Chef

Topics

- Overview of Chef
- Workstation Setup
- Organization Setup
- Test Node Setup
- Writing your First Cookbook
- Dissecting your first Chef Run
- Introducing the Node Object
- Attributes, Templates, Cookbook Dependencies

Topics

- Template Variables, Notifications, Controlling Idempotency
- Data bags, search
- Roles
- Environments
- Using Community Cookbooks
- Further Resources

Chef is Infrastructure As Code

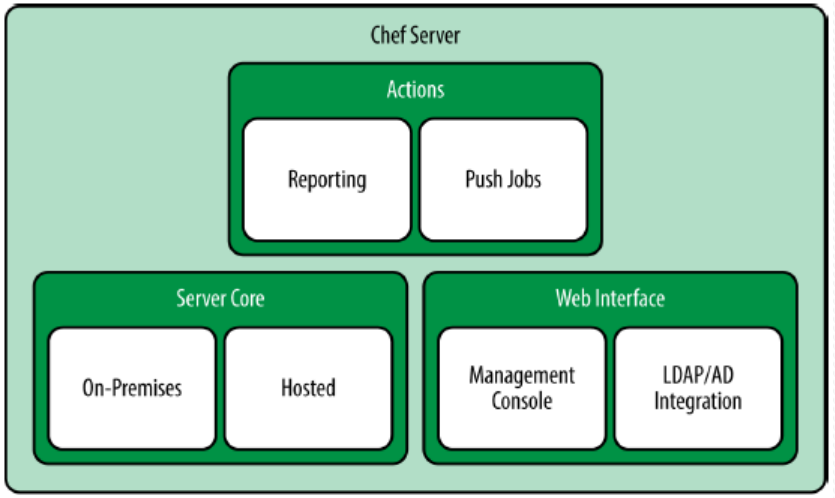
- Programmatically provision and configure software components
- Treat like any other code base
- Reconstruct business from code repository, data backup and compute resources

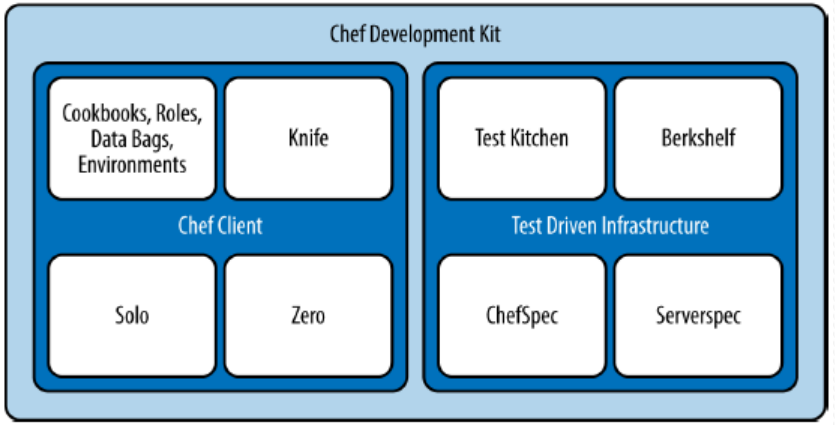
Configuration Code

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

Overview of Chef

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





Server Resources

- Networking
- Files
- Directories
- Symlinks
- Mounts
- Registry Keys
- Powershell Scripts
- Users
- Groups
- Packages
- Services
- File Systems

Declarative Interface to Resources

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

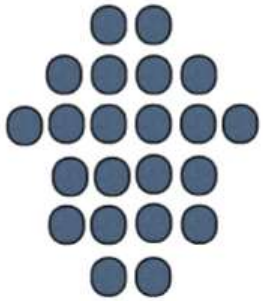
Managing Complexity

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

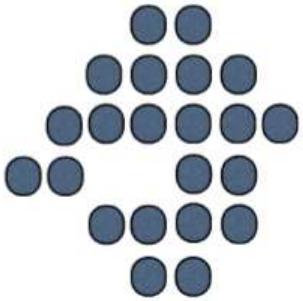
Organizations

My Infrastructure

Your Infrastructure



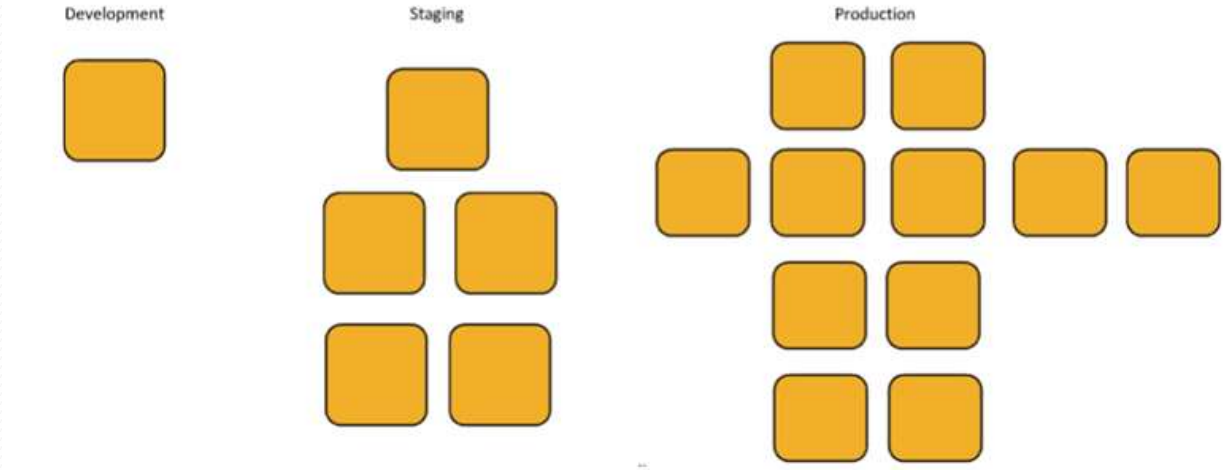
Their Infrastructure



Organizations

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

Environments



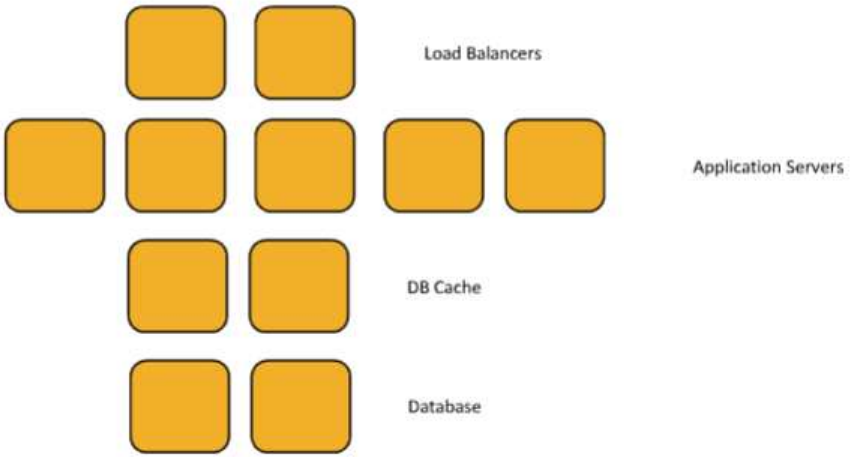
Environments

- Environment can represent your patterns, workflows, and can be used to model the life-stages of your applications
  - Development
  - Test
  - Staging
  - Production
  - Etc.
- Every organizations starts with a single environment

Environments Define Policy

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

Roles



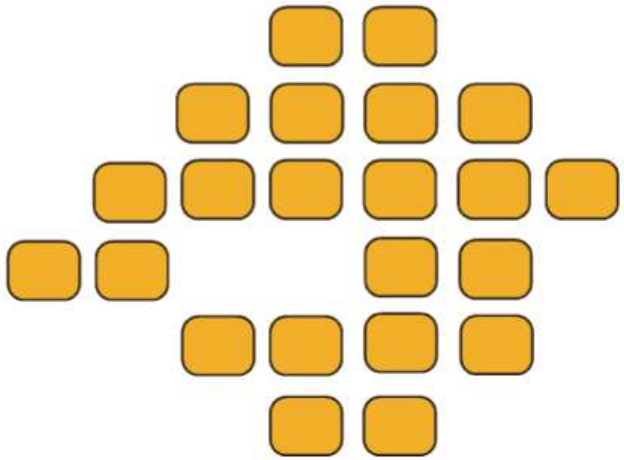
Roles

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

Roles Define Policy

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

Nodes



Nodes

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

Node

- Each Node will belong to one Organization
- Each Node will belong to one Environment
- Each Node will have zero or more roles

Nodes Adhere to Policy

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

Resources

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

Resource in Recipes

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

Recipes

- Configuration files that describe the resource and their desired state
- Recipes can
  - Install and configure software components
  - Mange files
  - Deploy applications
  - Execute other recipes
  - And more

Example Recipe

```
package "apache2"
```

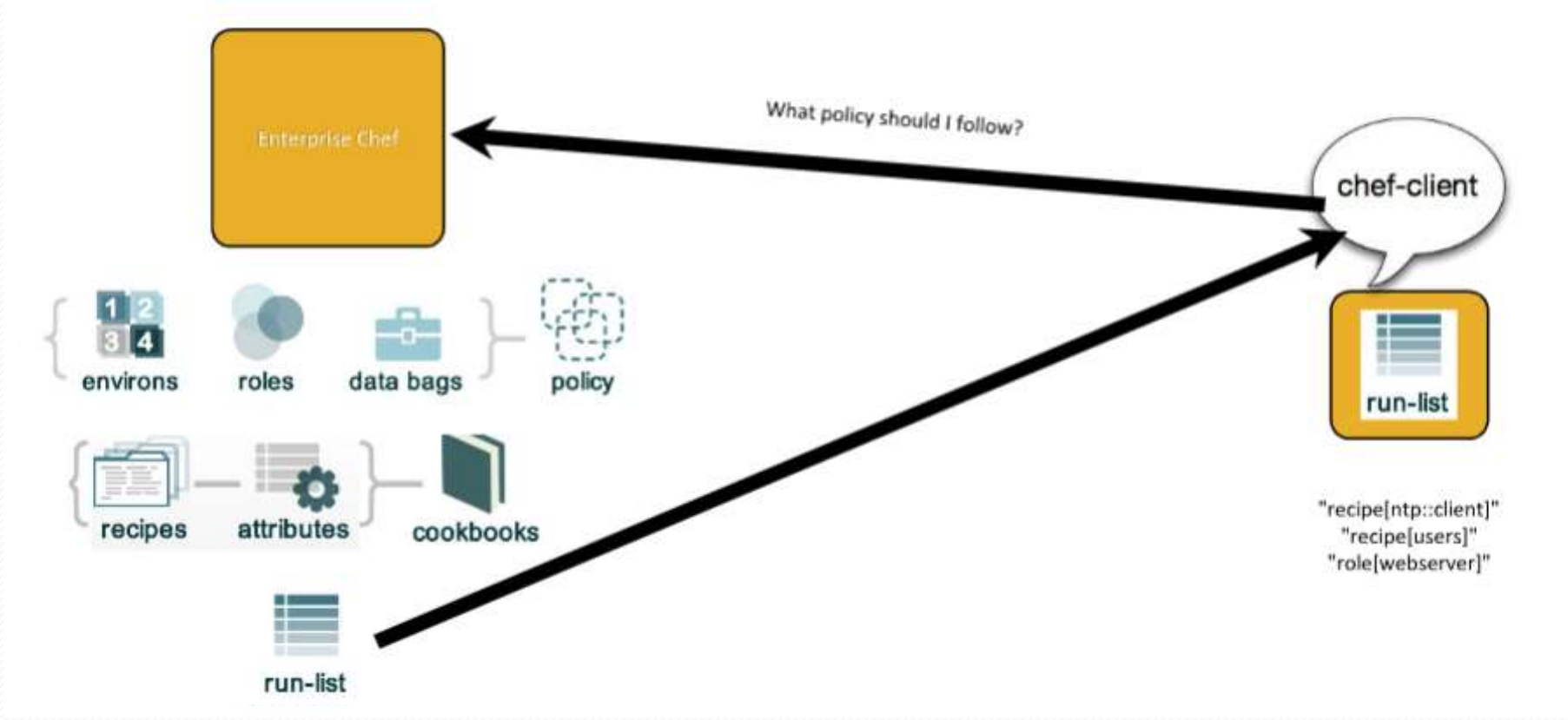
```
template "/etc/apache2/apache2.conf" do
  source "apache2.conf.erb"
  owner "root"
  group "root"
  mode "0644"
  variables(:allow_override => "All")
  notifies :reload, "service[apache2]"
end
```

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

Cookbooks

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

Run List



Run List Specifies Policy

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

Search

- Search for Nodes with Roles
- Find Topology Data
- IP Addresses
- Hostnames
- FQDNs

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
```

Manage Complexity

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

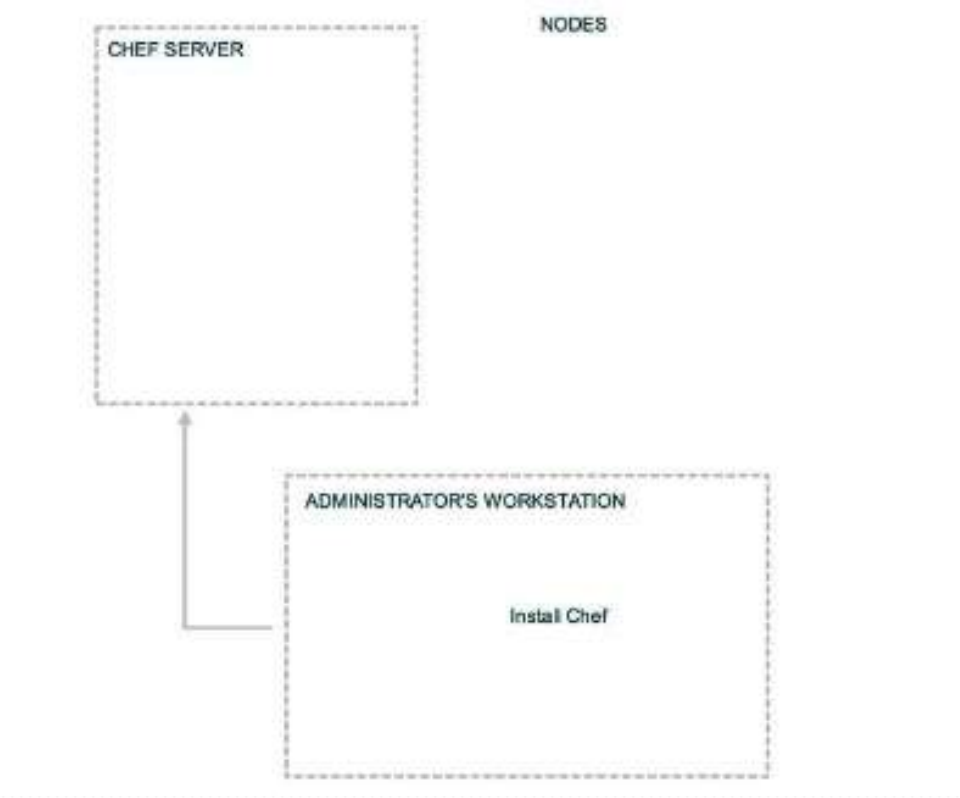
Configuration Drift

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

Workstation Setup

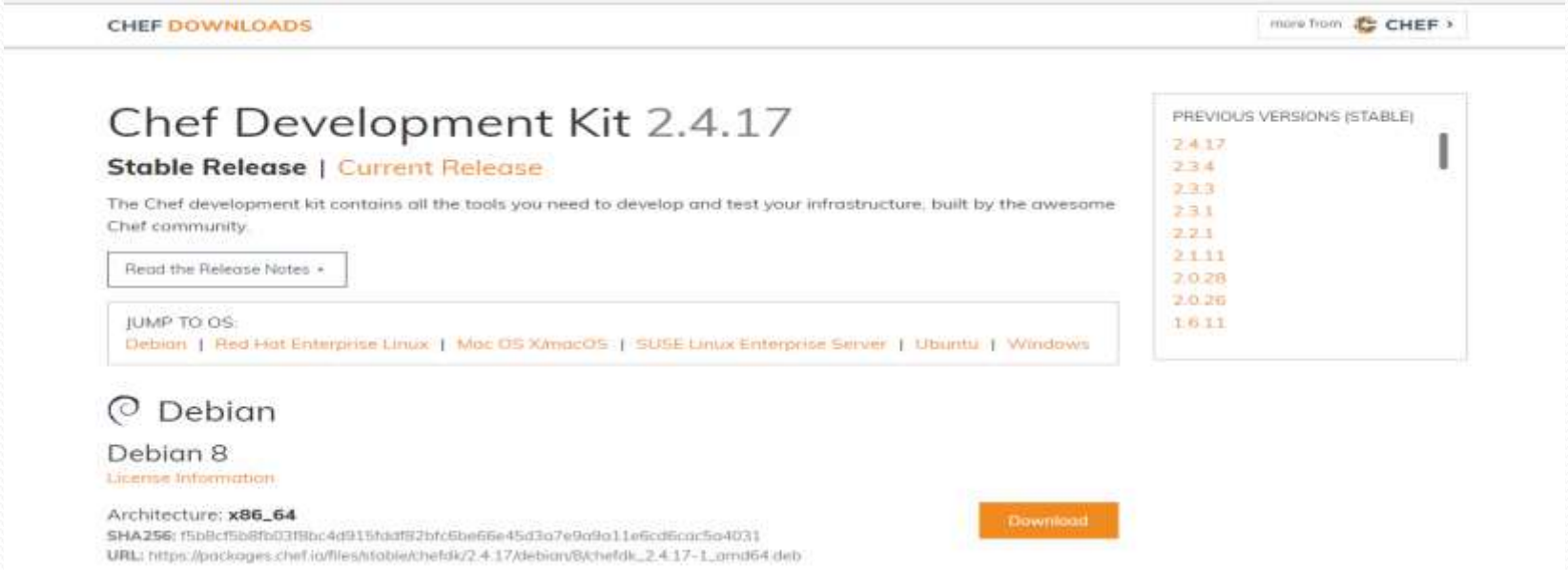
- Login to Enterprise Chef
- View your Organization in Enterprise Chef
- Describe Knife, the Chef command line utility
- Use Knife on your workstation

Chef Setup



Install ChefDK

- https://downloads.chef.io/chef-dk/

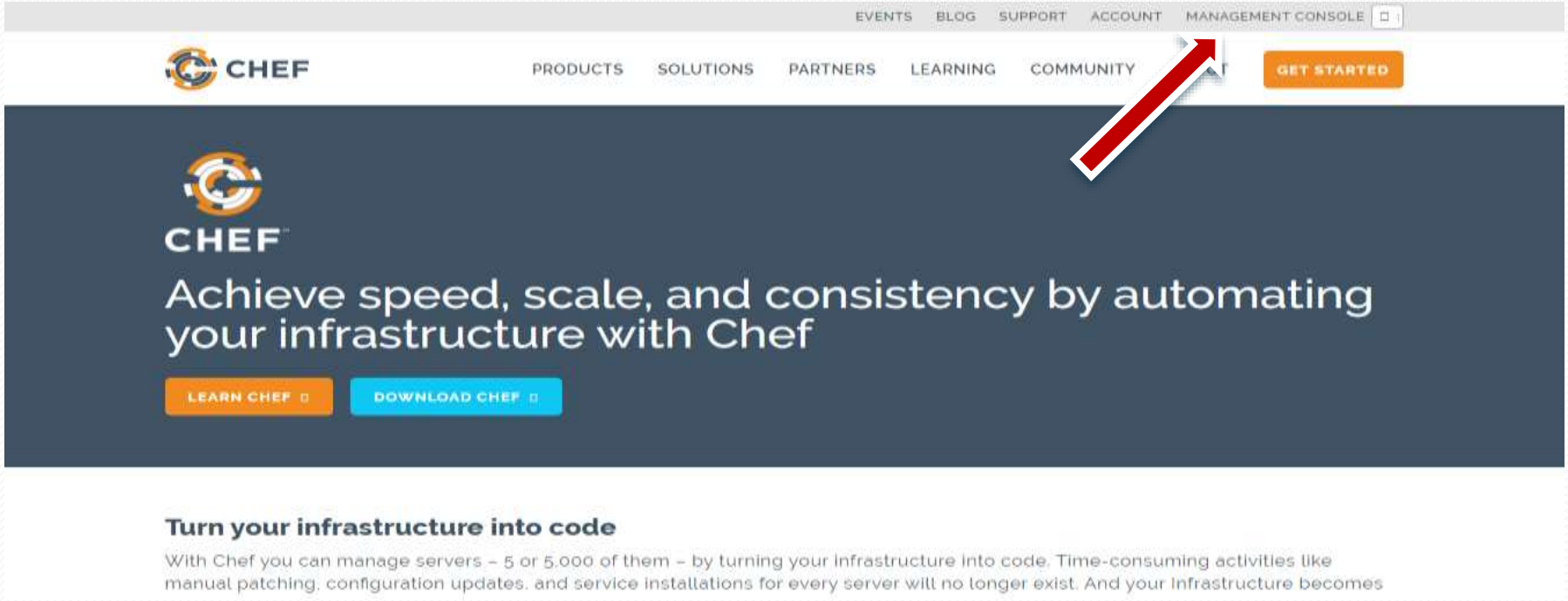


Installation

- Chef and all its dependencies are installed via an operating system specific package (omnibus installer)
- Installation includes
  - The Ruby language – used by Chef
  - Knife – Command line tool for administrators
  - Chef-client – Client application
  - Ohai – System profiler
  - … and more

Your Chef Server for This Class

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



Create a New Account

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

Download “Starter Kit” for Your Organization

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

What is the Starter Kit?

- The 'Starter Kit' is an archive file (e.g. chef-starter.zip) that contains
  - A sample Chef repository with a sample 'starter' cookbook
  - Configuration files allowing the workstation to talk to the Chef server using knife
  - We'll look at this in detail later…

Knife is the command-line tool for Chef

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

A quick tour of Chef-repo

- Every infrastructure managed by Chef has a Chef Repository (chef-repo)
- Type all commands in this class from the chef-repo directory
- Lets see what's in the chef-repo…

Tour the chef-repo

```
$ cd chef-repo
```

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

Tour the chef-repo

```
$ ls -al
```

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

What's inside the .chef directory?

EduRamp
The B2B Learning Specialist

```
$ ls .chef
```

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

What's inside the .chef directory?

- `knife.rb` is the configuration file for Knife
- The other two files are certificates for authentication with Chef Server

knife.rb

- Default location
  - `~/.chef/knife.rb`
  - `c:\users\You\.chef\` (Windows)
- Use a project specific configuration
  - `.chef/knife.rb` of the current directory
  - `chef-repo/.chef/knife.rb`

knife.rb

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

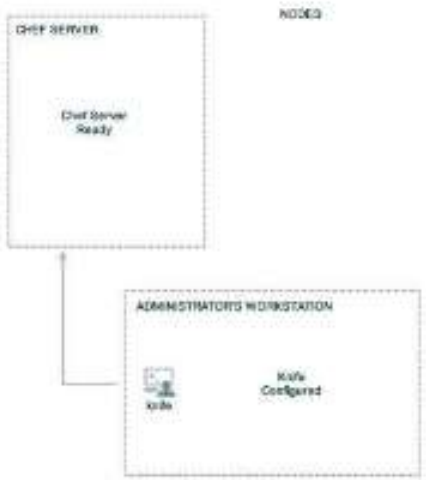
```
current_dir = File.dirname(__FILE__)
log_level                :info
log_location             STDOUT
node_name                "USERNAME"
client_key               "#{current_dir}/USERNAME.pem"
validation_client_name   "ORGNAME-validator"
validation_key           "#{current_dir}/ORGNAME-validator.pem"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

Verify Knife

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

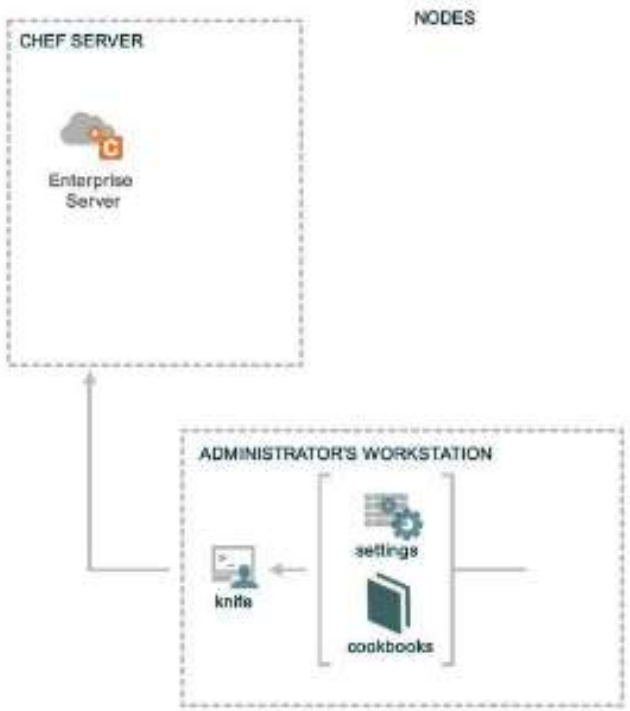
```
$ knife client list
ORGNAME-validator
```

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



knife Client List

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



Exercise: Run ‘knife help list’

```
$ knife help list
```

```
Available help topics are:
  bootstrap
  chef-shell
  client
  configure
  cookbook
  cookbook-site
  data-bag
  delete
  deps
  diff
  download
  edit
  environment
  exec
  list
```

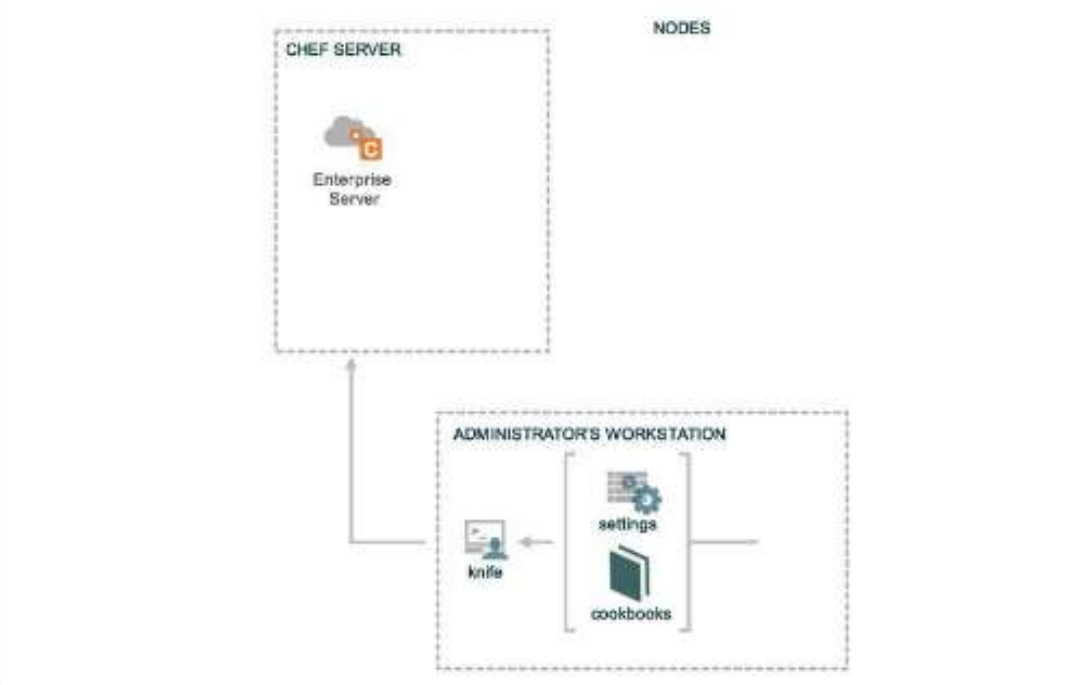
A few knife tips

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

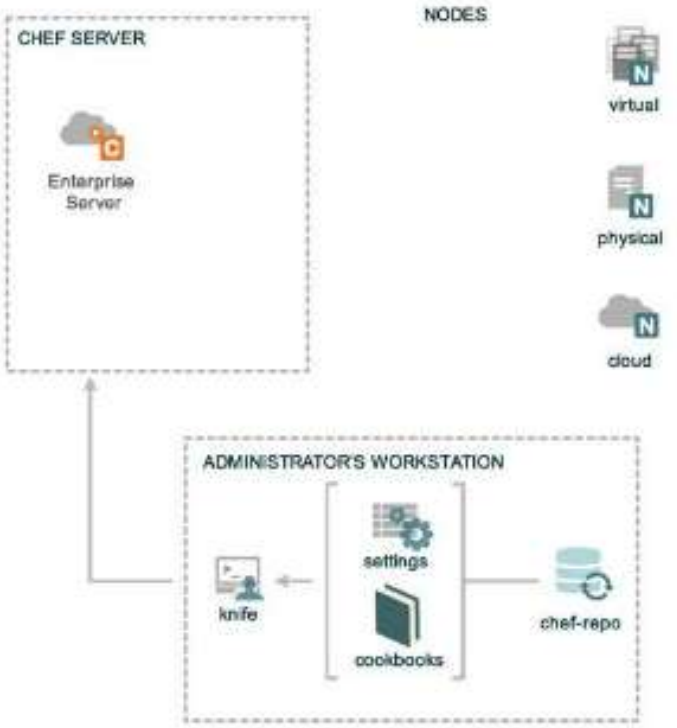
Best Practice: Use a test editor with project drawer

- 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++
- Download Sublime Text (www.sublimetest.com)

Chef Setup



Nodes



Nodes

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

Chef Manage

Bring Your On Node

- Use your Virtual Machine (VM) or Server
- The IP Address or public hostname
- An application for establishing an ssh connection
- ‘sudo’ or ‘root’ permissions on VM

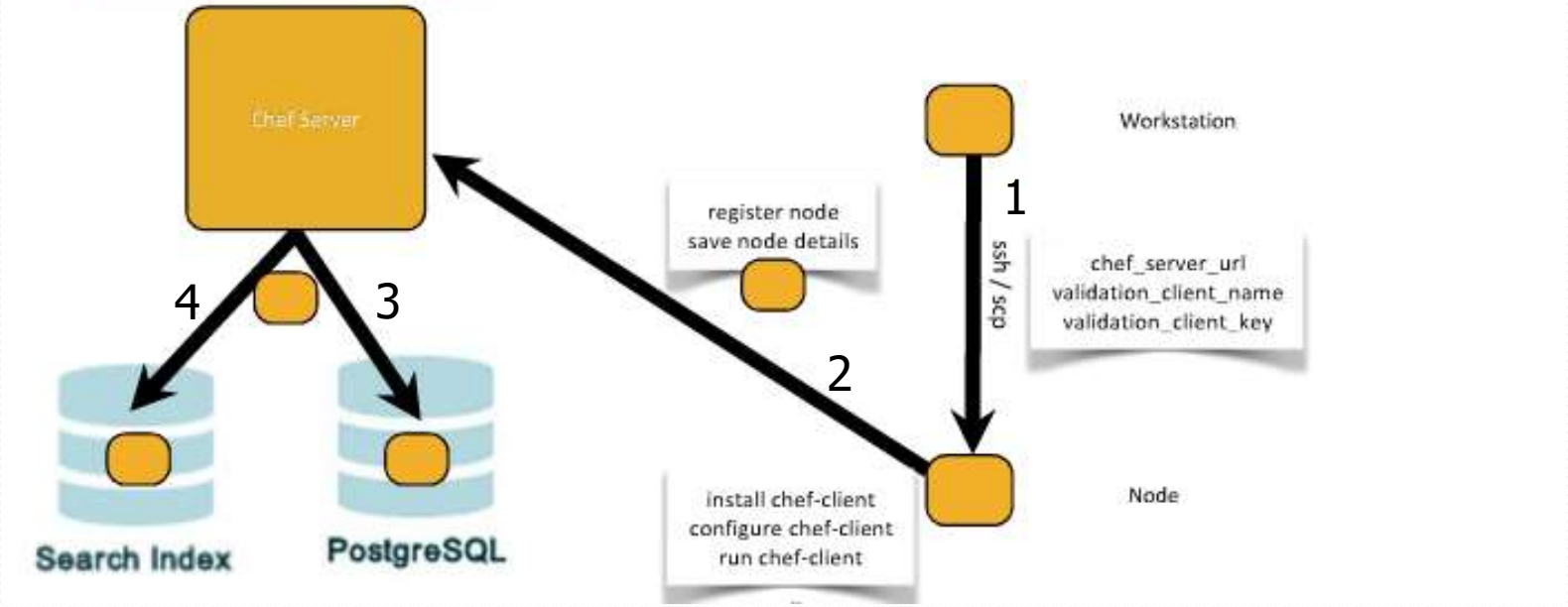
“Bootstrap” the Target instance

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

```
Bootstrapping Chef on uvo1qrwls0jdgs3blvt.vm.cld.sr
uvo1qrwls0jdgs3blvt.vm.cld.sr knife sudo password:
Enter your password:
...
...
uvo1qrwls0jdgs3blvt.vm.cld.sr Creating a new client identity for node1 using the
validator key.
uvo1qrwls0jdgs3blvt.vm.cld.sr resolving cookbooks for run list: []
uvo1qrwls0jdgs3blvt.vm.cld.sr Synchronizing Cookbooks:
uvo1qrwls0jdgs3blvt.vm.cld.sr Compiling Cookbooks...
uvo1qrwls0jdgs3blvt.vm.cld.sr [2014-01-28T11:03:14-05:00] WARN: Node node2 has an
empty run list.
uvo1qrwls0jdgs3blvt.vm.cld.sr Converging 0 resources
uvo1qrwls0jdgs3blvt.vm.cld.sr Chef Client finished, 0 resources updated
```

knife Bootstrap

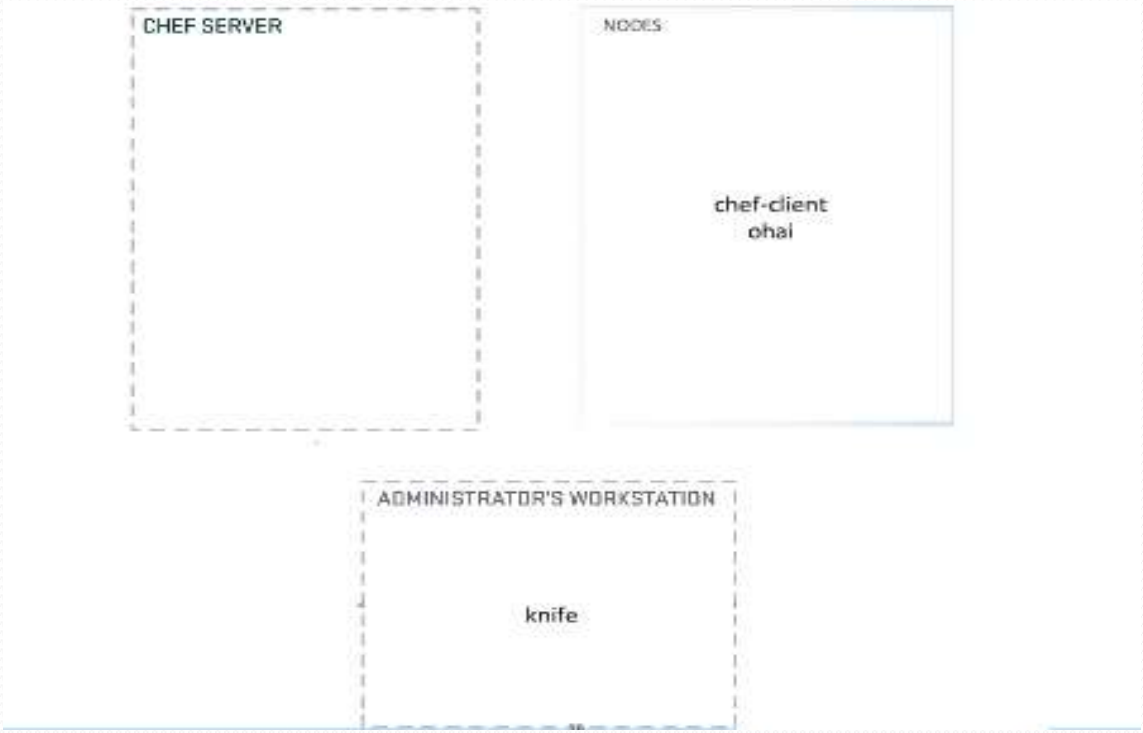
```
knife bootstrap HOSTNAME --sudo -x chef -P
chef -N "node1"
```



Installation

- Chef and all its dependencies are installed via an operating system specific package (omnibus installer)
- Installation includes
  - The Ruby language – used by Chef
  - Knife – Command line tool for administrators
  - Chef-client – Client application
  - Ohai – System profiler
  - … and more

Chef Setup



Verifying Your Target Instance's Chef-Client Configuration


```
$ ssh chef@<EXTERNAL_ADDRESS>

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

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

Examine /etc/chef/client.rb

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

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

Exercise: Change the log level on your test node


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

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

- Set the default log level for chef-client to `:info`
- More configuration options can be found on the docs site: http://docs.getchef.com/config_rb_client.html

View Node on Chef Server

- Login to your Hosted Enterprise Chef

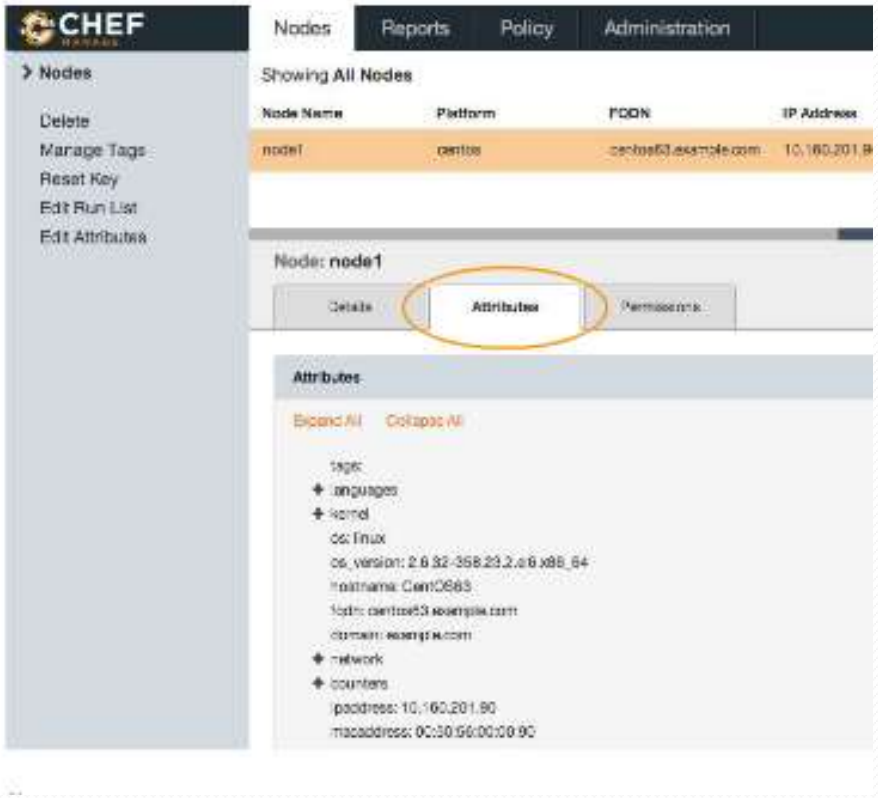
View Node on Chef Server

- Click the 'Details' tab

EduRamp
The B2B Learning Specialist

Vide Node on Chef Server

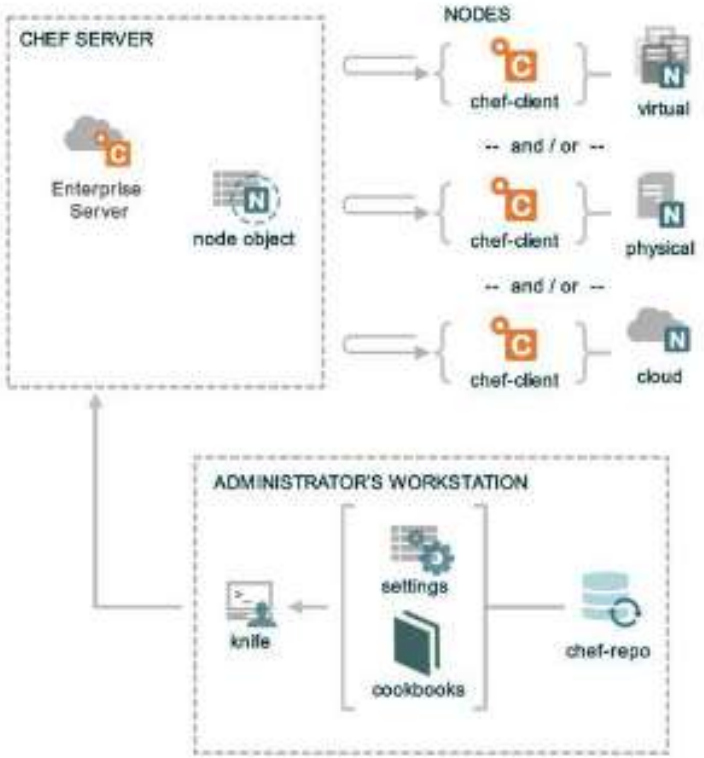
- Click the 'Attributes' tab



Node

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

Chef Setup



Chef Resources & Recipes

- Writing an Apache Cookbook
- Describe in detail a Cookbook
- Create a new Cookbook
- Explain a Recipe
- Describe how to use package, service and cookbook_file resources
- Upload a Cookbook to Chef Server
- Explain what a Run List is and how to set it for a Node via knife
- Explain output of a chef-client run

What's 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

Problem Statement

- We need a web server configured to server up our home page
- Success of this will be determined by seeing a home page in a web browser

Steps to Solution

- Install Apache
- Start the service
- Make sure it will start will server boots up
- Write home page code

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
```

Explore the Cookbook

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

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

Edit the Default Recipe

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

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

Chef Resources

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

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

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

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

Add Package Resource to Install Apache to Default Recipe

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

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

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

SAVE FILE!

Resource We Just Wrote

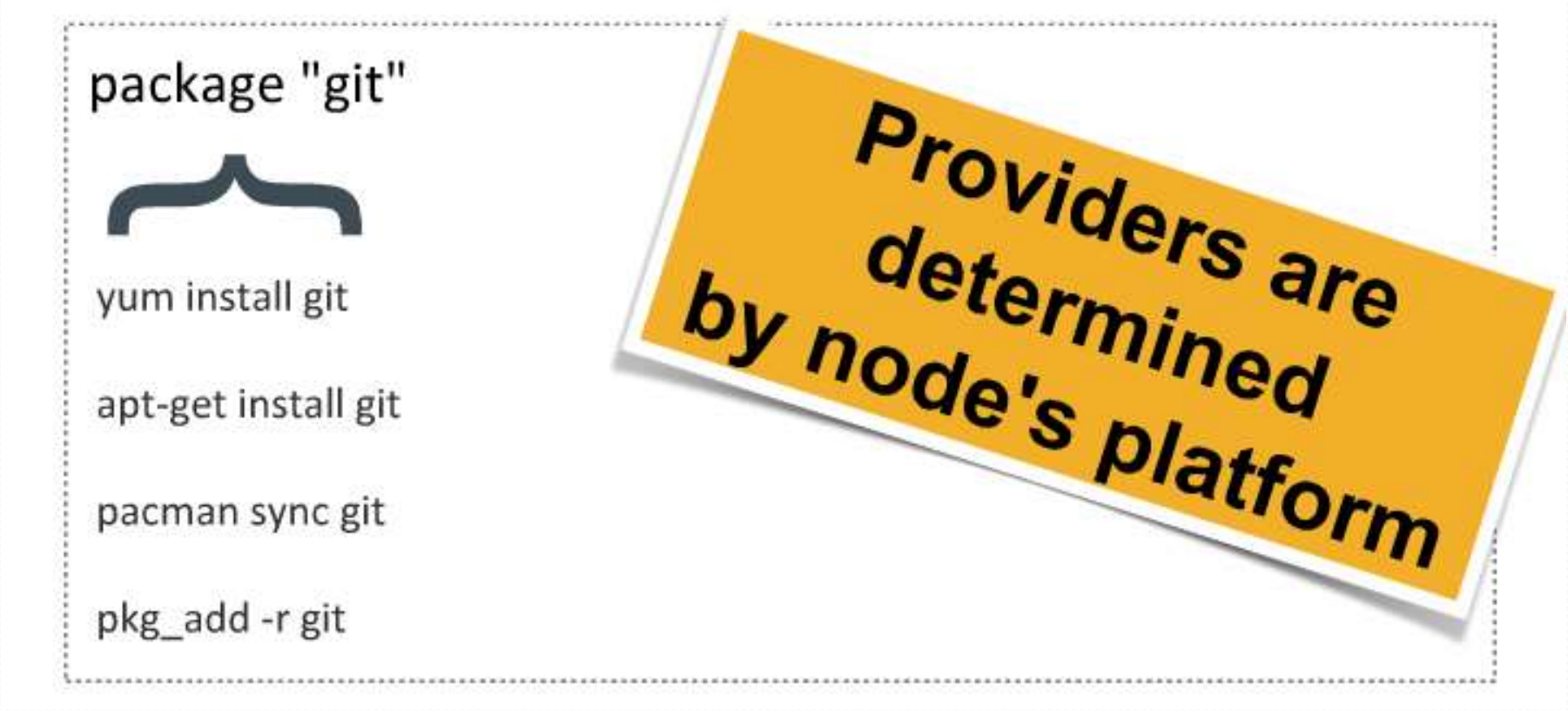
- Is a package resource
- Whose name is *httpd*
- With an install **action**

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

Notice We Did Not Say How to Install Package

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

Package Resource



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

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

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

package "httpd" do
  action :install
end

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

SAVE FILE!

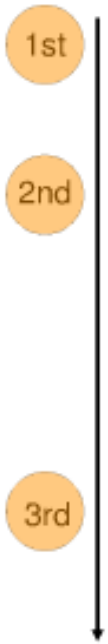
Resource We Just Wrote

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

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

Order Matters

- Resources are executed in order




```
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
```

Add cookbook_file resource to copy the home page

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

```
...

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

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

SAVE FILE!

Resource We Just Wrote

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

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

Full Content of Apache Recipe

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

package "httpd" do
  action :install
end

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

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

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

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

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

SAVE FILE!

Upload the Cookbook

```
$ knife cookbook upload apache
```

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

The Run List

- The Run List is the ordered set of recipes and roles that the Chef-client will execute on a node
  - Recipes are specified by “recipe[name]”
  - Roles are specified by “role[name]”

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


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

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

Exercise: Run Chef-client

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

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

Exercise: Verify the Home Page Works

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

You have just written your first Chef cookbook

Exercise: Reading the Output of a Chef-client Run


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

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

Exercise: Reading the Output of a Chef-client Run


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

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

Exercise: Reading the Output of a Chef-client Run


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

    - install version 2.2.15-29.el6.centos of package httpd
```

- Checks to see if the package httpd is installed

Exercise: Reading the Output of a Chef-client Run


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

    - enable service service[httpd]

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

    - start service service[httpd]
```

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

Idempotence

- Action on resources in Chef are designed to be **idempotent**
  - i.e they can be applied multiple times but the end result is still the same – like multiplying 1 by 1
- Chef is a "desired state configuration" system – if a resource is already configured, no action is taken
- This is called **convergence**

Exercise: Reading the Output of a Chef-client Run


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

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

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

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

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

Exercise: Reading the Output of a Chef-client Run


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

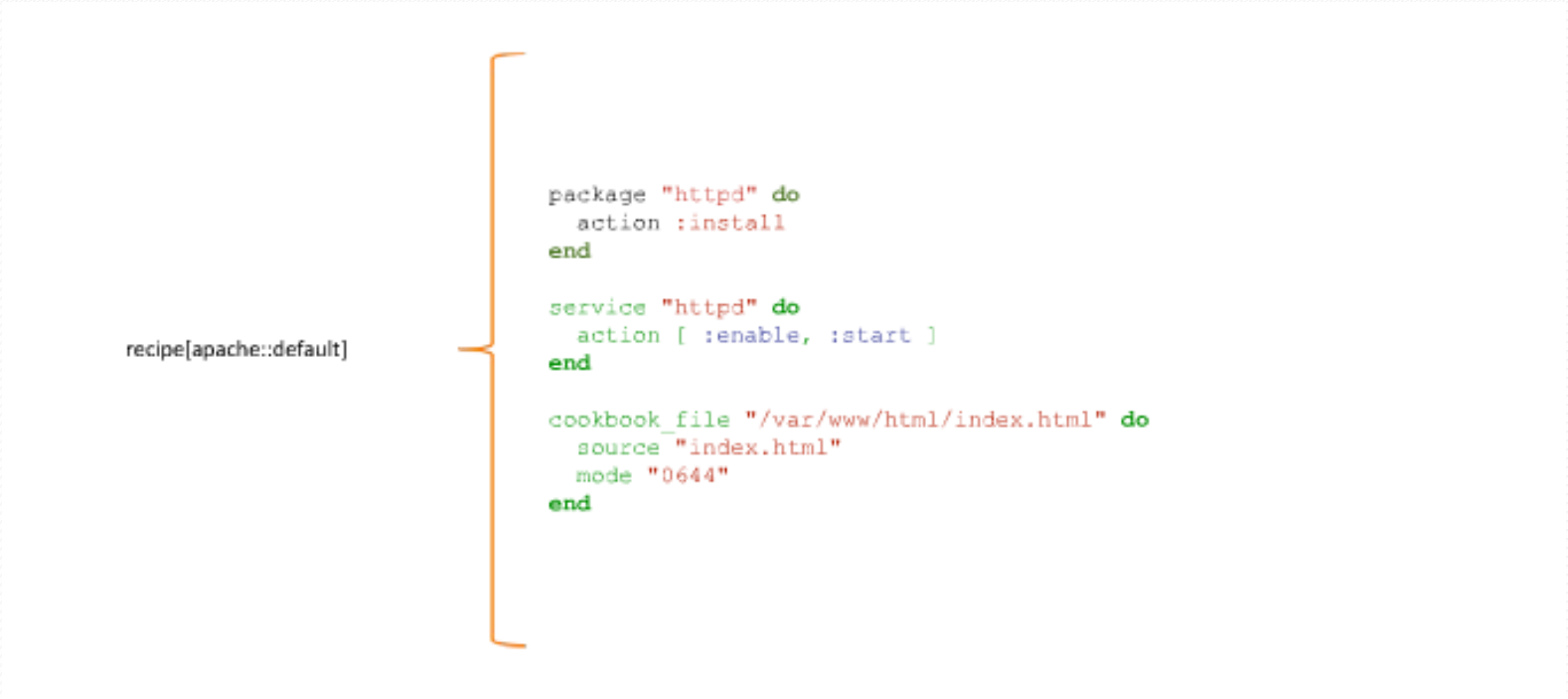
- Notice that a complete Chef-Run displays:
  - The time the client took to complete convergence
  - Status of report and exception handlers

Exercise: Re-run Chef-client

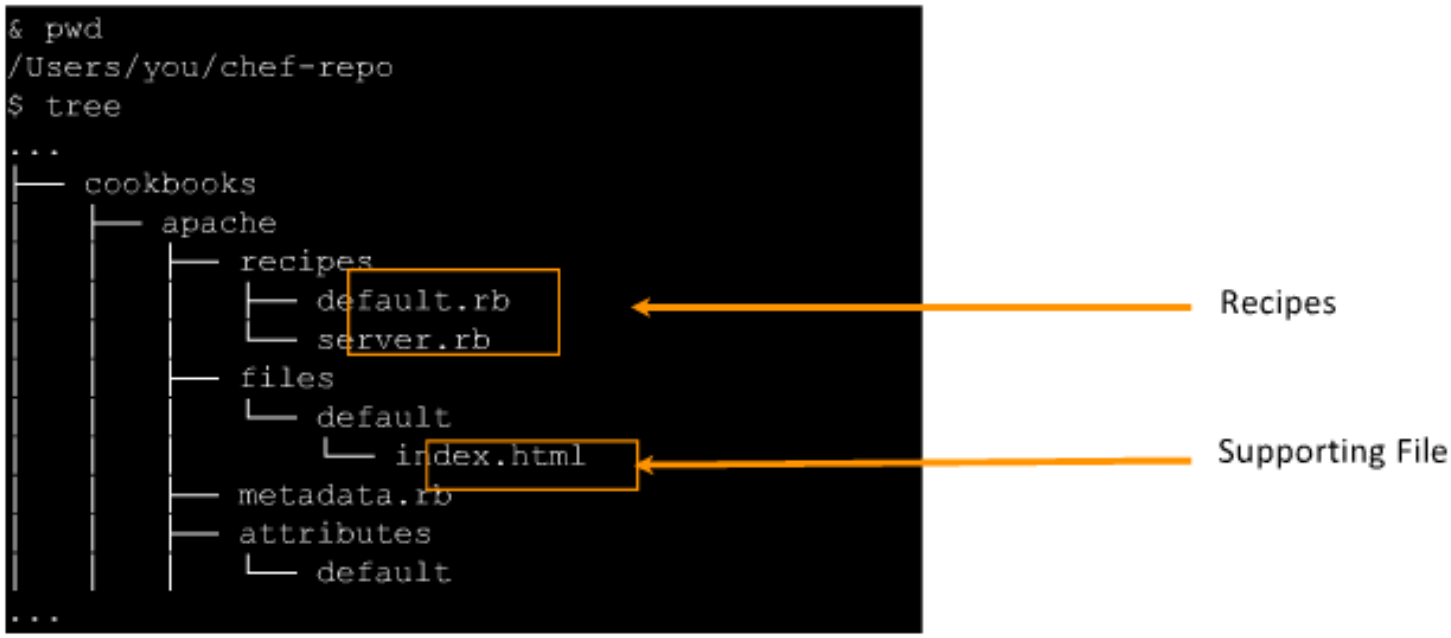
```
chef@node1:~$ sudo chef-client
```

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

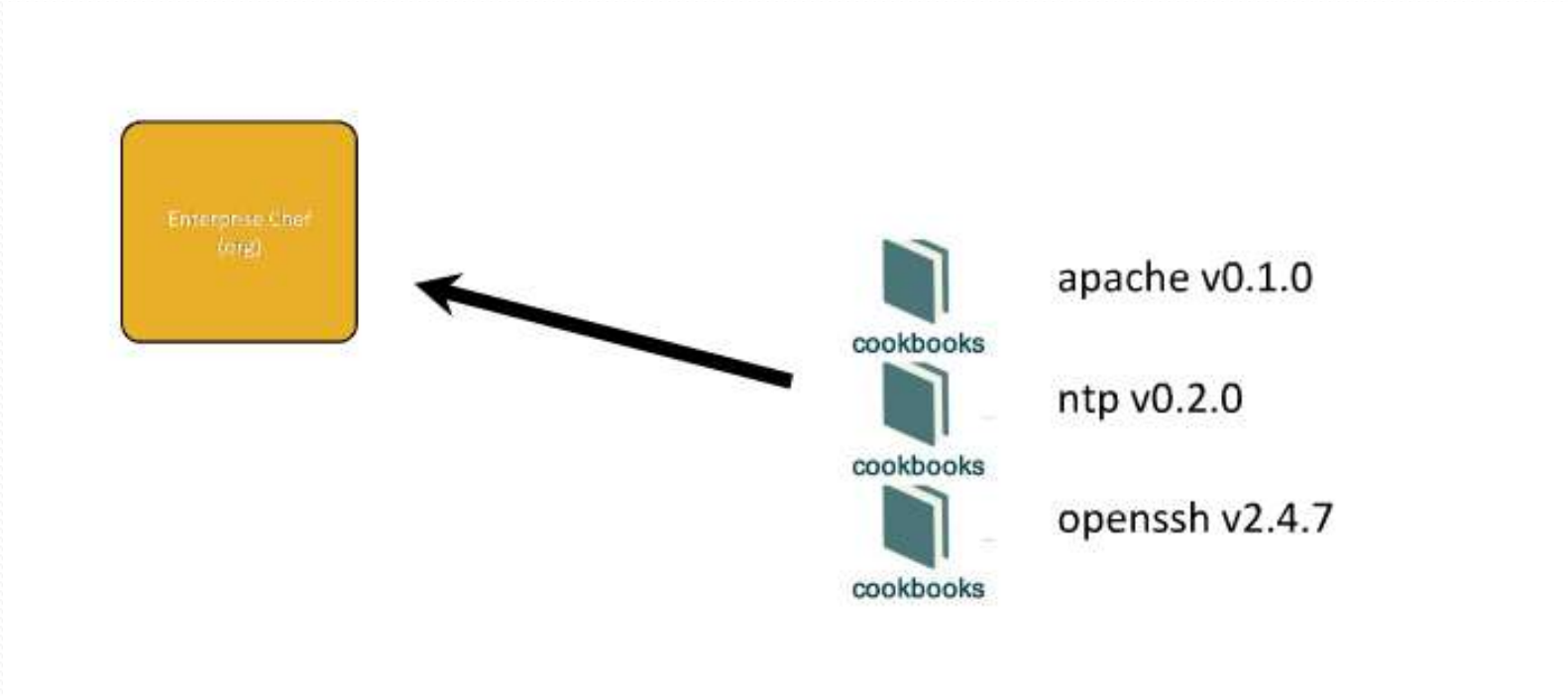
Recap – Resources are grouped into Recipes



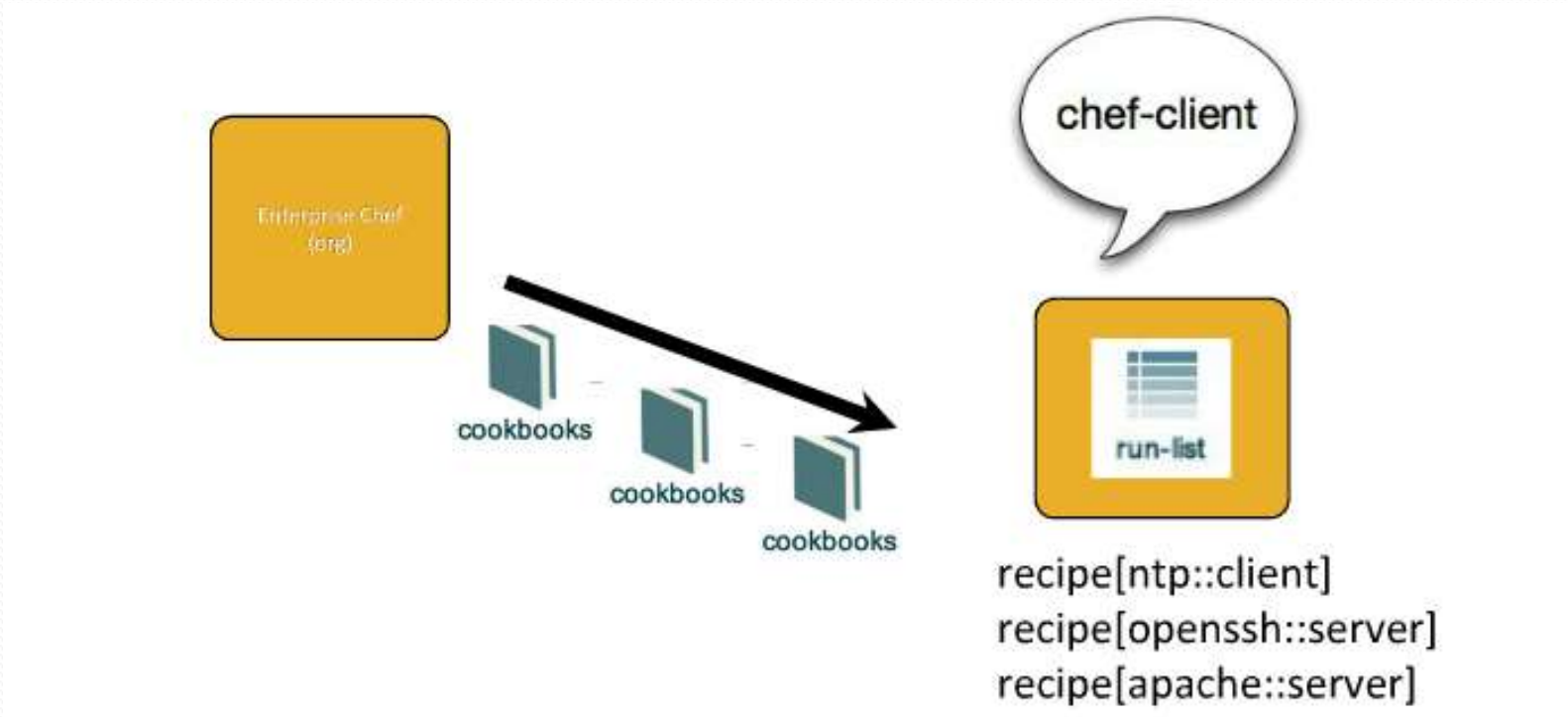
Cookbooks contain Recipes & Supporting Files



Cookbooks are Installed as Artifacts' on Chef Server

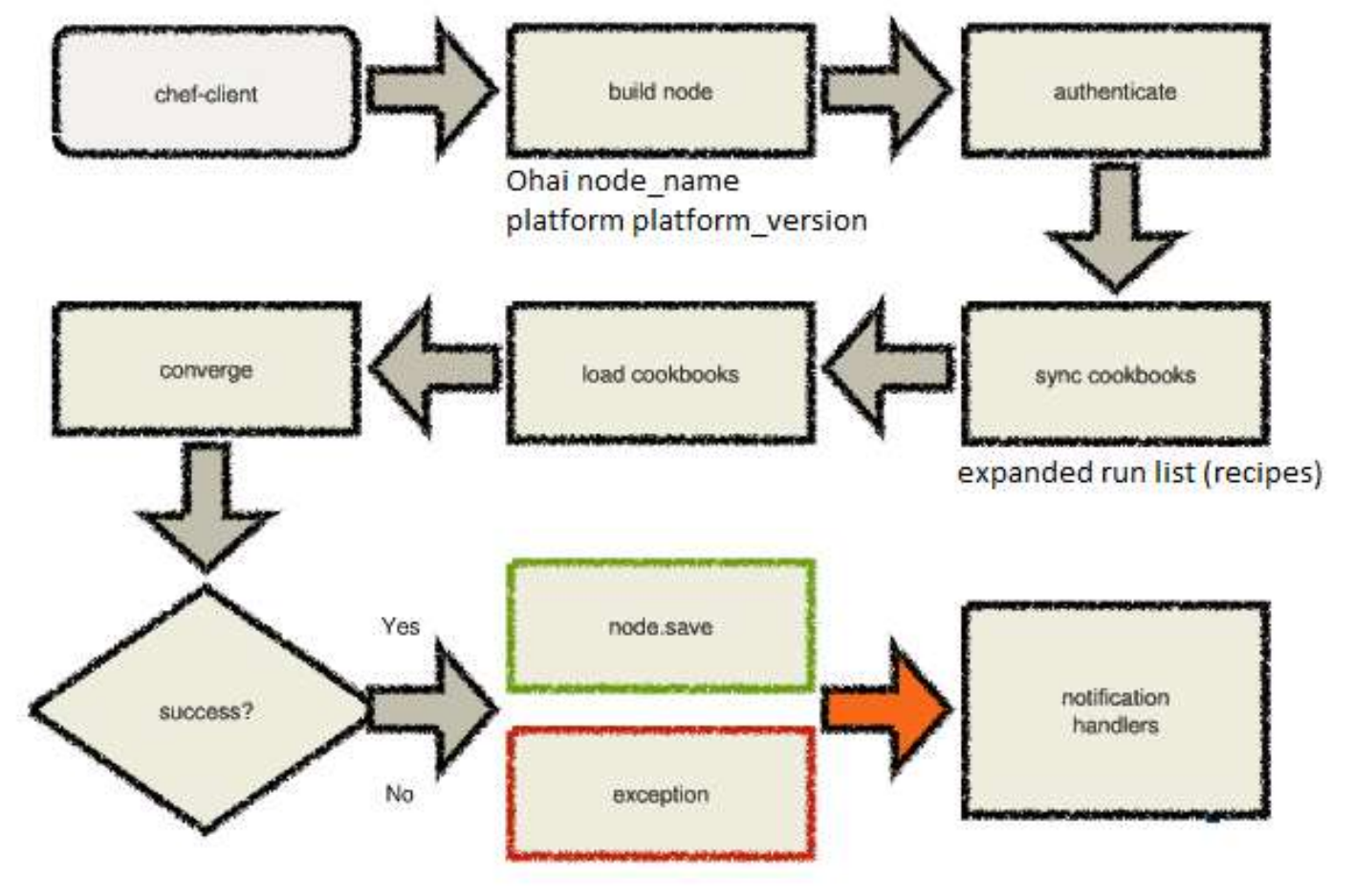


Nodes have run_lists made of Recipes



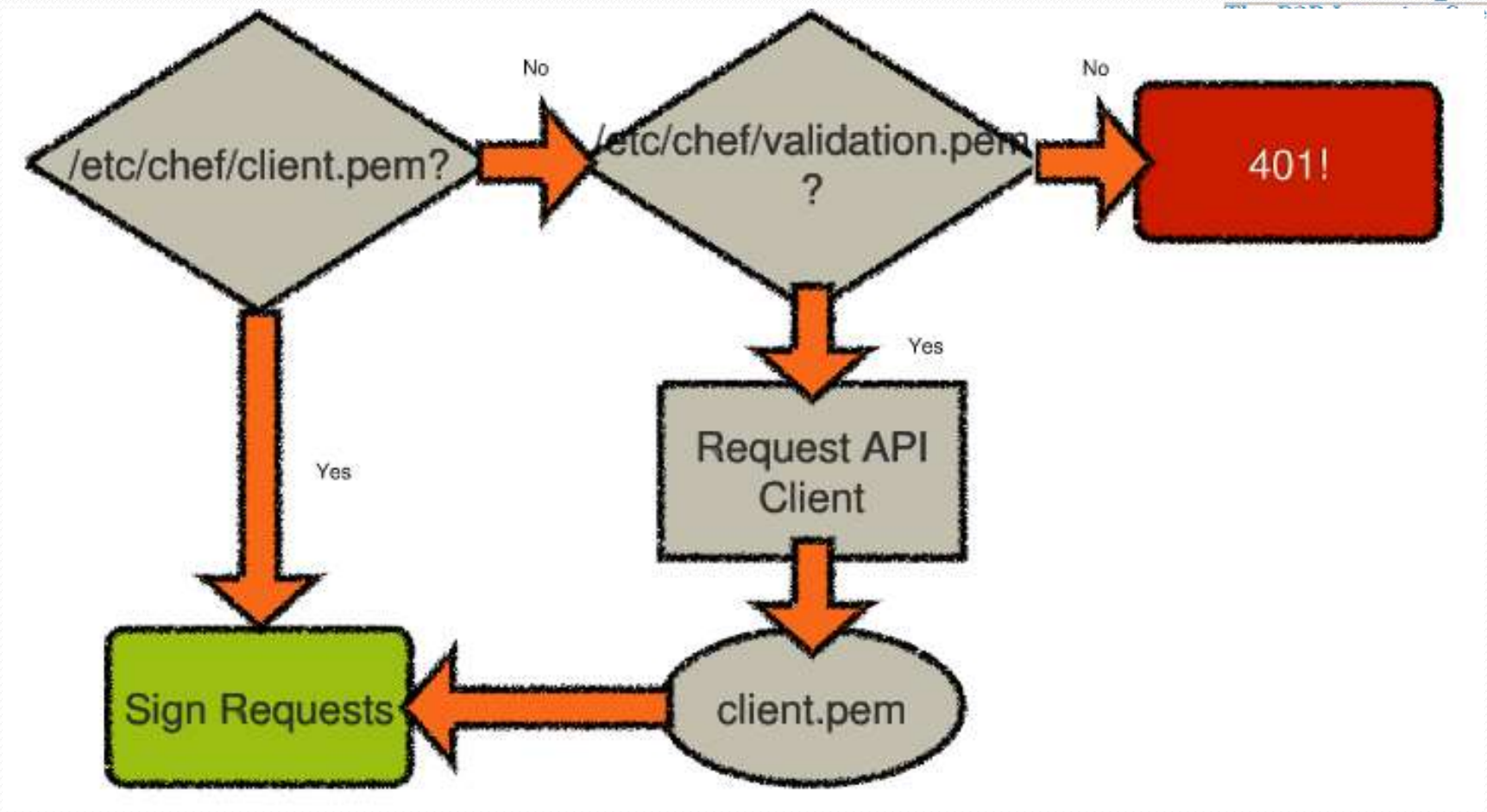
Dissecting your First Chef-client Run

- List all steps taken by Chef-client during a run
- Explain the basic security model of Chef
- Explain the concepts of Resource Collection



Private Keys

- Chef Server requires keys to authenticate
  - client.pem – private key for API client
  - validation.pem – private key for ORGNAME-validator


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

EduRamp ELS
The B2B Learning Specialist

Resource Collection

Multiphase Execution – Compile Phase

- During compile phase, Chef
  - Loads all Cookbooks from Run List
  - Reads all Recipes to build resource collection

Multiphase Execution – Execute Phase

- During compile phase, Chef takes the resource collection and for each resource it will
  - Check if the resource is in the required state
    - If ‘yes’ – do nothing
    - If ‘no’ – bring resource in line with the required state
  - Move on to next resource

Resource Collection Phase
Compile Phase

Recipe

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

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

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

Resource Collection

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

Resource Collection Phase
Execute Phase

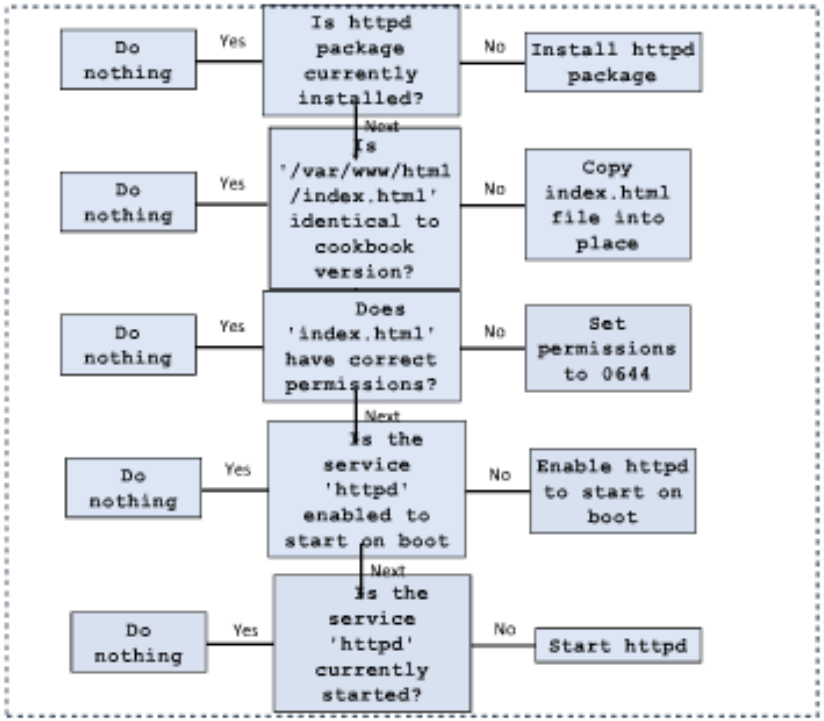
Recipe

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

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

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

Resource Collection

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

Execution



Recipe Order is Important

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

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

- These recipes are invoked in the following order

1. recipe[ntp::client]
2. recipe[openssh::server]
3. recipe[apache::server]

Resource Collection – Multiple Recipes

3. recipe[httpd::server]

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


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



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

Resource Collection

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

Final Resource Collection

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

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

Node Object

- Explain what Node object represents in Chef
- List the Nodes in an Organization
- Show details about a Node
- Describe what Node attributes are
- Retrieve a Node attribute directly, and via search

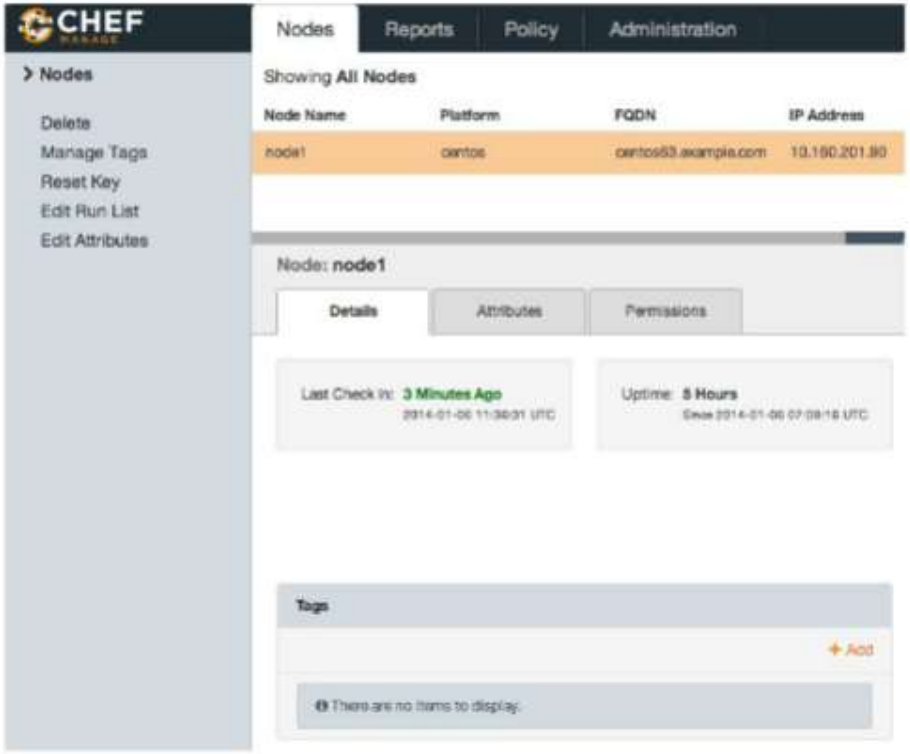
What is the Node Object

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

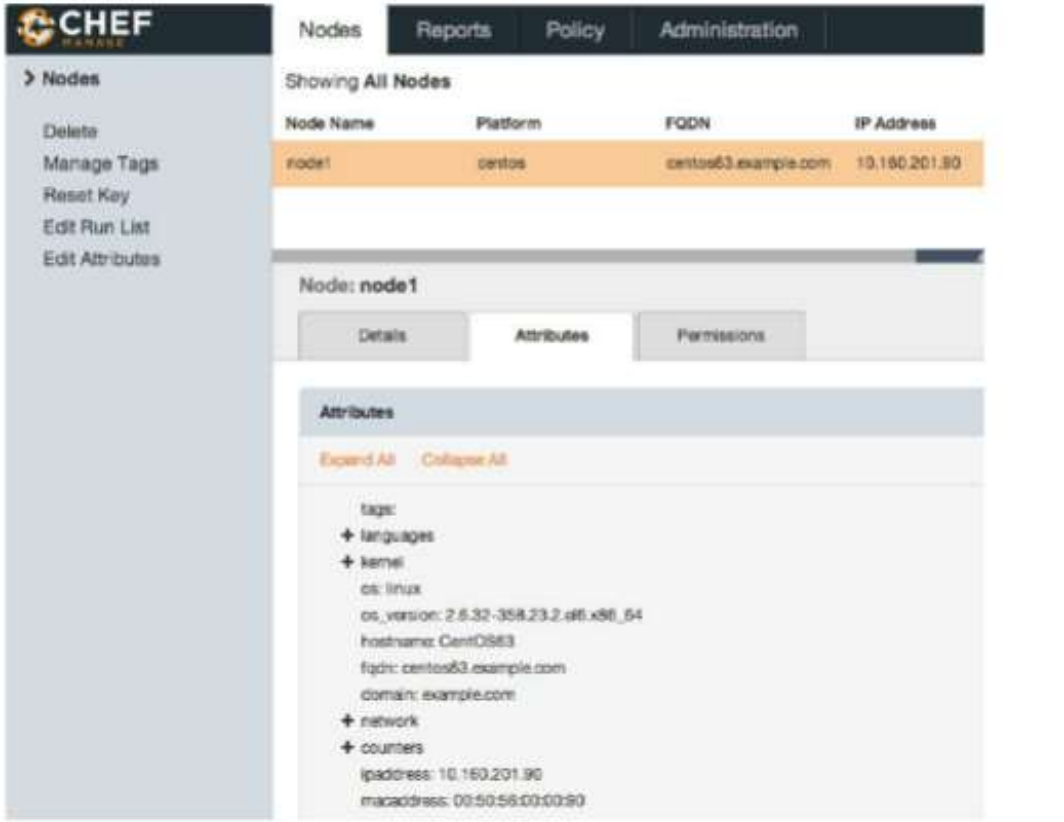
Node

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

View Node on Chef Server



View Node on Chef Server



Exercise: List Nodes

```
$ knife node list
```

```
node1
```

Exercise: List Clients

```
$ knife client list
```

```
ORGNAME-validator
node1
```

Each Node Must Have a Unique Name

- Every node must have a unique name within an Organization
- Chef defaults to the Fully Qualified Domain Name of the server i.e. in the format

```
server.domain.com
```

- We overrode it to “node1” to make typing easier

Exercise: Show Node Details

```
$ knife node show node1
```

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

What is the Node Object

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

Ohai


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

  },
  "perl": {
    "version": "5.14.2",
    "archname": "x86_64-
linux-gnu-thread-multi"
  },
  "python": {
    "version": "2.6.6",
    "builddate": "Jul 10
2013, 22:48:45"
  },
  "perl": {
    "version": "version",
    "archname": "x86_64-
linux-thread-multi"
  },
    "lua": {
      "version": "5.1.4"
},
```

```
"kernel": {
  "name": "Linux",
  "release": "3.2.0-32-virtual",
  "version": "#1 SMP Wed Oct 16
18:37:12 UTC 2013",
  "machine": "x86_64",
  "modules": {
    "isofs": {
      "size": "70066",
      "refcount": "2"
    },
    "des_generic": {
      "size": "16604",
      "refcount": "0"
    }
  },
  "os": "GNU/Linux"
},
"os": "linux",
"os_version": "2.6.32-
358.23.2.el6.x86_64",
"ohai_time": 1389105685.7735305,
```

```
"network": {
  "interfaces": {
    "lo": {
      "mtu": "16436",
      "flags": [
        "LOOPBACK", "UP","LOWER_UP"
      ],
      "encapsulation": "Loopback",
      "addresses": {
        "127.0.0.1": {
          "family": "inet",
          "netmask": "255.0.0.0",
          "scope": "Node"
        },
        "::1": {
          "family": "inet6",
          "scope": "Node"
        }
      },
    },
    "eth0": {
      "type": "eth",
      "number": "0",
```

EduRamp ELS
The B2B Learning Specialist

Exercise: Run Ohai on Node

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

Exercise: Show all Node Attributes

EduRamp
The B2B Learning Specialist

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

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

Default Attributes:

Override Attributes:

Automatic Attributes (Chal Data):
block_device:
  dm-0:
    removable: 0
    size:      26393472
```

Exercise: Show Raw Node Objects


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

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

Exercise: Show only FQDN Attributes


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

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

Exercise: Use Search to Find the same Data


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

```
1 items found

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

Setting Node Attributes

- Describe where & how to set attributes
- Explain the attribute merge order and precedence rules
- Declare an attribute with a Recipe and set its value

What are Attributes?

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

Attributes Sources

- Attributes can be set at various levels (in increasing order of precedence)
- Automatically on the Node itself (by Ohai)
- In roles
- In Environments
- In Cookbook Recipes
- In Cookbook attribute files

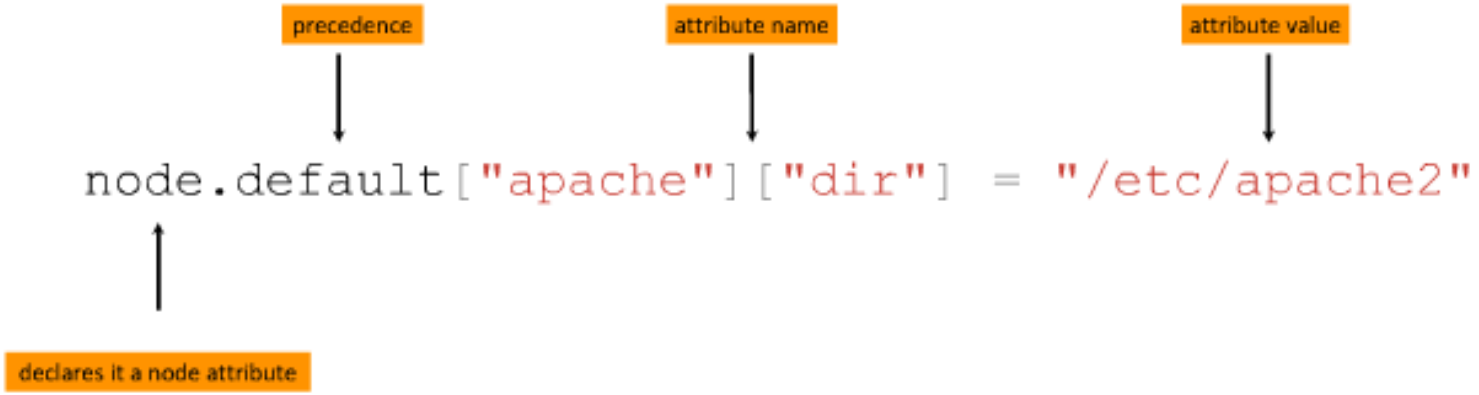
Setting Attributes in attribute files

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

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

Setting Attributes in Recipes

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



Problem Statement

- We have defined our “index.html” homepage in our Recipe, but we may want to change that without changing the Recipe
- Success of this will be determined by changing the homepage by changing an attribute value

Exercise: Set attribute in attributes file

EduRamp ELS
The B2B Learning Specialist

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

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

SAVE FILE!

- Set an attribute to a specific value in the attributes file

Exercise: Add index1.html to Cookbook's files/default directory

EduRamp ELS
The B2B Learning Specialist

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

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

SAVE FILE!

Exercise: Set Attribute in a Recipe

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

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

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

SAVE FILE!

Exercise: Upload the Cookbook

```
$ knife cookbook upload apache
```

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

Exercise: Re-run Chef Client

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

```
Starting Chef Client, version 11.8.2
resolving cookbooks for run list: ["apache"]
...
Recipe: apache::default
  * package[httpd] action install (up to date)
  * service[httpd] action enable (up to date)
  * service[httpd] action start (up to date)
  * cookbook_file[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 03fb1d to 478a29
        --- /var/www/html/index.html  2014-01-07 04:28:34.994965341 -0500
        +++ /tmp/.index.html20140107-12128-a6czsl  2014-01-07 04:31:24.049155232 -0500
        @@ -1,6 +1,8 @@
         <html>
        -<body>
        + <body>
            <h1>Hello, world!</h1>
        -</body>
        +    <h2>This is index1.html</h2>
        +    <p>We configured this in the attributes file</p>
        + </body>
         </html>
    - restore selinux security context

Chef Client finished, 1 resources updated
```

Exercise: Verifying New Homepage Works

EduRamp ELS
The B2B Learning Specialist

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

Exercise: Set attribute in Recipe

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

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

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

SAVE FILE!

Exercise: add index2.html to Cookbook's files/default directory

EduRamp ELS
The B2B Learning Specialist

OPEN IN EDITOR: cookbooks/apache/files/default/index2.html

```
<html>
 <body>
  <h1>Hello, world!</h1>
   <h2>This is index2.html</h2>
    <p>We configured this in the recipe</p>
 </body>
</html>
```

SAVE FILE!

Exercise: Update the Cookbook

```
$ knife cookbook upload apache
```

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

Exercise: Re-run Chef Client

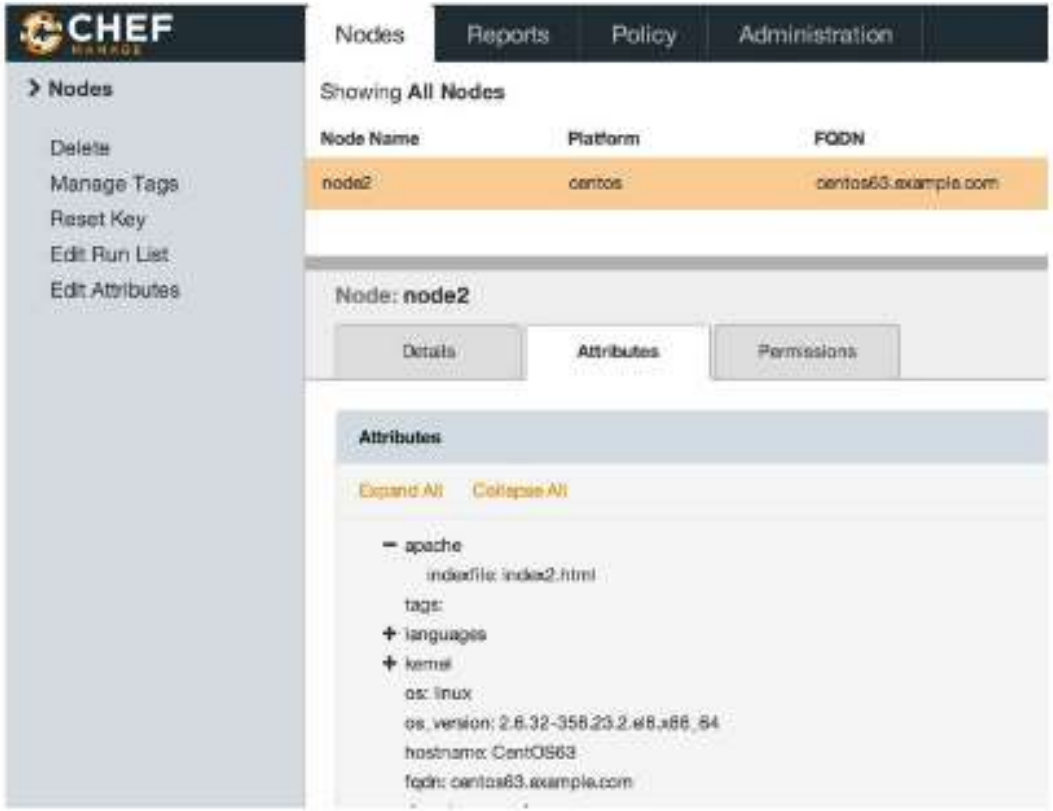
```
chef@node1:~$ sudo chef-client
```

```
Starting Chef Client, version 11.8.2
resolving cookbooks for run list: ["apache"]
...
Recipe: apache::default
  * package[httpd] action install (up to date)
  * service[httpd] action enable (up to date)
  * service[httpd] action start (up to date)
  * cookbook_file[/var/www/html/index.html] action create
    - update content in file /var/www/html/index.html from 478a29 to 153183
        --- /var/www/html/index.html  2014-01-07 04:31:24.049155232 -0500
        +++ /tmp/.index.html20140107-12296-11u8kv7 2014-01-07 04:32:45.605669431 -0500
        @@ -1,8 +1,8 @@
         <html>
          <body>
           <h1>Hello, world!</h1>
        -   <h2>This is index1.html</h2>
        -    <p>We configured this in the attributes file</p>
        +   <h2>This is index2.html</h2>
        +    <p>We configured this in the recipe</p>
          </body>
         </html>
    - restore selinux security context

Chef Client finished, 1 resources updated
```

Exercise: Verify New Homepage Works

- Open a web browser
- Recipe value has taken precedence

Exercise: Viewing Attributes view WebUI



Exercise: Viewing Attributes using knife


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

```
...
Attributes:
tags:

Default Attributes:
apache:
  indexfile: index2.html

Override Attributes:

Automatic Attributes (Ohai Data):
block_device:
  dm-0:
    removable: 0
    size:      31563776
...
```

Default Attribute Precedence

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

Attributes, Templates and Cookbook Dependencies

- Describe Cookbook Attribute files
- Use ERB Template in Chef
- Explain Attribute Precedence
- Describe Cookbook Metadata
- Specify Cookbook dependencies
- Perform Cookbook creation, upload and test loop

Problem Statement

- 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 of this will be determined by seeing the message when we log in to the test Node

Exercise: Create a Cookbook named ‘motd’


```
$ knife cookbook create motd
```

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

Exercise: Create a default.rb attribute file

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

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

SAVE FILE!

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

Exercise: Open the default Recipe in your editor

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

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

Exercise: Add template resource for /etc/motd

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

The template [/etc/motd] resource

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

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

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

SAVE FILE!

Exercise: Open motd.erb in your editor

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

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

SAVE FILE!

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

Templates are used for Almost All Configurations

- 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

Exercise: Upload the motd Cookbook


```
$ knife cookbook upload motd
```

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

Exercise: Create a Cookbook 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.erb attribute file

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

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

SAVE FILE!

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

Exercise: Upload the PCI Cookbook


```
$ knife cookbook upload pci
```

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

Exercise: Add the motd Recipe to your Node's Run List


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

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

Exercise: Add the motd Recipe to your Node's Run List


```
$ knife node show node1
```

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

Exercise: Re-run the Chef Client

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

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

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

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

FAIL!

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

OPEN IN EDITOR: cookbooks/motd/metadata.rb

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

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

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

Cookbook Metadata

OPEN IN EDITOR: cookbooks/motd/metadata.rb

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

Cookbook Attributes are Applied for all Downloaded Cookbooks

- Cookbooks downloaded as dependencies will have their attributes files evaluated
- Even if there is no Recipe from the Cookbook in the Run List

Exercise: Upload the motd Cookbook


```
$ knife cookbook upload motd
```

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

Exercise: Re-run the Chef Client

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

```
Forking chef instance to converge...
Starting Chef Client, version 11.8.2
*** Chef 11.8.2 ***
Chef-client pid: 15152
Run List is [recipe[apache], recipe[motd]]
Run List expands to [apach  motd]
Starting Chef Run for node
…
    - update content in file /etc/ from  b0c4  62eb
        (new content is bina , d f o ut ppres d)
    - restore selinux securit  co ext

Chef Run complete in 6.8103587 conds
Removing cookbooks/apache/files/default/index2.html from the cache; it is no longer needed by chef-client.
Running report handlers
Report handlers complete
Chef Client finished, 1 resources updated
Sending resource update report (run-id: f923449e-cc7b-45f7-a9fb-ac1da3653557)
```

WIN!

Exercise: Check your Work

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

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

Template Variables, Notifications, Controlling Idempotency

- Use the **execute** resource
- Control idempotence manually with `not_if` and `only_if`
- Navigate the Resource page on docs.getchef.com
- Describe the Directory resource
- Implement resource notifications
- Explain what Template Variables are and how to use them
- Use Ruby variables loops and string expansion

Problem Statement

- We need to deploy multiple custom home pages running on different ports
- The success of this will be determined by being able to view our custom home page

Exercise: Change the Cookbook's version number in the metadata

OPEN IN EDITOR: cookbooks/apache/metadata.rb

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

SAVE FILE!

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

Exercise: Edit the attribute file default.rb

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

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

SAVE FILE!

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

Exercise: Open the default Recipe in your editor

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

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

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

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

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

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

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

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

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

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

SAVE FILE!

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

Execute resources are generally not idempotent

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

Enter the not_if and only_if meta parameter


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

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

Exercise: Iterate over each apache site

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

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

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

- Delete the cookbook_file resource

Exercise: Iterate over each apache site


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

- Calling `.each` loops over each site

```
"apache"  "sites"  "clowns"     "port"    80
"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`"

Template Variables

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

Exercise: Add a directory resource to create the document_root

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

Exercise: Add the directory resource


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

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

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


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

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

Don’t forget the last “end”

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

Exercise: Add custom.erb to your templates directory

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

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

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

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

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

SAVE FILE!

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

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

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

SAVE FILE!

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

Exercise: Upload the Apache Cookbook


```
$ knife cookbook upload apache
```

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

Exercise: Verify the two sites are working

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

- Recipes contain the pattern for how to so something ("How we deploy apache virtual hosts")

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

Recipe Inclusion, Data Bags, Search

- Explain what Data Bags are and how they are used
- Describe the User and Group resources
- Use the `include_recipe` directive
- Describe the role Search plays in recipes

Problem Statement

- Employees should have local user accounts created on servers along with custom groups
- The success of this will be checked by adding new employees and groups to server dynamically

Where should we store the user data?

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

Introducing Data Bags

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

Make a data_bags directory

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

(No output)

Exercise: Create a data bag named users


```
$ knife data_bag create users
```

```
Created data_bag[users]
```

Exercise: Create a user item in the users data bag

OPEN IN EDITOR: data_bags/users/bobo.json

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

SAVE FILE!

Exercise: Create the data bag item


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

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

Exercise: Create another user in the user data bag

OPEN IN EDITOR: data_bags/users/frank.json

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

SAVE FILE!

Exercise: Create the data bag item


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

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

Exercise: Show all the items in the user data bag


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

```
2 items found

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

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

Exercise: Find Bobo's shell in Chef


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

```
1 items found

data_bag_item_users_bobo:
  shell: /bin/bash
```

Exercise: Create a data bag named groups


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

```
Created data_bag[groups]
```

Exercise: Create a group item in the group data bag

OPEN IN EDITOR: data_bags/groups/clowns.json

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

SAVE FILE!

Exercise: Create the data bag item


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

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

Exercise: Show all the groups in Chef


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

```
1 items found

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

Exercise: Create a Cookbook named ‘users’


```
$ knife cookbook create users
```

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

Exercise: Open the default recipe in your editor

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

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

SAVE FILE!

Exercise: Open the default recipe in your editor


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

include_recipe "users::groups"
```

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

Exercise: Open the default recipe in your editor


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

include_recipe "users::groups"
```

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

Best Practice: Use include_recipe liberally

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

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

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

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

SAVE FILE!

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

Exercise: Upload the users Cookbook

EduRamp
The B2B Learning Specialist

```
$ knife cookbook upload users
```

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

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


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

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

Exercise: Re-run the Chef Client

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

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

    - create user user[bobo]

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

    - create user user[frank]

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

    - create group[clowns]
...
```

Exercise: Verify the users and groups exist


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

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

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

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

Recap

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

Roles

- Explain what Roles are and how they are used to provide clarity
- Discuss the Role Ruby DSL
- Show a Role with knife
- Explain how merge order affects the precedence hierarchy
- Describe nested Roles

What is a Role?

- So far, we have been just adding recipes directly to a single node
- But that is not how your infrastructure works – this 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 buts it is nice to be able to see them evolve in your source control history

Exercise: Create the webserver role

OPEN IN EDITOR: roles/webserver.rb

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

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

SAVE FILE!

Exercise: Create the webserver role

OPEN IN EDITOR: roles/webserver.rb

- You can set default node attributes within a role.

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

SAVE FILE!

Exercise: Create the Role

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

```
Updated Role webserver!
```

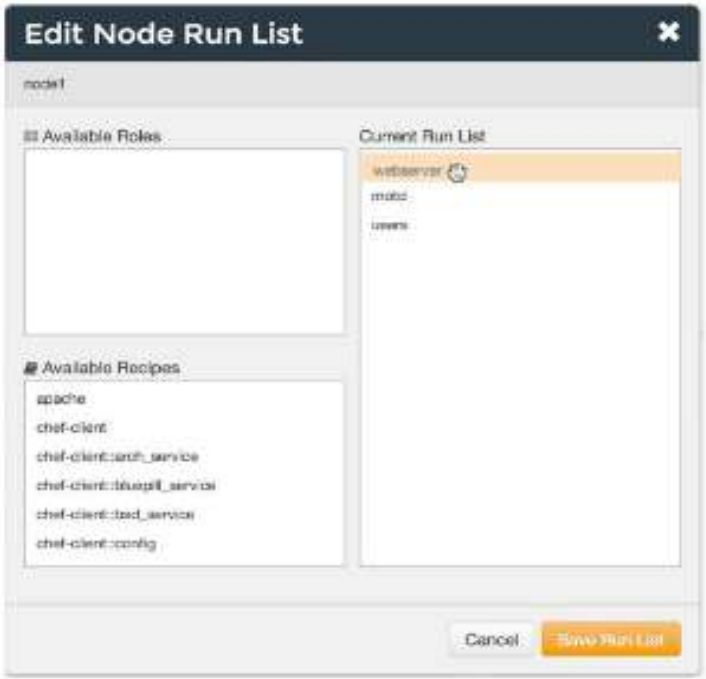
Exercise: Show the Role with knife


```
$ knife role show webserver
```

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

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

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



Exercise: Re-run the Chef Client

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

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

Exercise: Re-run the Chef Client

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

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

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

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

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

    - update content in file /srv/apache/admin/index.html from none to 8abbb3
      --- /srv/apache/admin/index.html  2014-01-07 06:39:51.762119942 -0500
      +++ /tmp/chef-rendered-template20140107-17953-1suqbi2014-01-07 06:39:51.764120008 -0500
      @@ -1 +1,8 @@
      +<html>
      +  <body>
      +    <h1>Welcome to Chef</h1>
      +    <h2>We love admin</h2>
      +    10.160.201.90:8000
      +  </body>
      +</html>[2014-01-07T06:39:51-05:00] INFO: template[/srv/apache/admin/index.html] mode changed to 644

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

Node Attributes that are hashes are merged

- The apache cookbooks attribute file contains:

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

- While our role has...

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

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


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

```
1 items found

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

Exercise: Edit the webserver role

OPEN IN EDITOR: roles/webserver.rb

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

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

SAVE FILE!

Exercise: Create the role

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

```
Updated Role webserver!
```

Exercise: Re-run the Chef Client

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

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

    - update content in file /etc/httpd/conf.d/bears.conf from 30610b to 92ele4
        --- /etc/httpd/conf.d/bears.conf 2014-01-07 05:13:42.450113970 -0500
        +++ /tmp/chef-rendered-template20140107-18263-5563t5 2014-01-07 06:57:48.091397610 -0500
        @@ -1,6 +1,6 @@
        -  Listen 81
        +  Listen 8081

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

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

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


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

```
1 items found

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

Best Practice: Have “base” roles

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

Exercise: Create the base role

OPEN IN EDITOR: roles/base.rb

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

SAVE FILE!

Exercise: Create the role

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

```
Updated Role base!
```

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

OPEN IN EDITOR: roles/webserver.rb

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

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

SAVE FILE!

Exercise: Update the role

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

```
Updated Role webserver!
```

Exercise: Re-run the Chef Client

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

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

Best Practice: Be explicit about what you ned or expect

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

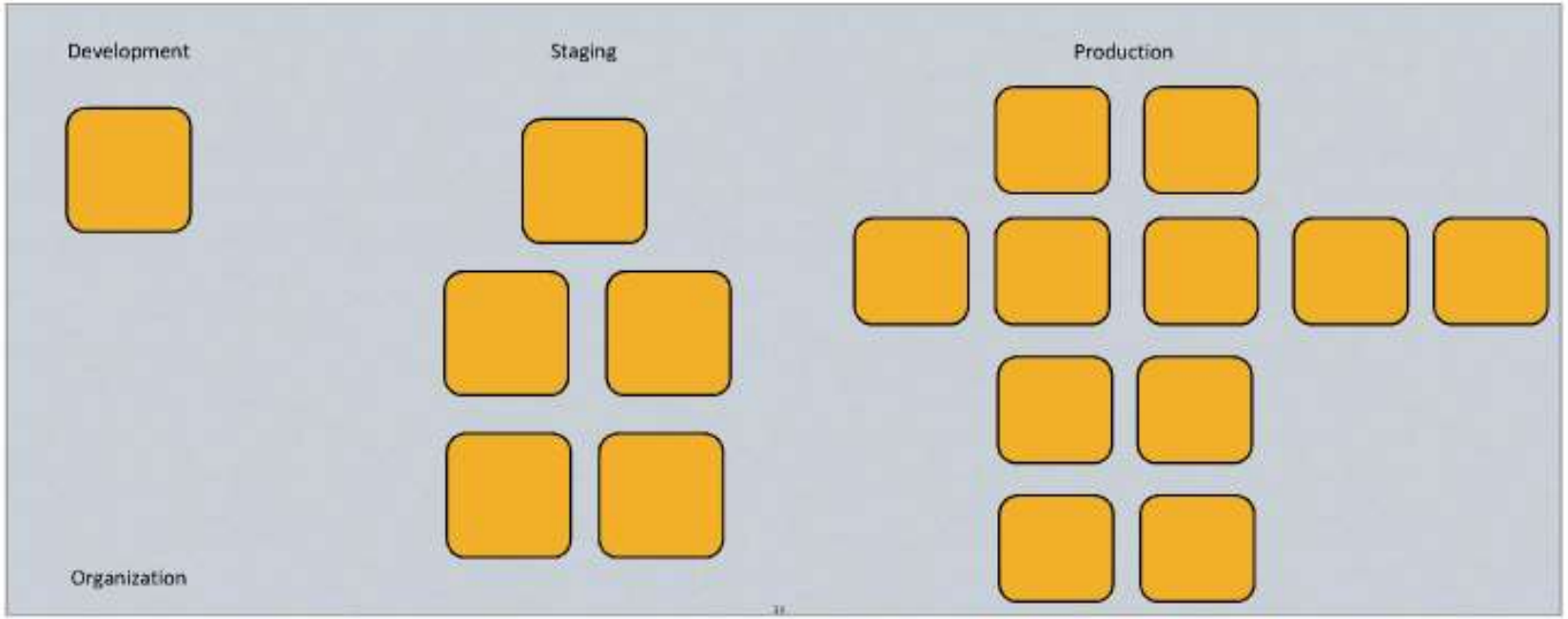
Exercise: Set the run list to just role [webserver]

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

Environments

- Describe what an Environment is and how it is different from an Organization
- Set Cookbook version constraints
- Explain when to set attributes in an evironment

Environments



Environments

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

Exercise: Use knife to show the available Cookbook versions


```
$ knife cookbook show apache
```

```
apache    0.2.0   0.1.0
```

Exercise: List current environments

EduRamp ELS
The B2B Learning Specialist

```
$ knife environment list
```

```
_default
```

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

Make an environments directory


```
$ mkdir environments
```

```
(No output)
```

Exercise: Create a Development Environment

OPEN IN EDITOR: environments/dev.rb

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

SAVE FILE!

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

Cookbook Version Constraints

- = Equal to
- There are other options but equality is the recommended practice
- Learn more at

http://docs.getchef.com/chef/essentials_cookbook_versions.html

Exercise: Create the Development Environment


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

```
Updated Environment dev
```

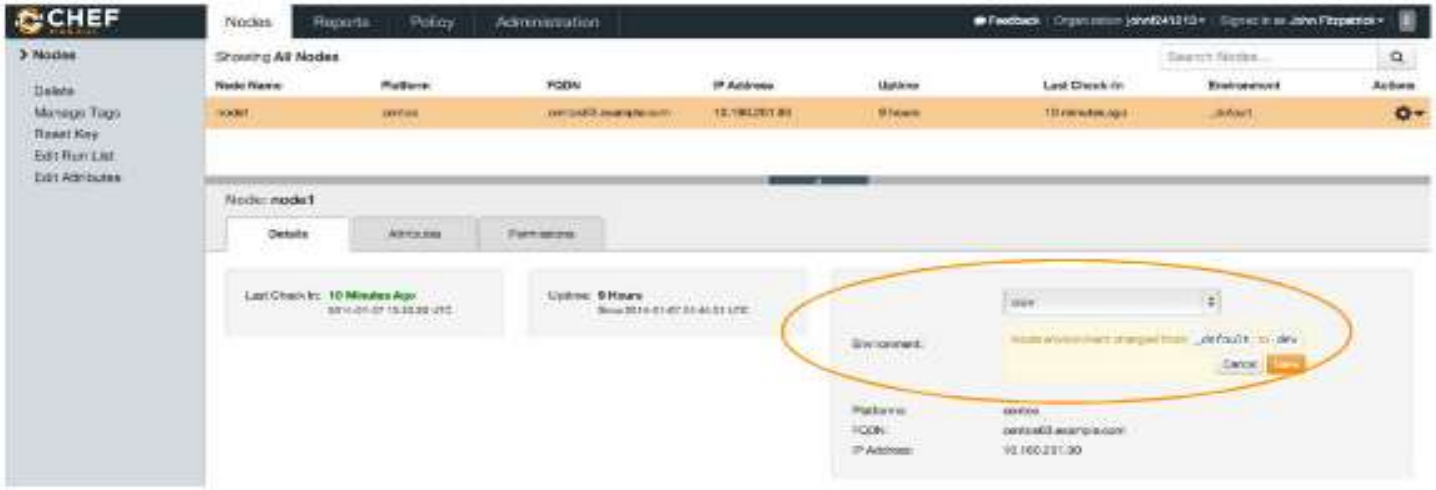
Exercise: Show your Development Environment


```
$ knife environment show dev
```

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

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

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



Exercise: Re-run the Chef Client

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

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

Exercise: Create a production environment

OPEN IN EDITOR: environments/production.rb

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

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

SAVE FILE!

Exercise: Create the production environment


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

```
Updated Environment production
```

Exercise: Re-run the Chef Client

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

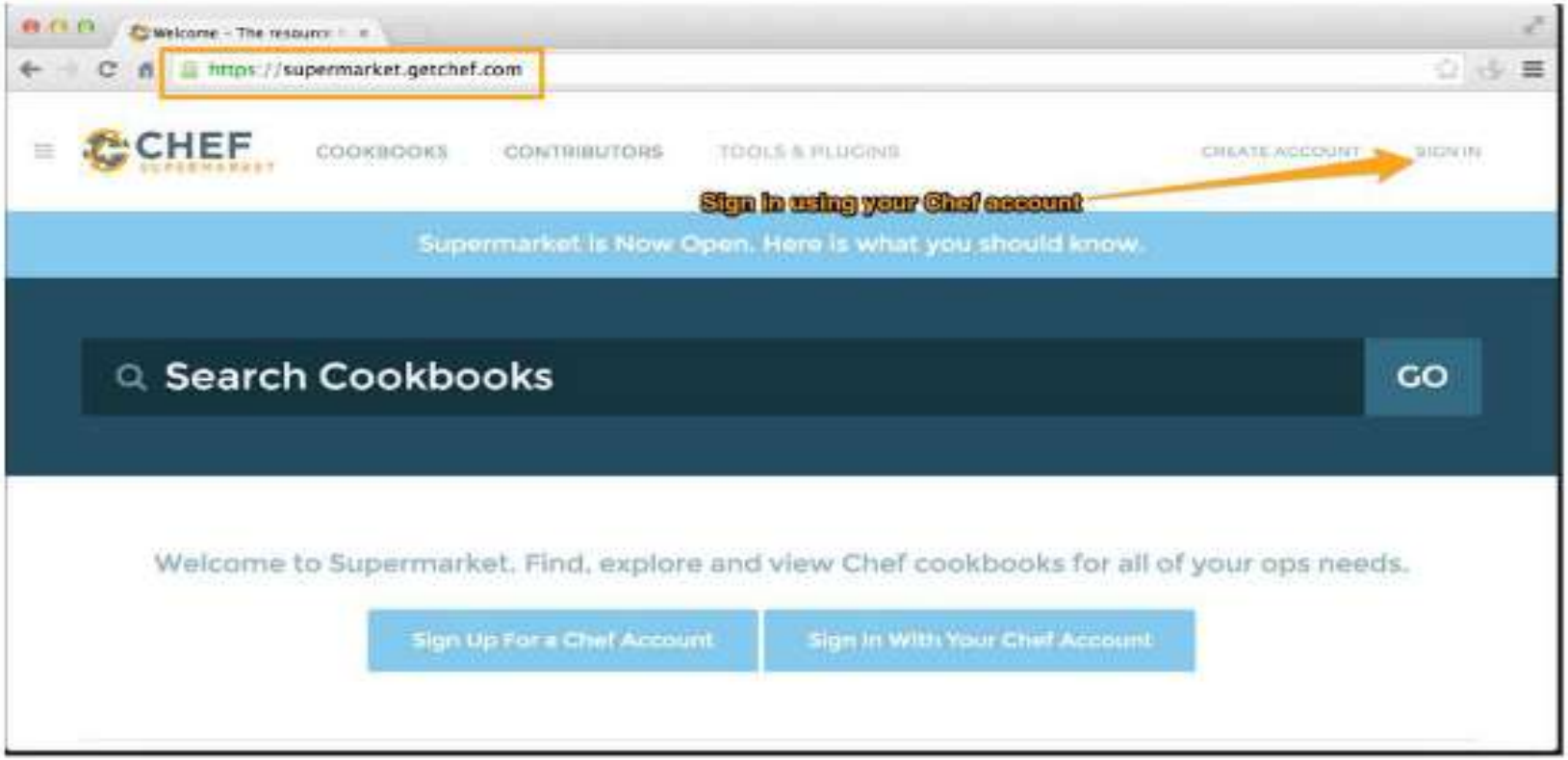
```
INFO: Loading cookbooks [apache, motd, pci, users]
Synchronizing Cookbooks:
...
Recipe: motd::default
  * template[/etc/motd] action create[2014-01-07T08:40:00-05:00] INFO: Processing template[/etc/motd] action create
(motd::default line 9)
[2014-01-07T08:40:00-05:00] INFO: template[/etc/motd] backed up to /var/chef/backup/etc/motd.chef-20140107084000.070561
[2014-01-07T08:40:00-05:00] INFO: template[/etc/motd] updated file contents /etc/motd

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

Chef Supermarket

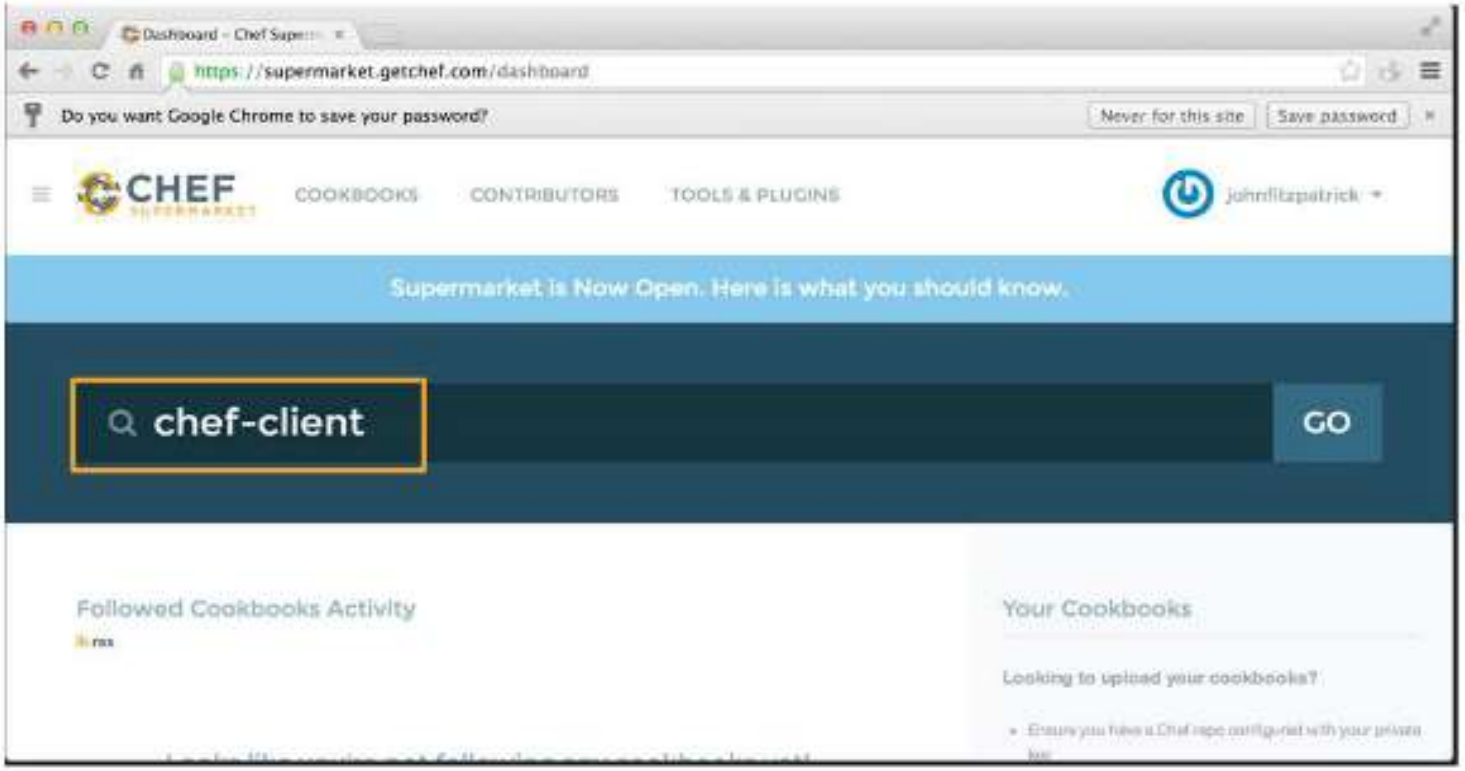
- Find, preview and download Cookbooks from the Chef Supermarket community site
- Use knife to work with the Chef Supermarket site API
- Download, extract, examine and implement Cookbooks from the Chef Supermarket

EduRamp ELS
The B2B Learning Specialist

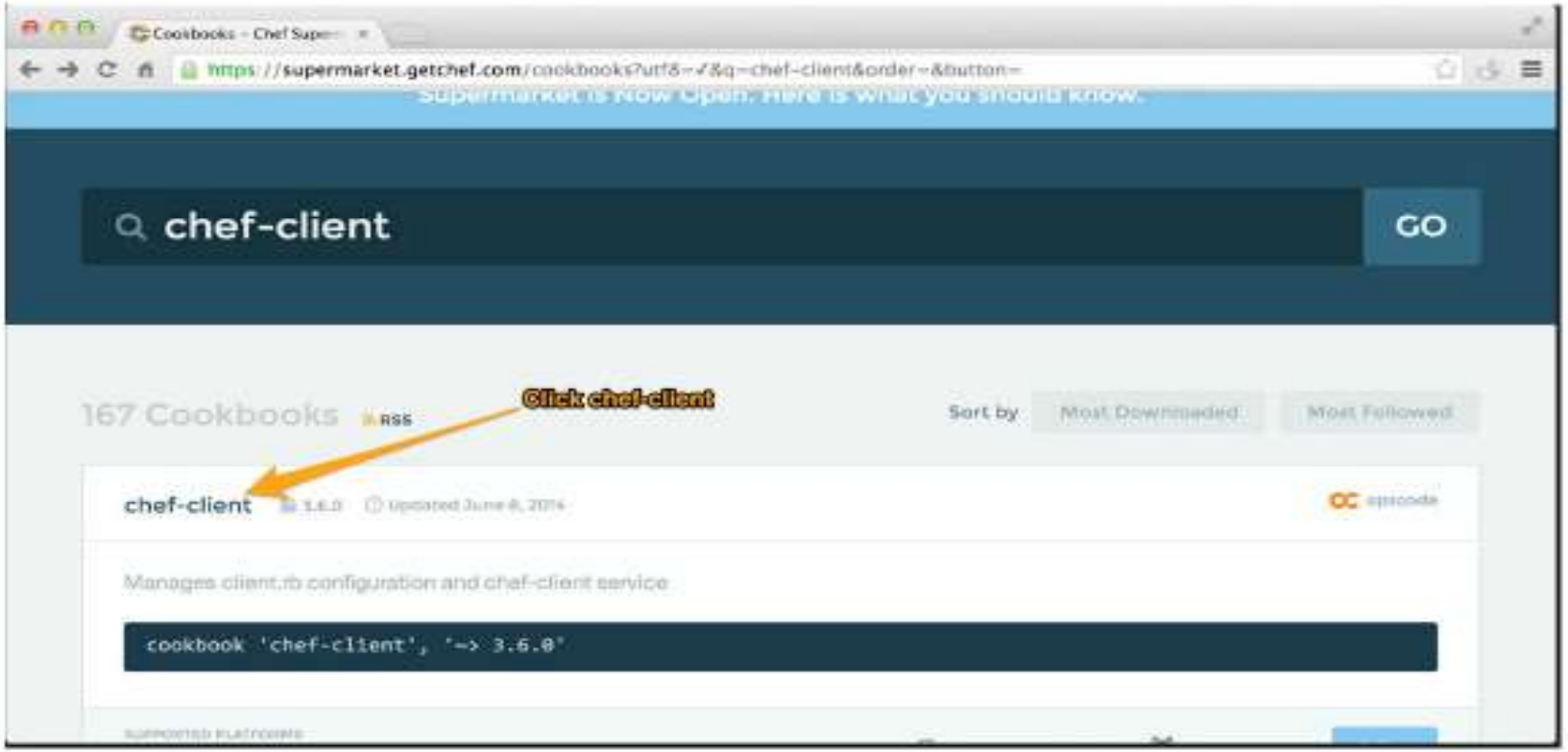
Exercise: Log into Supermarket



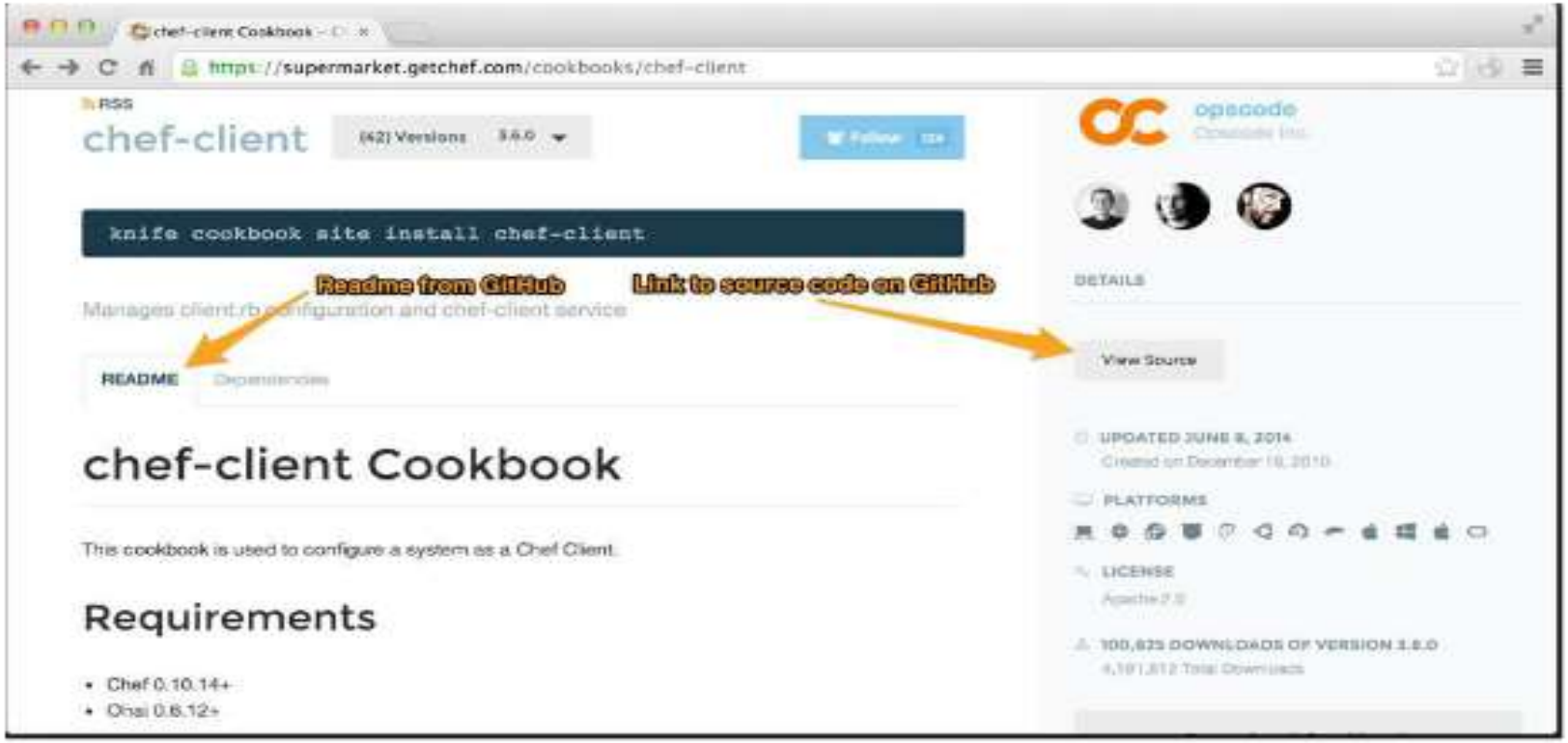
Exercise: Search for chef-client



Search Results



Viewing a Cookbook



You can download Cookbooks directly from the site…

- You can download Cookbooks directly from the Supermarket site, but:
  - It does not put them in your Chef Repository
  - It is not fast if you know what you are looking for (click, click…)
  - It is not necessarily fast if you **don't** know what you are looking for
  - You are 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 ntp Cookbook

NTP Cookbook

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

Exercise: Download the ntp Cookbook


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

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

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

Exercise: Extract the ntp Cookbook

EduRamp
The B2B Learning Specialist

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

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

Examining the ntp Cookbook

- The Cookbook is quite flexible but for this exercise we are just interested in the most basic use, an NTP client
  - default recipe

Exercise: View the ntp::default receipe


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

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

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

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

Exercise: Upload the ntp Cookbook

EduRamp
The B2B Learning Specialist

```
$ knife cookbook upload ntp
```

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

Exercise: Add the ntp recipe to the base role

OPEN IN EDITOR: roles/base.rb

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

SAVE FILE!

Exercise: Upload the base role

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

```
Updated Role base!
```

Exercise: Re-run the Chef Client

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

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

    - restart service service[ntpd]
```