Community Cookbooks

Find, Explore and View Chef Cookbooks

CHEF

Objectives

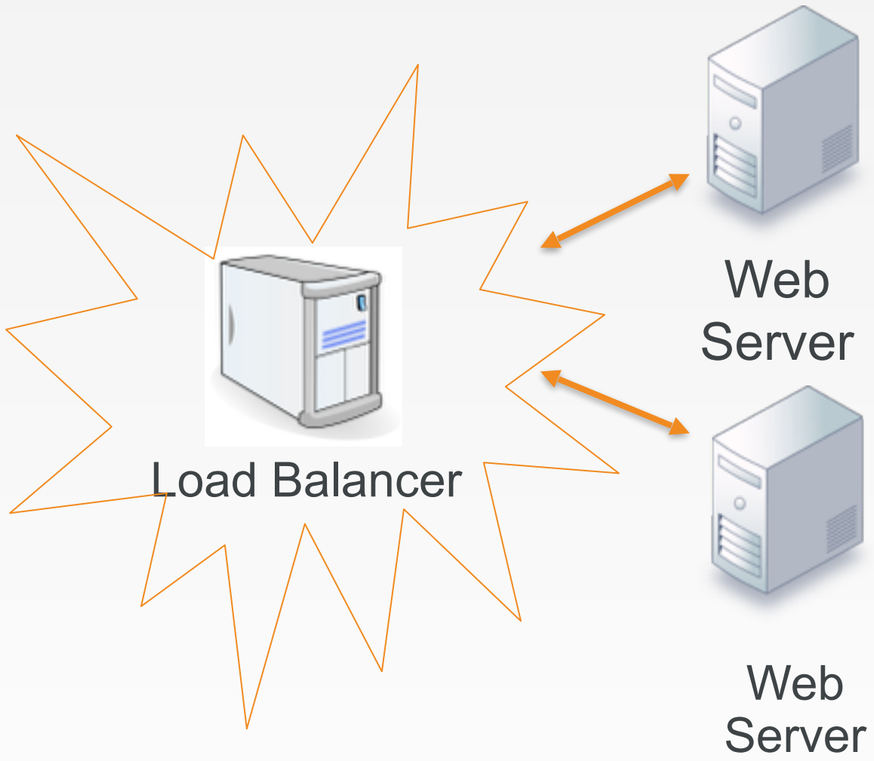
After completing this module, you should be able to

- Find cookbooks on the Chef Super Market
- Create a wrapper cookbook
- Replace the existing default values
- Upload a cookbook to Chef Server
- Bootstrap a new node that runs the cookbook

Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

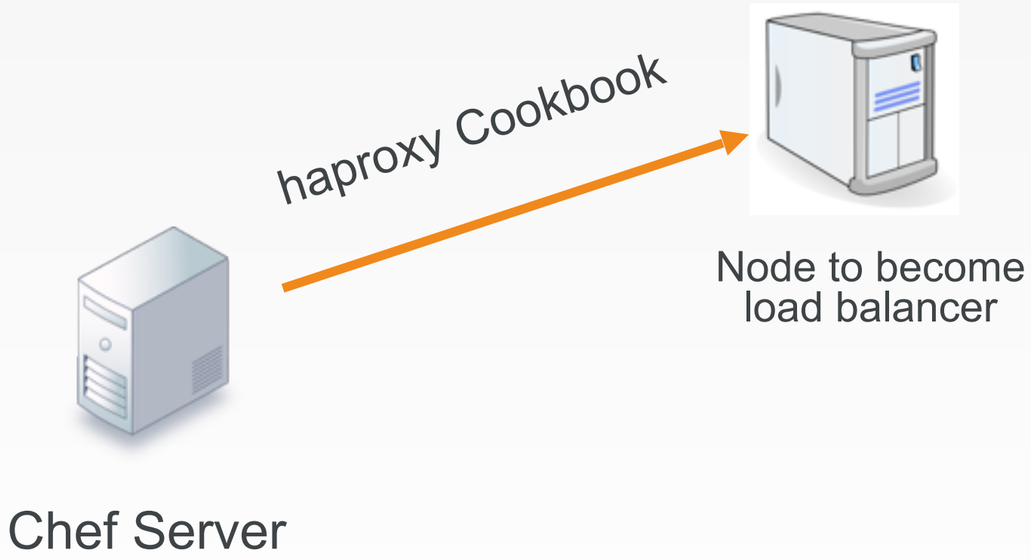
Receives requests and relays them to other systems.



Load Balancer

Work that needs to be accomplished to setup a load balancer within our infrastructure:

- Write a haproxy (load balancer) cookbook.
- We will need to establish a new node within our organization to which we apply that cookbook.



Community Cookbooks

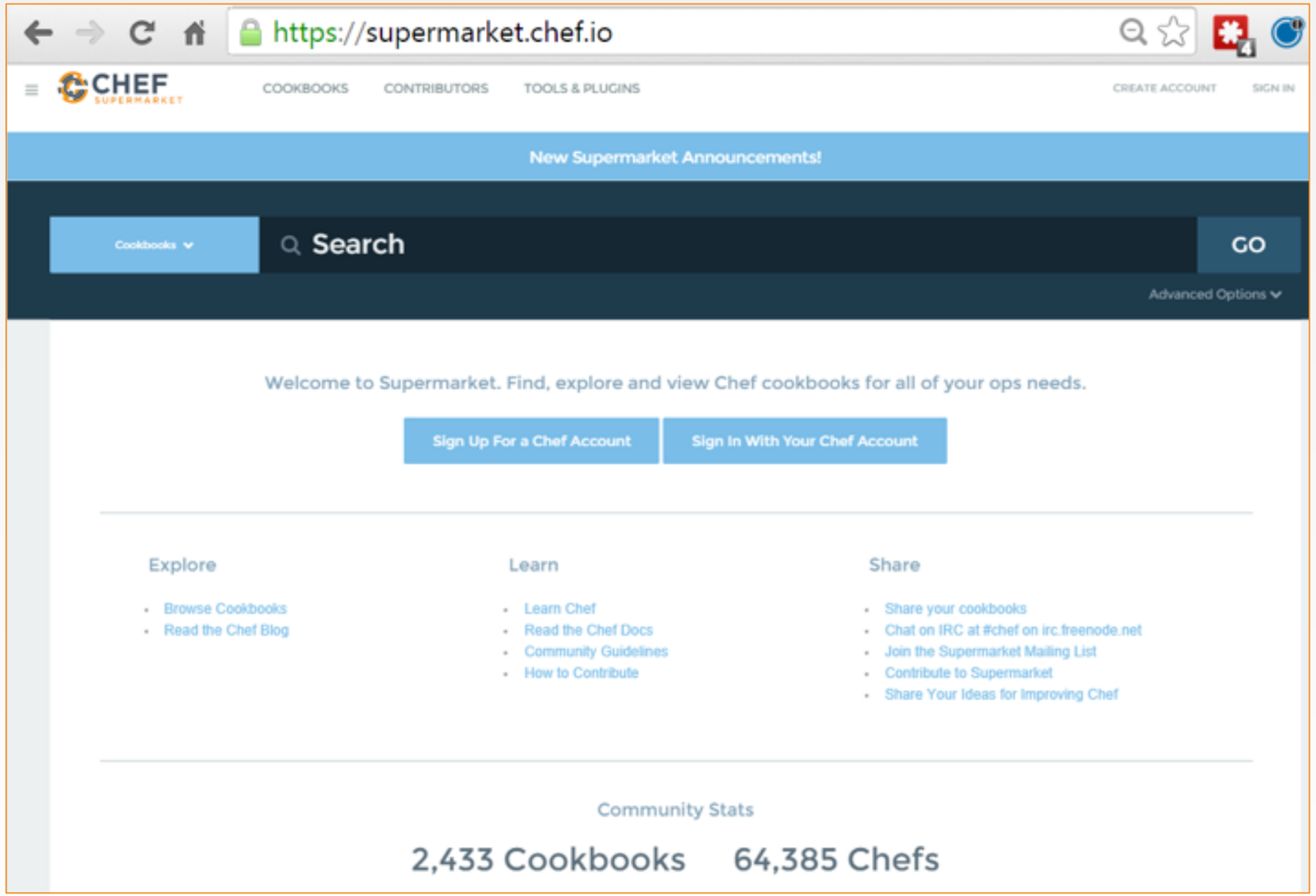
Someone already wrote that cookbook?

Available through the community site called Supermarket/

https://supermarket.chef.io

Community Cookbooks

- Community cookbooks are managed by individuals.
- Chef does not verify or approve cookbooks in the Supermarket.
- Cookbooks may not work for various reasons.
- Still, there are real benefits to community cookbooks.



Load Balancer

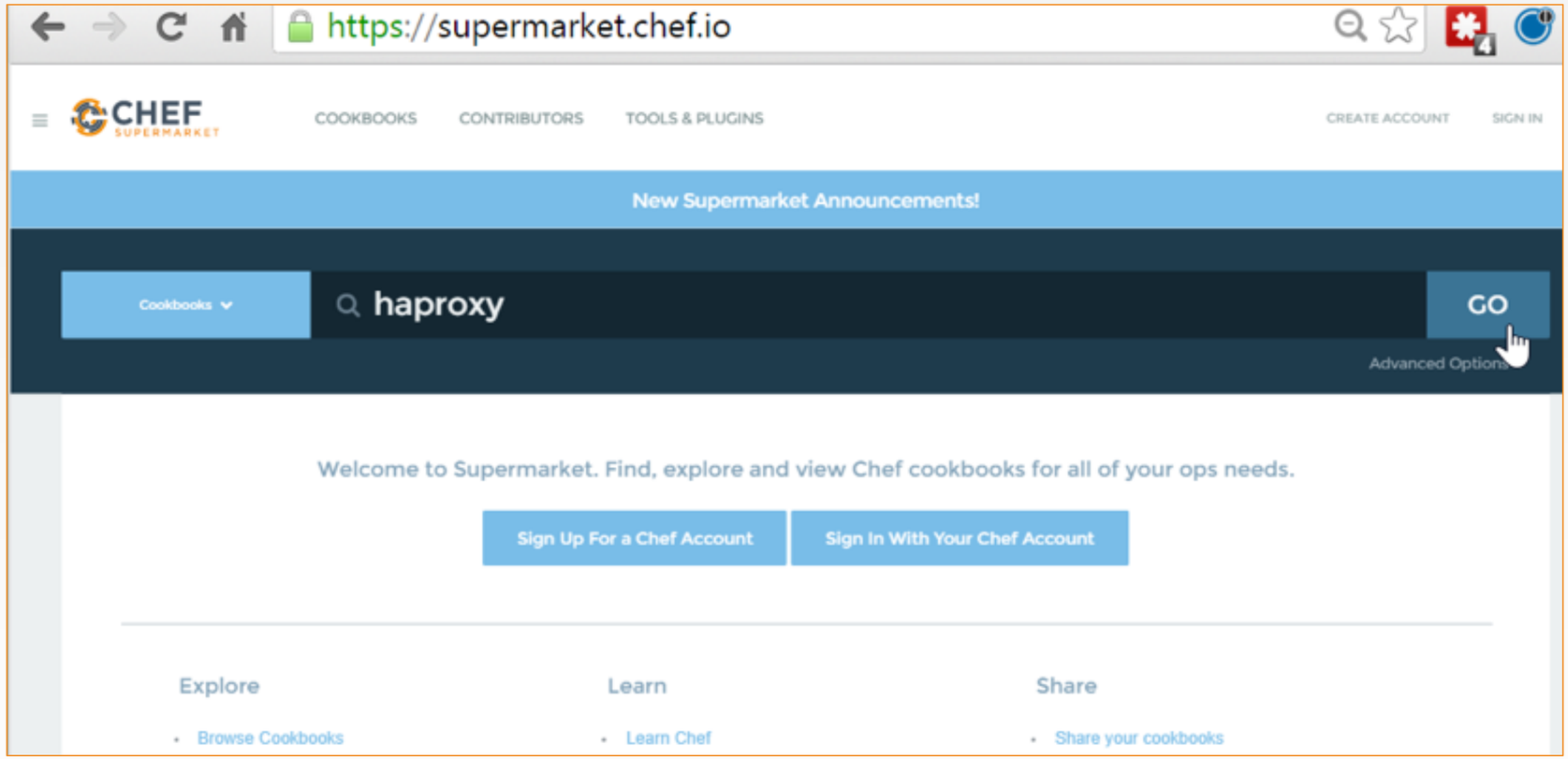
Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ❑ Find or Create a Cookbook to Manage a load balancer
- ❑ Configure the load balancer to send traffic to the new node
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the haproxy (load balancer) cookbook

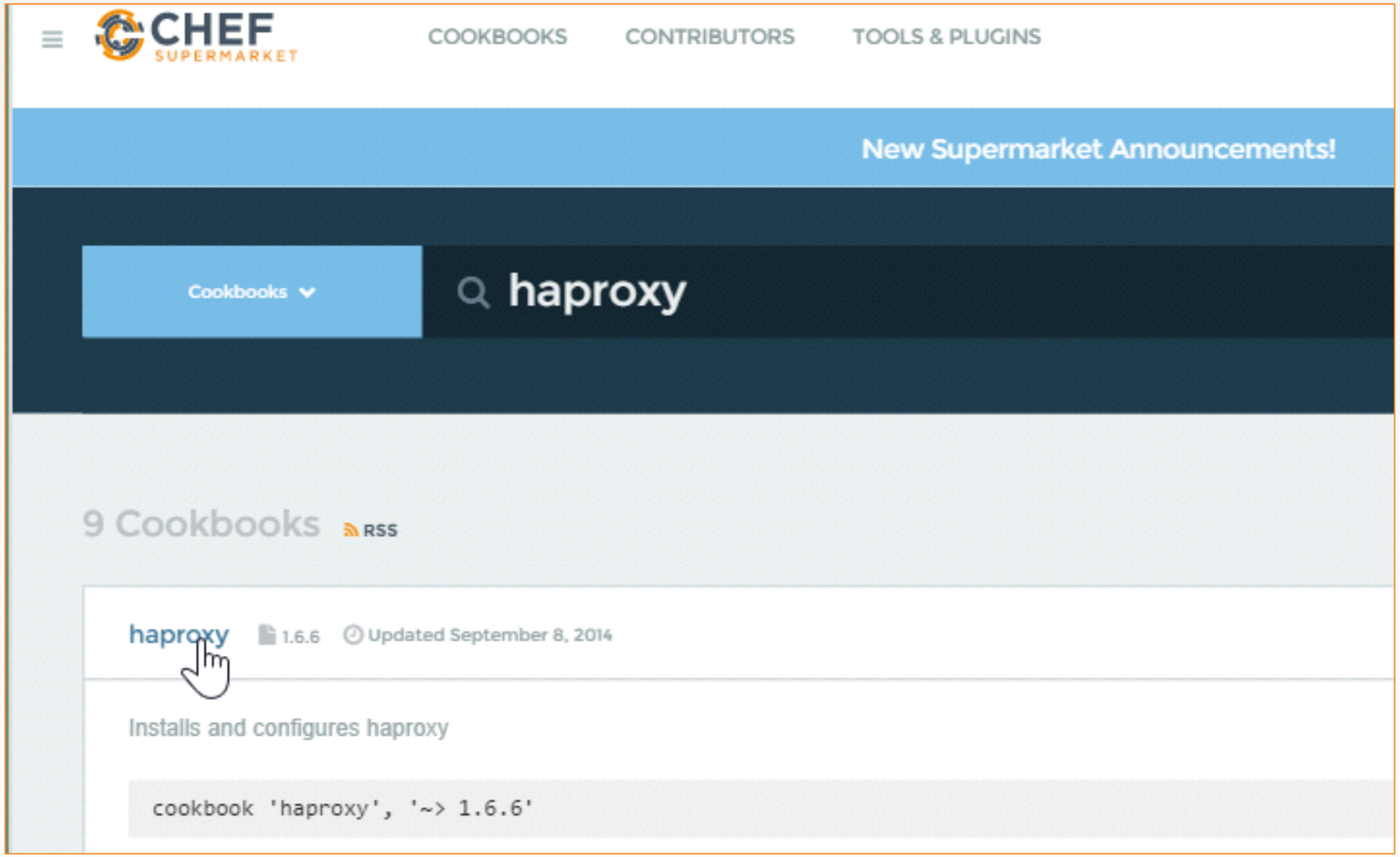
GL: Searching in the Supermarket

1. From the https://supermarket.chef.io page, type **haproxy** in the search field and then click the **GO** button.



GL: Searching in the Supermarket

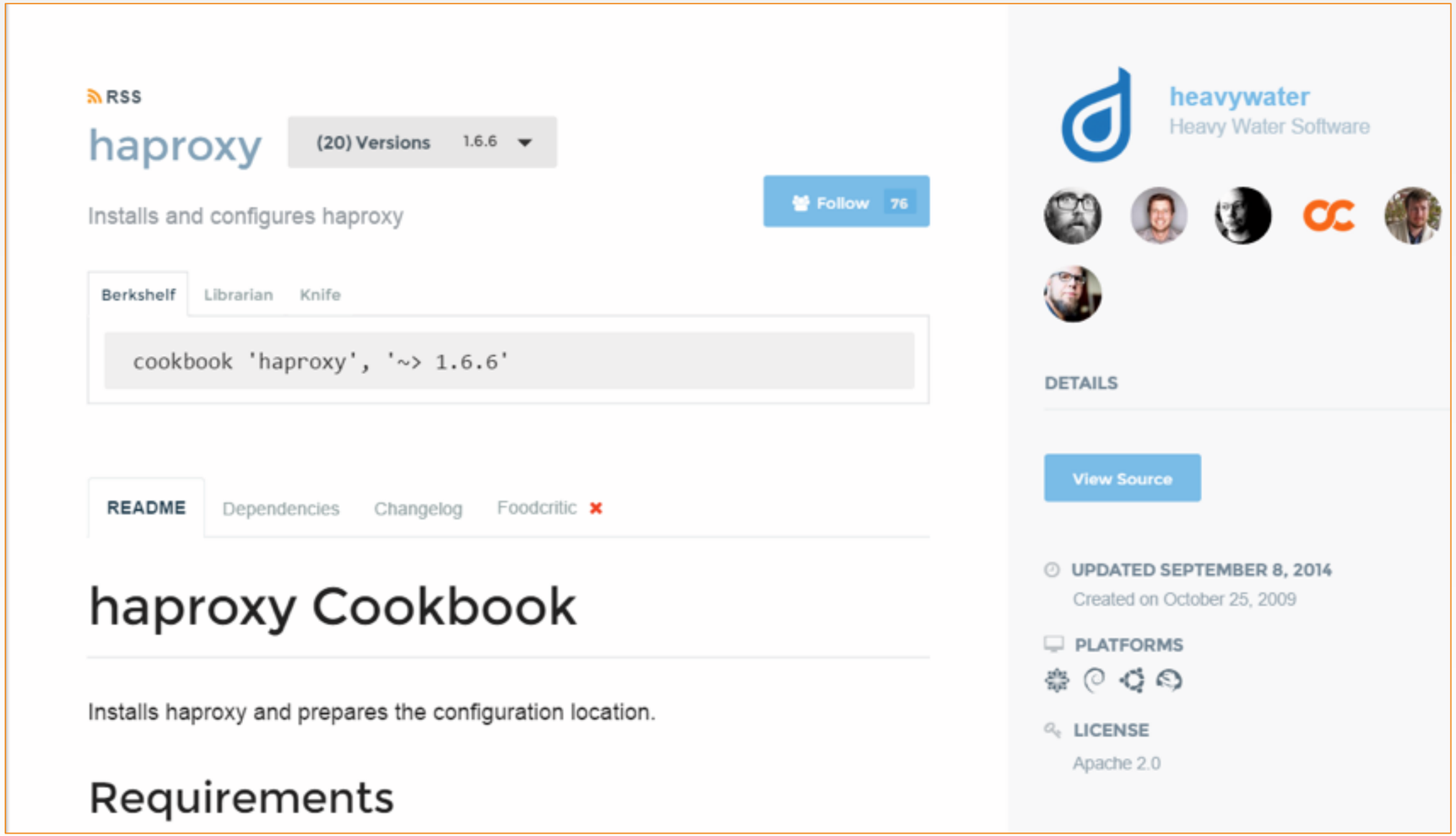
2. Click the resulting **haproxy** link.



Supermarket Cookbooks

On the right-hand side we can see the individuals that maintain the cookbook...

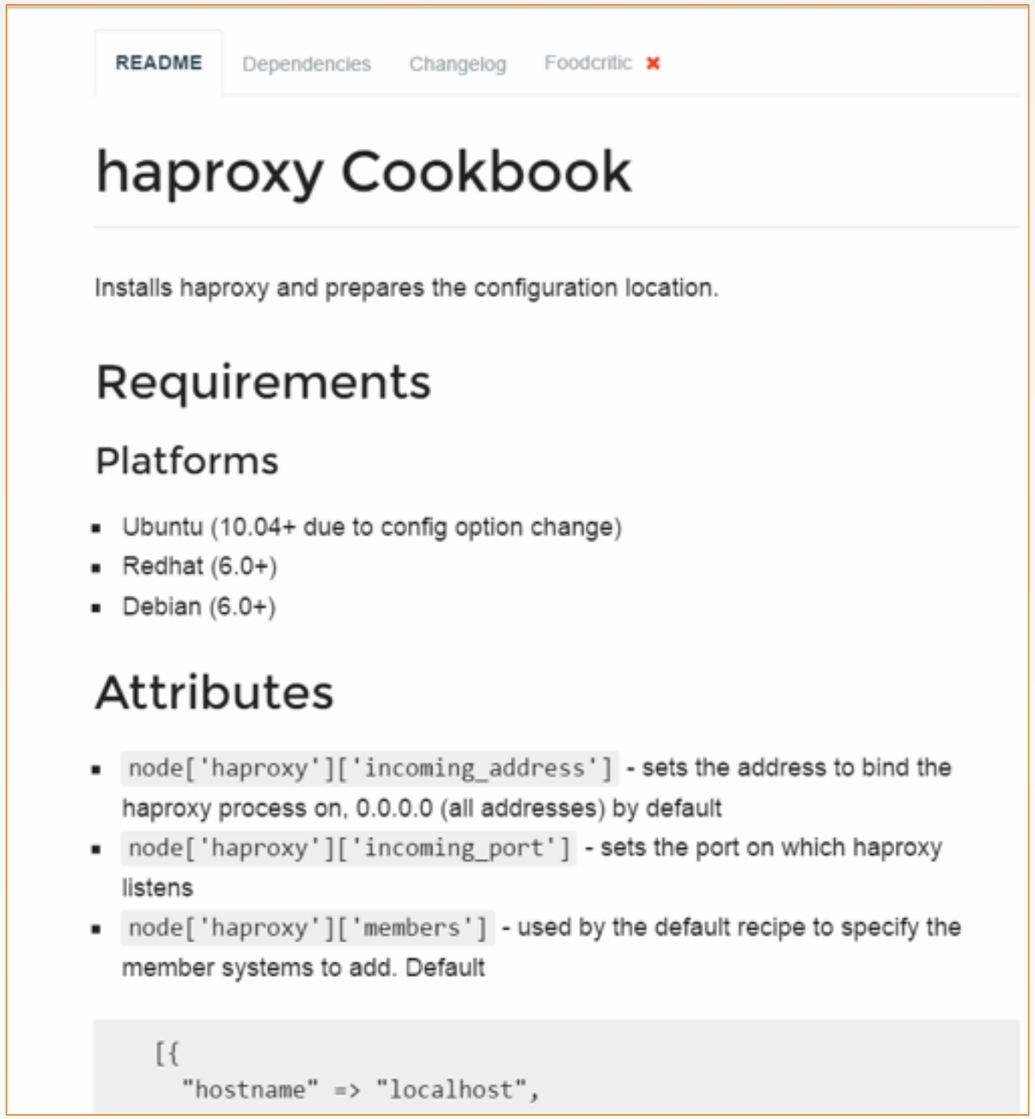
On the left, we are presented with the various ways we can install the cookbook...



Supermarket Cookbooks

The area to focus most of your attention from the beginning is the README.

Reading and understanding the README at a glance is difficult. It is a skill that comes with time.



README Dependencies Changelog Foodcritic ✖

haproxy Cookbook

Installs haproxy and prepares the configuration location.

Requirements

Platforms

- Ubuntu (10.04+ due to config option change)
- Redhat (6.0+)
- Debian (6.0+)

Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
```


CHEF

Supermarket Cookbooks

These node attributes are different than the automatic ones defined by Ohai.

Attributes defined in a cookbook are not considered automatic.

Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4000,
  "ssl_port" => 4000
}, {
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4001,
  "ssl_port" => 4001
}]
```

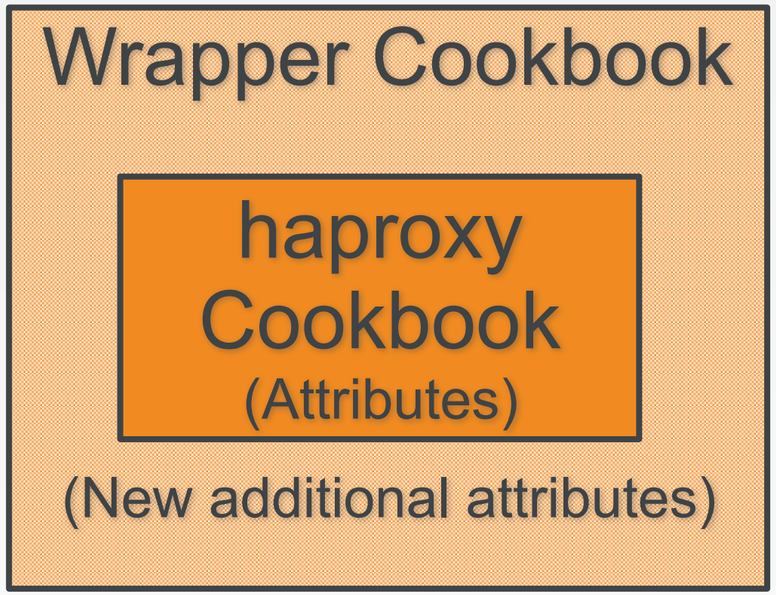
- `node['haproxy']['member_port']` - the port that member systems will be listening on if not otherwise

https://docs.chef.io/attributes.html

Supermarket Cookbooks

A wrapper cookbook is a new cookbook that encapsulates the functionality of the original cookbook.

It defines new default values for the recipes.



https://docs.chef.io/supermarket.html#wrapper-cookbooks

https://www.chef.io/blog/2013/12/03/doing-wrapper-cookbooks-right/

GL: CD and Generate the Cookbook

```
$ cd ~/chef-repo
$ chef generate cookbook cookbooks/myhaproxy
```

```
Compiling Cookbooks...
Recipe: code_generator::cookbook
  * directory[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy] action create
    - create new directory C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy
  * template[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb] action
create_if_missing
    - create new file C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/metadata.rb
    - update content in file C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/
metadata.rb from none to 899276
    (diff output suppressed by config)
  * template[C:/Users/sdelfante/chef-repo/cookbooks/myhaproxy/README.md] action
create_if_missing
```

GL: Create a Dependency in the Cookbook

```
~/chef-repo/cookbooks/myhaproxy/metadata.rb
```

```
name                'myhaproxy'
maintainer          'The Authors'
maintainer_email    'you@example.com'
license             'all_rights'
description         'Installs/Configures myhaproxy'
long_description    'Installs/Configures myhaproxy'
version             '0.1.0'

depends 'haproxy', '~> 1.6.6'
```

Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ✓ Find or create a cookbook to manage a load balancer
- ❑ Configure the load balancer to send traffic to the new node
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the haproxy cookbook

Supermarket Cookbooks

Currently, the haproxy cookbook assumes that there are two different services running on the localhost at port 4000 and port 4001.

In a moment, you'll need to change that.

Attributes

- `node['haproxy']['incoming_address']` - sets the address to bind the haproxy process on, 0.0.0.0 (all addresses) by default
- `node['haproxy']['incoming_port']` - sets the port on which haproxy listens
- `node['haproxy']['members']` - used by the default recipe to specify the member systems to add. Default

```
[{
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4000,
  "ssl_port" => 4000
}, {
  "hostname" => "localhost",
  "ipaddress" => "127.0.0.1",
  "port" => 4001,
  "ssl_port" => 4001
}]
```

- `node['haproxy']['member_port']` - the port that member systems will be listening on if not otherwise

https://docs.chef.io/supermarket.html#wrapper-cookbooks

CHEF

GL: Capture Node's Public Host Name and IP

```
$ knife node show --help
```

```
knife node show NODE (options)
    -a ATTR1 [--attribute ATTR2] ,   Show one or more attributes
        --attribute
    -s, --server-url URL             Chef Server URL
        --chef-zero-host HOST        Host to start chef-zero on
        --chef-zero-port PORT        Port (or port range) to start chef-zero on.
Port ranges
    -k, --key KEY                    API Client Key
        --[no-]color                 Use colored output, defaults to false on
Windows, true
    -c, --config CONFIG              The configuration file to use
        --defaults                   Accept default values for all questions
    -d, --disable-editing            Do not open EDITOR, just accept the data as is
    -e, --editor EDITOR              Set the editor to use for interactive commands
```

GL: Capture Node's Public Host Name and IP

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

```
node1:
  ipaddress: 172.31.8.68
```

Amazon EC2 Instances

The IP address and host name are unfortunately not how we can address these nodes within our recipes.

GL: Capture Node's Public Host Name and IP

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

```
node1:
  cloud:
    local_hostname:  ip-172-31-8-68.ec2.internal
    local_ipv4:      172.31.8.68
    private_ips:     172.31.8.68
    provider:        ec2
    public_hostname: ec2-54-175-46-24.compute-1.amazonaws.com
    public_ips:      54.175.46.24
    public_ipv4:     54.175.46.24
```

GL: Edit the myhaproxy/recipes/default.rb

```
~/chef-repo/cookbooks/myhaproxy/recipes/default.rb
```

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2016 The Authors, All Rights
Reserved.



include_recipe 'haproxy::default'
```

GL: Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
#

node['haproxy']['members'] = [
{
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4000,
    'ssl_port' => 4000
  }, {
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4001,
    'ssl_port' => 4001
}]

include_recipe 'haproxy::default'
```

GL: Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node['haproxy']['members'] = [
  {
    'hostname' => 'localhost',
    'ipaddress' => '127.0.0.1',
    'port' => 4000,
    'ssl_port' => 4000
  },
  {
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',
    'ipaddress' => '52.8.71.11',
    'port' => 80,
    'ssl_port' => 80
  }]

include_recipe 'haproxy::default'
```

GL: Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',
    'ipaddress' => '52.8.71.11',
    'port' => 80,
    'ssl_port' => 80
  }]

include_recipe 'haproxy::default'
```

GL: Edit the myhaproxy/recipes/default.rb

~/chef-repo/cookbooks/myhaproxy/recipes/default.rb

```
node.default['haproxy']['members'] = [{
    'hostname' => 'ec2-52-8-71-11.us-west-1.compute.amazonaws.com',
    'ipaddress' => '52.8.71.11',
    'port' => 80,
    'ssl_port' => 80
  }]

include_recipe 'haproxy::default'
```

Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the haproxy cookbook

Lab: Upload the Cookbook

- ❑ Upload the cookbook to the Chef Server

Lab: Upload the Cookbook

```
$ cd ~/chef-repo/cookbooks/myhaproxy
```

Lab: Upload the Cookbook

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myhaproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using build-essential (2.2.3)
Using cpu (0.2.0)
Using haproxy (1.6.6)
Using myhaproxy (0.1.0) from source at .
```

Lab: Upload the Cookbook

```
$ berks upload
```

```
Uploaded build-essential (2.2.3) to: 'https://api.opscode.com:443/organizations/
steveessentials2'
Uploaded cpu (0.2.0) to: 'https://api.opscode.com:443/organizations/steveessentials2'
Uploaded haproxy (1.6.6) to: 'https://api.opscode.com:443/organizations/steveessentials2'
Uploaded myhaproxy (0.1.0) to: 'https://api.opscode.com:443/organizations/steveessentials2'
```

Lab: Verify the Cookbook Upload

```
$ knife cookbook list
```

```
apache              0.2.1
build-essential     2.2.3
cpu                 0.2.0
haproxy             1.6.6
myhaproxy           0.1.0
workstation         0.2.1
```

Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- ✓ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the haproxy cookbook

Lab: Bootstrap a Load Balancer

- ❑ Bootstrap a new node
- ❑ Update the run list of the new node to include the wrapper proxy server cookbook
- ❑ SSH to that system and run chef-client
- ❑ Verify that traffic to the load balancer is relayed to the web server.

Lab: Bootstrap a New Node

```
$ knife bootstrap FQDN2 -x USER -P PWD --sudo -N node2
```

```
Creating new client for node2
Creating new node for node2
Connecting to ec2-54-210-192-12.compute-1.amazonaws.com
ec2-54-210-192-12.compute-1.amazonaws.com Starting first Chef Client run...
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list: []
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-192-12.compute-1.amazonaws.com [2016-09-16T17:13:10+00:00] WARN:
Node node2 has an empty run list.
ec2-54-210-192-12.compute-1.amazonaws.com Converging 0 resources
ec2-54-210-192-12.compute-1.amazonaws.com
ec2-54-210-192-12.compute-1.amazonaws.com Running handlers:
```

Lab: Validate the New Node

```
$ knife node show node2
```

```
Node Name:   node2
Environment: _default
FQDN:        ip-172-31-0-128.ec2.internal
IP:          54.210.192.12
Run List:
Roles:
Recipes:
Platform:    centos 6.6
Tags:
```

Lab: Define the Run List

```
$ knife node run_list add node2 "recipe[myhaproxy]"
```

```
node2:
  run_list: recipe[myhaproxy]
```

Lab: Validate the Run List

```
$ knife node show node2
```

```
Node Name:   node2
Environment: _default
FQDN:        ip-172-31-0-128.ec2.internal
IP:          54.210.192.12
Run List:    recipe[myhaproxy]
Roles:
Recipes:
Platform:    centos 6.6
Tags:
```

SSH Woes

Logging into both systems is a pain. We can use another knife tool to allow us to send commands to all of our nodes.

GL: Using knife ssh

```
$ knife ssh --help
```

```
knife ssh QUERY COMMAND (options)
    -a, --attribute ATTR             The attribute to use for opening the connection
- default depends on the context
    -s, --server-url URL             Chef Server URL
        --chef-zero-host HOST        Host to start chef-zero on
        --chef-zero-port PORT        Port (or port range) to start chef-zero on.
Port ranges like 1000,1010 or 8889-9999 will try all given ports until one works.
    -k, --key KEY                    API Client Key
        --[no-]color                 Use colored output, defaults to false on
Windows, true otherwise
    -C, --concurrency NUM            The number of concurrent connections
    -c, --config CONFIG              The configuration file to use
        --defaults                   Accept default values for all questions
```

GL: Define the Run List

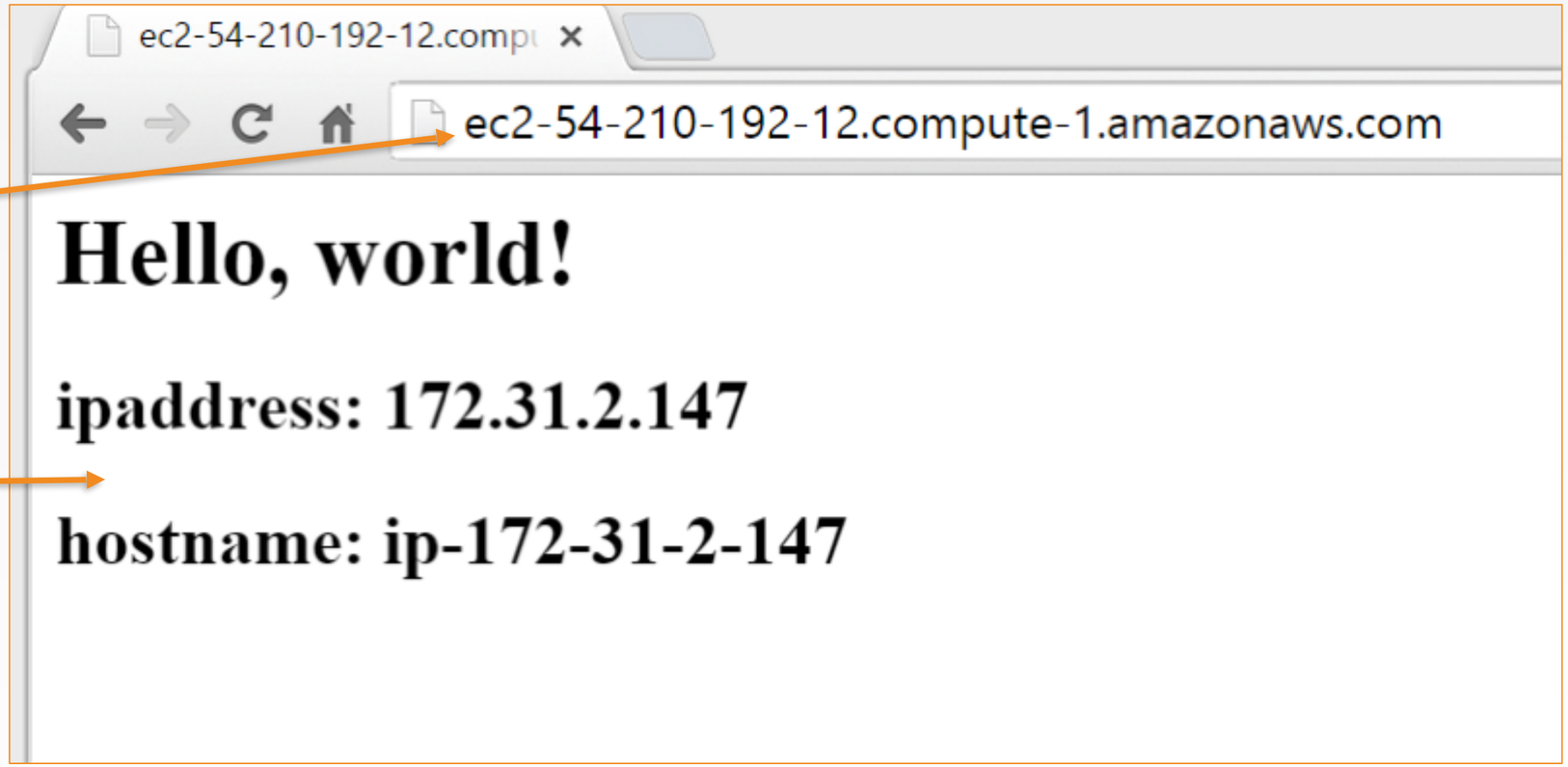
```
$ knife ssh "*:*" -x USERNAME -P PASSWORD "sudo chef-client"
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list:
["apache"]
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-54-175-46-24.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com    - apache
ec2-54-175-46-24.compute-1.amazonaws.com  Compiling Cookbooks...
ec2-54-175-46-24.compute-1.amazonaws.com  Converging 3 resources
ec2-54-175-46-24.compute-1.amazonaws.com  Recipe: apache::server
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com   - build-essential
ec2-54-210-192-12.compute-1.amazonaws.com   - cpu
```

GL: Testing Your Websites

URL of load balancer.

Output from the web server.



Lab: Bootstrap a Load Balancer

- ✓ Bootstrap a new node
- ✓ Update the run list of the new node to include the wrapper proxy server cookbook
- ✓ SSH to that system and run chef-client
- ✓ Verify that traffic to the load balancer is relayed to the web server.

Load Balancer

Adding a load balancer will allow us to better grow our infrastructure.

Objective:

- ✓ Find or create a cookbook to manage a load balancer
- ✓ Configure the load balancer to send traffic to the new node
- ✓ Upload cookbook to Chef Server
- ✓ Bootstrap a new node that runs the haproxy cookbook

Discussion

What are the benefits and drawbacks of the Chef Super Market?

Is your team able to leverage community cookbooks? Is the team able to contribute to community cookbooks?

Why do you use a wrapper cookbook? When might you decide to not wrap the cookbook?

DISCUSSION

Q&A

What questions can we help you answer?

- Chef Super Market
- Wrapper Cookbooks
- Node Attributes
- knife ssh

CHEF

CHEF™