Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

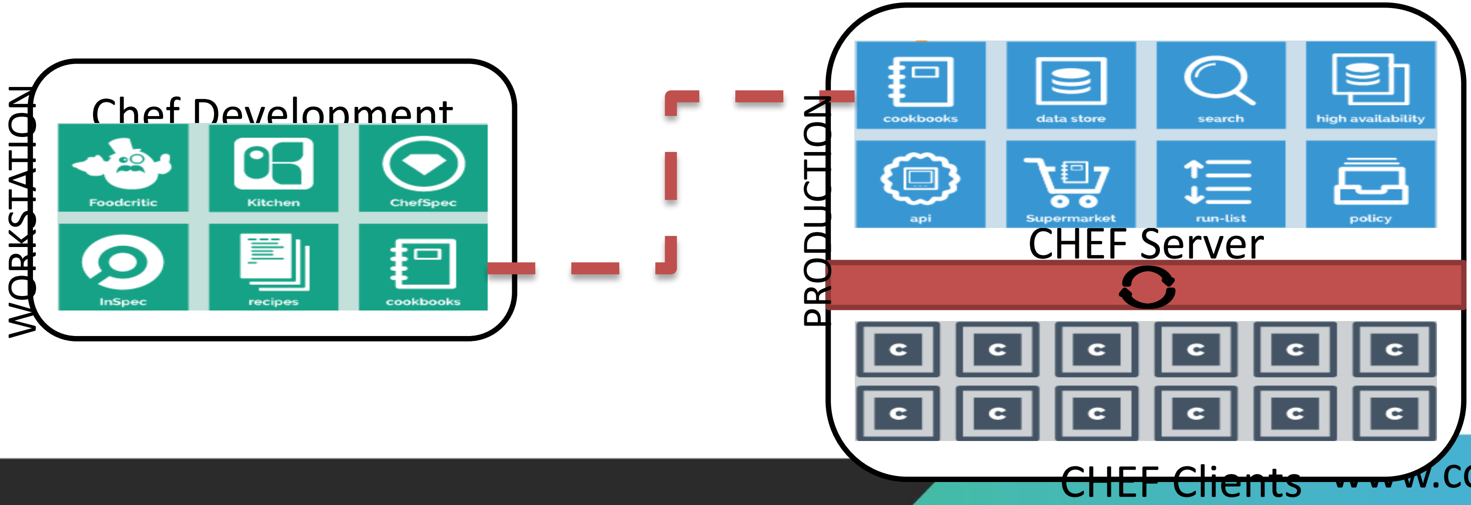
Chef for configuration management

Chef

- Chef is a configuration management tool written in Ruby and Erlang
- Was written to manage Linux but later versions also support Microsoft Windows
- In February 2013, Opscode released version 11 of Chef
- It uses a pure-Ruby to write system configuration "recipes"
- Integrates with cloud-based platforms such as Internap, Amazon EC2, Google Cloud Platform, OpenStack, SoftLayer, Microsoft Azure and Rackspace
- Support for includes AIX, RHEL/CentOS, FreeBSD, OS X, Solaris, Microsoft Windows and Ubuntu platforms
- Additional client platforms supported include Arch Linux, Debian and Fedora
- Chef Server can be on RHEL/CentOS, Oracle Linux, and Ubuntu
- Chef can run in client/server mode or standalone configuration named "chef-solo"

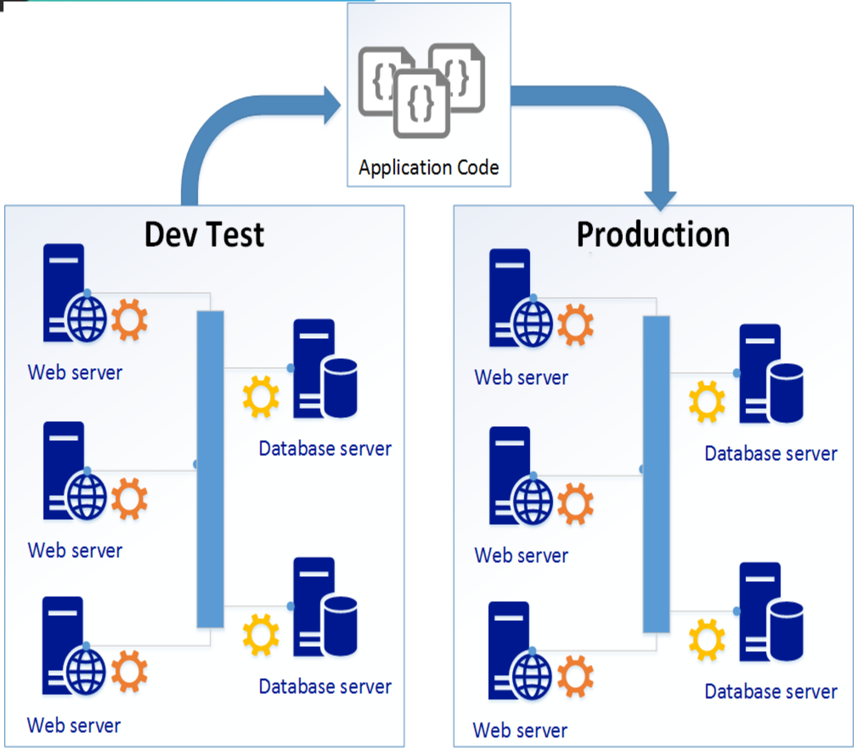
Chef Architecture

- Chef Development Kit has tools to develop and test your infrastructure automation code
- Infrastructure as code automation code is developed locally on workstation and then deployed in production
- Chef Server is a central repository for Chef cookbooks and have information about every node being managed
- Chef client runs on each node and securely communicates with the Chef server to get the latest configuration instructions for that node
- Chef cookbooks have code for desired state of infrastructure
- Chef node is a physical machine or virtual machine in network being managed by the Chef server



Chef is Infrastructure as Code

- ‘Infrastructure as a code’ is a modern approach to manage infrastructure
- Infrastructure as Code (IaC) is essential to DevOps
- Computing and network infrastructure are defined in code
    - This can be stored in source control systems
    - The Puppet integration tool is an example of IaC



Traditional Approach to Manage Infrastructure

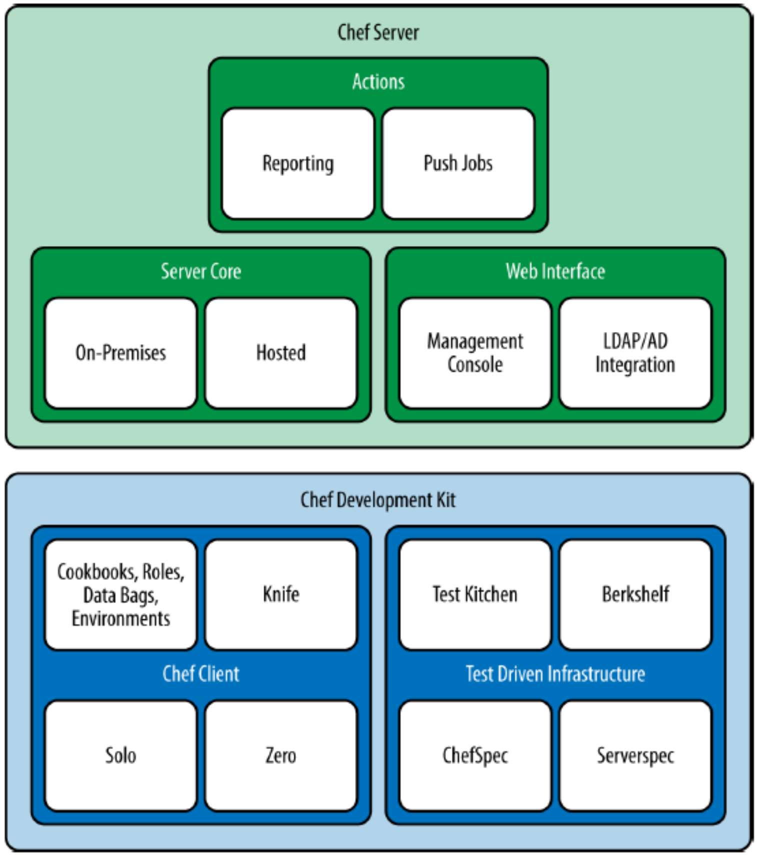
- Releasing a new service used to be complex
- Need to find or purchase hardware
- Expensive and often time consuming
- Hardware needs to be configured to support the applications
- Software installation and configuration is time consuming
- Difficult to automate

Cloud Platforms

- Cloud platforms change everything
- Need to set up an account and a payment method
- Now have a dynamic infrastructure
- Servers can be created and destroyed by software commands

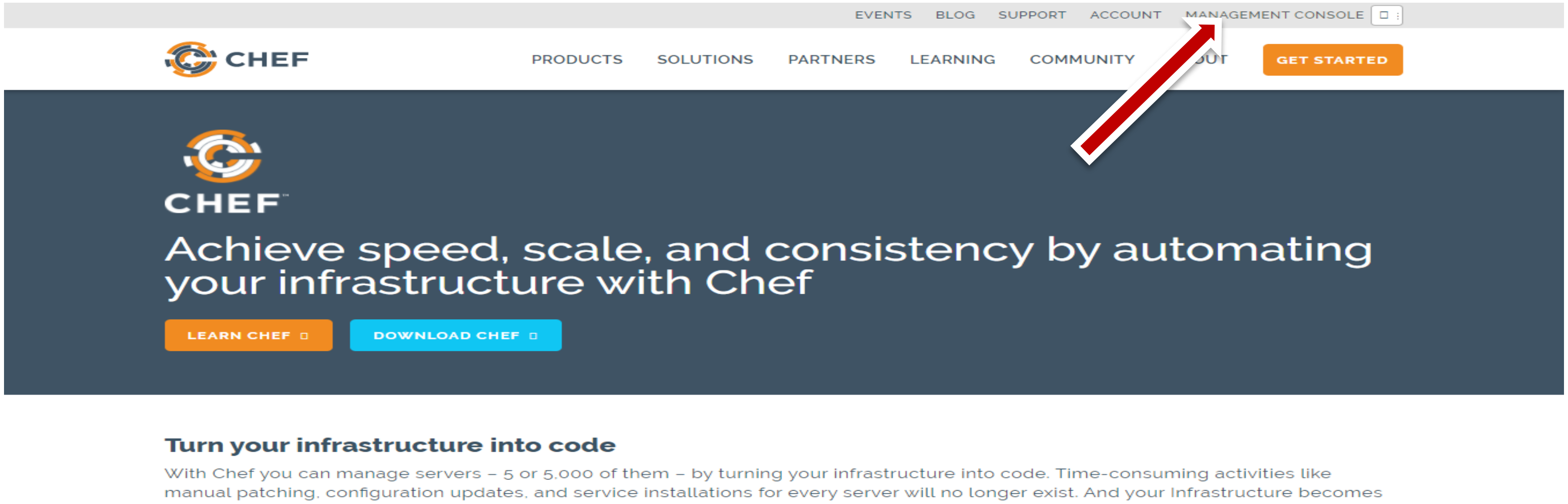
Chef Overview

- We will define the following in context of Chef:
  - Server
  - Node
  - Resource
  - Recipe
  - Cookbook
  - Run List
  - Roles
  - Search



Chef Server

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



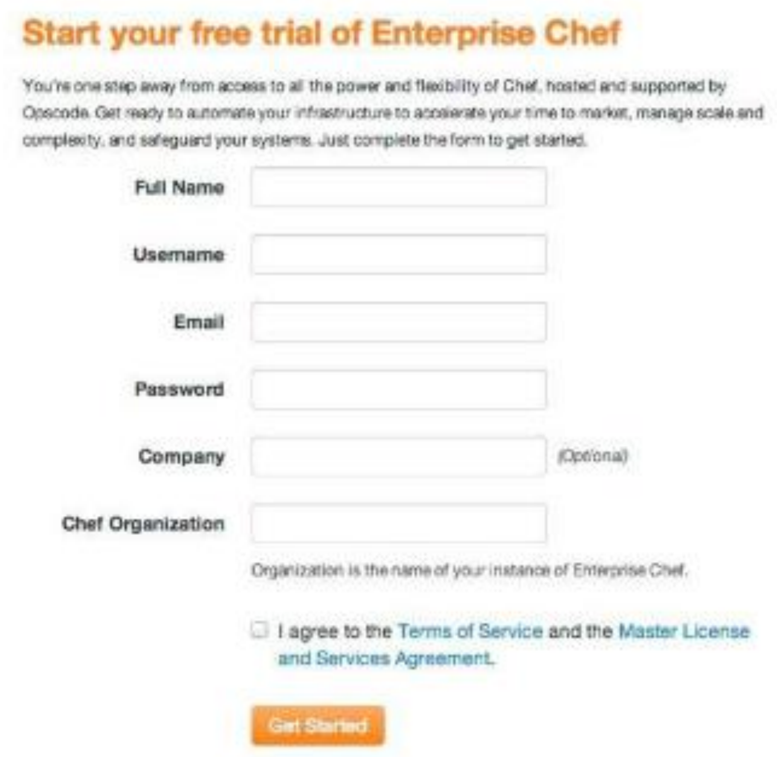
Achieve speed, scale, and consistency by automating your infrastructure with Chef

Turn your infrastructure into code

With Chef you can manage servers – 5 or 5,000 of them – by turning your infrastructure into code. Time-consuming activities like manual patching, configuration updates, and service installations for every server will no longer exist. And your Infrastructure becomes

Create a New Account

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



Start your free trial of Enterprise Chef

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

Full Name		
Username		
Email		
Password		
Company		(Optional)
Chef Organization		

Organization is the name of your instance of Enterprise Chef.

☐ I agree to the Terms of Service and the Master License and Services Agreement.

Get Started

Chef Organization

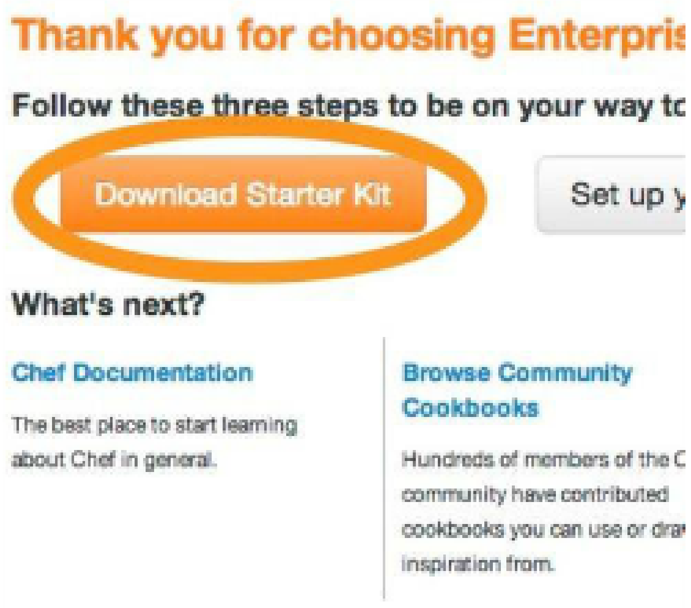
- An organization is the top-level entity for role-based access control in the Chef server
- Each organization contains the default groups (admins, clients, and users, plus billing_admins for the hosted Chef server), at least one user and at least one node (on which the chef-client is installed)
- The Chef server supports multiple organizations
- Organizations are completely independent tenants of Enterprise Chef
- Share nothing with other organizations
- May represent different
    - Companies
    - Business units
    - Departments

Create a New Organization

- From Chef Server console menu create a new organization of your choice

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)



Thank you for choosing Enterpris

Follow these three steps to be on your way tc

What's next?

Chef Documentation

The best place to start learning about Chef in general.

Browse Community Cookbooks

Hundreds of members of the C community have contributed cookbooks you can use or dra inspiration from.

What is 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

Chef Node

- 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.

Nodes in Organizations

- 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

Chef 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

Resources 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

Server Resources

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

Declarative Interface for 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

Chef Recipe

- 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
```

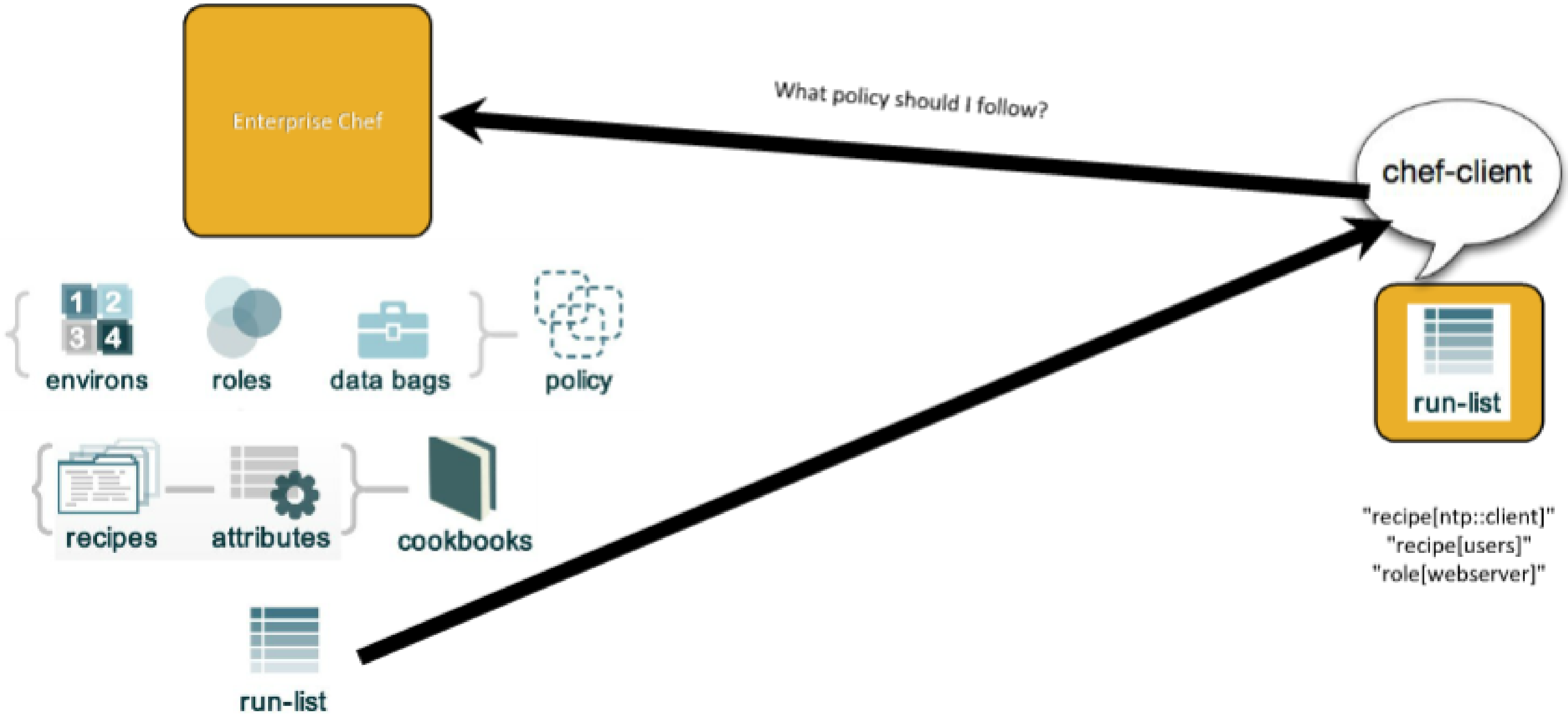
Chef Cookbooks

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

Chef Run List

- A run-list defines all of the information necessary for Chef to configure a node into the desired state
- A run-list is:
  - An ordered list of roles and/or recipes that are run in the exact order defined in the run-list; if a recipe appears more than once in the run-list, the chef-client will not run it twice
  - Always specific to the node on which it runs; nodes may have a run-list that is identical to the run-list used by other nodes
  - Stored as part of the node object on the Chef server
  - Maintained using knife and then uploaded from the workstation to the Chef server, or maintained using Chef Automate

Chef Run List



"recipe[ntp::client]"
"recipe[users]"
"role[webserver]"

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

Chef Roles

- A role is a way to define certain patterns and processes that exist across nodes in an organization as belonging to a single job function
- Each role consists of zero (or more) attributes and a run-list
- Each node can have zero (or more) roles assigned to it
- When a role is run against a node, the configuration details of that node are compared against the attributes of the role, and then the contents of that role's run-list are applied to the node's configuration details
- When a chef-client runs, it merges its own attributes and run-lists with those contained within each assigned role

Chef 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

Chef Search

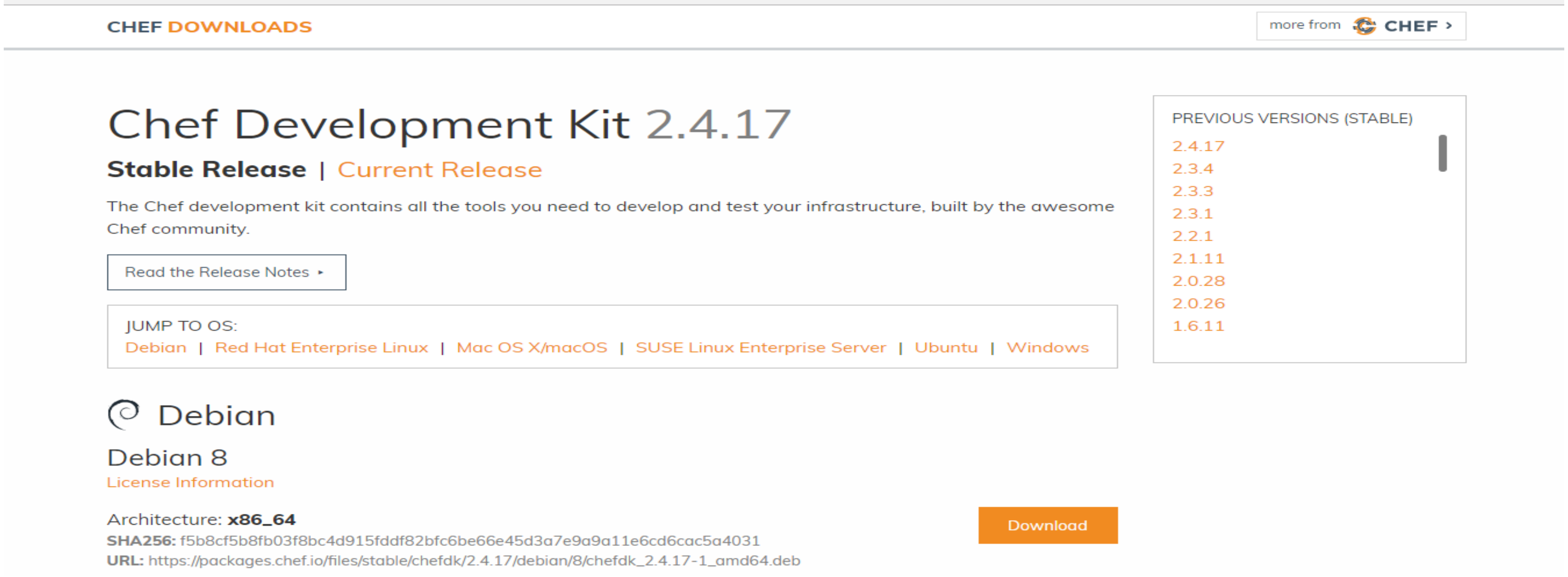
- Search indexes allow queries to be made for any type of data that is indexed by the Chef server, including data bags (and data bag items), environments, nodes, and roles. A defined query syntax is used to support search patterns like exact, wildcard, range, and fuzzy
- Search for Nodes with Roles
- 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
```

Install Chef DK

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



Chef Development Kit 2.4.17

Stable Release | Current Release

The Chef development kit contains all the tools you need to develop and test your infrastructure, built by the awesome Chef community.

JUMP TO OS:
Debian | Red Hat Enterprise Linux | Mac OS X/macOS | SUSE Linux Enterprise Server | Ubuntu | Windows

Debian

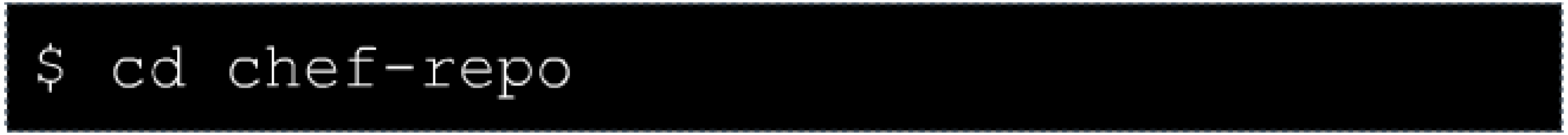
Debian 8
License Information

Architecture: **x86_64**
SHA256: f5b8cf5b8fb03f8bc4d915fddf82bfc6be66e45d3a7e9a9a11e6cd6cac5a4031
URL: https://packages.chef.io/files/stable/chefdk/2.4.17/debian/8/chefdk_2.4.17-1_amd64.deb

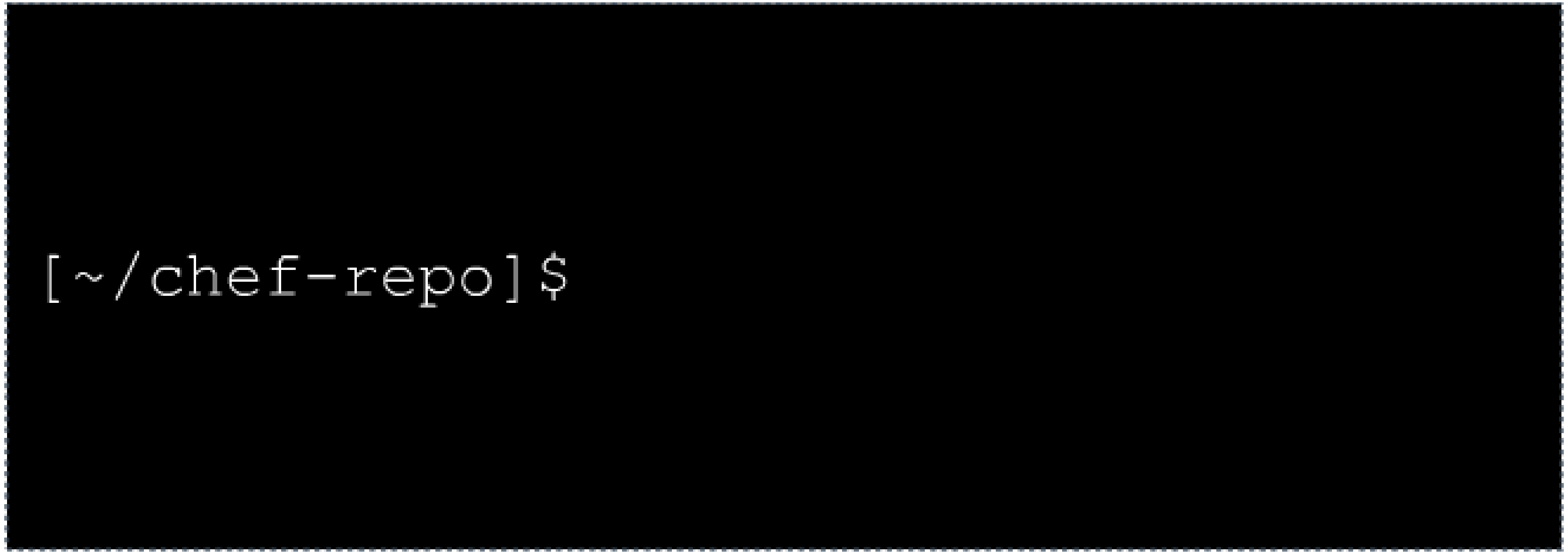
PREVIOUS VERSIONS (STABLE)
2.4.17
2.3.4
2.3.3
2.3.1
2.2.1
2.1.11
2.0.28
2.0.26
1.6.11

Chef Repo

- Chef DK is stalled under folder chef-repo on your workstation



```
$ cd chef-repo
```



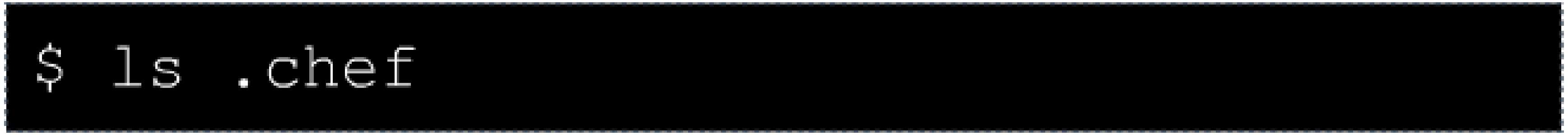
```
[~/chef-repo]$
```

Chef Repo

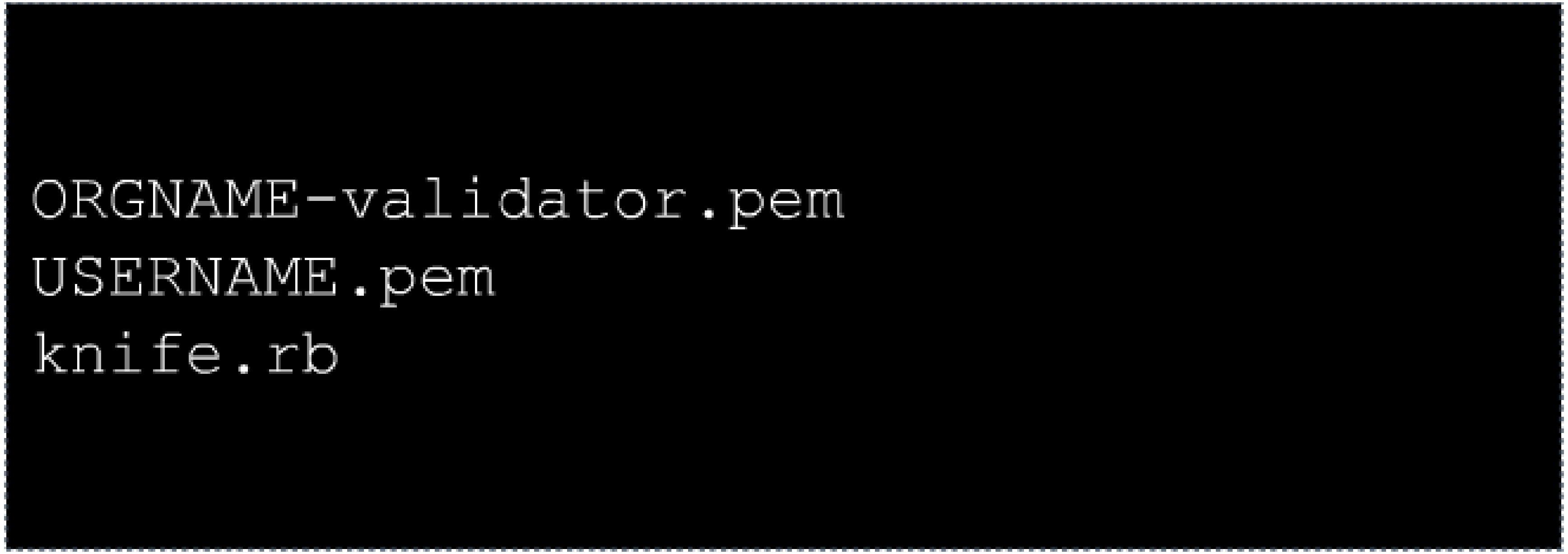
```
$ ls -al
```

```
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 is Inside .chef Folder



```
$ ls .chef
```



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

What is Inside .chef Folder

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

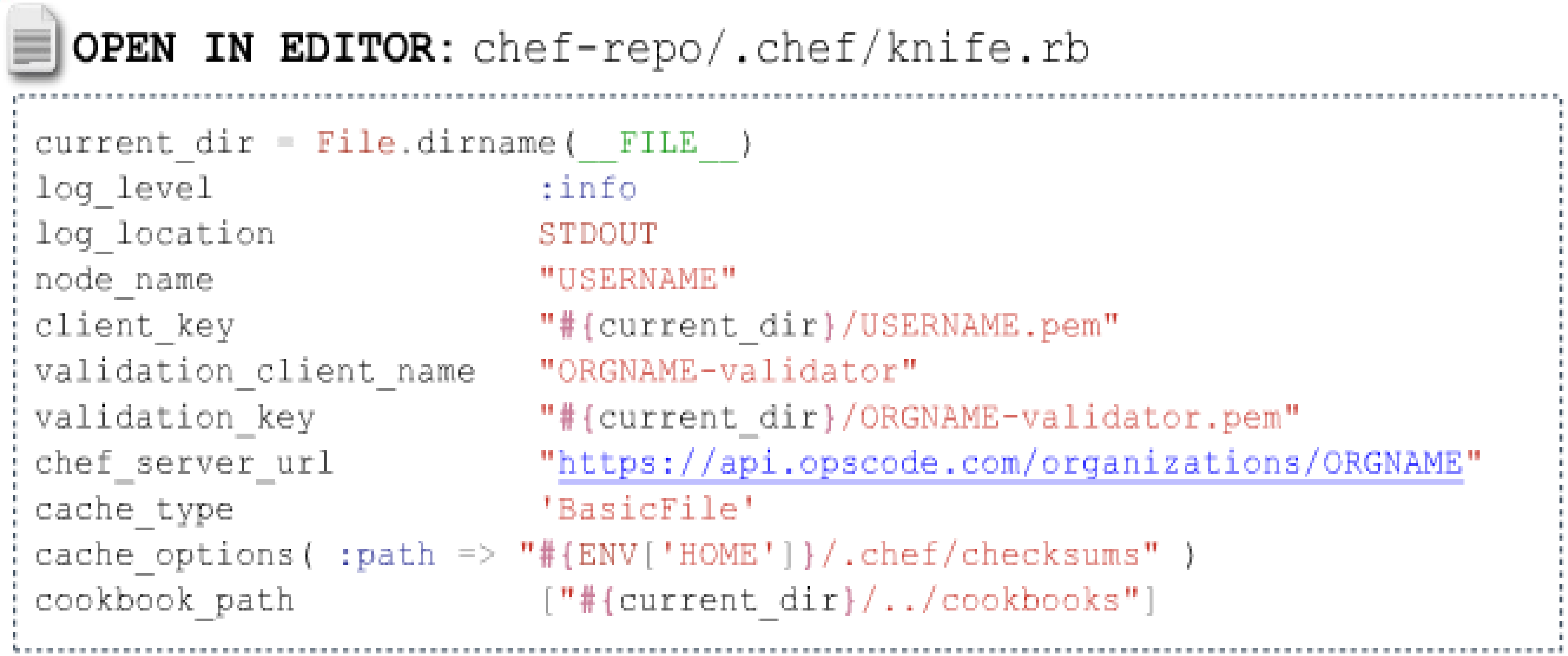
Knife

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

Knife.rb

- Default location
  - `~/.chef/knife.rb`
  - `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"]
```

Knife help list

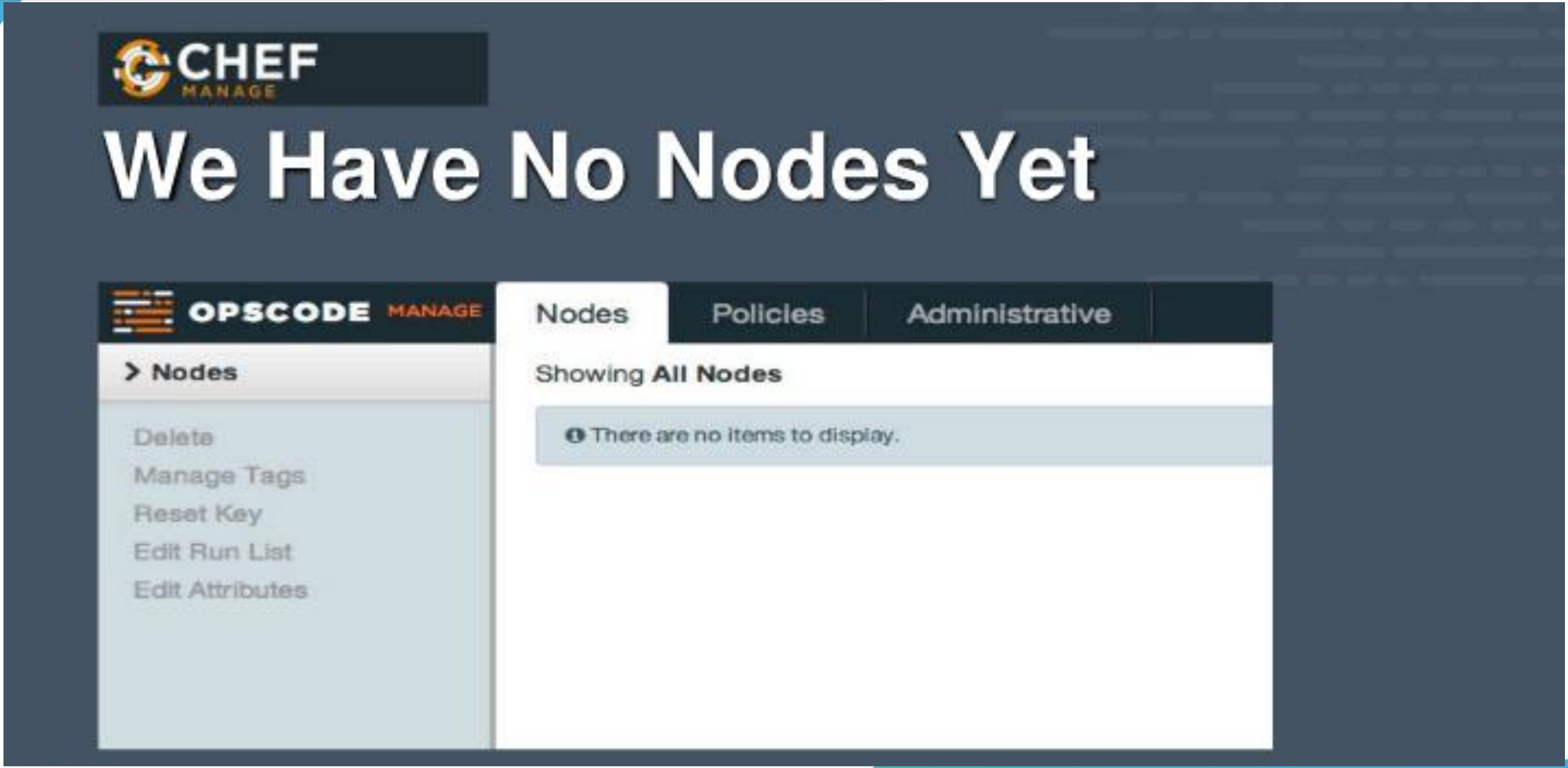
```
$ knife help list
```

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

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

Chef Manage


CHEF MANAGE

We Have No Nodes Yet

Showing **All Nodes**

There are no items to display.

Bootstrap a Node

```
knife bootstrap ADDRESS --ssh-user ubuntu --sudo --identity-file IDENTITY_FILE --node-name node1-ubuntu
```

```
knife bootstrap ADDRESS --ssh-user USER --ssh-password 'PASSWORD' --sudo --use-sudo-password --node-name node1-ubuntu --run-list 'recipe[learn_chef_apache2]'
```

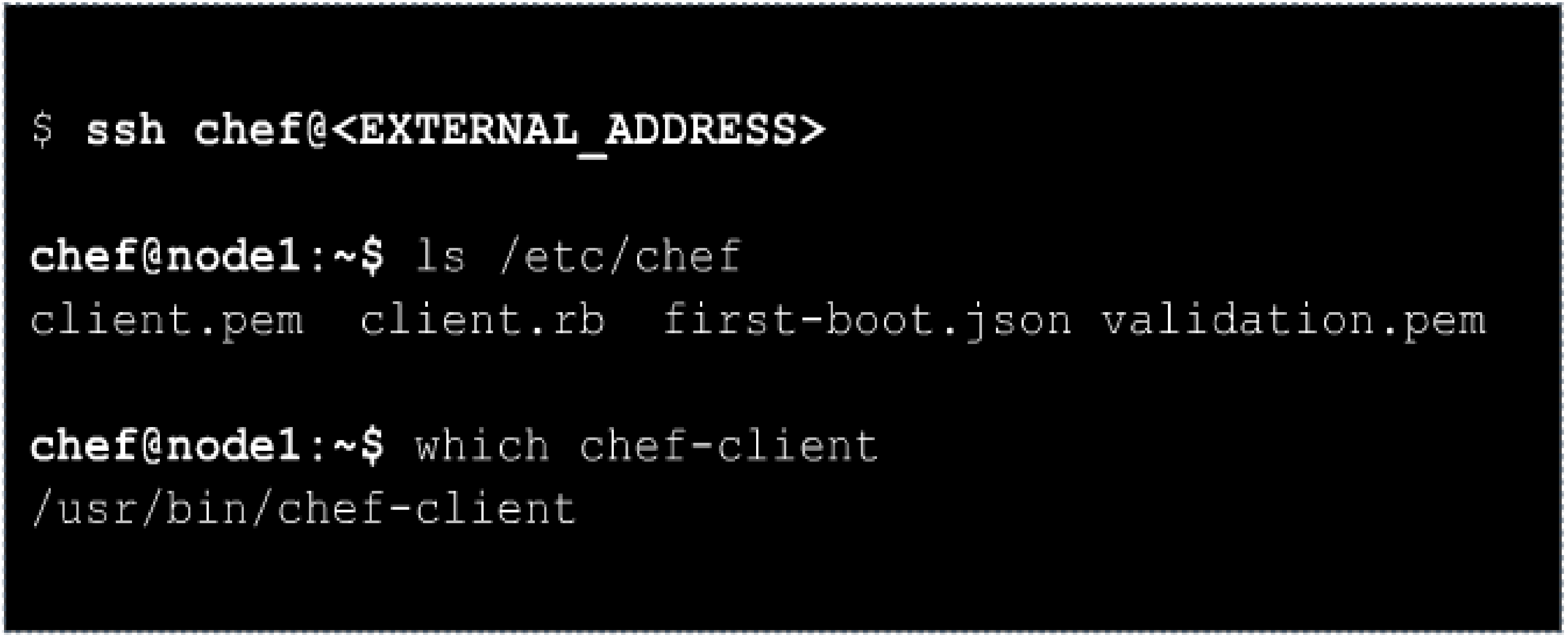
```
knife bootstrap localhost --ssh-port PORT --ssh-user vagrant --sudo --identity-file IDENTITY_FILE --node-name node1-ubuntu --run-list 'recipe[learn_chef_apache2]'
```

```
knife bootstrap windows winrm ADDRESS --winrm-user USER --winrm-password 'PASSWORD' --node-name node1-windows
```

Chef Installation on Node

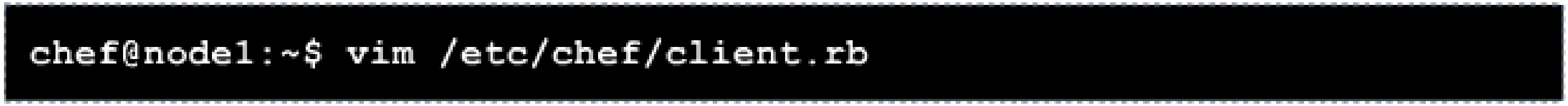
- Chef and all its dependencies are installed via an operating system specific package (omnibus installer)
- Installation includes
    - The Ruby language – used by Chef
    - Knife – Command line tool for administrators
    - Chef-client – Client application
    - Ohai – System profiler
    - … and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

Verifying Chef Installation on Node

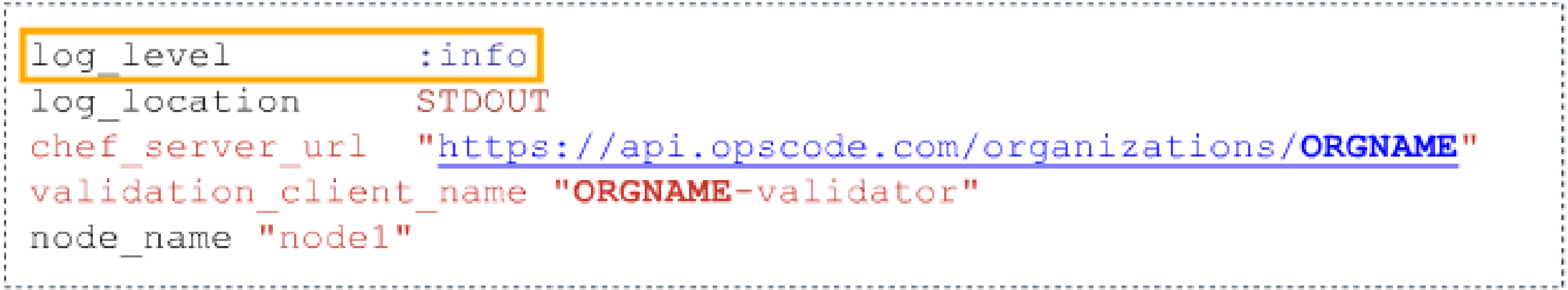


```
$ ssh chef@<EXTERNAL_ADDRESS>

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

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

Change Log Level on Your Node



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

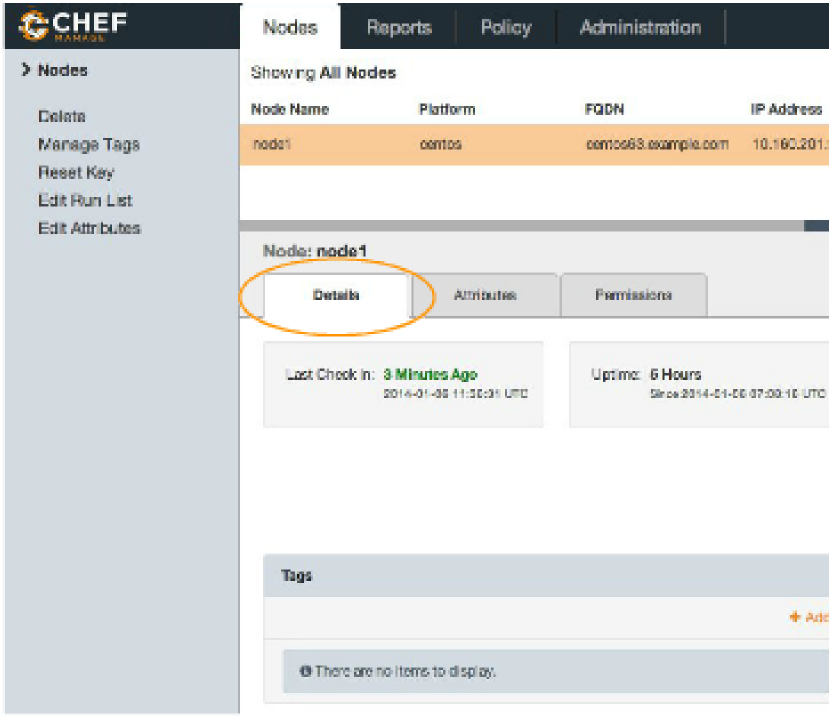


```
log_level              :info
log_location           STDOUT
chef_server_url        "https://api.opscode.com/organizations/ORGNAME"
validation_client_name "ORGNAME-validator"
node_name "node1"
```

- 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

- Click the 'Details' tab



Showing All Nodes

Node Name	Platform	FQDN	IP Address
node1	centos	centos63.example.com	10.160.201.

Node: node1

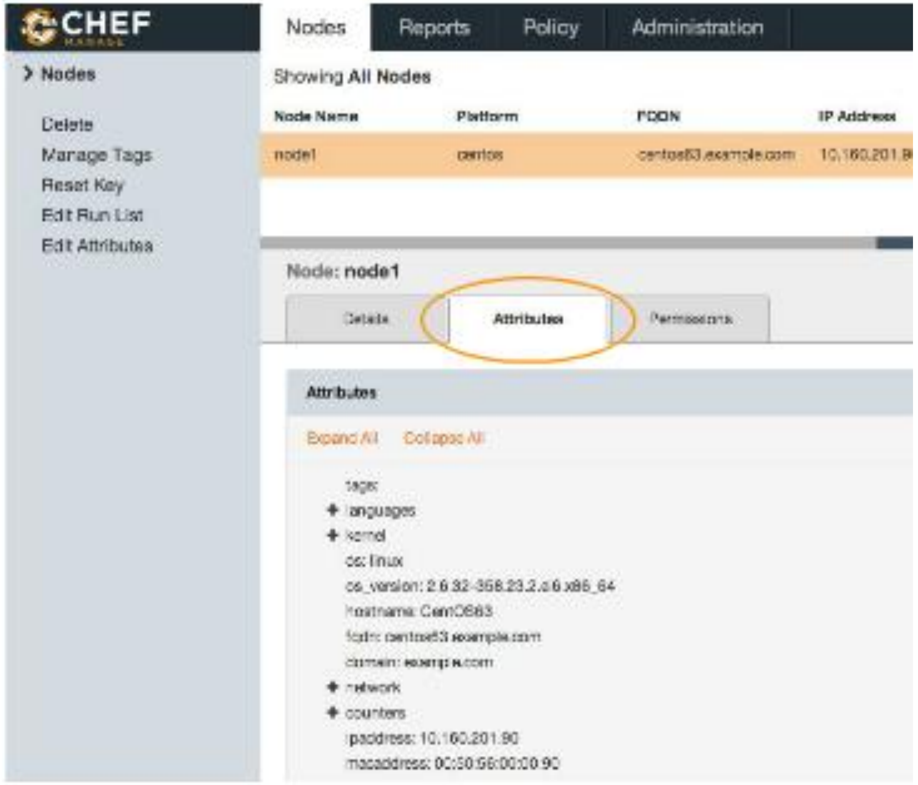
Last Check In: 3 Minutes Ago 2014-01-06 11:30:31 UTC

Uptime: 5 Hours Since 2014-01-06 07:00:16 UTC

There are no items to display.

View Node on Chef Server

- Click the 'Attributes' tab



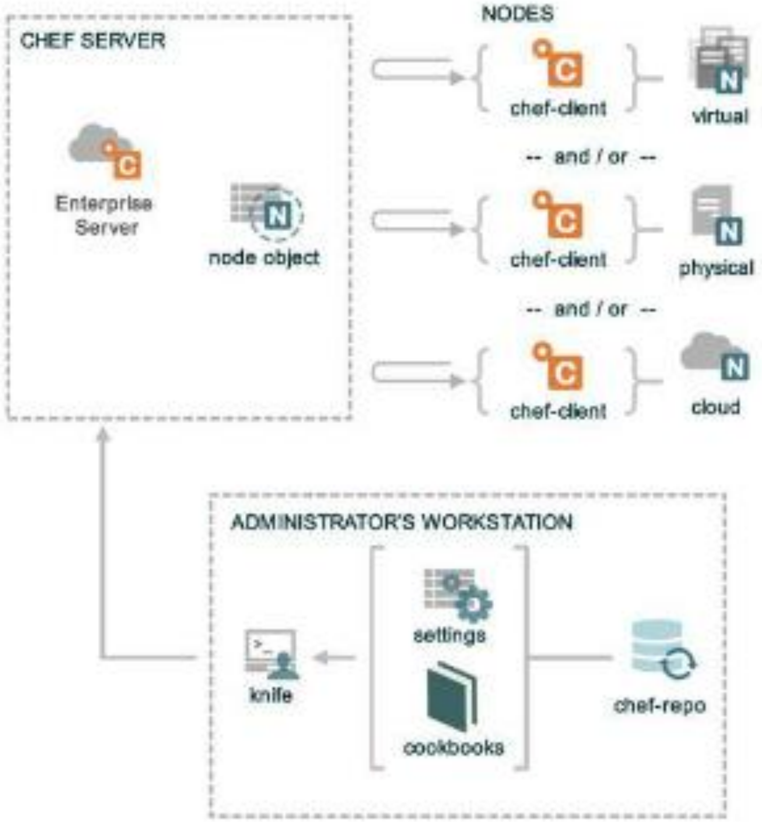
Node Name	Platform	FQDN	IP Address
node1	centos	centos63.example.com	10.160.201.9

```
  tags:
+ languages
+ kernel
  os: linux
  os_version: 2.6.32-358.23.2.el6.x86_64
  hostname: CentOS63
  fqdn: centos63.example.com
  domain: example.com
+ network
+ counters
  ipaddress: 10.160.201.90
  macaddress: 00:50:56:00:00:90
```

Node

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

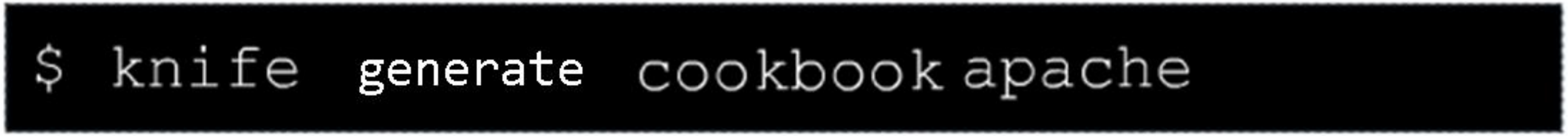
Chef Setup So Far



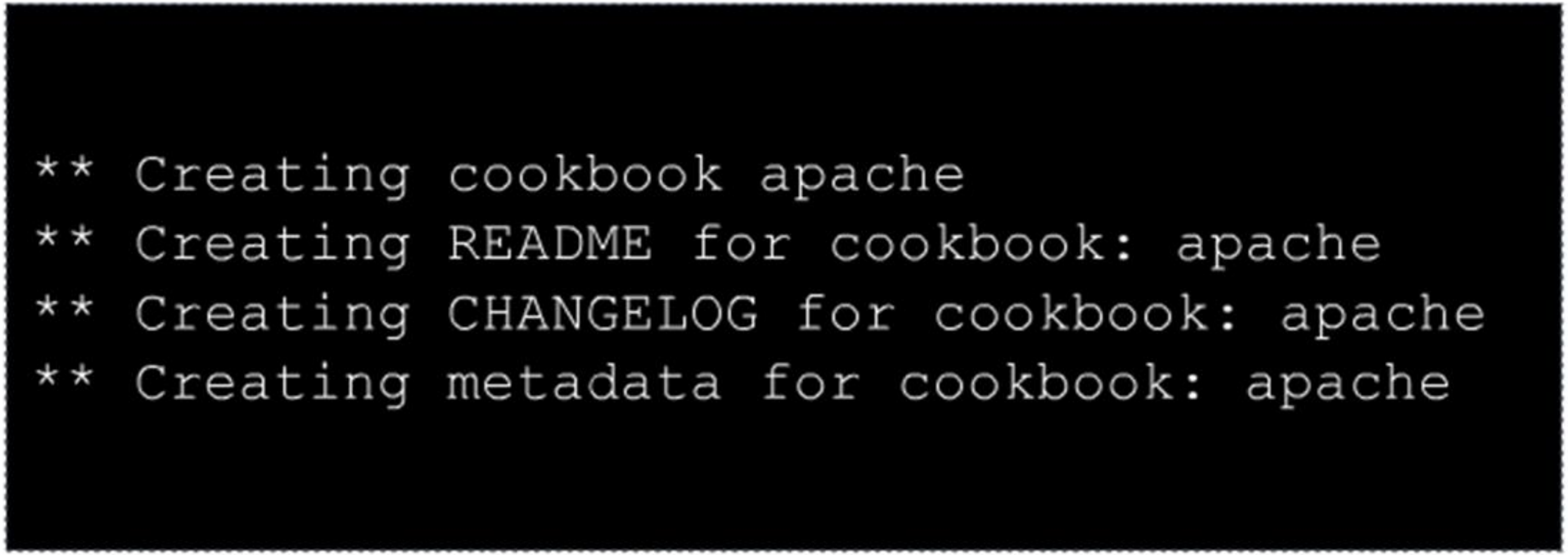
Problem Statement

- 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

Create a New Cookbook



```
$ knife generate cookbook 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 Default Receipe

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

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

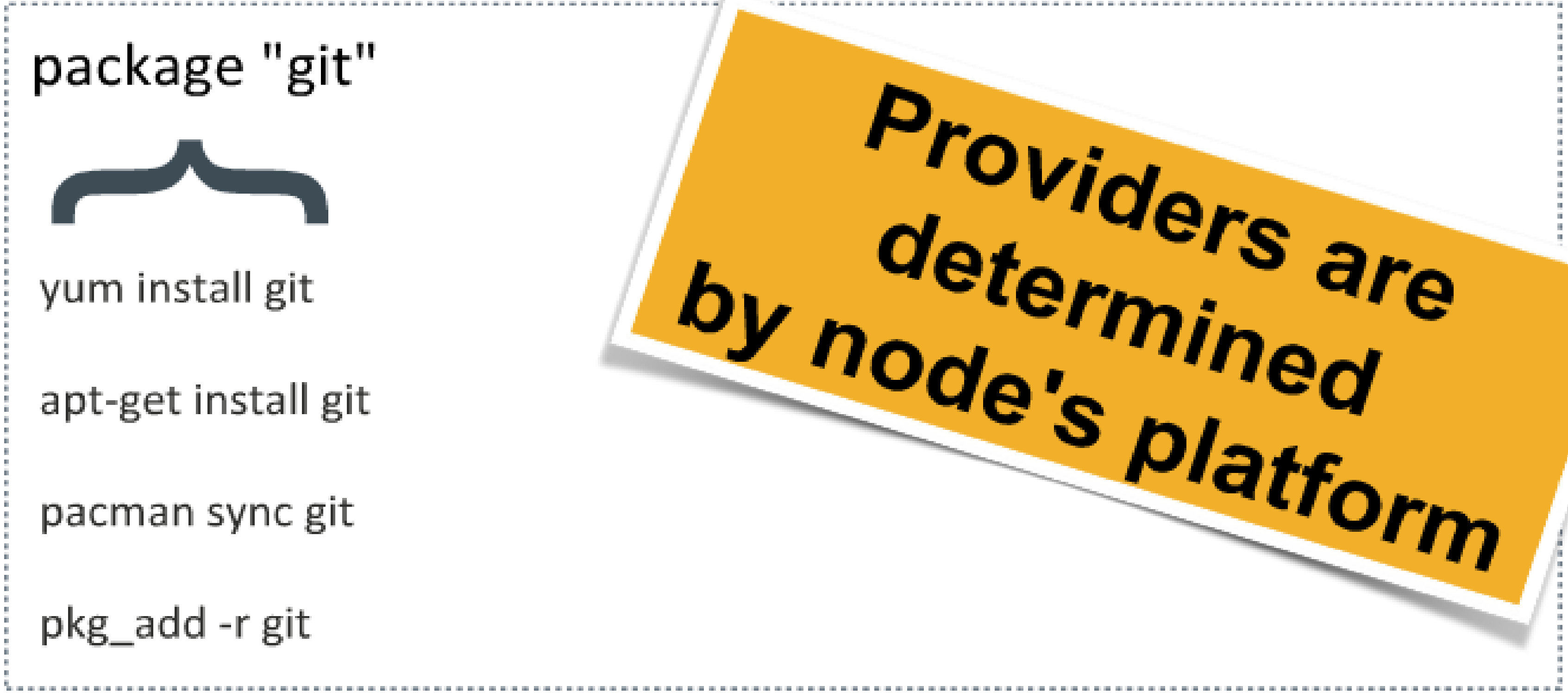
- Is a package resource
- Whose name is *httpd*
- With an install **action**

```
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 Resources



package "git"

yum install git

apt-get install git

pacman sync git

pkg_add -r git

Providers are determined by node's platform

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!

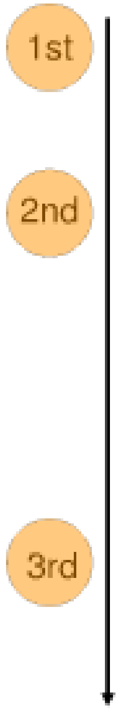
Resource We Just Wrote

- 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
```

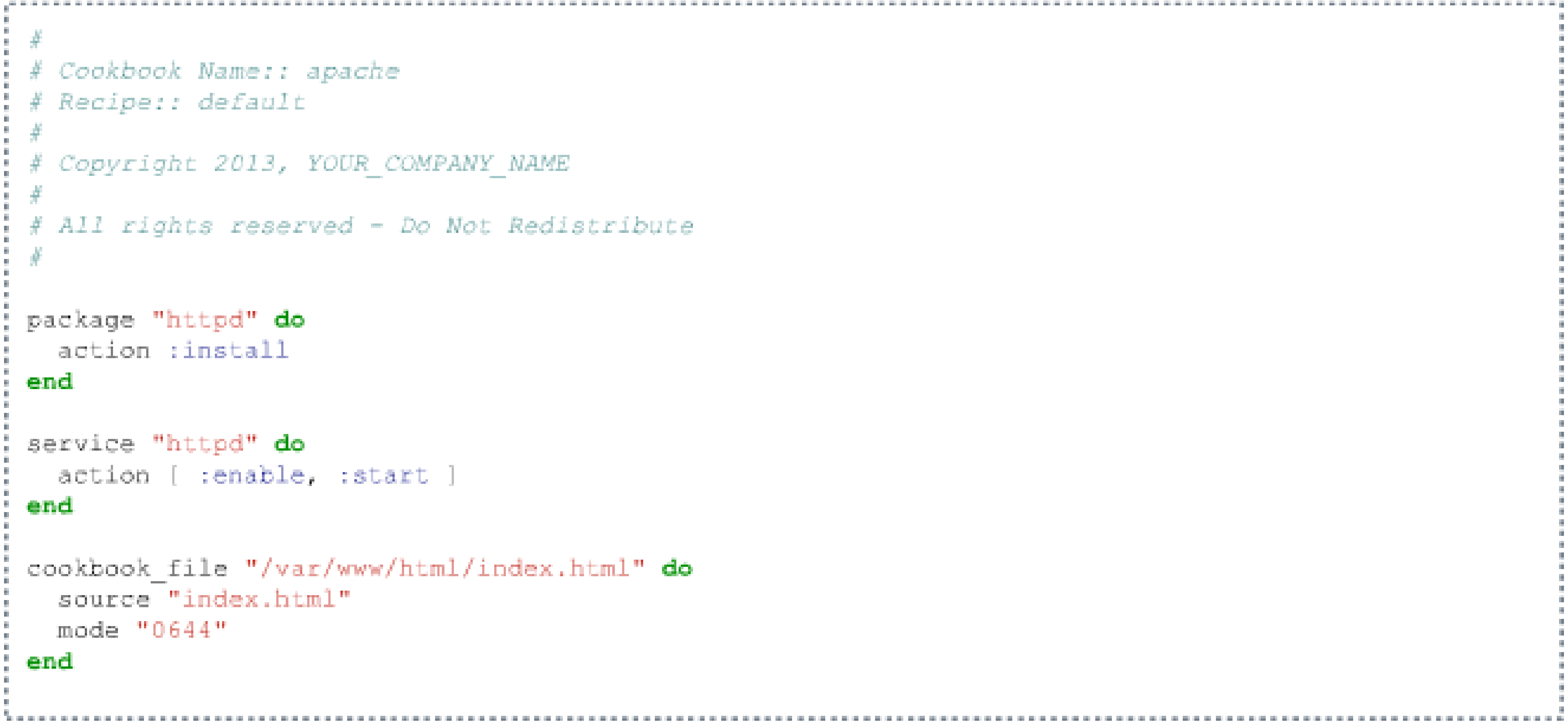
SAVE FILE!

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 Receipe



```
#
# 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
```

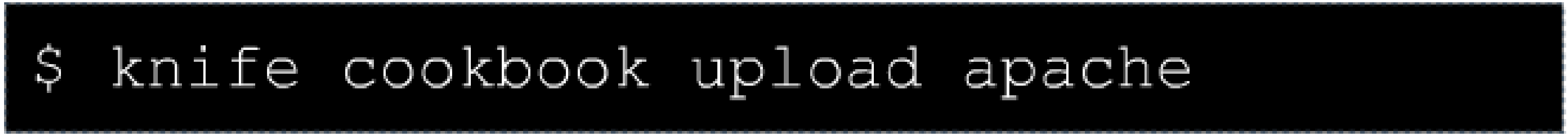
Add Index.html to Your Cookbook

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

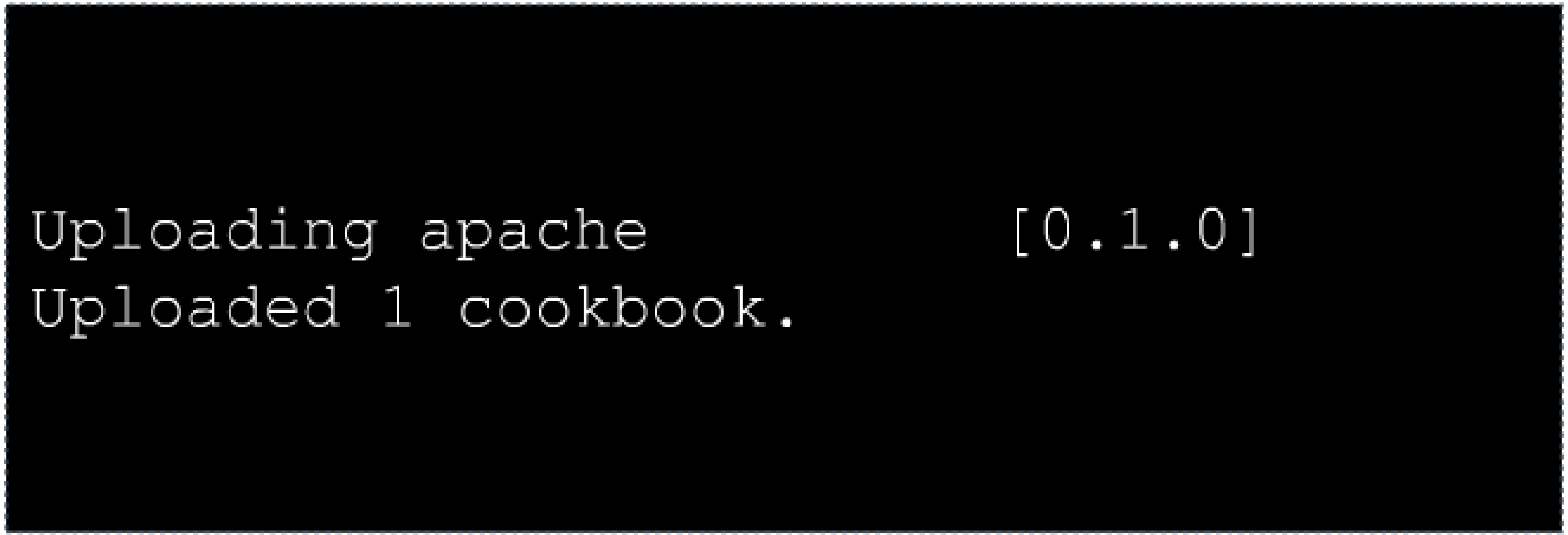
```
<html>
<body>
  <h1>Hello, world!</h1>
</body>
</html>
```

SAVE FILE!

Upload Cookbook to Chef Server



```
$ knife cookbook upload apache
```



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

The Run List

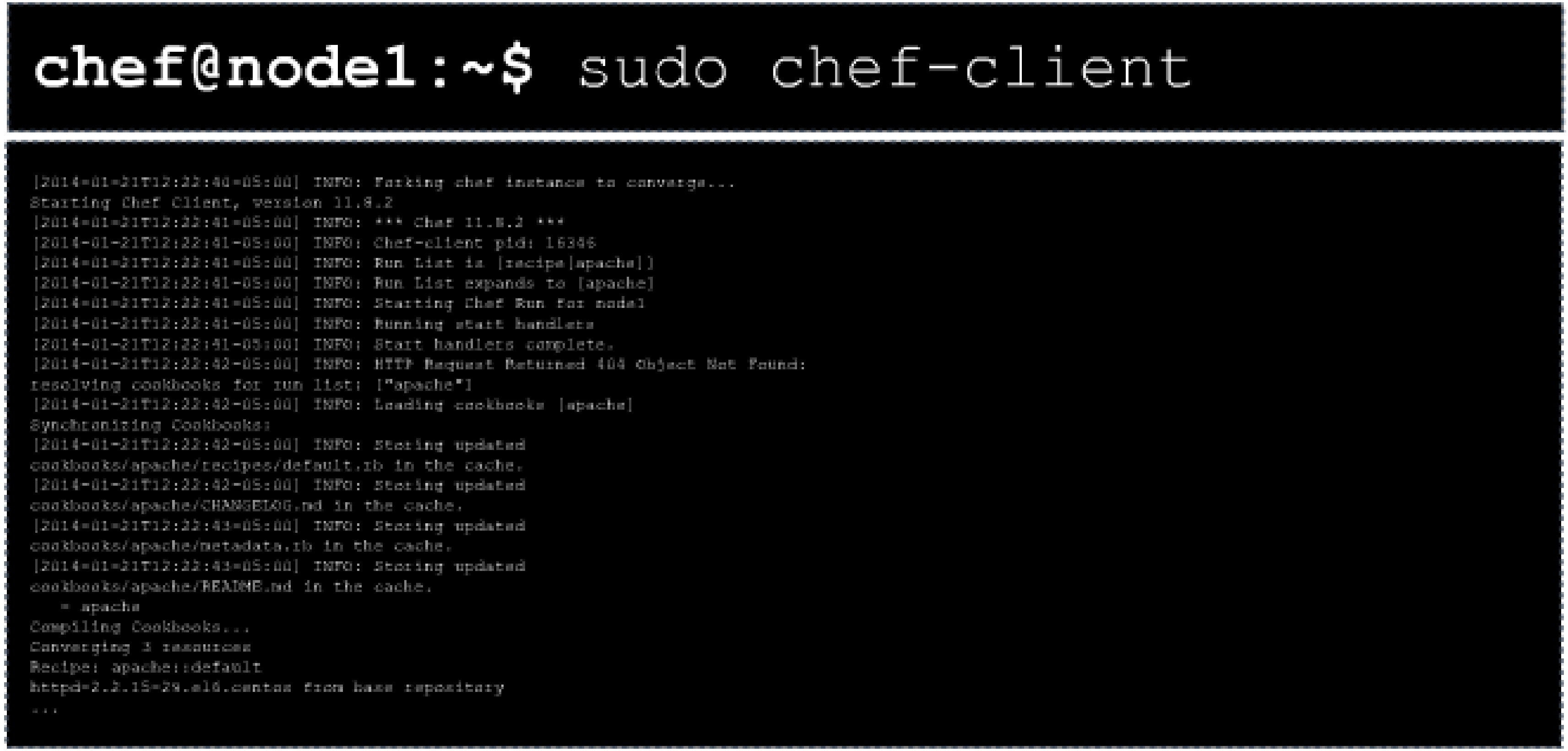
- The Run List is the ordered set of recipes and roles that the Chef-client will execute on a node
  - Recipes are specified by “recipe[name]”
  - Roles are specified by “role[name]”

Add apache recipe to test node's run list

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

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

Run Chef-client From Node

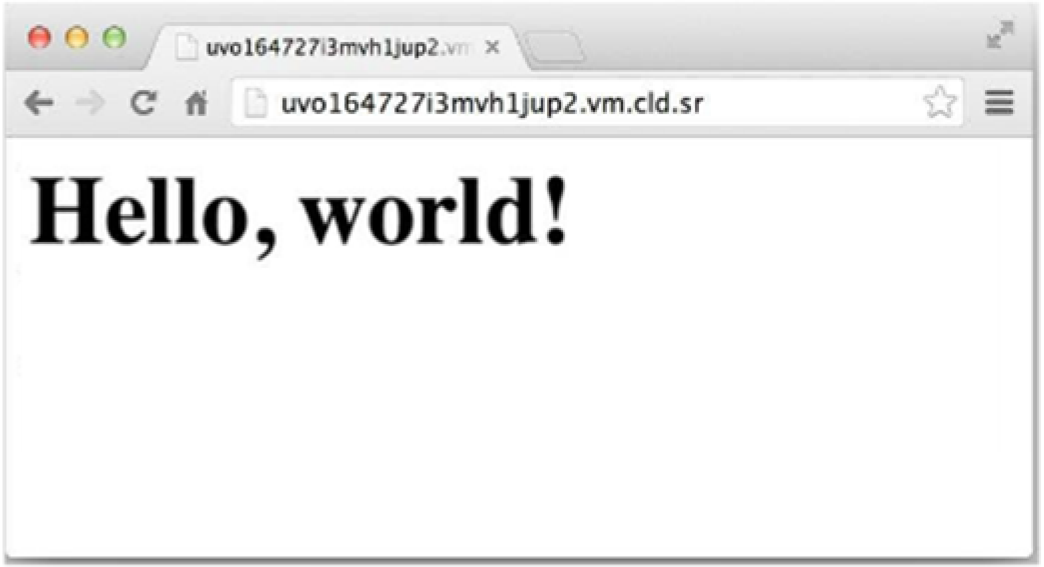


```
chef@node1:~$ sudo chef-client
```

```
[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
...
```

Verify the Home Page Works On Node

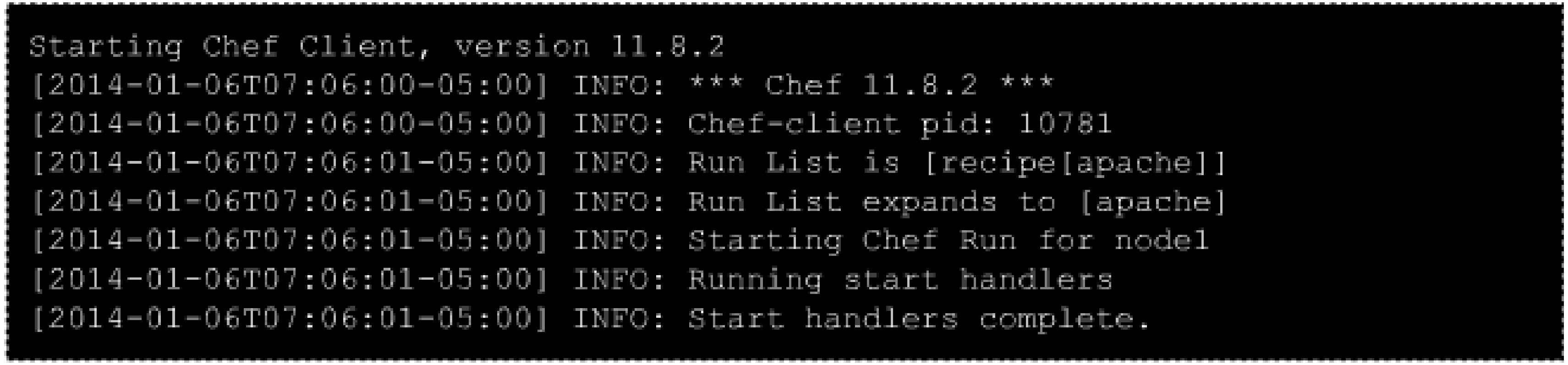
- Open a browser on your node
- Type localhost:8080
- You should see the apache home page



uvo164727i3mvh1jup2.vm.cld.sr

Hello, world!

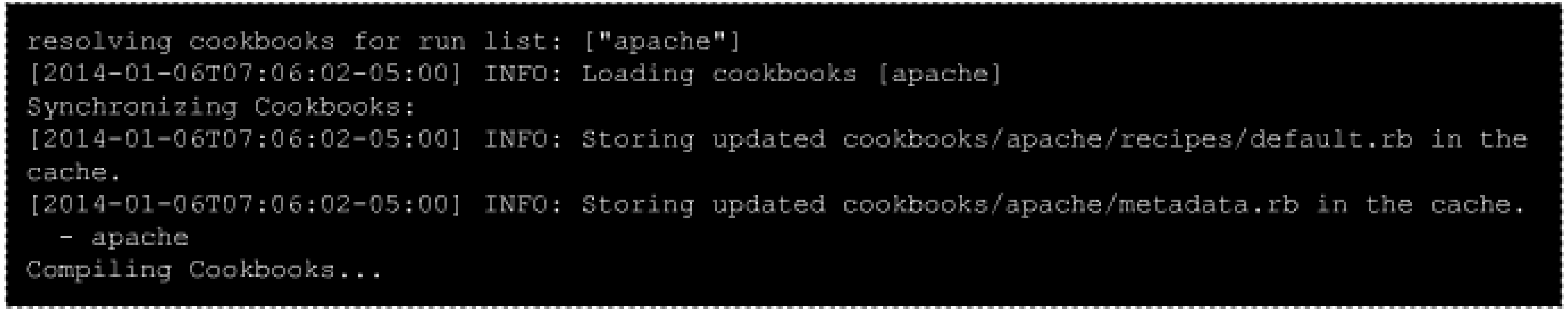
Reading the Output of a Chef-client Run



```
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

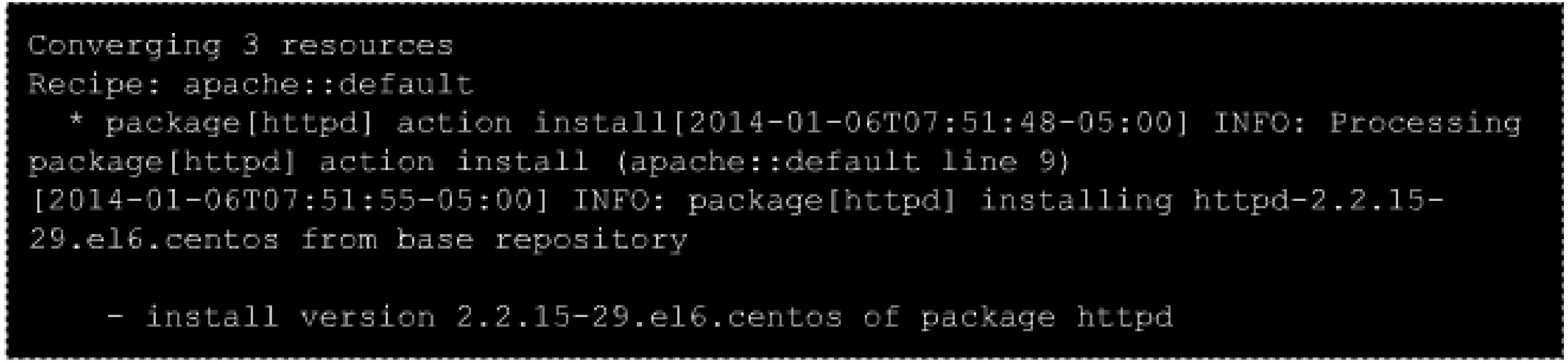
Reading the Output of a Chef-client Run



```
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

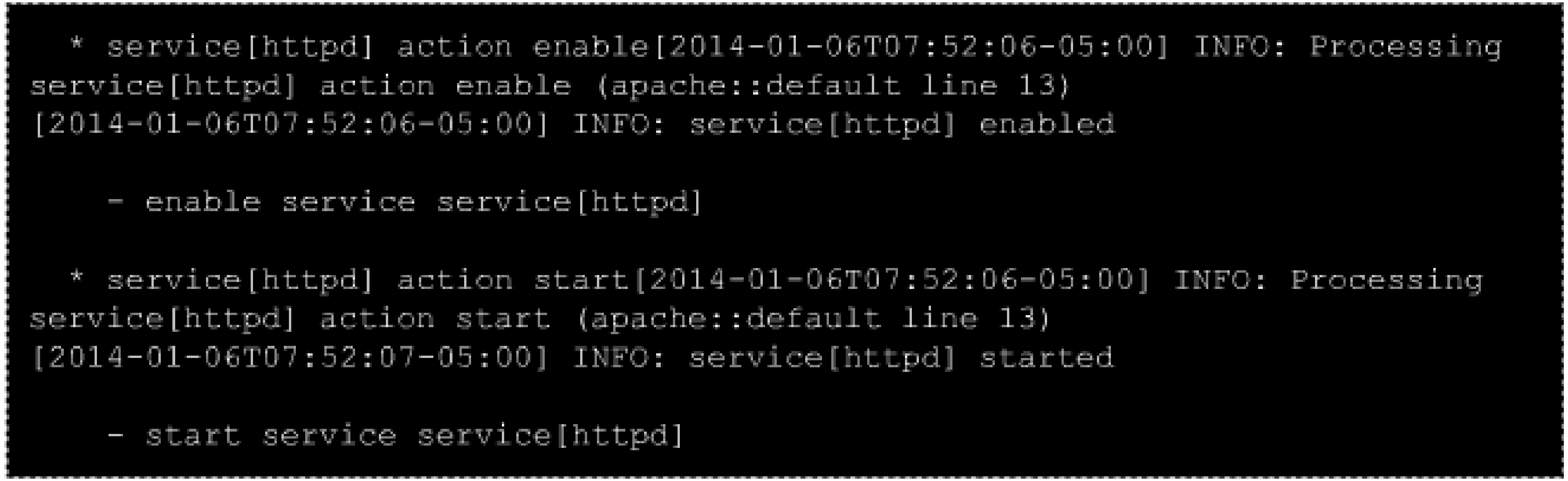
Reading the Output of a Chef-client Run



```
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

Reading the Output of a Chef-client Run



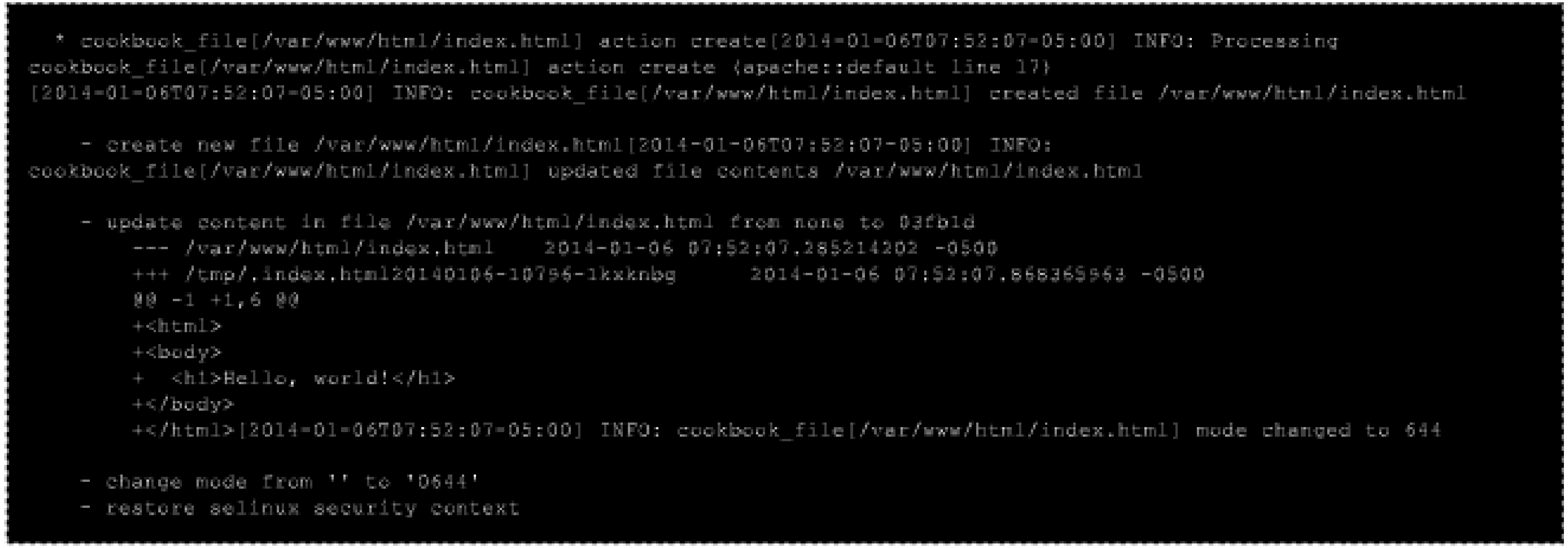
```
  * service[httpd] action enable[2014-01-06T07:52:06-05:00] INFO: Processing
service[httpd] action enable (apache::default line 13)
[2014-01-06T07:52:06-05:00] INFO: service[httpd] enabled

    - 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**

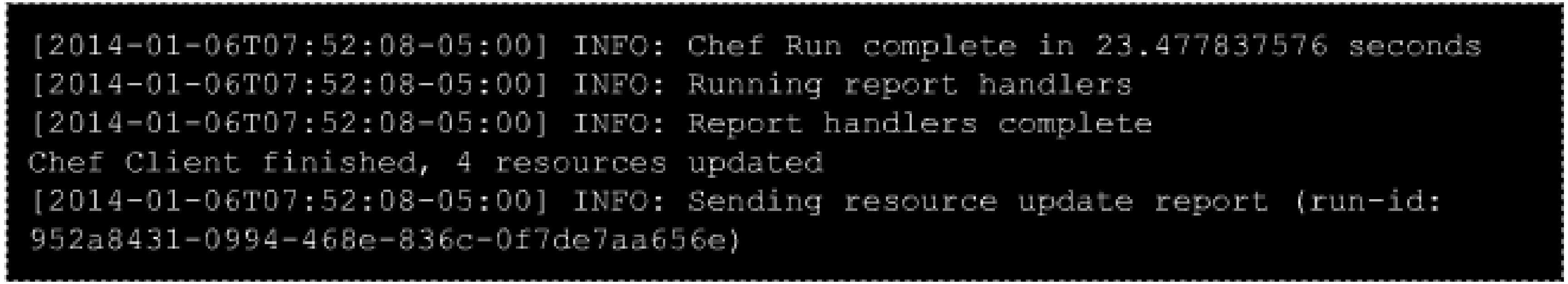
Reading the Output of a Chef-client Run



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

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

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

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

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

Reading the Output of a Chef-client Run



```
[2014-01-06T07:52:08-05:00] 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)
```

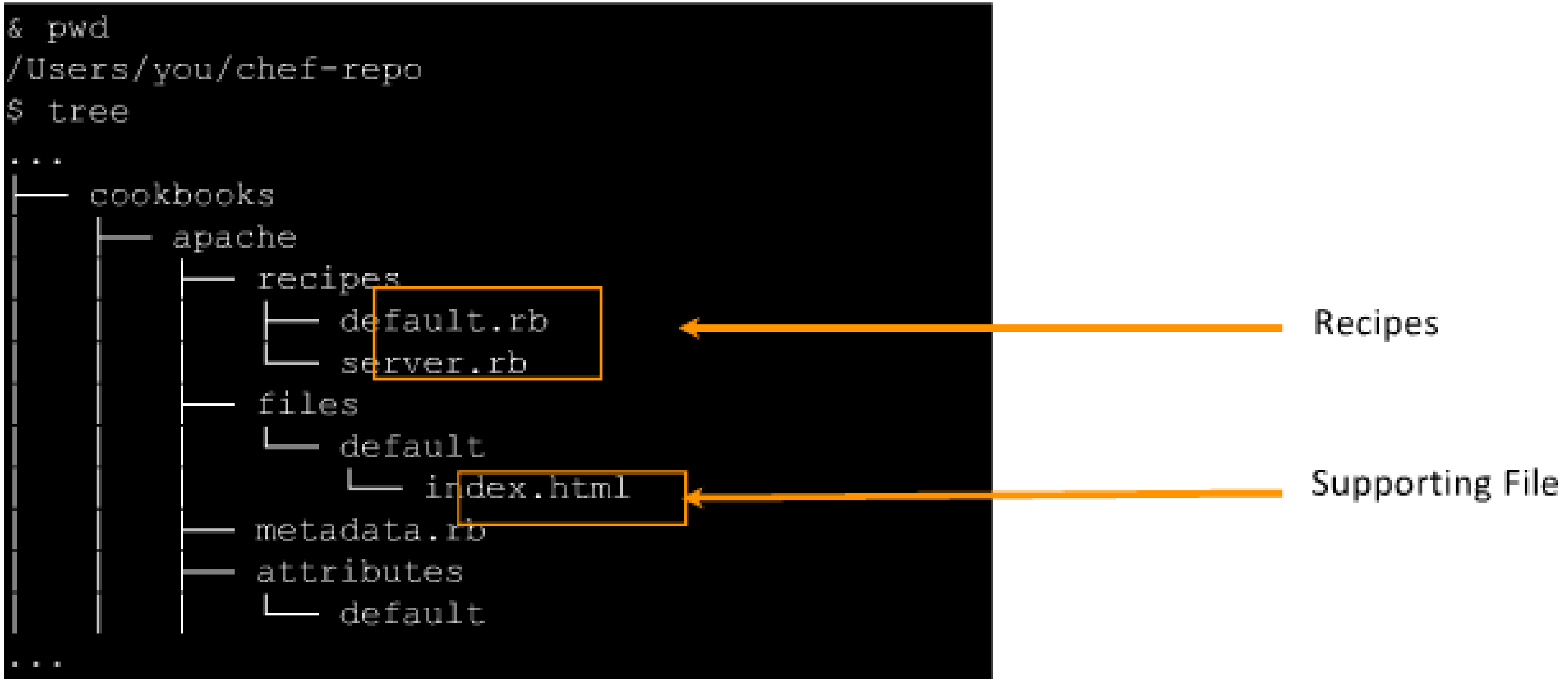
- Notice that a complete Chef-Run displays:
  - The time the client took to complete convergence
  - Status of report and exception handlers

Recap

recipe[apache::default]

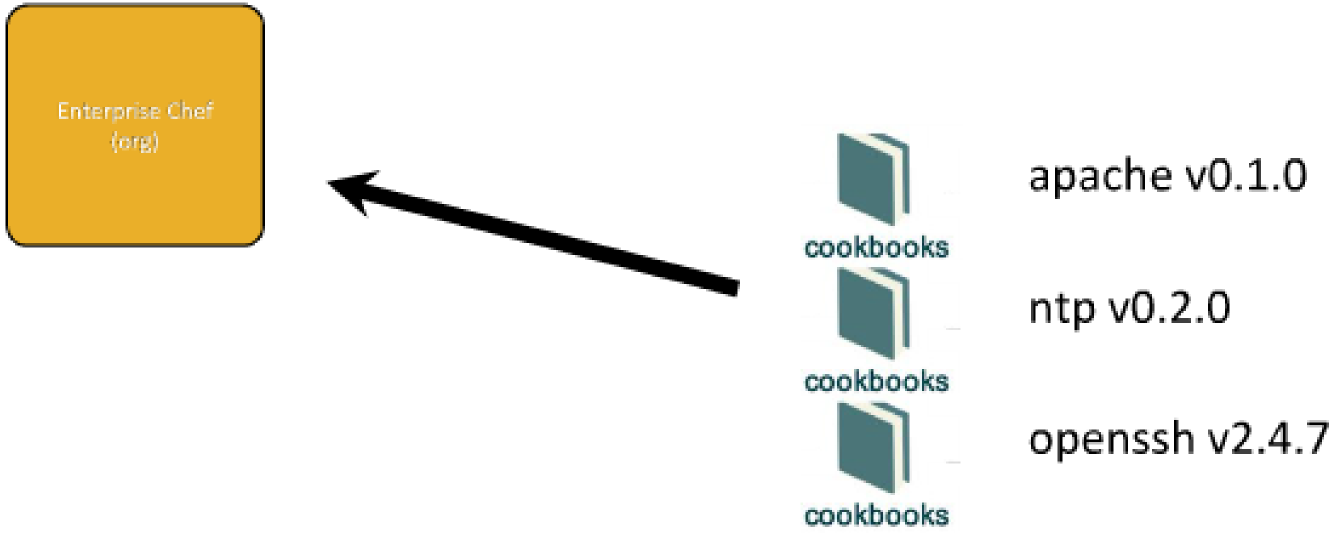
```
package "httpd" do
  action :install
end

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

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

Cookbooks contain Recipes & Supporting Files

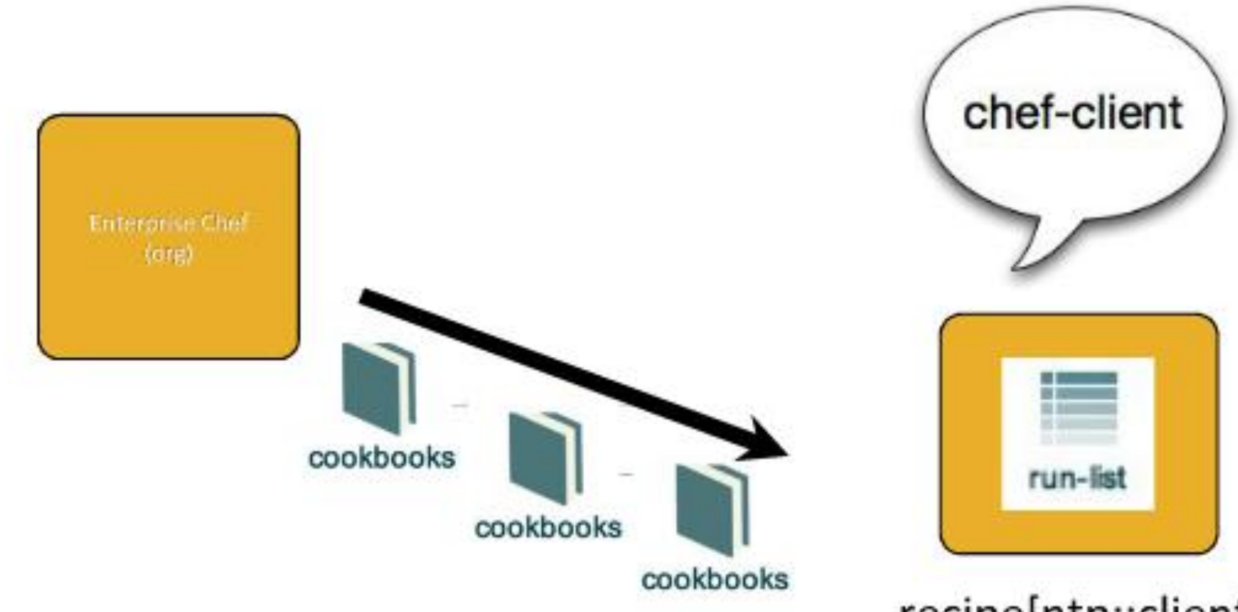


```
& pwd
/Users/you/chef-repo
$ tree
...
├── cookbooks
│   ├── apache
│   │   ├── recipes
│   │   │   ├── default.rb
│   │   │   └── server.rb
│   │   ├── files
│   │   │   └── default
│   │   │       └── index.html
│   │   ├── metadata.rb
│   │   ├── attributes
│   │   │   └── default
...
```

Cookbooks are Installed as Artifacts' on Chef Server



Nodes have run_lists made of Recipes

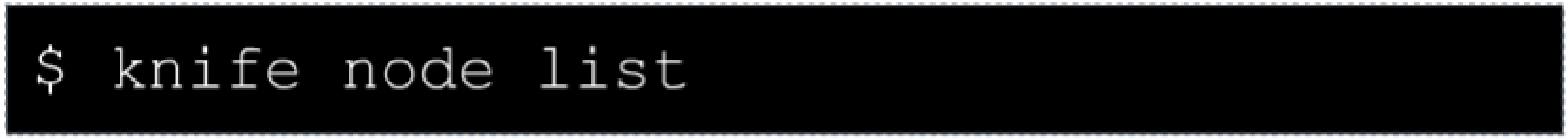


recipe[ntp::client]
recipe[openssh::server]
recipe[apache::server]

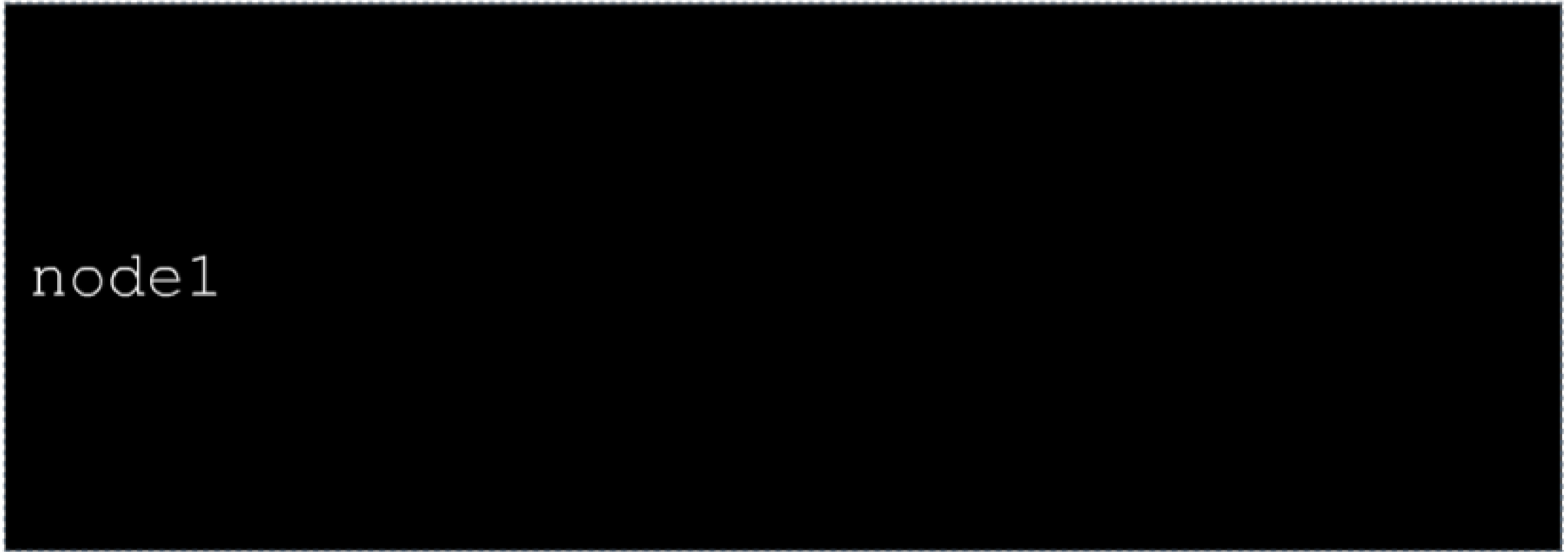
Chef Environments

- An environment is a way to map an organization’s real-life workflow to what can be configured and managed when using Chef server
- Every organization begins with a single environment called the _default environment, which cannot be modified (or deleted)
- Additional environments can be created to reflect each organization’s patterns and workflow. For example, creating production, staging, testing, and development environments
- Generally, an environment is also associated with one (or more) cookbook versions

List Nodes



```
$ knife node list
```



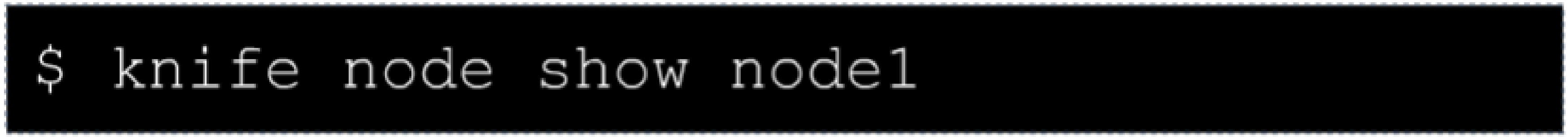
```
node1
```

List Clients

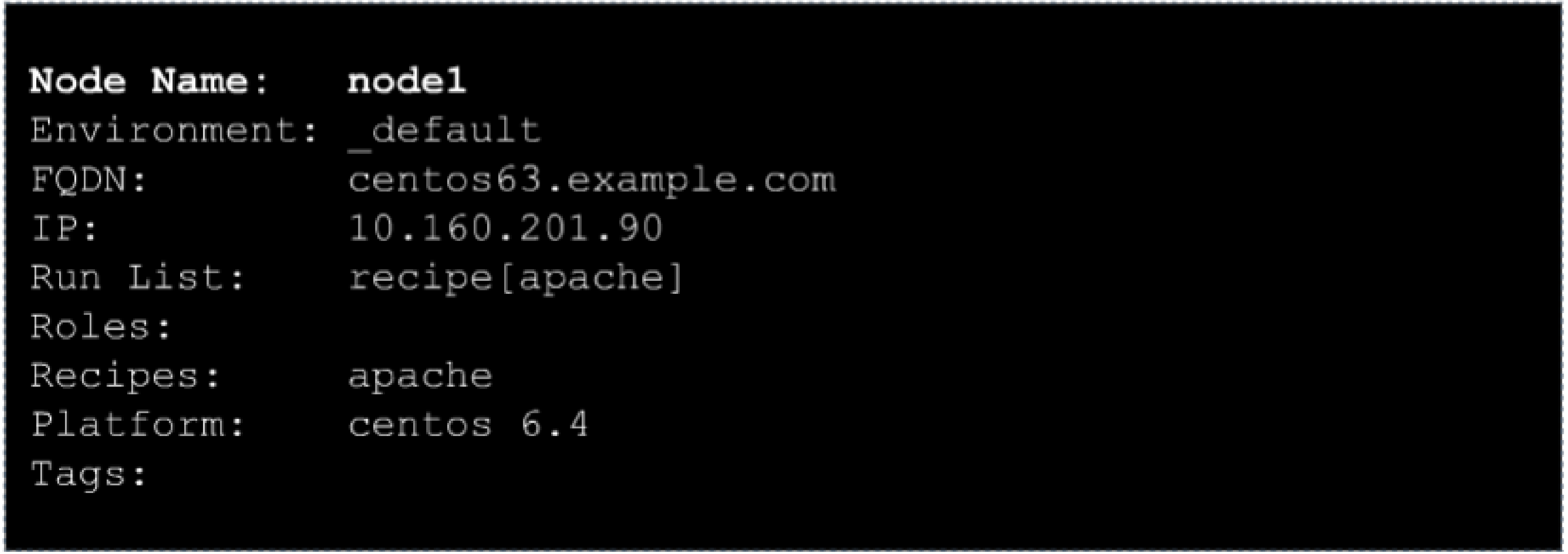
```
$ knife client list
```

```
ORGNAME-validator
node1
```

Show Node Details

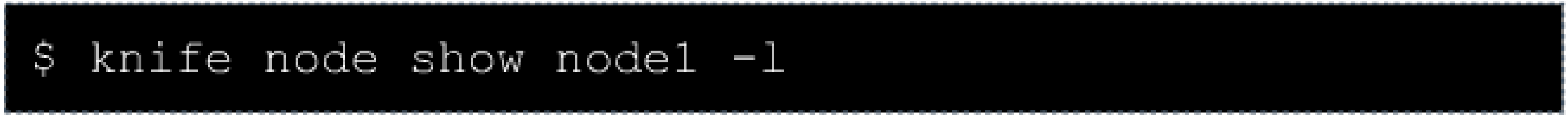


```
$ knife node show node1
```

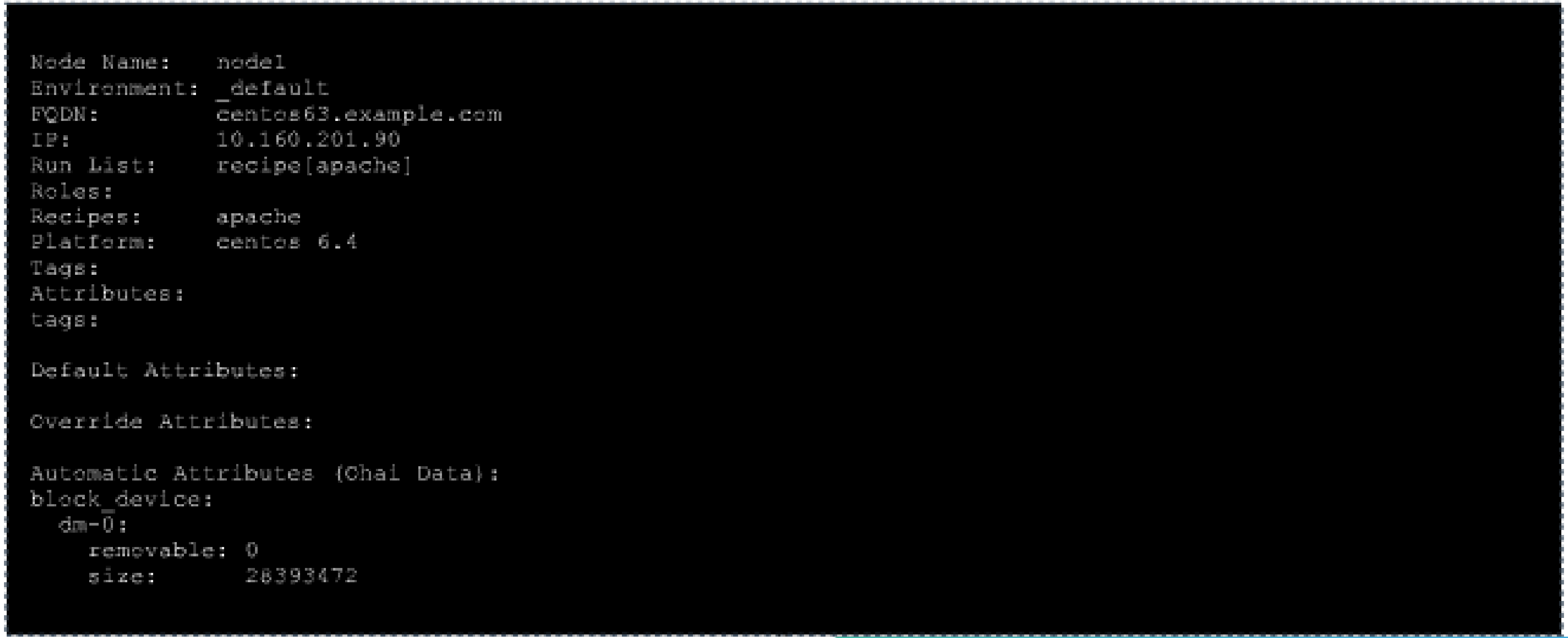


```
Node Name:   node1
Environment: _default
FQDN:        centos63.example.com
IP:          10.160.201.90
Run List:    recipe[apache]
Roles:
Recipes:     apache
Platform:    centos 6.4
Tags:
```

Show all Node Attributes

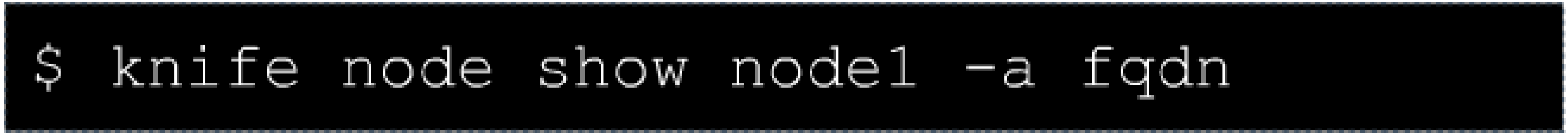


```
$ knife node show node1 -l
```

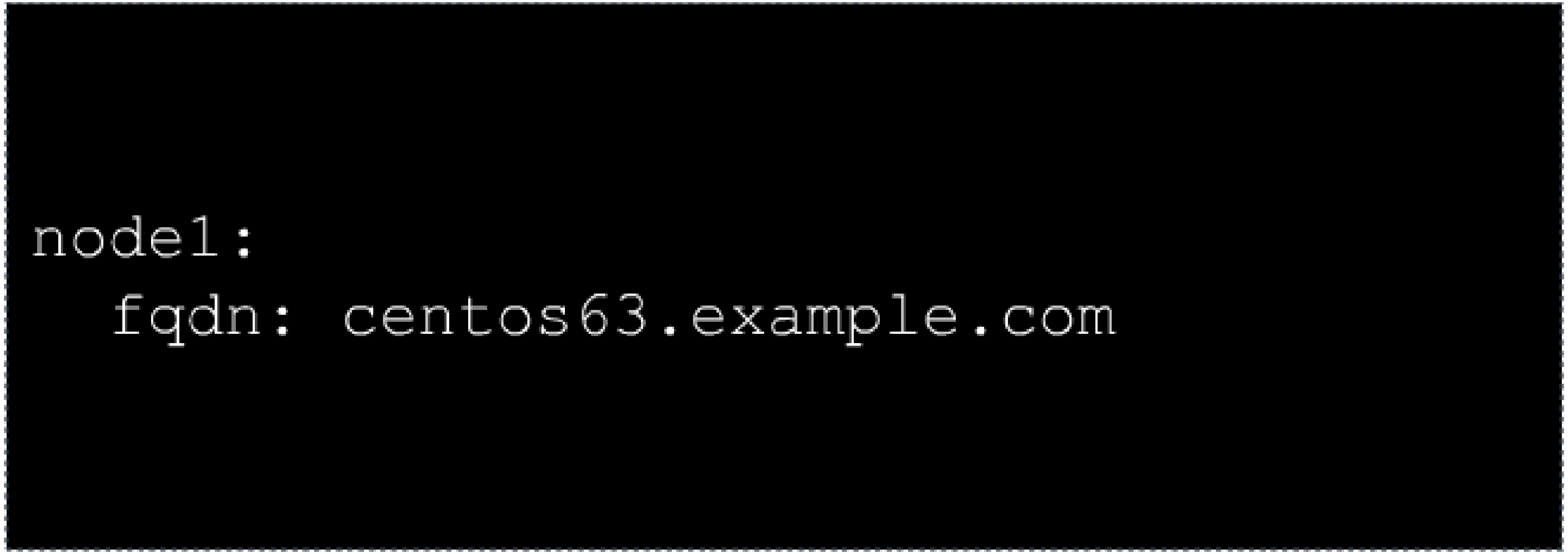


```
Node Name:   node1
Environment: _default
FQDN:        centos63.example.com
IP:          10.160.201.90
Run List:    recipe[apache]
Roles:
Recipes:     apache
Platform:    centos 6.4
Tags:
Attributes:
tags:

Default Attributes:

Override Attributes:

Automatic Attributes (Chai Data):
block_device:
  dm-0:
    removable: 0
    size:      28393472
```

Show only FQDN Attributes

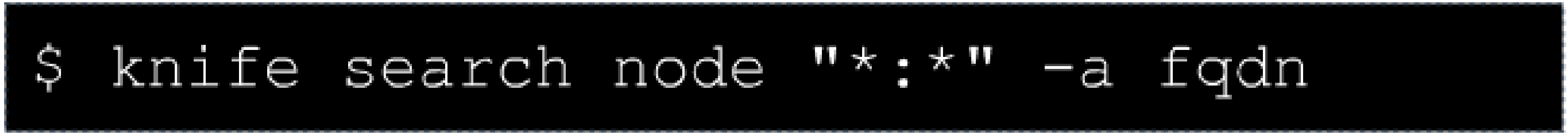


```
$ knife node show node1 -a fqdn
```

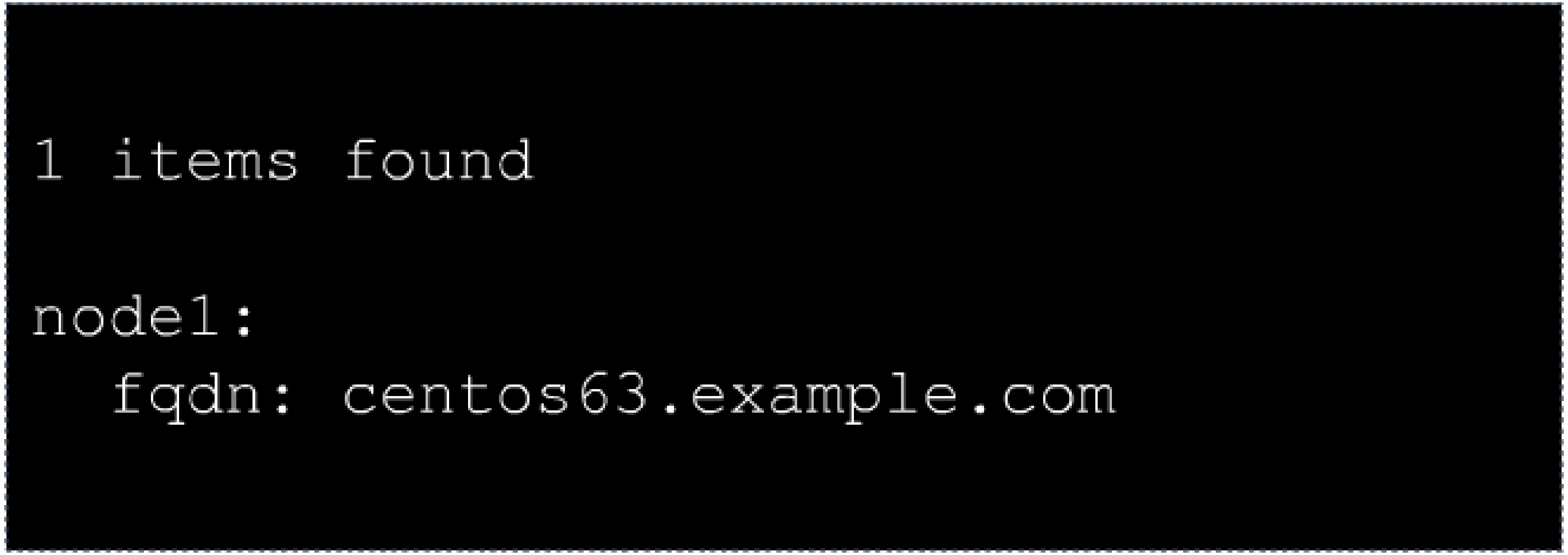


```
node1:
  fqdn: centos63.example.com
```

Use Search to Find the same Data



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



```
1 items found

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

Chef 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

Cognixia

A Collabera LEARNING SOLUTIONS COMPANY

THANK YOU

www.cognixia.com