Load Balancer

Scaling up

Objectives

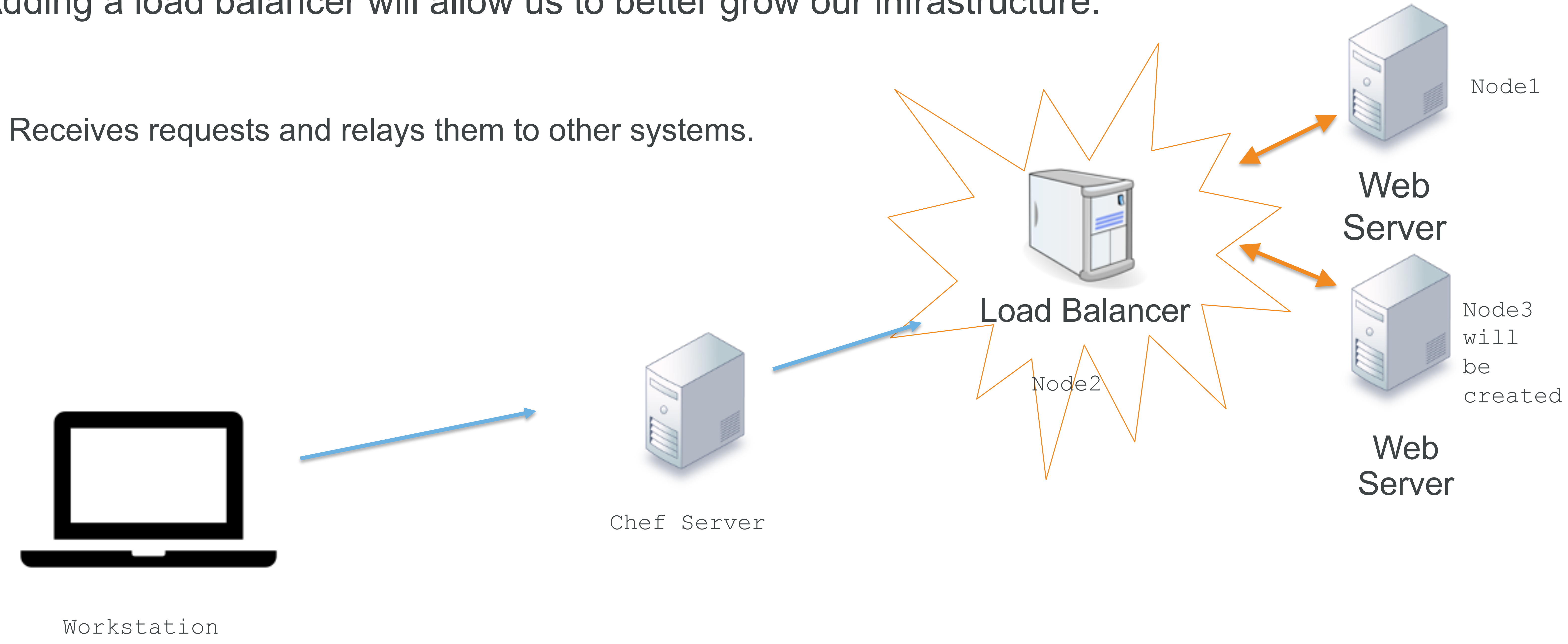
After completing this module, you should be able to

- ➢ Create a haproxy cookbook
- ➢ Add traffic route to existing webserver in node1
- ➢ Upload haproxy cookbook to chef server
- ➢ Bootstrap a node2 for haproxy

Load Balancer

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

Receives requests and relays them to other systems.



Manage Multiple Nodes

Our site has just gotten super busy so we now need to manage multiple nodes.

GL: Scaling up

Our site has just got super busy with multiple web servers – so we now need a load balancer.

Objective:

- ❑ Create a load balancer cookbook
- ❑ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the load balancer cookbook

GL: Generate HAProxy Cookbook

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

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

GL: Edit haproxy cookbook's default recipe

`chef-repo/cookbooks/haproxy/recipes/default.rb`

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

package 'haproxy'

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
end

service 'haproxy' do
  action [:start, :enable]
end
```

CHEF

DISCUSSION

Configure HAProxy

We need to configure HAProxy to route traffic to our web server.

Have a look at how HAProxy is configured

http://bit.ly/1Xoai9R

GL: Generate the Template

```
$ chef generate template cookbooks/haproxy haproxy.cfg
```

```
Compiling Cookbooks...
Recipe: code_generator::template
  * directory[cookbooks/haproxy/templates/default] action create
    - create new directory cookbooks/haproxy/templates/default
  * template[cookbooks/haproxy/templates/default/haproxy.cfg.erb] action create
    - create new file cookbooks/haproxy/templates/default/haproxy.cfg.erb
    - update content in file cookbooks/haproxy/templates/default/haproxy.cfg.erb
from none to e3b0c4
    (diff output suppressed by config)
```

CHEF

GL: Configuring haproxy.cfg.erb

`cookbooks/haproxy/templates/haproxy.cfg.erb`

```
...
frontend  main *:80
    acl url_static       path_beg       -i /static /images /javascript /stylesheets
    acl url_static       path_end       -i .jpg .gif .png .css .js

    use_backend static          if url_static
    default_backend             app

backend static
    balance     roundrobin
    server      static 127.0.0.1:4331 check

backend app
    balance     roundrobin
    server app <<IP ADDRESS>>:80 weight 1 maxconn 100 check
```

Copy the bit.ly URL from below the slide in your participant guide:

http://bit.ly/1Xoai9R

DISCUSSION

Deriving the Public IP Address in AWS

Let's stop for a moment and discuss how we can add a webserver's public IP Addresses to haproxy.cfg using the `cloud` attribute.

CHEF

HAProxy Configuration Should Look Like This

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

We need to add the webserver's IP Addresses to haproxy.cfg

Doing it manually seems wrong

Should be able to Search all desired webserver IPs !!!

HAProxy Configuration Should Look Like This

```
...
backend app
  balance roundrobin
  server app0 <<node1 IP ADDRESS>>:80 weight 1 maxconn 100 check
  server app1 <<node2 IP ADDRESS>>:80 weight 1 maxconn 100 check
```

We will do search capability in haproxy later, let's hard code now and test haproxy cookbook

But we need IP, you can use node1 IP, but there is a better way

DISCUSSION

Introducing Chef Server SOLR search

Chef server use SOLR for indexing and searching.
Node objects, attributes, cookbook details, recipe details – all gets indexed in SOLR and can be searched.
To search IP, we will use “knife search node”

Knife search for IPs

```
$ knife search node "policy_group:prod AND policy_name:company_web"
```

```
1 items found


Node Name:    node1
Policy Name:   company_web
Policy Group: prod
FQDN:         ip-172-31-18-33.us-east-2.compute.internal
IP:           3.16.131.207
Run List:     recipe[company_web::default]
Recipes:      company_web::default, apache::default, apache::server
Platform:     centos 7.8.2003
Tags:
```

DISCUSSION

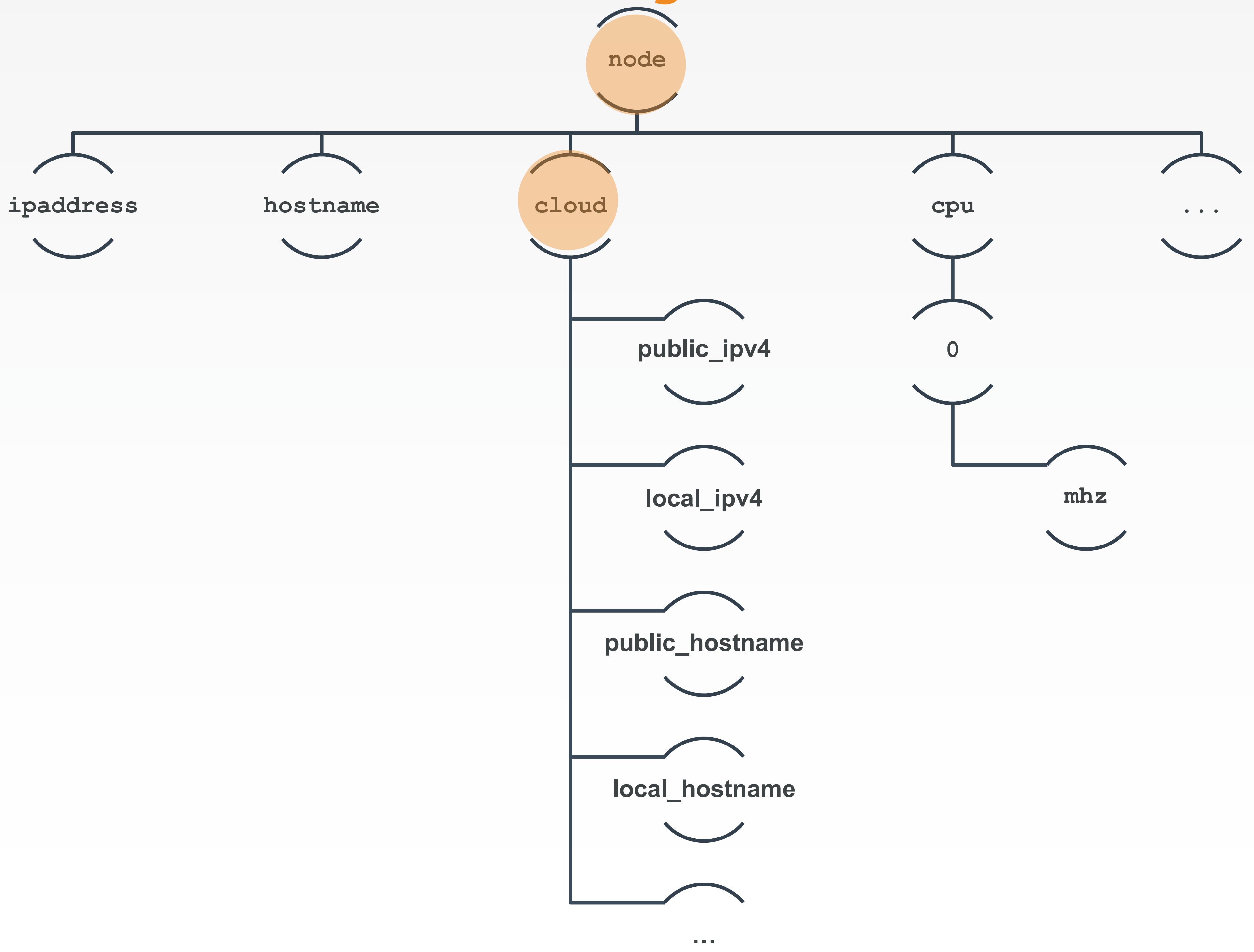
Amazon EC2 Instances

We can't use the `ipaddress` attribute within our recipes – they're on the private (internal) network and we need external access.

In a previous section we looked at the node object.

Nodes in EC2 have some specific networking attributes in their node object – some private & some public.

EC2 and the Node Object



Individual Node's EC2 Information

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

```
node1:
  cloud:
    local_hostname:  ip-172-31-29-218.ec2.internal
    local_ipv4:      172.31.29.218
    private_ips:     172.31.29.218
    provider:        ec2
    public_hostname: ec2-54-88-185-159.compute-1.amazonaws.com
    public_ips:      54.88.185.159
    public_ipv4:     54.88.185.159
```

Individual Node's EC2 Information

```
$ knife node show node1 -a cloud.public_ipv4
```

```
node1:
  cloud.public_ipv4: 3.16.131.207
```

Individual Node's Public IP Address

```
$ knife search node "policy_group:prod AND
policy_name:company_web" -a cloud.public_ipv4
```

```
1 items found

node1:
  cloud.public_ipv4: 3.16.131.207
```

HAProxy Configuration Should Look Like This

```
...
backend app
  balance roundrobin
  server app 3.16.131.207:80 weight 1 maxconn 100 check
```

Scaling up

Our site has just got super busy with multiple web servers – so we now need a load balancer.

✓ Create a load balancer cookbook
❑ Upload cookbook to Chef Server
❑ Bootstrap a new node that runs the load balancer cookbook

Upload the Policyfile

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

LOCAL

Generate Policyfile

```
$ chef generate policyfile haproxy
```

```
Resolving cookbook dependencies...
Fetching 'haproxy' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using haproxy (0.1.0) from source at .
```

Change Policyfile content

```
$ vi ~/chef-repo/haproxy.rb
```

```
# A name that describes what the system you're building with Chef does.
name 'haproxy'

# Where to find external cookbooks:
default_source :supermarket

# run_list: chef-client will run these recipes in the order specified.
run_list 'haproxy::default'

# Specify a custom source for a single cookbook:
cookbook 'haproxy', path: 'cookbooks/haproxy'
```

Install Policyfile

```
$ chef install haproxy.rb
```

```
Building policy haproxy
Expanded run list: recipe[haproxy::default]
Caching Cookbooks...
Installing haproxy >= 0.0.0 from path

Lockfile written to /home/centos/chef-repo/haproxy.lock.json
Policy revision id: 648817382c99f23416f7ff2a3d14da7e812bf6d184bf6e4f810ed49084ba6de1
```

LOCAL

CHEF

Push Policyfile to Server

```
$ chef push prod haproxy.lock.json
```

```
Uploading policy haproxy (648817382c) to policy group prod
Uploaded haproxy 0.1.0 (d699a0c0)
```

CHEF™

Verify the Policyfile Upload

```
$ chef show-policy
```

```
apache
* prod:   ab5d900157


company_web
* prod:   c5590c5d36


haproxy
* prod:   648817382c


workstation
* prod:   2e5817026e
```

Lab: Scaling up

Our site has just got super busy with multiple web servers – so we now need a load balancer.

Objective:

- ✓ Create a load balancer cookbook
- ✓ Upload cookbook to Chef Server
- ❑ Bootstrap a new node that runs the load balancer cookbook

Bootstrap a Load Balancer Node

```
$ knife bootstrap <ip_node2> -x centos --sudo -i <aws.pem path> -N node2 --
policy-name haproxy --policy-group prod
```

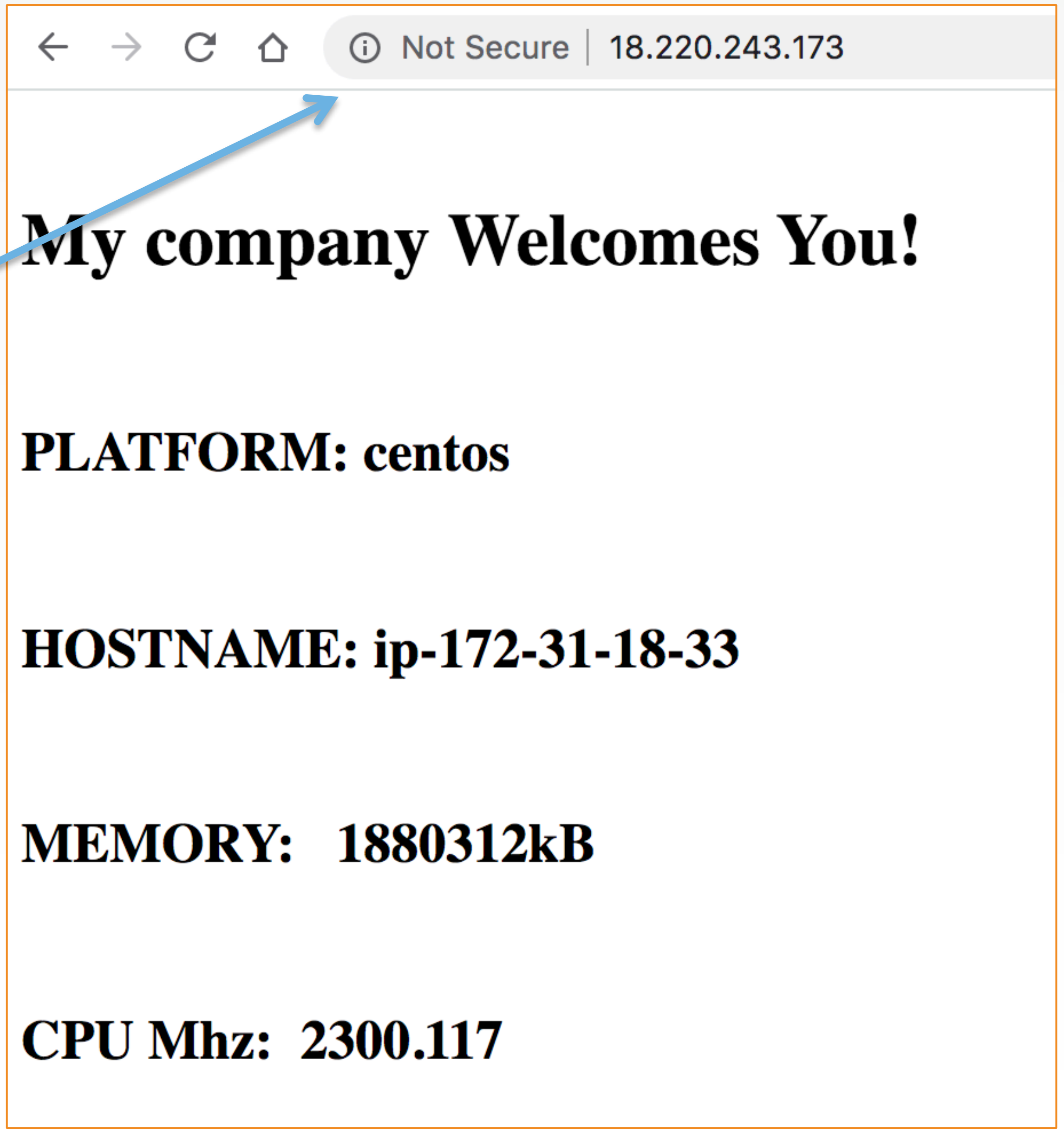
```
This will bootstrap a node, and also apply haproxy policy
```

Validate the New Node

```
$ knife node show node2
```

```
Node Name:    node2
Policy Name:  haproxy
Policy Group: prod
FQDN:         ip-172-31-19-183.us-east-2.compute.internal
IP:           18.220.243.173
Run List:     recipe[haproxy::default]
Recipes:      haproxy::default
Platform:     centos 7.8.2003
Tags:
```

Validate Load Balancer


Not Secure | 18.220.243.173

My company Welcomes You!

PLATFORM: centos

HOSTNAME: ip-172-31-18-33

MEMORY: 1880312kB

CPU Mhz: 2300.117

Load balancer IP

Routing to webserver

Getting Second Webserver

Objective:

- Bootstrap third node – our second websever
- Manually change loadbalancer
- Upload to server
- Test Load balancer

Bootstrap a web server Node

```
$ knife bootstrap <ip_node3> -x centos --sudo -i <aws.pem path> -N node3 --
policy-name company_web --policy-group prod
```

```
This will bootstrap a node, and also apply haproxy policy
```

Validate the New Node

```
$ knife node show node3
```

```
Node Name:    node3
Policy Name:   company_web
Policy Group: prod
FQDN:         ip-172-31-24-5.us-east-2.compute.internal
IP:           52.15.221.52
Run List:     recipe[company_web::default]
Recipes:      company_web::default, apache::default, apache::server
Platform:     centos 7.8.2003
Tags:
```

Use this IP
Add to loadbalancer

Upgrading Load Balancer

Objective:

- Manually change loadbalancer
- Upload to server
- Test Load balancer

HAProxy Configuration Should Look Like This

```
$ ~/chef-repo/cookbooks/haproxy/templates/haproxy.cfg.erb
```

```
...
backend app
  balance roundrobin
  server app0 3.16.131.207:80 weight 1 maxconn 100 check
  server app1 52.15.221.52:80 weight 1 maxconn 100 check
```

Upload the metadata.rb

```
$ ~/chef-repo/cookbooks/haproxy/metadata.rb
```

```
name 'haproxy'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'All Rights Reserved'
description 'Installs/Configures haproxy'
long_description 'Installs/Configures haproxy'
version '0.2.0'
chef_version '>= 14.0'
```

Change version, very important !!

Go to chef-repo

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

LOCAL

Update Policyfile

```
$ chef update haproxy.rb
```

```
Attributes already up to date
Building policy haproxy
Expanded run list: recipe[haproxy::default]
Caching Cookbooks...
Installing haproxy >= 0.0.0 from path

Lockfile written to /home/centos/chef-repo/haproxy.lock.json
Policy revision id:
f501a623a03dec00478641e67ac60a72ccc813888daf22e1a42783ff4182e061
```

LOCAL

Push Policyfile to Chef Server

```
$ chef push prod haproxy.lock.json
```

```
Uploading policy haproxy (f501a623a0) to policy group prod
Uploaded haproxy 0.2.0 (ae0efc25)
```

Updated version now pushed to the chef server

CHEF

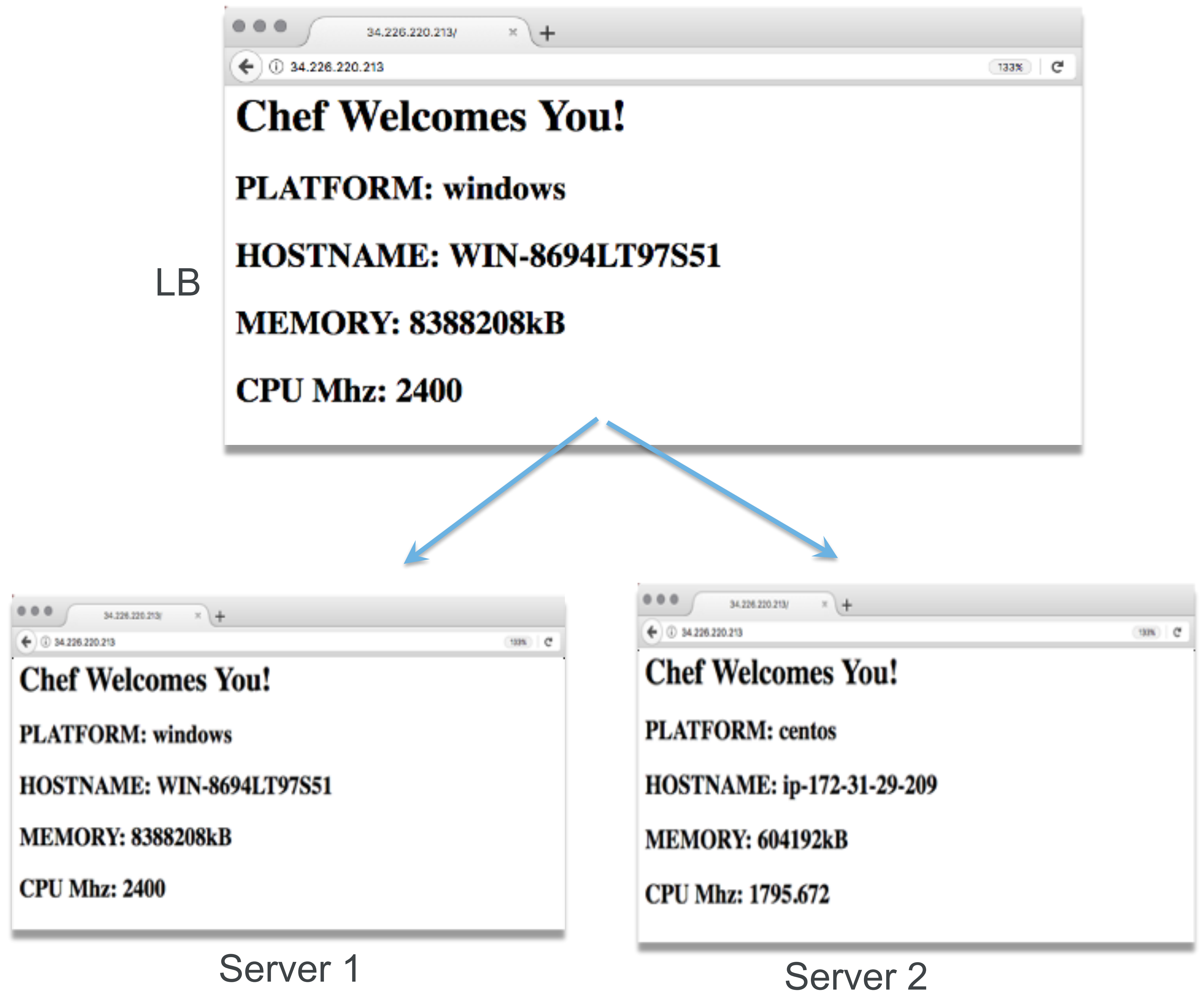
Apply new changes to LB node

```
$ knife ssh <ip_load balancer> -m -x centos -i
<aws.pem path> "sudo chef-client"
```

```
Building policy haproxy
Expanded run list: recipe[haproxy::default]
Caching Cookbooks...
Installing haproxy >= 0.0.0 from path

Lockfile written to /home/centos/chef-repo/haproxy.lock.json
Policy revision id: 648817382c99f23416f7ff2a3d14da7e812bf6d184bf6e4f810ed49084ba6de1
```

Lab: Test the Load Balancer



```
34.226.220.213/
34.226.220.213
133%
```

Chef Welcomes You!

PLATFORM: windows

HOSTNAME: WIN-8694LT97S51

MEMORY: 8388208kB

CPU Mhz: 2400

Chef Welcomes You!

PLATFORM: windows

HOSTNAME: WIN-8694LT97S51

MEMORY: 8388208kB

CPU Mhz: 2400

Chef Welcomes You!

PLATFORM: centos

HOSTNAME: ip-172-31-29-209

MEMORY: 604192kB

CPU Mhz: 1795.672

Note:
Haproxy reads from cache.
Thus, any change in config needs a restart of the service.
You can ssh to the LB node and run "sudo systemctl restart haproxy" to restart the haproxy service

-- We will do this in a much elegant way in the next chapter, as it needs some background of **knife search**

CHEF™