ssh role:base "sudo chef-client" \
  -x USERNAME -P PASSWORD -a cloud.public_ipv4
```

- Knife SSH performs a search for nodes on the Chef Server with the query "role:base".
- Knife opens an SSH connection to the nodes' IP address (-a ipaddress or -a cloud.public_ipv4).
- The SSH session will connect as the user (-x opscode) with the password (-P opscode).
- The command "sudo chef-client" is passed to SSH and run on the node.

OPSCODE | Chef

Look at the Node

```
> knife node show target1
Node Name:   target1
Environment: _default
FQDN:        target1
IP:          10.12.13.201
Run List:    role[base], role[monitoring]
Roles:       base, monitoring
Recipes:     apt, nagios::client, chef-
client::delete_validation, nagios::server
Platform:    ubuntu 12.04
```

What happened?

```
knife ssh role:base "sudo chef-client" \
  -x USERNAME -P PASSWORD -a cloud.public_ipv4
```

- What is that “cloud.public_ipv4”?

OPSCODE | Chef

Verify the system

```
> ssh opscode@IPADDRESS
opscode@IPADDRESS's password:

opscode@target1:~$ sudo ohai
```

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"
  },
```

Chef 101 Terminology

OPSCODE | Chef

chef-client runs on your systems

OPSCODE | Chef

chef-client talks to a Chef Server

OPSCODE | Chef

API Clients authenticate with RSA keys

The server has the public key

OPSCODE | Chef

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

OPSCODE | Chef

Nodes have data called *attributes*

Attributes are indexed for *search* on the Chef Server

OPSCODE | Chef

Nodes have a **Run List**

The list of roles or recipes to apply *in order*

OPSCODE | Chef

Roles have a **Run List**

The list of roles or recipes to apply *in order*

OPSCODE | Chef

Chef manages **Resources** on the Node

OPSCODE | Chef

Resources are **declarative** descriptions of system resources

Resources describe **what**, not how.

OPSCODE | Chef

Resources are configured with *idempotent action* through **Providers**

OPSCODE | Chef

Providers know *how* to perform the **actions** to configure a Resource

OPSCODE | Chef

Recipes are lists of resources

Resources are applied in the order
they're written in recipes

OPSCODE | Chef

Cookbooks are *packages* for Recipes

OPSCODE | Chef

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

Exercise: motd-tail cookbook

OPSCODE | Chef

Exercise: motd-tail cookbook

Policy statement: Show which roles are applied by Chef on this node when users login with SSH.

New concepts:

- Node object methods
- Template resources
- ERb syntax

- Download the motd-tail cookbook.
- Extract it to the cookbooks directory.
- Upload it to the Chef Server.
- Add "recipe[motd-tail]" to the base role's run list.
- Run Chef on the target managed node.

OPSCODE | Chef

Exercise: chef-client cookbook

```
> knife cookbook site download motd-tail

> tar -zxvf motd-tail*.tar.gz -C cookbooks

> knife cookbook upload motd-tail
```

FAIL!

OPSCODE | Chef

Add to base role run list

```
name "base"
 description "Base role applied to all nodes."
 run_list(
   "recipe[chef-client::delete_validation]",
   "recipe[motd-tail]"
 )
```

OPSCODE | Chef

Exercise: chef-client cookbook

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

cookbooks/motd-tail/recipes/default.rb

```
template "/etc/motd.tail" do
 source "motd.tail.erb"
 group "root"
 owner "root"
 mode 00644
 backup 0
end
```

ERb Syntax

- To embed a value in an ERb template:
  - Start with <%=
  - Write the Ruby expression (e.g., node attribute)
  - End with %>
- This inserts the result of the value in the output file

ERb Syntax

- Use any Ruby statement in a template
  - Starting with <% evaluates the expression, but does not insert the result
  - Ending with -%> does not insert a line in the resulting file

cookbooks/motd-tail/templates/default/motd.tail.erb

```
***
Chef-Client - <%= node.name %>
Hostname: <%= node['cloud'] ? node['cloud']['public_hostname'] : node['fqdn'] %>
<% if ! Chef::Config[:solo] -%>
Chef Server: <%= Chef::Config[:chef_server_url] %>
<% end -%>
 <% if node.chef_environment != '_default' -%>
 Environment: <%= node.chef_environment %>
 <% end -%>
 Last Run: <%= ::Time.now %>

Roles:
<% node['roles'].each do |role| -%>
  <%= role %>
<% end -%>
***
```

cookbooks/motd-tail/templates/default/motd.tail.erb

```
***
Chef-Client - <%= node.name %>
Hostname: <%= node['cloud'] ? node['cloud']['public_hostname'] : node['fqdn'] %>
<% if ! Chef::Config[:solo] -%>
Chef Server: <%= Chef::Config[
<% end -%>
 <% if node.chef_environment != '_default' -%>
 Environment: <%= node.chef_environment %>
 <% end -%>
 Last Run: <%= ::Time.now %>

Roles:
<% node['roles'].each do |role| -%>
  <%= role %>
<% end -%>
***
```

Node method

Node method

cookbooks/motd-tail/templates/default/motd.tail.erb

```
***
Chef-Client - <%= node.name %>
Hostname: <%= node['cloud'] ? node['cloud']['public_hostname'] : node['fqdn'] %>
<% if ! Chef::Config[:solo] -%>
Chef Server: <%= Chef::Config[:chef_server_url] %>
<% end -%>
 <% if node.chef_environment != '_default' -%>
 Environment: <%= node.chef_environment %>
 <% end -%>
 Last Run:

Roles:
<% node['roles'].each do |role| -%>
  <%= role %>
<% end -%>
***
```

Node attribute

Node attribute

OPSCODE | Chef

cookbooks/motd-tail/templates/default/motd.tail.erb

```
***
Chef-Client - <%= node.name %>
Hostname: <%= node['cloud'] ? node['cloud']['public_hostname'] : node['fqdn'] %>
<% if ! Chef::Config[:solo] -%>
Chef Server: <%= Chef::Config[:chef_server_url] %>
<% end -%>
 <% if node.chef_environment != '_default' -%>
 Environment: <%= node.chef_environment %>
 <% end -%>
 Last Run: <%= ::Time.now %>

Roles:
<% node['roles'].each do |role| -%>
  <%= role %>
<% end -%>
***
```

Ruby method

Ruby loop

OPSCODE | Chef

Template Location

- Chef templates go in the "templates/default" directory of the cookbook.
- The "source" resource attribute indicates the file in this directory.
- Templates can be loaded from other cookbooks using the "cookbook" resource attribute.

OPSCODE | Chef

Look at the Node

```
> knife node show target1
Node Name:   target1
Environment: _default
FQDN:        target1
IP:          10.12.13.201
Run List:    role[base], role[monitoring]
Roles:       base, monitoring
Recipes:     apt, nagios::client, chef-
client::delete_validation, motd-tail, nagios::server
Platform:    ubuntu 12.04
```

OPSCODE | Chef

Look at the node

```
> ssh opscode@IPADDRESS
opscode@IPADDRESS's password:
***
Chef-Client: target1
Hostname: target1
Chef Server: https://chef.local/organizations/training
Last Run: 2013-02-07 05:35:31 +0000

Roles:
  base
  monitoring
***
```

Exercise: sudo cookbook

Exercise: sudo cookbook

Policy statement: User privileges will be managed through sudoers entries.

New concepts:

- Attribute priority
- Setting attributes in Cookbooks and Roles
- Using attributes in a template
- Ruby array iteration
- Package resource
- File backups

#protip: Log in and su to root!

- Download the sudo cookbook.
- Extract it to the cookbooks directory.
- Upload it to the Chef Server.
- Add "recipe[sudo]" to the base role's run list.
- Modify sudo-specific attributes in the base role.
- Run Chef on the target managed node.

OPSCODE | Chef

Exercise: chef-client cookbook

```
> knife cookbook site download sudo

> tar -zxvf sudo*.tar.gz -C cookbooks

> knife cookbook upload sudo
```

OPSCODE | Chef

Add sudo to base role run list

```
name "base"
description "Base role applied to all nodes."
run_list(
  "recipe[chef-client::delete_validation]",
  "recipe[motd-tail]",
  "recipe[sudo]"
)
```

OPSCODE | Chef

Add attributes for sudo to base role

```
default_attributes(
  "authorization" => {
    "sudo" => {
      "users" => ["opscode"],
      "groups" => ["admin","sudo"],
      "passwordless" => true
    }
  }
)
```

OPSCODE | Chef

Exercise: chef-client cookbook

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

cookbooks/sudo/attributes/default.rb

```
default['authorization']['sudo']['groups']              = []
default['authorization']['sudo']['users']               = []
default['authorization']['sudo']['passwordless']        = false
default['authorization']['sudo']['include_sudoers_d'] = false
default['authorization']['sudo']['agent_forwarding']  = false
```

OPSCODE | Chef

cookbooks/sudoers/recipes/default.rb

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

if node['authorization']['sudo']['include_sudoers_d']
  directory '/etc/sudoers.d' { ... }
  cookbook_file '/etc/sudoers.d/README' { ... }
end

template '/etc/sudoers' do
  source 'sudoers.erb'
  mode '0440'
  owner 'root'
  group 'root'
  variables(:sudoers_groups => node['authorization']['sudo']['groups'],
            :sudoers_users => node['authorization']['sudo']['users'],
            :passwordless => node['authorization']['sudo']['passwordless'],
            :include_sudoers_d => node['authorization']['sudo']['include_sudoers_d']
            :agent_forwarding => node['authorization']['sudo']['agent_forwarding'])
end
```

cookbooks/sudo/templates/default/sudoers.erb

```
root            ALL=(ALL) ALL

<% @sudoers_users.each do |user| -%>
<%= user %>    ALL=(ALL) <%= "NOPASSWD:" if @passwordless %>ALL
<% end -%>

# Members of the sysadmin group may gain root privileges
%sysadmin      ALL=(ALL) <%= "NOPASSWD:" if @passwordless %>ALL

<% @sudoers_groups.each do |group| -%>
# Members of the group '<%= group %>' may gain root privileges
%<%= group %> ALL=(ALL) <%= "NOPASSWD:" if @passwordless %>ALL
<% end -%>

<%= '#includedir /etc/sudoers.d' if @include_sudoers_d  %>
```

Updating Templates

```
* template[/etc/sudoers] action create
  - update template[/etc/sudoers] from 3bc623 to 7efd61
      --- /etc/sudoers  2012-01-31 15:56:42.000000000 +0000
      +++ /tmp/chef-rendered-template20130207-959-1rjg870 2013-02-07 03:51:04.126880000 +0000
      @@ -1,29 +1,23 @@
```

OPSCODE | Chef

File Content Updates

- "file", "template", "cookbook_file" and "remote_file"
- Default backup location is /var/chef/backup, configurable with "file_backup_path" in /etc/chef/client.rb
- 5 backups are kept by default, change this with the "backup" parameter in the resource.

OPSCODE | Chef

Verify sudo permissions

```
% ssh opscode@IPADDRESS
opscode@IPADDRESS's password:

opscode@target1:~$ sudo -l
Matching Defaults entries for opscode on this host:
    !lecture, tty_tickets, !fqdn

User opscode may run the following commands on this
host:
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
```

Exercise: users cookbook

Exercise: users cookbook

Policy statement: All sysadmins will be managed users from a common data bag.

New concepts:

- Data bags
- Encapsulating repeated functionality in custom resources (LWRP)
- Data driven recipes

- Download the users cookbook.
- Extract it to the cookbooks directory.
- Upload it to the Chef Server.
- Add "recipe[users::sysadmins]" to the base role's run list.
- Create a data bag of user items in JSON.
- Run Chef on the target managed node.

OPSCODE | Chef

Exercise: chef-client cookbook

```
> knife cookbook site download users

> tar -zxvf users*.tar.gz -C cookbooks

> knife cookbook upload users
```

OPSCODE | Chef

Add users to base role

```
name "base"
description "Base role applied to all nodes."
run_list(
  "recipe[chef-client::delete_validation]",
  "recipe[chef-client]",
  "recipe[motd-tail]",
  "recipe[sudo]",
  "recipe[users::sysadmins]"
)
```

OPSCODE | Chef

Create data_bags/users/yourusername.json

```
{
   "id": "yourusername",
   "groups": ["sysadmin"],
   "uid": 2001,
   "shell": "/bin/bash",
   "comment": "Your Name"
}
```

OPSCODE | Chef

Upload the data bag to the Chef Server

```
> knife data bag create users

> knife data bag from file users yourusername.json

> knife data bag show users yourusername
```

```
users_manage "sysadmin" do
   group_id 2300
   action [ :remove, :create ]
end
```

Detailed discussion about LWRP is outside scope of this workshop.

users_manage LWRP

- Search the "users" data bag for search_group ("sysadmin").
- Creates a group for the user.
- Creates the user.
- Creates user's .ssh directory and creates an authorized_keys with public SSH keys.
- Creates the specified group ("sysadmin")

Detailed discussion about LWRP is outside scope of this workshop.

Verify the system

```
  * users_manage[sysadmin] action create
Recipe: <Dynamically Defined Resource>
  * user[yourusername] action create
    - create user user[yourusername]
  * directory[/home/yourusername/.ssh] action create
    - create new directory /home/yourusername/.ssh
    - change mode from '' to '0700'
    - change owner from '' to 'yourusername'
    - change group from '' to 'yourusername'
* group[sysadmin] action create
    - create group[sysadmin]
```

Add SSH public key to data bag item

```
> ssh-keygen -t rsa -f chef-workshop
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in chef-workshop.
Your public key has been saved in chef-workshop.pub.
```

OPSCODE | Chef

Add SSH public key to data bag item

```
{
  "id": "yourusername",
  "groups": ["sysadmin"],
  "uid": 2001,
  "shell": "/bin/bash",
  "comment": "Your Name",
  "nagios": {
    "email": "you@example.com"
  },
  "ssh_keys": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxsj... SNIP"
}
```

Upload data bag and run Chef

- Upload the data bag item again using the previous command.
- Run chef-client on the target managed node.

Verify the system

```
* template[/home/yourusername/.ssh/authorized_keys] action
create
  - create template[/home/yourusername/.ssh/
authorized_keys]
      --- /tmp/chef-tempfile20130212-24676-14thg55
2013-02-12 10:05:38.032047311 +0000
      +++ /tmp/chef-rendered-template20130212-24676-tpk3ow
2013-02-12 10:05:38.032047311 +0000      @@ -0,0 +1,4 @@
      +# Generated by Chef for
ip-10-145-232-156.ec2.internal
      +# Local modifications will be overwritten.
      +
      +ssh-rsa AAAA...local
```

Verify the system

```
> ssh -i chef-workshop yourusername@IPADDRESS

yourusername@target1:~$ id
uid=2001(yourusername) gid=2001(yourusername)
groups=2001(yourusername),2300(sysadmin)

yourusername@target1:~$ sudo -l

User yourusername may run the following commands on
this host:
    (ALL) NOPASSWD: ALL
```

Additional Topics

Version Control
Ruby
Testing Recipes
Get Involved!
Chef Development

- **USE SOMETHING.**
- Distributed Version Control
- Git, GitHub, BitBucket
  - http://git-scm.com
  - https://github.com
  - https://bitbucket.org
- Workflows, CI

- Recipe DSL
- Libraries, "LWRPs" and more
- Knife plugins
- Report/exception handlers
- chef-shell

- Chef 10.14+, "why run" mode
- Test Kitchen (RubyGem)
- Vagrant
  - http://vagrantup.com
- Minitest - cookbook, handler
- Cucumber - cucumber-chef
  - http://www.cucumber-chef.org/

OPSCODE | Chef

Get Involved

- Community Site:
  - community.opscode.com
- IRC: #chef, #chef-hacking
  - irc.freenode.net
- Mailing list:
  - lists.opscode.com
- ChefConf, Community Summits, User Groups, Hack days and more

- Apache 2 Software License
- Continually growing number of contributors!
- **Development** repositories:
  - http://github.com/opscode
  - http://github.com/opscode-cookbooks

Questions?

Further Resources

OPSCODE | Chef

Further Resources

- http://opscode.com/
- http://learnchef.com
- http://community.opscode.com/
- http://docs.opscode.com
- http://wiki.opscode.com/
- http://lists.opscode.com
- http://youtube.com/user/Opscode

OPSCODE | 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

OPSCODE | Chef

Get Involved Locally

- Add some URLs for local MeetUp groups

OPSCODE-WORKSHOP - Save 20%

Exercise: Database Connection

Exercise: Database Connection

Policy statement: The database connection should be dynamically generated based on a search and encrypted credentials

New concepts:

- Creating Cookbooks
- Search
- Environments
- Encrypted Data bags

OPSCODE | Chef

Exercise: Database Connection

- Create an Environment
- Update your node's Environment
- Create an encrypted data bag item with database credentials
- Create a cookbook
- Write a file that uses
  - Search for the host
  - Encrypted Data Bag for the Credentials

database.yml

```
staging:
  host:       foo.example.com
  username:   yourusername
  password:   yourpassword
```

OPSCODE | Chef

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
  - http://wiki.opscode.com/display/chef/Community+Plugins

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

	Attribute Files	Node/ Recipe	Environment	Role
Default	1	2	3	4
Force Default	5	6		
Normal	7	8		
Override	9	10	**12**	11
Force Override	13	14		
Automatic	15			