Set Up a Node & Write a Cookbook

Chef Fundamentals Webinar Series

training@opscode.com

OPSCODE™

Nathen Harvey

- Technical Community Manager at Opscode
- Co-host of the Food Fight Show Podcast

- @nathenharvey

- nharvey@opscode.com

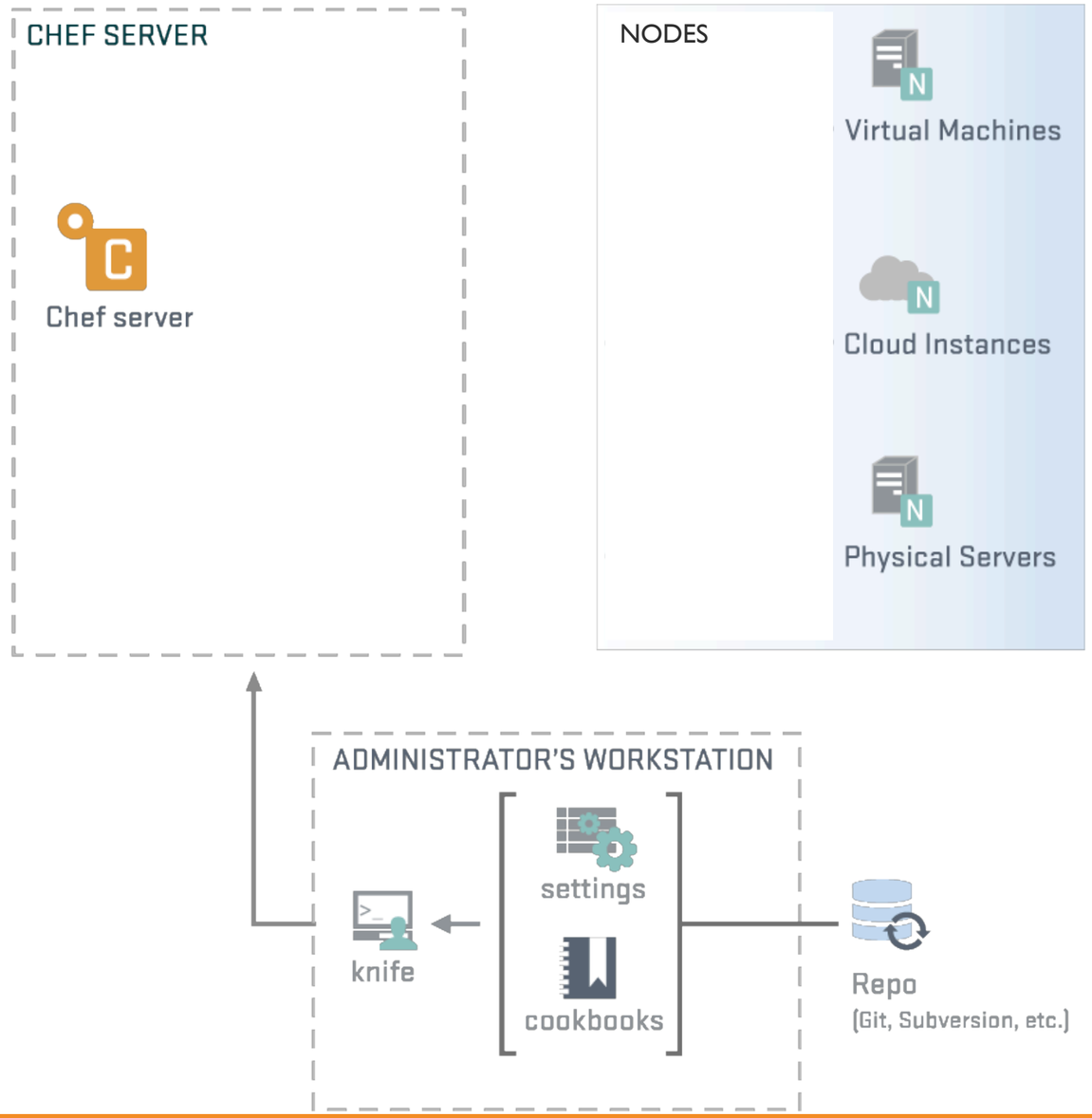
Node Setup

Setup a Node to manage

Lesson Objectives

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

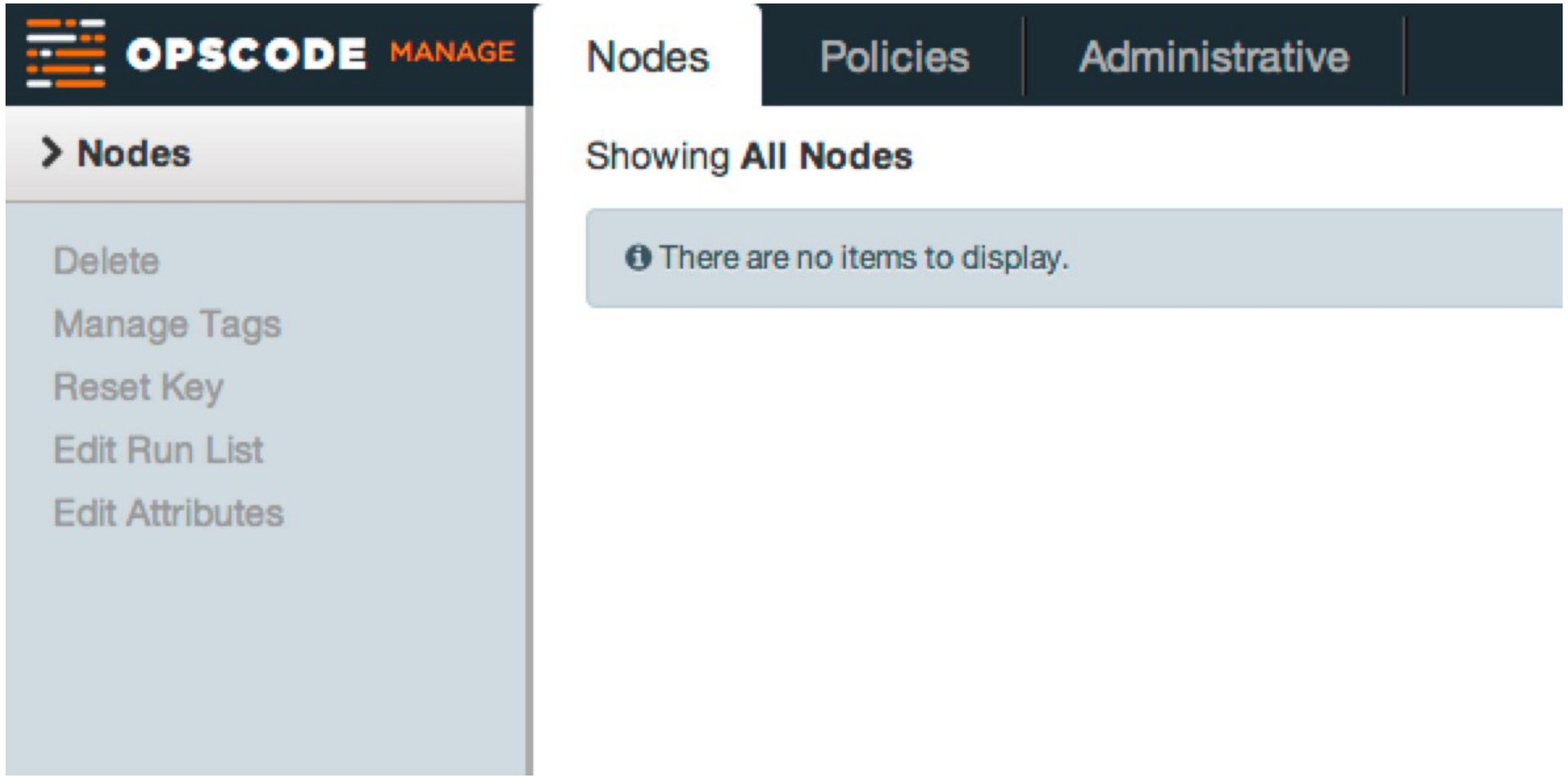
Nodes


CHEF SERVER
Chef server
NODES
Virtual Machines
Cloud Instances
Physical Servers
ADMINISTRATOR'S WORKSTATION
knife
settings
cookbooks
Repo
(Git, Subversion, etc.)

Nodes

- Nodes represent the servers in your infrastructure these may be
  - Physical or virtual servers
  - Hardware that you own
  - Compute instances in a public or private cloud

We Have No Nodes Yet



Training Node

- The labs require a node to be managed
- We allow for four different options
  - Bring your own Node
  - Use Vagrant from the Starter Kit
  - Launch an instance of a public AMI on EC2
  - Use the Chef Fundamentals training lab

learnchef.com

Welcome to #learnchef

#learnchef is a broad collection of resources specifically designed to help you learn C

Chef Fundamentals Webinar Series

Join our free weekly webinar series aimed at guiding you down the #learnchef path.

This series will prepare key development, engineering, and operations staff to use Ch
units in the course has hands on exercises to reinforce the material. You will learn Che
the end of the class, you will have a code repository that can be used and modified t
business problems.

Join Opscode Community Manager Nathen Harvey as he covers:

- Week 1 - Overview of Chef
- Week 3 - Setup a Node & Write Your First Cookbook (31-Oct-2013)
- Week 5 - Working with Templates (14-Nov-2013)
- Week 6 - Roles, Community Cookbooks, & Further Resources (21-Nov-2013)

Training Lab Requirements

Training Lab Requirements:

You will need a server or virtual machine in order to complete the training lab exercises. There are four different options that you may use.

To participate in the labs for module 3, you will need the following

- Hostname or IP Address of your training lab server
- SSH Username
- SSH Password
- SSH Port (default is 22)

Watch th Chef Fundamentals Webinar - Training Lab Setup video or more information on setting up your trai

BETA Chef Training Lab

- Login to the Lab
  - https://use.cloudshare.com/
- Make sure your environment is ready
- Runtime: 24 Hours
- Auto-suspend after: 1 Hour
- Storage time: 7 Days

Your Node

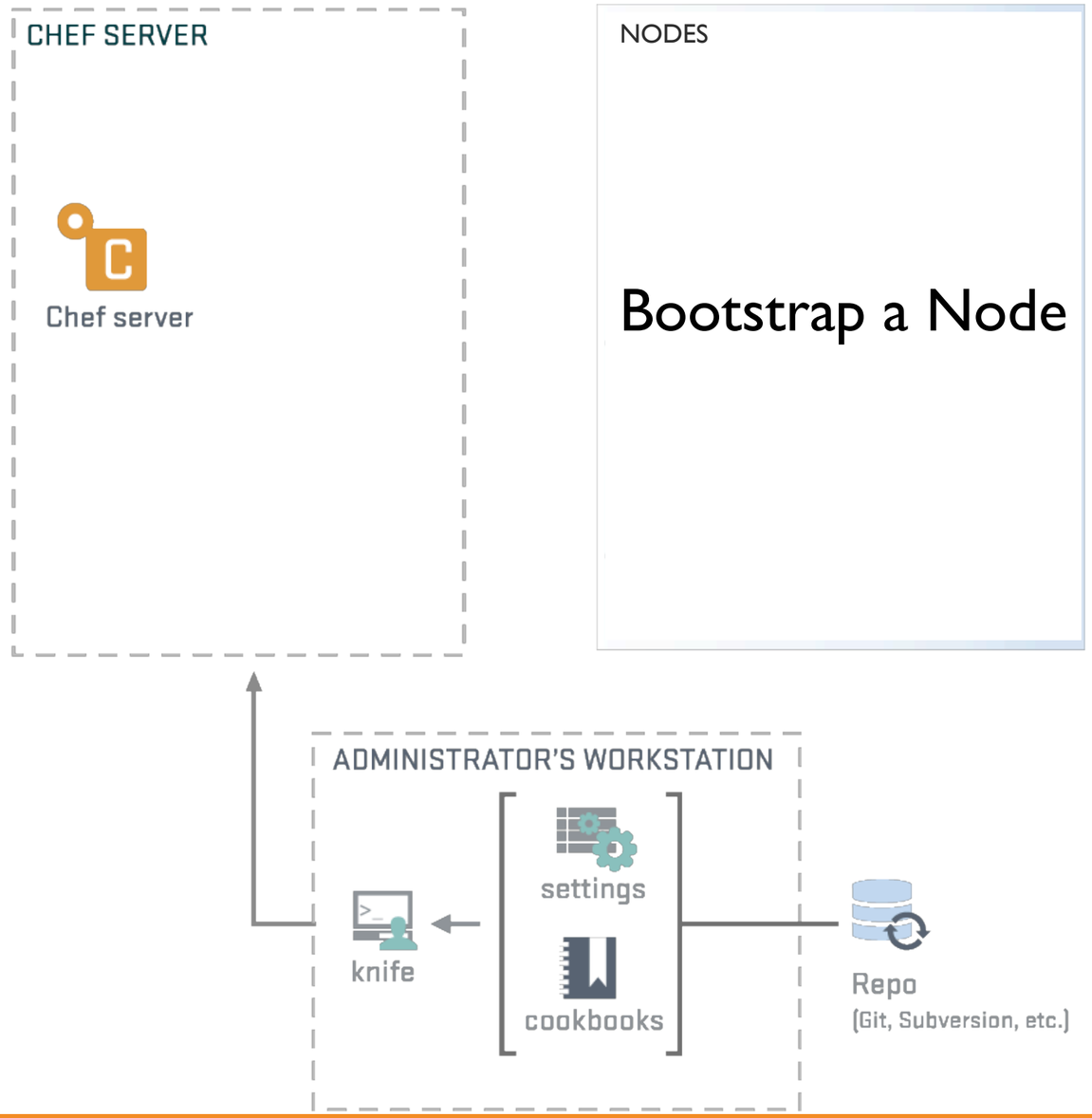
- Hostname or IP Address
- SSH Username
- SSH Password
- SSH Port (default is 22)

- SSH credentials for the Training Lab & the EC2 AMI
  - `username:  opscode`
  - `password:  opscode`

Checkpoint

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

Checkpoint


CHEF SERVER
Chef server
NODES
Bootstrap a Node
ADMINISTRATOR'S WORKSTATION
settings
knife
cookbooks
Repo
(Git, Subversion, etc.)

Bootstrap the Target Instance

```
$ knife bootstrap --help
```

```
knife bootstrap FQDN (options)
        --sudo                       Execute the bootstrap via sudo
    -x, --ssh-user USERNAME          The ssh username
    -P, --ssh-password PASSWORD      The ssh password
    -p, --ssh-port PORT              The ssh port
    -N, --node-name NAME             The Chef node name for your new node
    -r, --run-list RUN_LIST          Comma separated list of roles/recipes to apply
```

knife bootstrap

- HOSTNAME or IP Address of your machine
- `--sudo`
- `-x YOUR_SSH_USERNAME`
- `-P YOUR_SSH_PASSWORD`
- `-p YOUR_SSH_PORT (defaults to 22)`
- `-N "target1"`

knife bootstrap - Lab or AMI

- HOSTNAME or IP Address of your machine
- `--sudo`
- `-x opscode`
- `-P opscode`
- `-N "target1"`

- No need for `-p`, uses the default ssh port

Bootstrap the Target Instance

```
$ knife bootstrap IPADDRESS --sudo -x opscode -P opscode -N “target1”
```

```
Bootstrapping Chef on ec2-54-211-119-145.compute-1.amazonaws.com
ec2-54-211-119-145.compute-1.amazonaws.com knife sudo password:
Enter your password:
...
...
ec2-54-211-119-145.compute-1.amazonaws.com Converging 0 resources
ec2-54-211-119-145.compute-1.amazonaws.com
ec2-54-211-119-145.compute-1.amazonaws.com Chef Client finished, 0
resources updated
ec2-54-211-119-145.compute-1.amazonaws.com
```


local workstation
managed node
(VM)

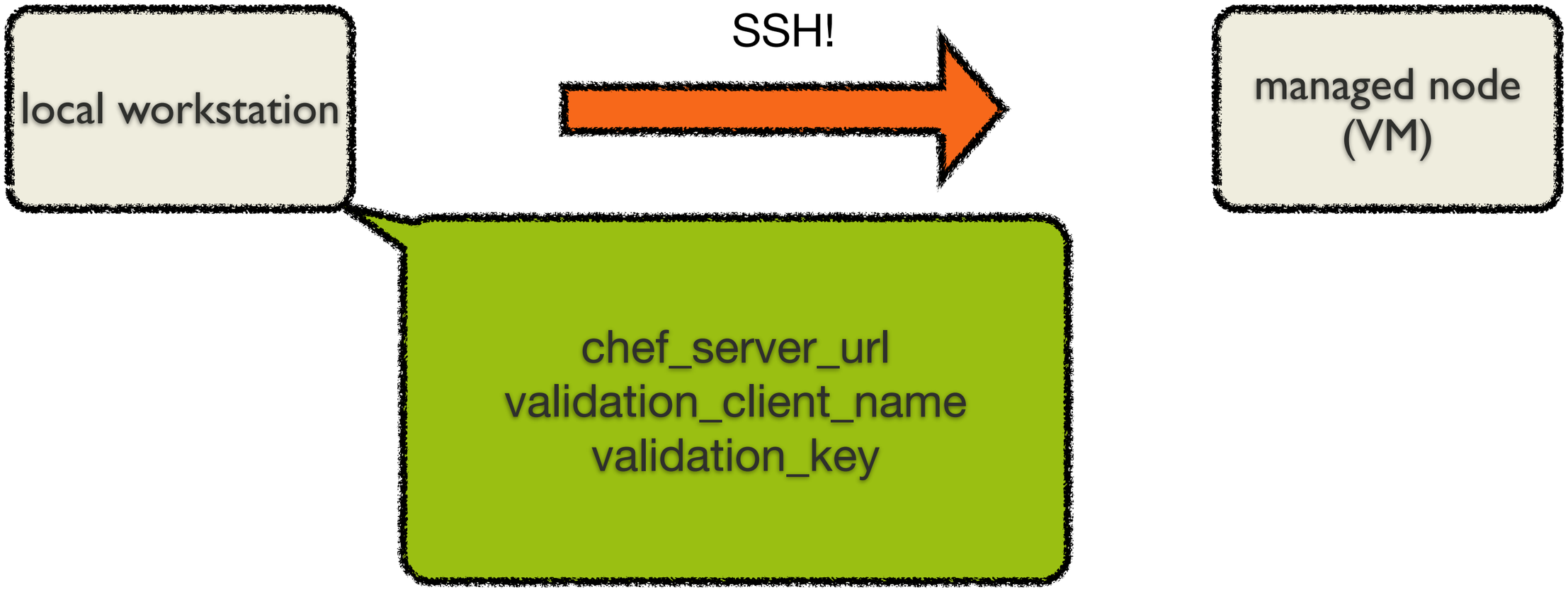
```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N target1
```

local workstation

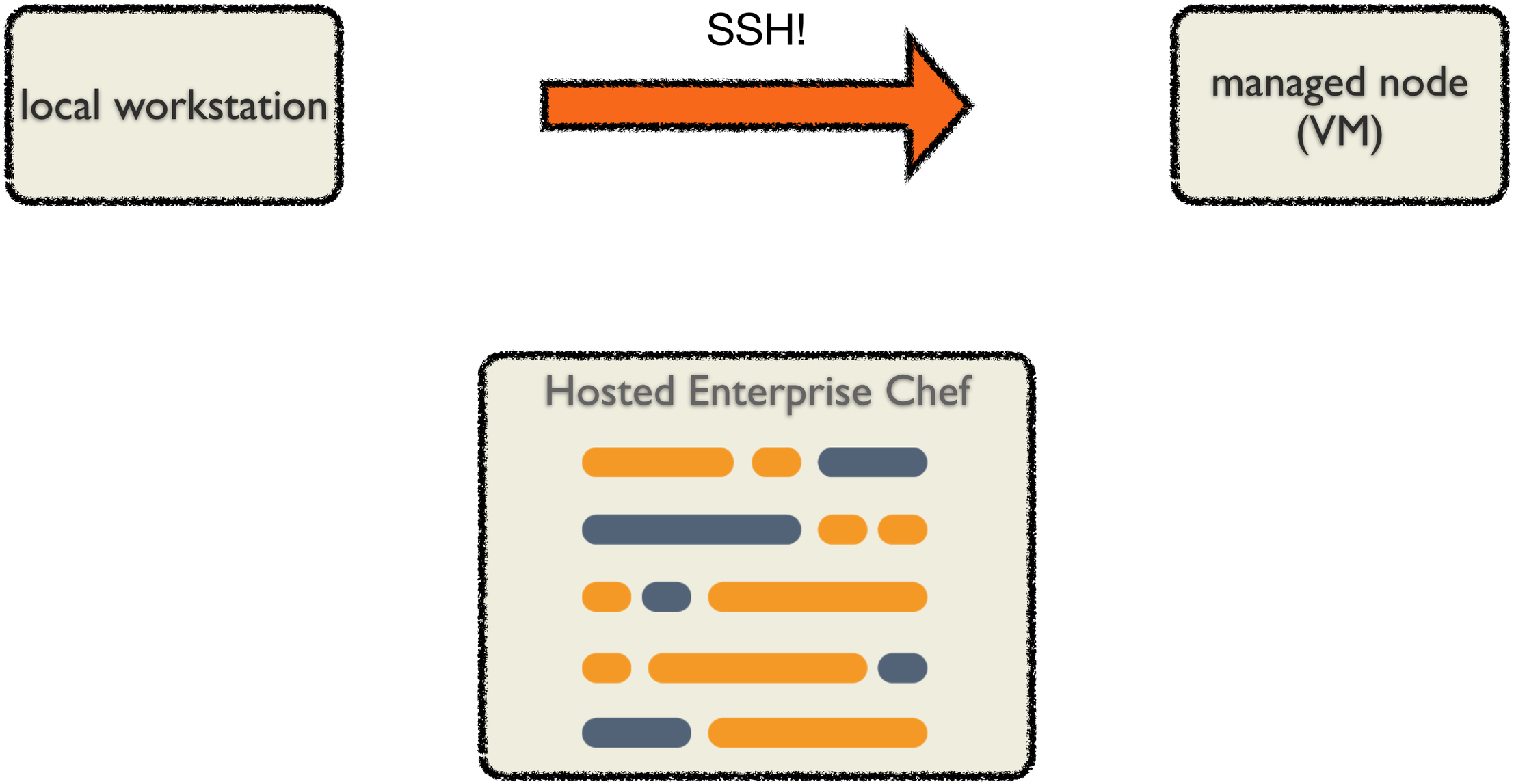
managed node
(VM)

```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N target1
```

```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N target1
```



```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N target1
```



```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N target1
```

local workstation

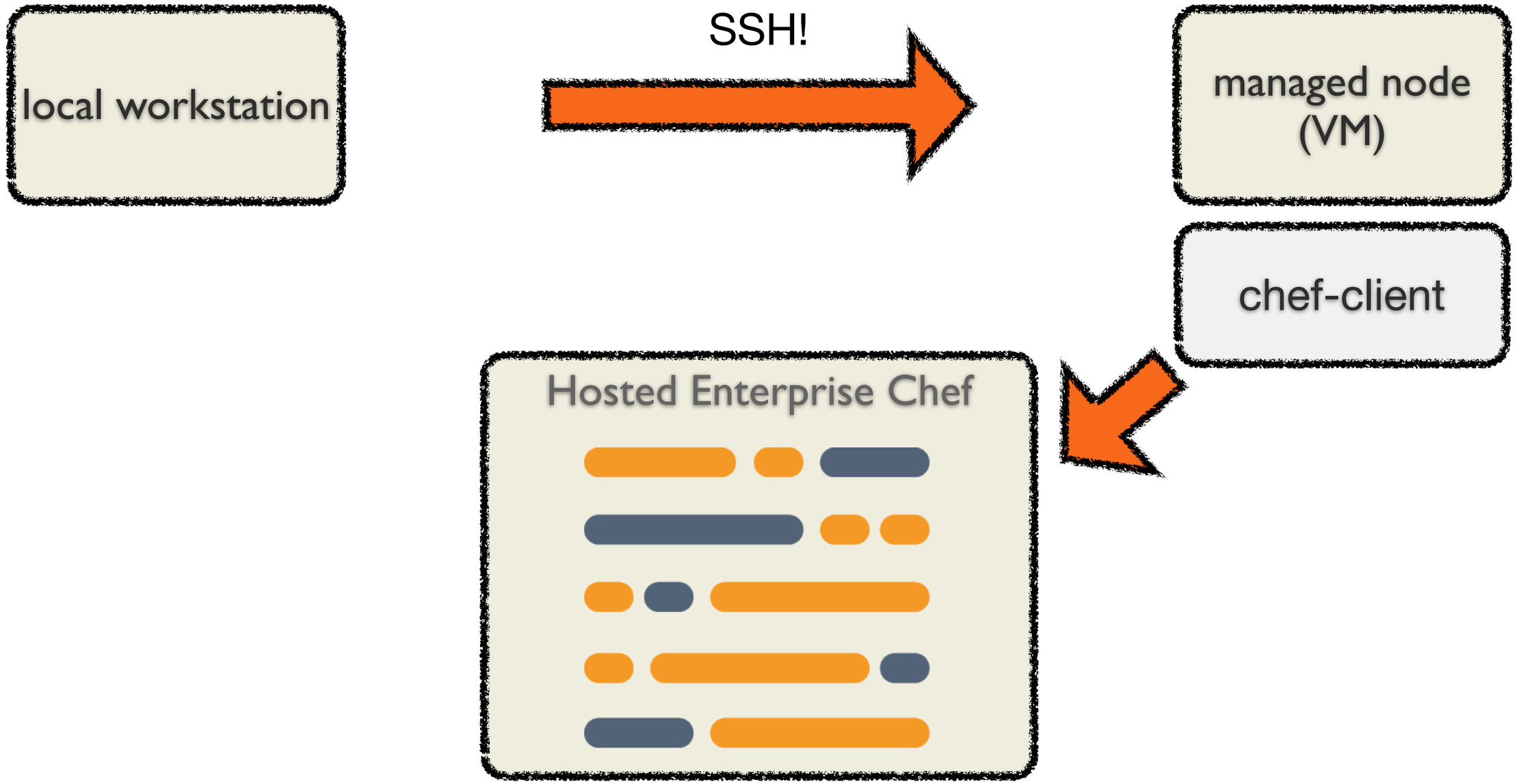
SSH!

managed node (VM)

bash -c '
install chef
configure client
run chef'

Hosted Enterprise Chef

```
$ knife bootstrap IPADDRESS --sudo -x USERNAME -P PASSWORD -N target1
```

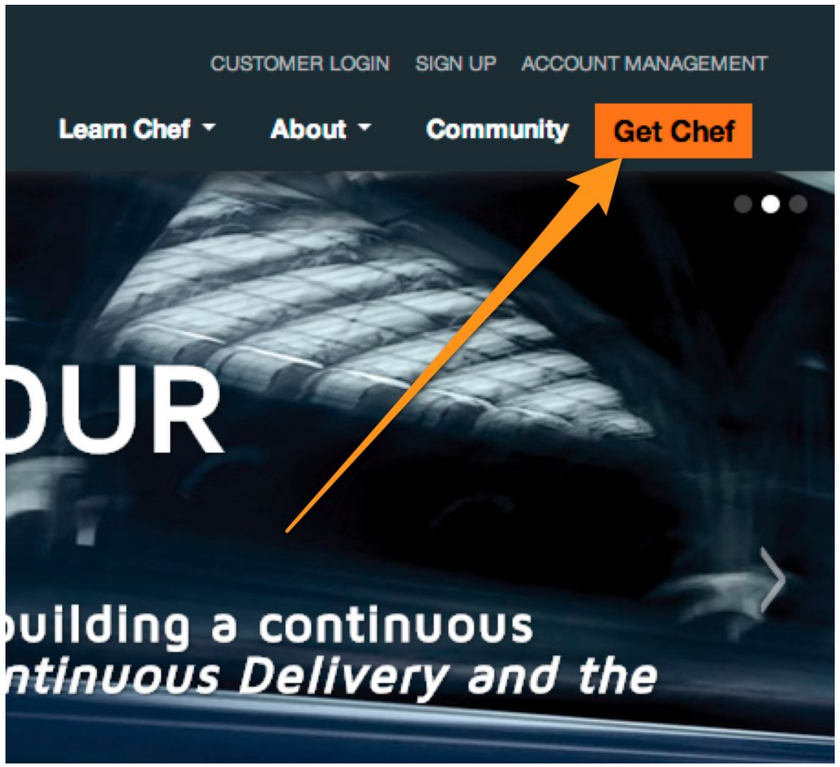


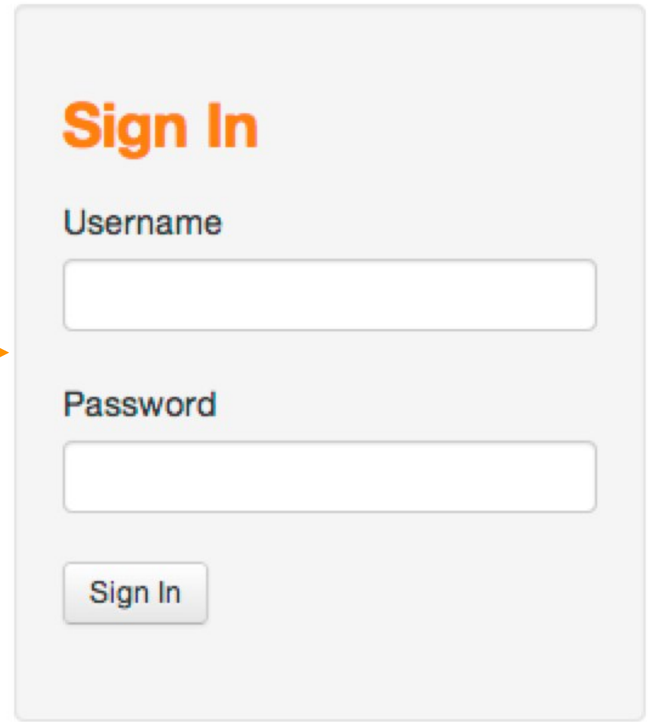
What just happened?

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

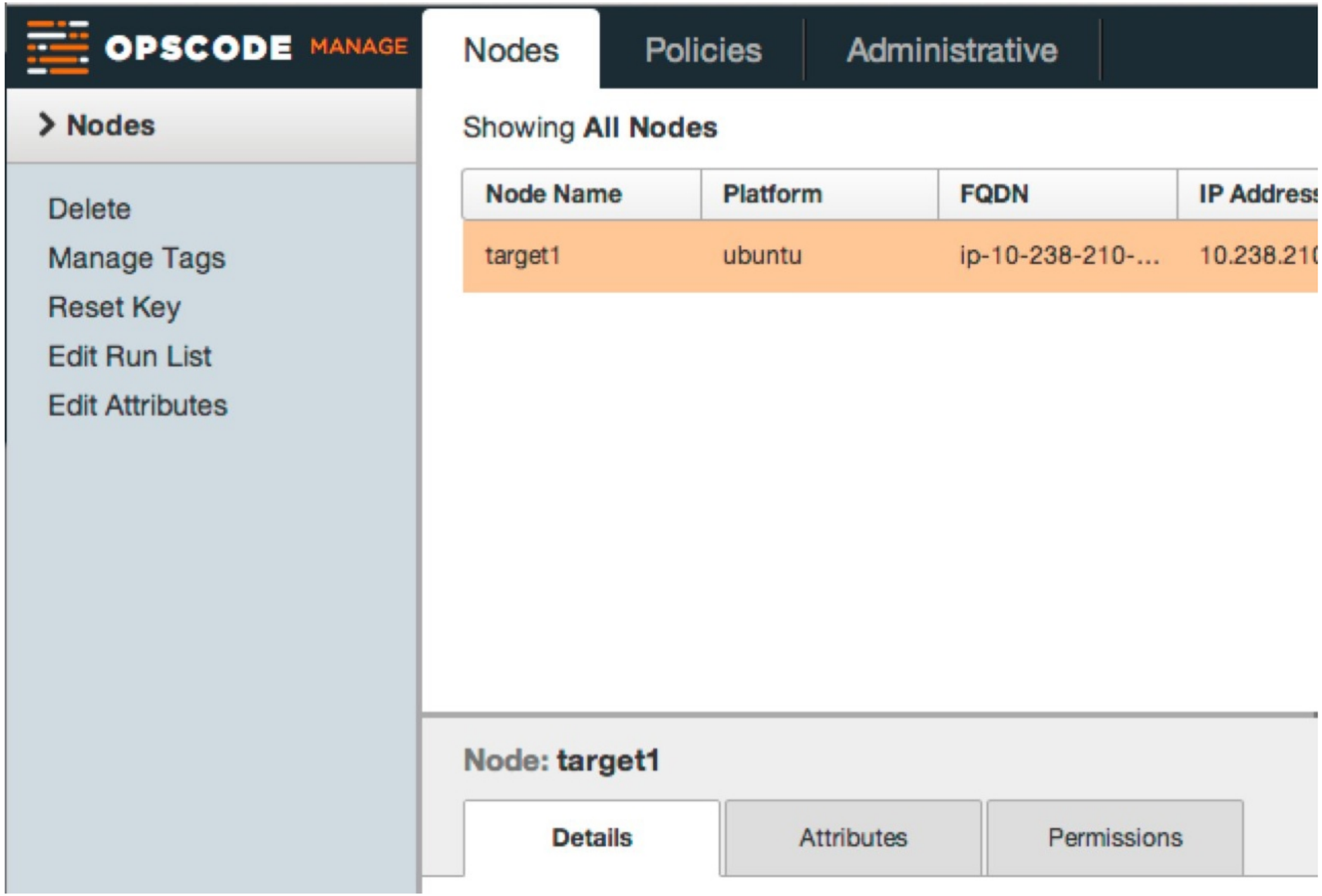
View Node on Chef Server

- Login to your Hosted Enterprise Chef

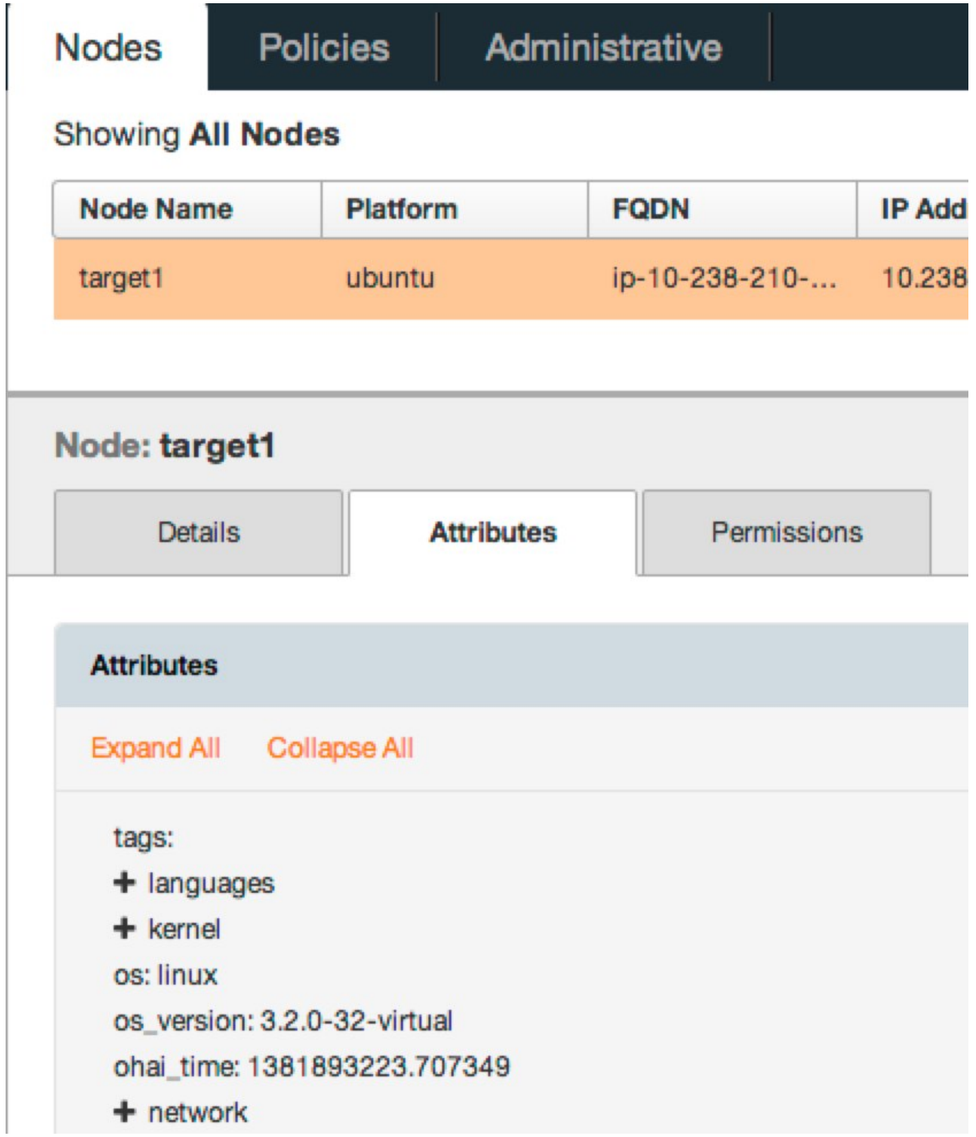




View Node on Chef Server



View Node on Chef Server



Node

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

Ohai

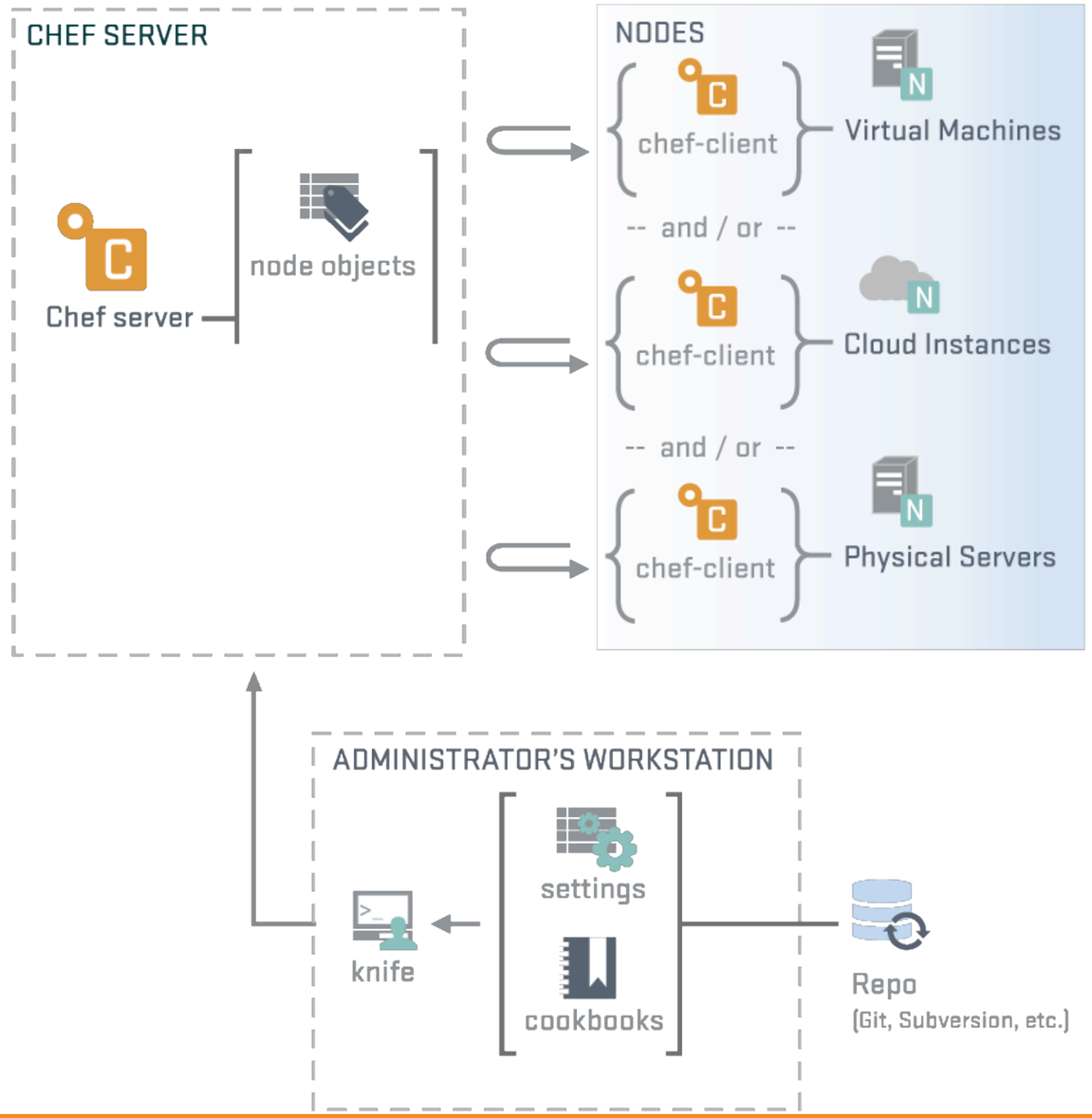

```
"languages": {
  "ruby": {

  },
  "perl": {
    "version": "5.14.2",
    "archname": "x86_64-
linux-gnu-thread-multi"
  },
  "python": {
    "version": "2.7.3",
    "builddate": "Aug 1
2012, 05:14:39"
  },
  "php": {
    "version":
"5.3.10-1ubuntu3.6",
    "builddate": "(cli)
(built: Mar"
  }
},
```

```
"kernel": {
  "name": "Linux",  "release":
"3.2.0-32-virtual",
  "version": "#51-Ubuntu SMP Wed
Sep 26 21:53:42 UTC 2012",
  "machine": "x86_64",
  "modules": {
    "isofs": {
      "size": "40257",
      "refcount": "0"
    },
    "acpiphp": {
      "size": "24231",
      "refcount": "0"
    }
  },
  "os": "GNU/Linux"
},
"os": "linux",
"os_version": "3.2.0-32-virtual",
"ohai_time": 1369328621.3456137,
```

```
"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",
```

Checkpoint


CHEF SERVER
Chef server
node objects
NODES
chef-client
Virtual Machines
-- and / or --
chef-client
Cloud Instances
-- and / or --
chef-client
Physical Servers
ADMINISTRATOR'S WORKSTATION
knife
settings
cookbooks
Repo
(Git, Subversion, etc.)

Write a Cookbook

Packages, Cookbook Files, and Services

OPSCODE™

Lesson Objectives

- After completing the lesson, you will be able to
  - Describe what a **cookbook** is
  - Create a new cookbook
  - Explain what a **recipe** is
  - Describe how to use the **package**, **service**, and **cookbook_file** resources
  - **Upload a cookbook** to the Chef Server
  - Explain what a **run list** is, and how to set it for a node

What is a cookbook?

- A cookbook is like a “package” for Chef recipes.
  - It contains all the recipes, files, templates, libraries, etc. required to configure a portion of your infrastructure
- Typically they map 1:1 to a piece of software or functionality.

The Problem and the Success Criteria

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

Required steps

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

Exercise: Create a new Cookbook

```
$ knife cookbook create apache
```

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

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
#
```

Exercise: Add a package resource to install Apache to 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
#

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

SAVE FILE!

Chef Resources

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

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

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

Chef Resources

- Have a type

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

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

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

Chef Resources

- Have a type
- Have a name

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

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

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

Chef Resources

- Have a type
- Have a name
- Have parameters

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

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

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

Chef Resources

- Have a type
- Have a name
- Have parameters
- Take action to put the resource into the desired state

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

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

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

Chef Resources

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

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

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

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

So the resource we just wrote...

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

So the resource we just wrote...

- Is a **package** resource

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

So the resource we just wrote...

- Is a **package** resource
- Whose **name** is *apache2*

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

So the resource we just wrote...

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

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

Notice we didn’t say how to install the package

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

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

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

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

package "apache2" do
  action :install
end

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

SAVE FILE!

So the resource we just wrote...

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

So the resource we just wrote...

- Is a **service** resource

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

So the resource we just wrote...

- Is a **service** resource
- Whose **name** is *apache2*

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

So the resource we just wrote...

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

```
service "apache2" 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
```

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

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

```
...

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

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

SAVE FILE!

So the resource we just wrote...

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

So the resource we just wrote...

- Is a **cookbook_file** resource

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

So the resource we just wrote...

- Is a **cookbook_file** resource
- Whose **name** is */var/www/index.html*

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

So the resource we just wrote...

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

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

Full contents of the apache recipe

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

package "apache2" do
  action :install
end

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

cookbook_file "/var/www/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>
```

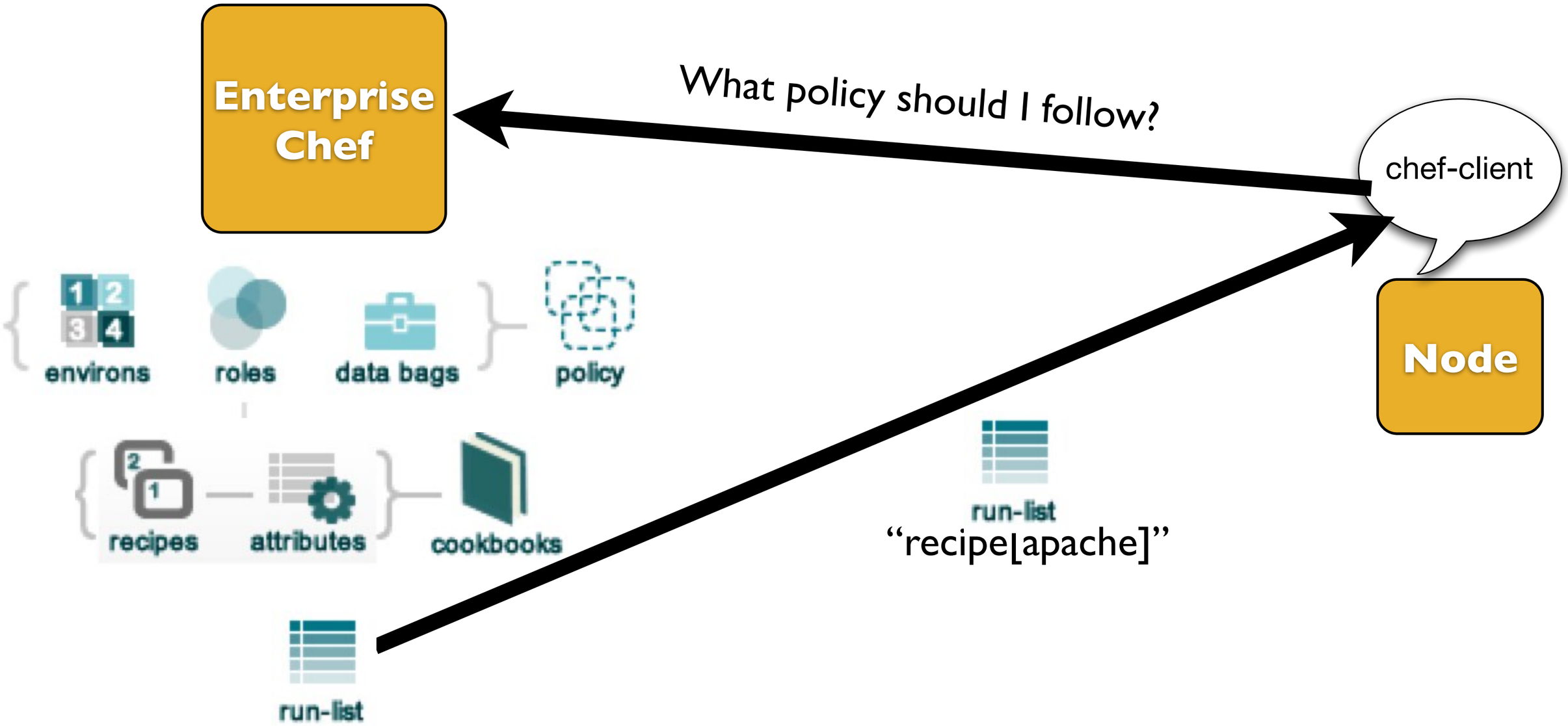
SAVE FILE!

Exercise: Upload the cookbook

```
$ knife cookbook upload apache
```

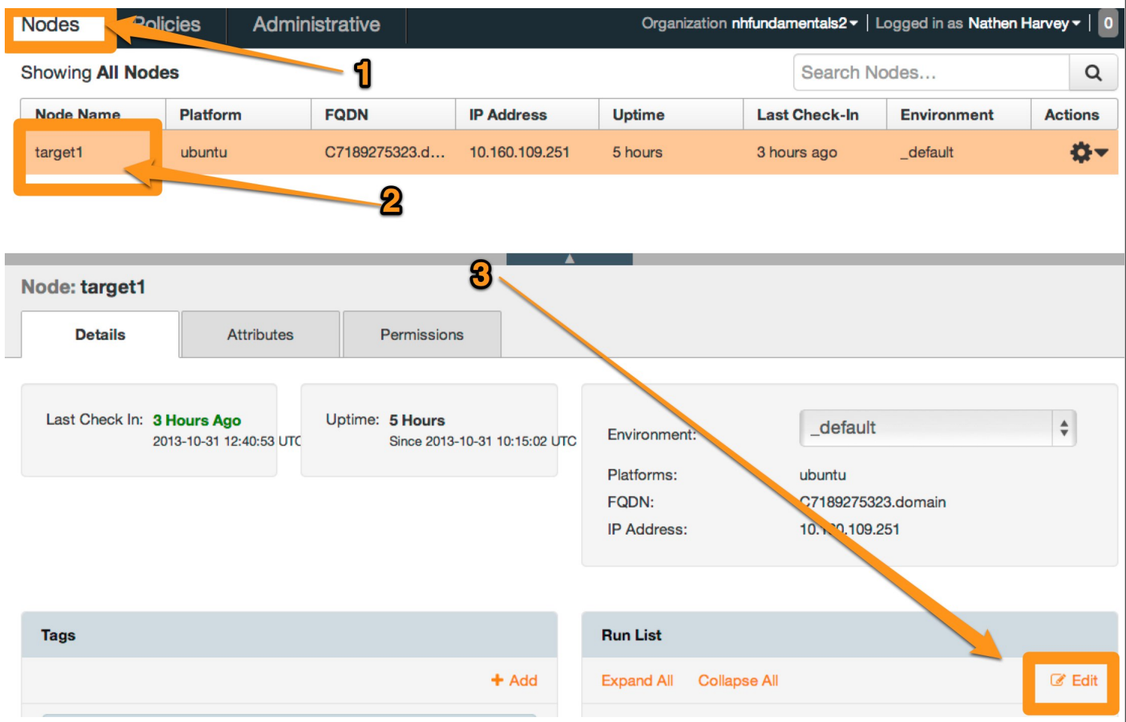
```
Uploading apache                [0.1.0]
Uploaded 1 cookbook.
```

Run List


Enterprise Chef
What policy should I follow?
chef-client
Node
environs
roles
data bags
policy
recipes
attributes
cookbooks
run-list
run-list
“recipe[apache]”

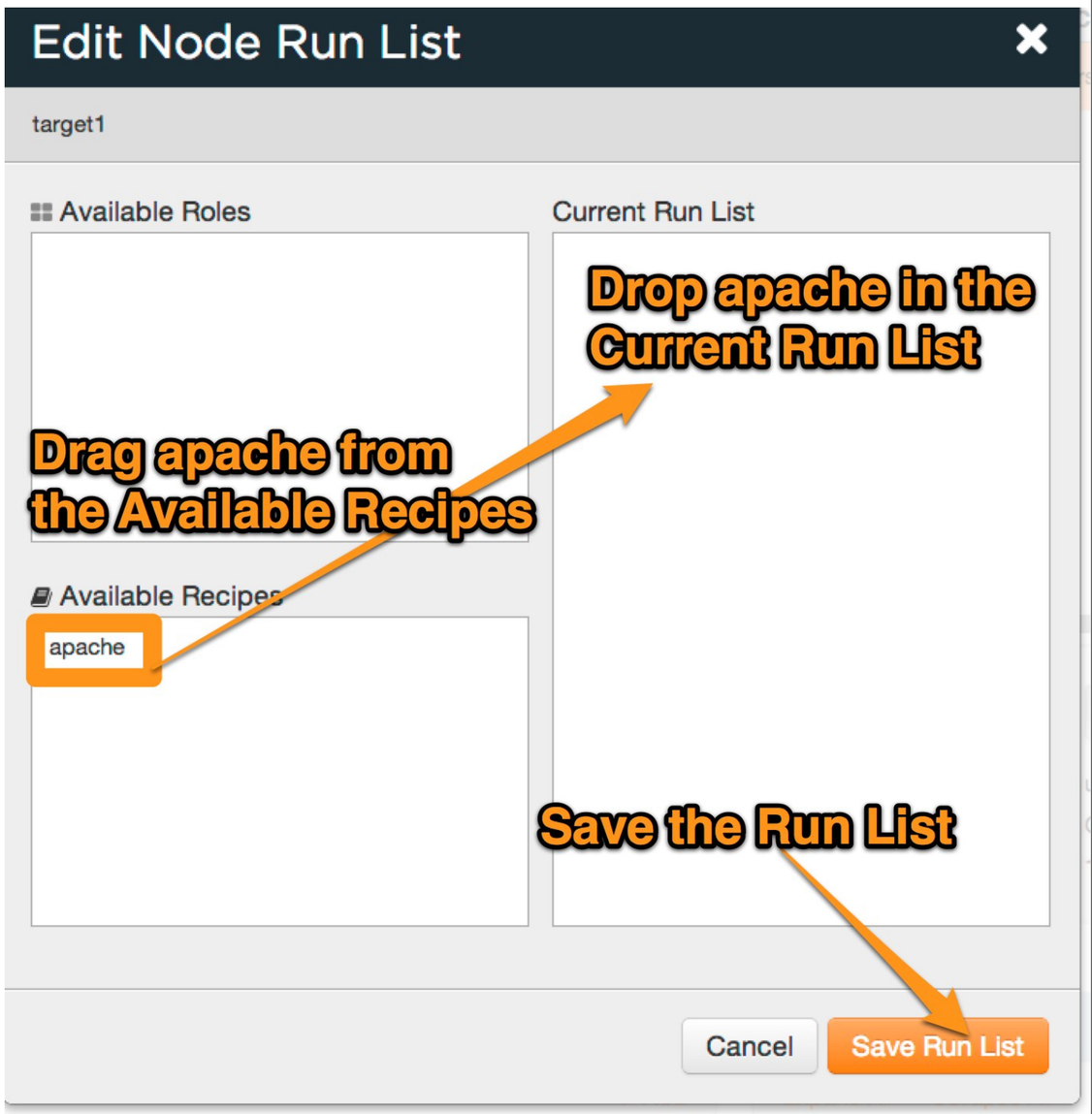
Update the Run List

- Login to Enterprise Hosted Chef
- Select the "Nodes" tab
- Select your Node
- Edit the Run List



Update the Run List

- Drag
- Drop
- Save



Exercise: Run the chef-client on your test node

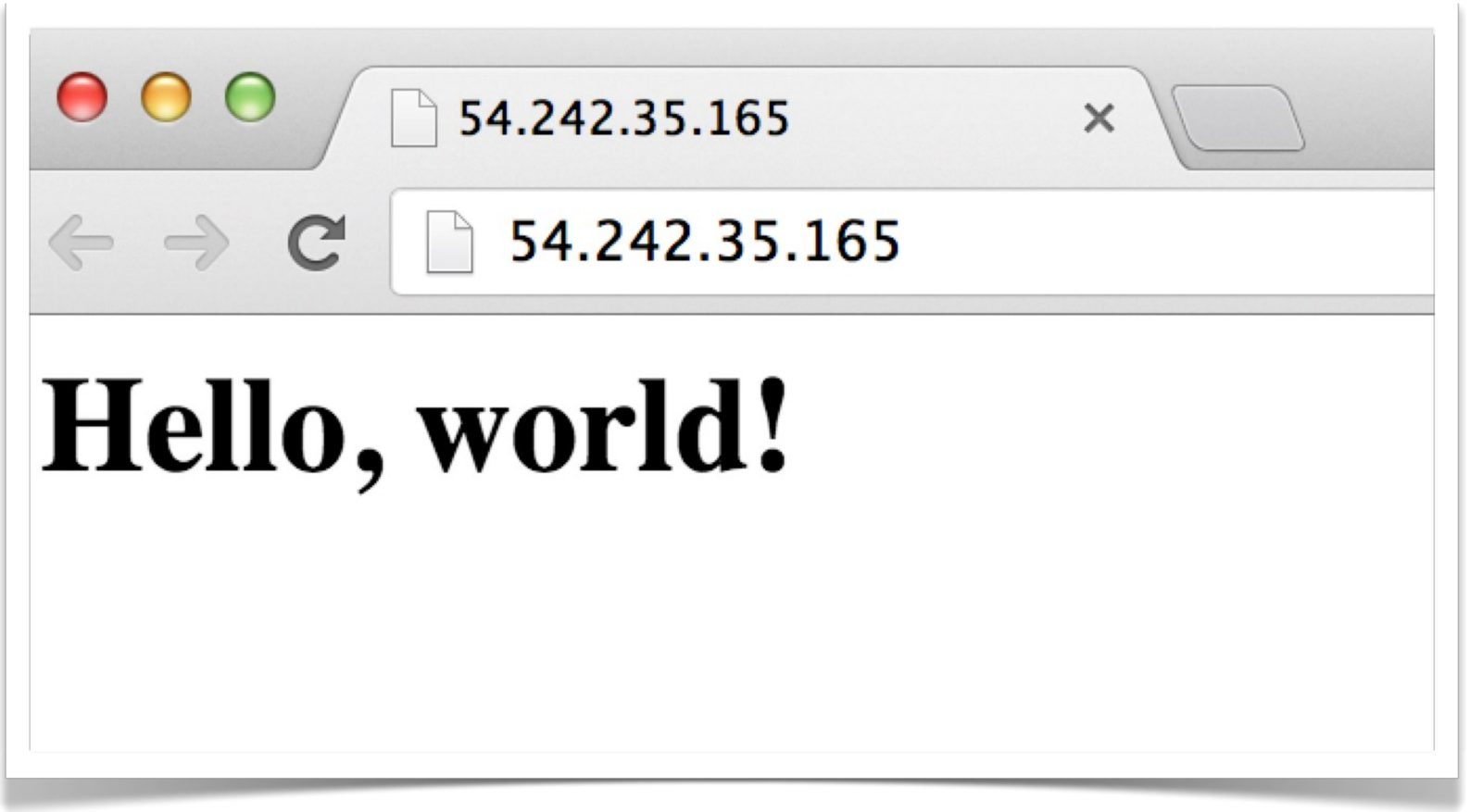
```
opscode@target1:~$ sudo chef-client
```

```
Starting Chef Client, version 11.4.4
[2013-06-25T04:20:22+00:00] INFO: *** Chef 11.4.4 ***
[2013-06-25T04:20:23+00:00] INFO: [inet6] no default interface, picking the first ipaddress
[2013-06-25T04:20:23+00:00] INFO: Run List is [recipe[apache]]
[2013-06-25T04:20:23+00:00] INFO: Run List expands to [apache]
[2013-06-25T04:20:23+00:00] INFO: Starting Chef Run for target1
[2013-06-25T04:20:23+00:00] INFO: Running start handlers
[2013-06-25T04:20:23+00:00] INFO: Start handlers complete.
resolving cookbooks for run list: ["apache"]
[2013-06-25T04:20:24+00:00] INFO: Loading cookbooks [apache]
Synchronizing Cookbooks:
[2013-06-25T04:20:24+00:00] INFO: Storing updated cookbooks/apache/recipes/default.rb in the cache.
[2013-06-25T04:20:24+00:00] INFO: Storing updated cookbooks/apache/recipes/tmp.rb in the cache.
[2013-06-25T04:20:24+00:00] INFO: Storing updated cookbooks/apache/CHANGELOG.md in the cache.
[2013-06-25T04:20:25+00:00] INFO: Storing updated cookbooks/apache/metadata.rb in the cache.
[2013-06-25T04:20:25+00:00] INFO: Storing updated cookbooks/apache/README.md in the cache.
  - apache
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::default
  * package[apache2] action install[2013-06-25T04:20:25+00:00] INFO: Processing package[apache2] action install (apache::default line 9)

    - install version 2.2.22-1ubuntu1 of package apache2
```

Exercise: Verify that the home page works

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

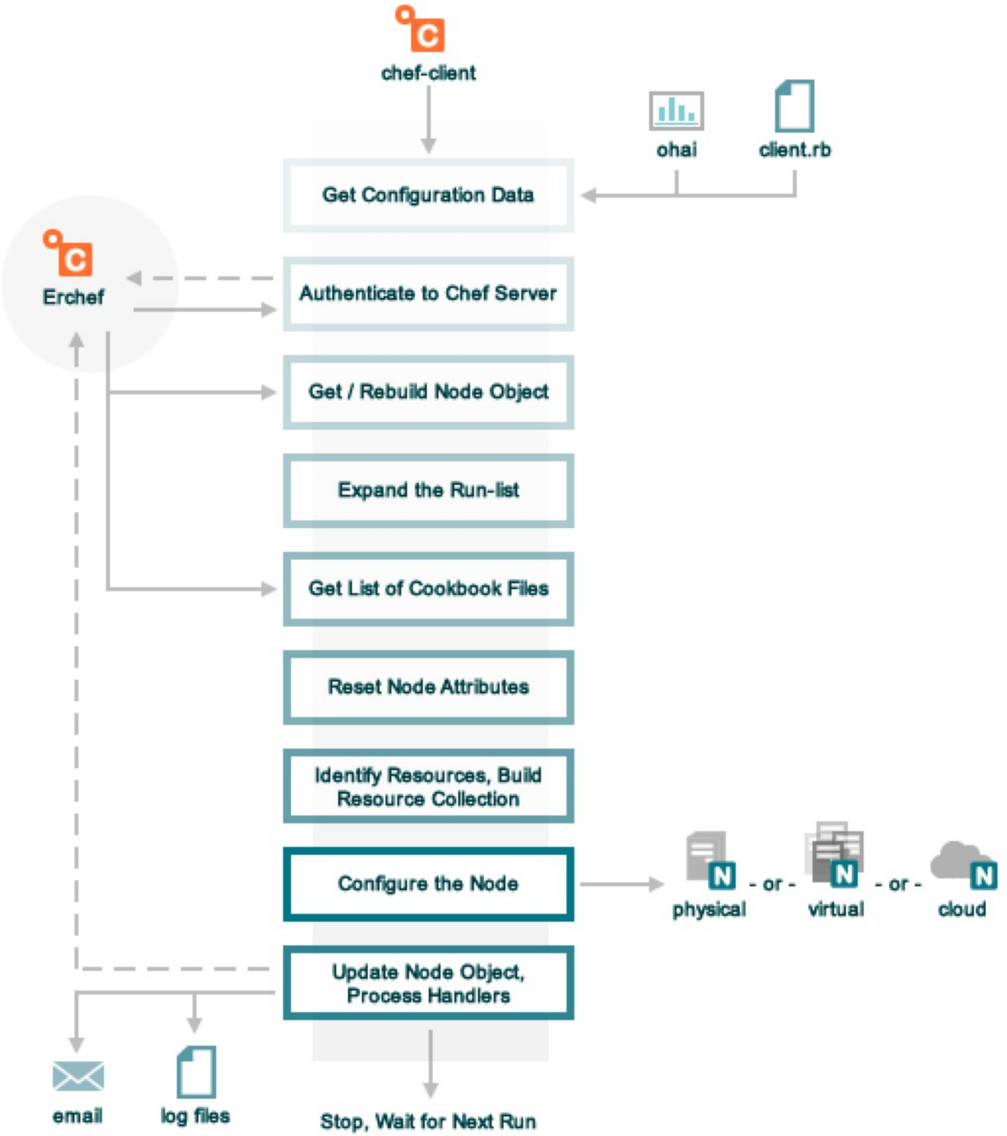


Congratulate yourself!

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

Next Week

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



Thank You

- Nathen Harvey
- Technical Community Manager at Opscode

- @nathenharvey

- nharvey@opscode.com