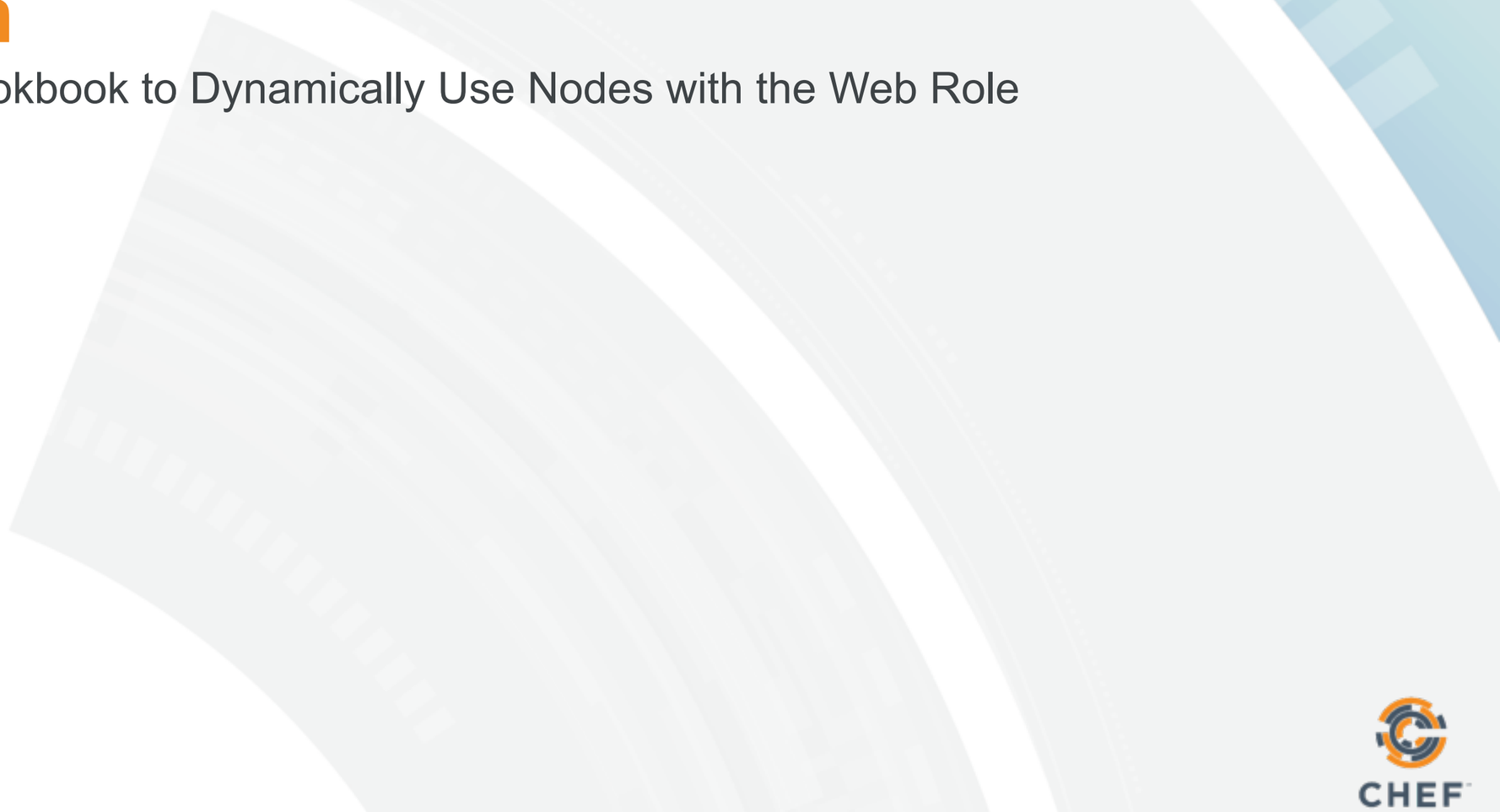
Search

Update a Cookbook to Dynamically Use Nodes with the Web Role



Objectives

After completing this module, you should be able to

- Describe the query syntax used in search
- Build a search into your recipe code
- Create a Ruby Array and Ruby Hash
- Update the myhaproxy wrapper cookbook (for the load balancer) to dynamically use nodes with the web role

Search

So far we have seen how Chef is able to manage the policy of the nodes.

We have two web servers and one load balancer.

Search

To add new servers as load balancer members, we would need to bootstrap a new web server and then update our load balancer's myhaproxy cookbook recipe.

That seems inefficient to have to update a cookbook recipe.

The Chef Server and Search

Chef Server maintains a representation of all the nodes within our infrastructure that can be searched on.

Search is a service discovery tool that allows us to query the Chef Server.

Chef Server

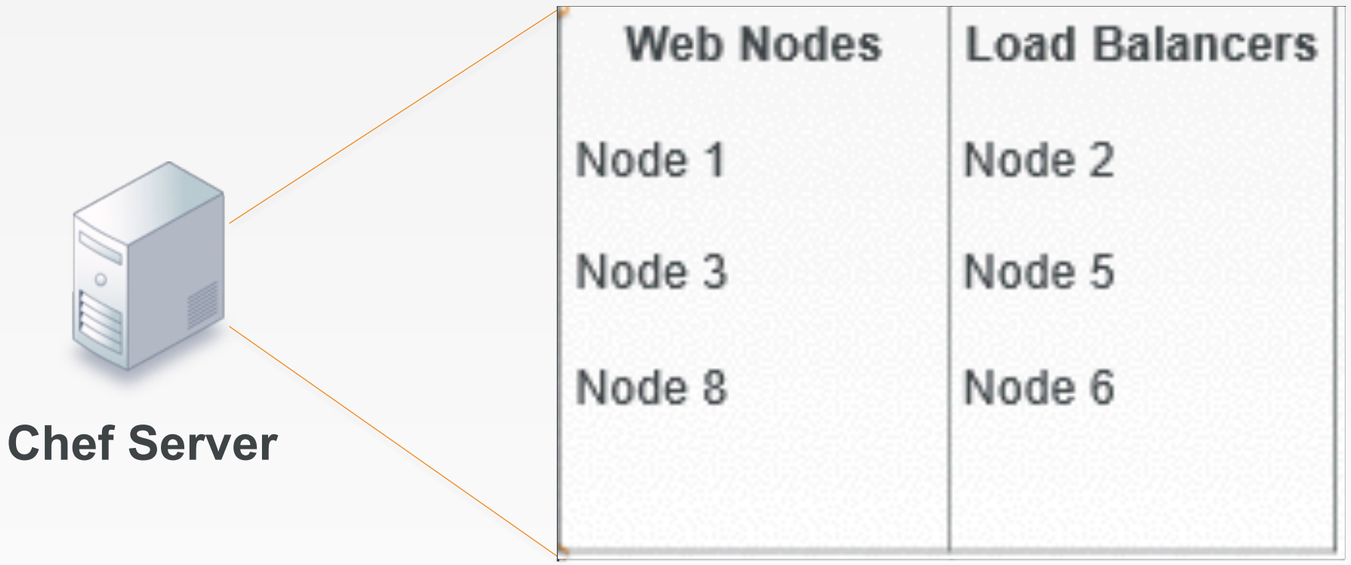
Web Nodes	Load Balancers
Node 1	Node 2
Node 3	Node 5
Node 8	Node 6

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

https://docs.chef.io/chef_search.html#search-indexes

The Chef Server and Search

We can ask the Chef Server to return all the nodes or a subset of nodes based on the query syntax that we provide it through `knife search` or within our recipes through `search`.



CHEF

Search Criteria

The search criteria that we have been using up to this point is "*:*"

Querying and returning every node is not what we need to solve our current problem.

Scenario: We want only to return a subset of our nodes--only the nodes that are webservers.

Search Syntax

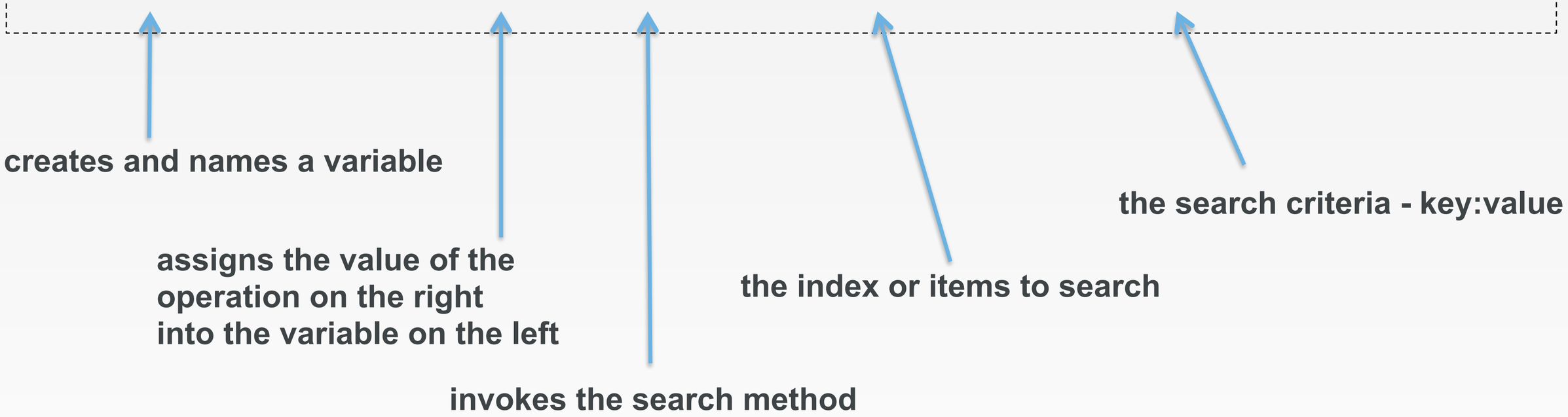
A search query is comprised of two parts: the key and the search pattern. A search query has the following syntax:

key:search_pattern

...where key is a field name that is found in the JSON description of an indexable object on the Chef server and search_pattern defines what will be searched for,

Search Syntax within a Recipe

```
all_web_nodes = search('node','role:web')
```



CHEF

Search Syntax within a Recipe

```
all_web_nodes = search('node','role:web')
```

Search the Chef Server for all node objects that have the role equal to 'web' and store the results into a local variable named "all_web_nodes'.

Hard Coding Example

~/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
  },
  {
    'hostname'  => 'ec2-54-176-64-173.us-west-1.compute.amazonaws.com',
    'ipaddress' => '54.175.46.48',
    'port' => 80,
    'ssl_port' => 80
  }
]
include_recipe 'haproxy::default'
```

GL: Dynamic Web Load Balancer

Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!

Objective:

- ❑ Update the myhaproxy cookbook to dynamically use nodes with the web role

GL: Showing node1 Cloud Attributes

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

```
node1:
  cloud:
    local_hostname:  ip-10-198-51-26.us-west-1.compute.internal
    local_ipv4:      10.198.51.26
    private_ips:     10.198.51.26
    provider:        ec2
    public_hostname: ec2-204-236-155-223.us-west-1.compute.amazonaws.com
    public_ips:      204.236.155.223
    public_ipv4:     204.236.155.223
```

GL: Showing node3 Cloud Attributes

```
$ knife node show node3 -a cloud
```

```
node3:
  cloud:
    local_hostname:  ip-10-197-105-148.us-west-1.compute.internal
    local_ipv4:      10.197.105.148
    private_ips:     10.197.105.148
    provider:        ec2
    public_hostname: ec2-54-176-64-173.us-west-1.compute.amazonaws.com
    public_ips:      54.176.64.173
    public_ipv4:     54.176.64.173
```

GL: Remove the Hard-coded Members

~/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
  },
  {
    'hostname'  => 'ec2-54-176-64-173.us-west-1.compute.amazonaws.com',
    'ipaddress' => '54.175.46.48',
    'port' => 80,
    'ssl_port' => 80
  }
]
include_recipe 'haproxy::default'
```

GL: Use Search to Identify the Members

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

```
all_web_nodes = search('node','role:web')

#TODO: Convert all found nodes into hashes with ipaddress,
#      hostname, port, ssl_port
#TODO: Assign all the hashes to the node's haproxy members
#      attribute.

include_recipe 'haproxy::default'
```

Creating an Array to Store the Converted Members

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

```
all_web_nodes = search('node','role:web')

members = []

#TODO: Convert all found nodes into hashes with ipaddress,
#      hostname, port, ssl_port

node.default['haproxy']['members'] = members

include_recipe 'haproxy::default'
```

Populating the Members with Each New Member

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

```
all_web_nodes = search('node','role:web')

members = []

all_web_nodes.each do |web_node|
  member = {}
  # TODO: Populate the hash with hostname, ipaddress, port, and
  #       ssl_port
  members.push(member)
end

node.default['haproxy']['members'] = members

include_recipe 'haproxy::default'
```

Populating the Hash with Node Details

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

```
# ... BEFORE THE LOOP IN THE RECIPE ...

all_web_nodes.each do |web_node|
  member = {
    'hostname' => web_node['cloud']['public_hostname'],
    'ipaddress' => web_node['cloud']['public_ipv4'],
    'port' => 80,
    'ssl_port' => 80
  }
  members.push(member)
end

# ... AFTER THE LOOP IN THE RECIPE ...
```

The Final Recipe

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

```
all_web_nodes = search('node','role:web')

members = []

all_web_nodes.each do |web_node|
  member = {
    'hostname' => web_node['cloud']['public_hostname'],
    'ipaddress' => web_node['cloud']['public_ipv4'],
    'port' => 80,
    'ssl_port' => 80
  }
  members.push(member)
end

node.default['haproxy']['members'] = members

include_recipe 'haproxy::default'
```

Dynamic Web Load Balancer

Every time we create a web node we need to update our load balancer (myhaproxy) cookbook. That doesn't feel right!

Objective:

- ✓ Update the myhaproxy cookbook to dynamically use nodes with the web role

Lab: Upload the Cookbook

- ❑ Update the major version of the myhaproxy cookbook
- ❑ Upload the cookbook
- ❑ Run chef-client on the load balancer node
- ❑ Verify that the load balancer node relays requests to both web nodes

Lab: Update the Version Number

~/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             '1.0.0'

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

Lab: CD and Install Dependencies

```
$ cd ~/chef-repo/cookbooks/myhaproxy
$ 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 (1.0.0) from source at .
```

Lab: Upload the Cookbook

```
$ berks upload
```

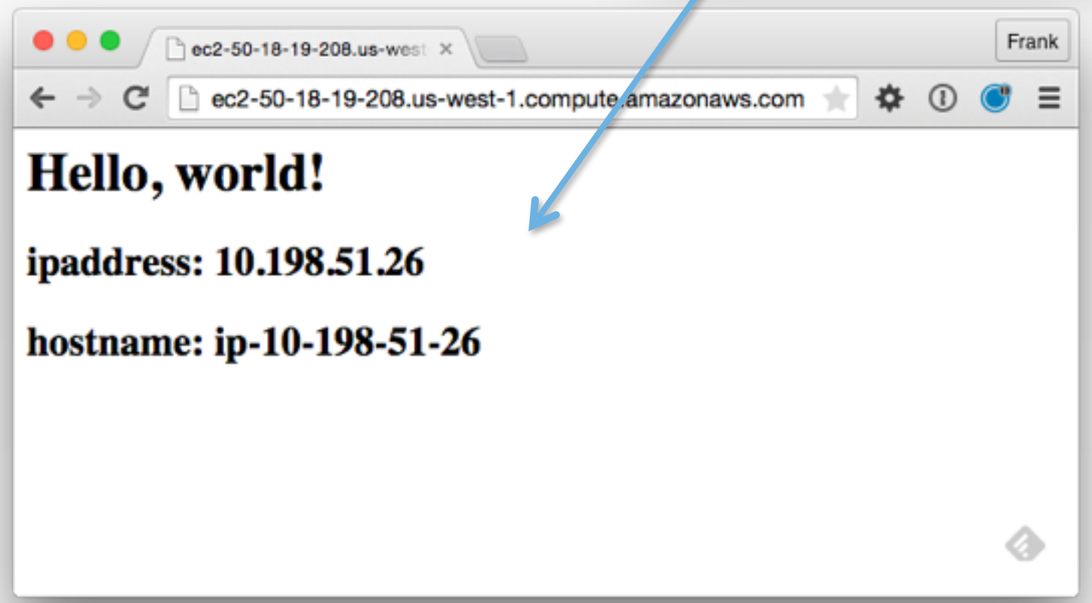
```
Uploaded build-essential (2.2.3) to: 'https://api.opscode.com:443/organizations/
steveessentials2'
Uploaded cpu (0.2.0) to: 'https://api.opscode.com:443/organizations/steveessentials2'
Uploaded haproxy (1.6.6) to: 'https://api.opscode.com:443/organizations/
steveessentials2'
Uploaded myhaproxy (1.0.0) to: 'https://api.opscode.com:443/organizations/
steveessentials2'
PS C:\Users\sdelfante\chef-repo\cookbooks\myhaproxy>
```

Lab: Run the 'knife ssh' Command

```
$ knife ssh "role:load_balancer" -x USER -P PWD "sudo chef-client"
```

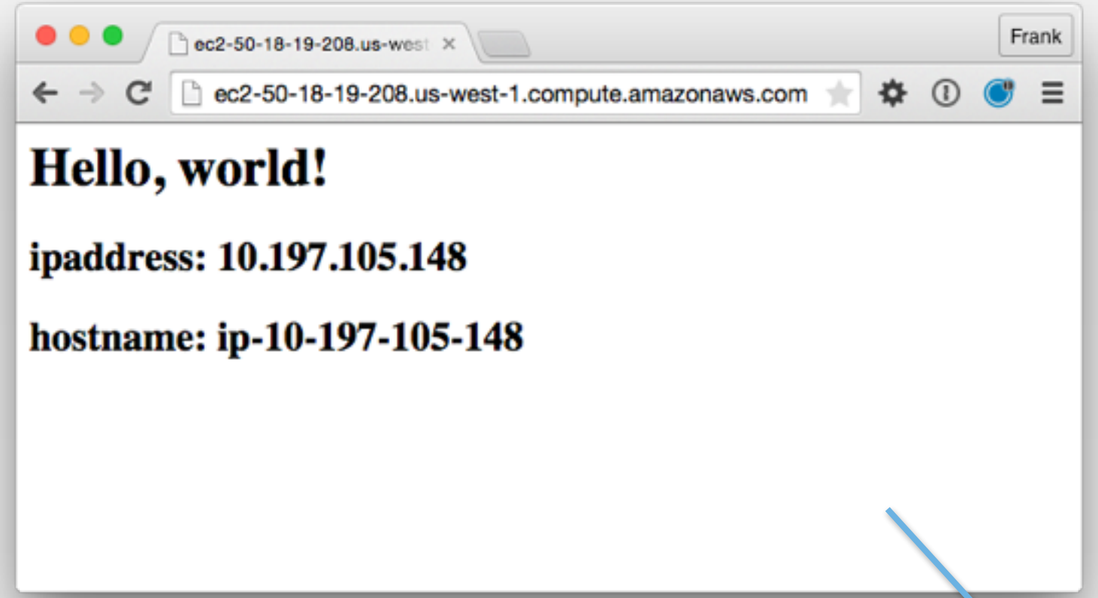
```
ec2-54-210-192-12.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-192-12.compute-1.amazonaws.com resolving cookbooks for run list: ["myhaproxy"]
ec2-54-210-192-12.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-192-12.compute-1.amazonaws.com   - build-essential
ec2-54-210-192-12.compute-1.amazonaws.com   - cpu
ec2-54-210-192-12.compute-1.amazonaws.com   - haproxy
ec2-54-210-192-12.compute-1.amazonaws.com   - myhaproxy
ec2-54-210-192-12.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-192-12.compute-1.amazonaws.com Converging 9 resources
ec2-54-210-192-12.compute-1.amazonaws.com Recipe: haproxy::install_package
ec2-54-210-192-12.compute-1.amazonaws.com   * yum_package[haproxy] action install (up to date)
```


ec2-204-236-155-223.us-w
Frank
ec2-204-236-155-223.us-west-1.compute.amazonaw...
Hello, world!
ipaddress: 10.198.51.26
hostname: ip-10-198-51-26

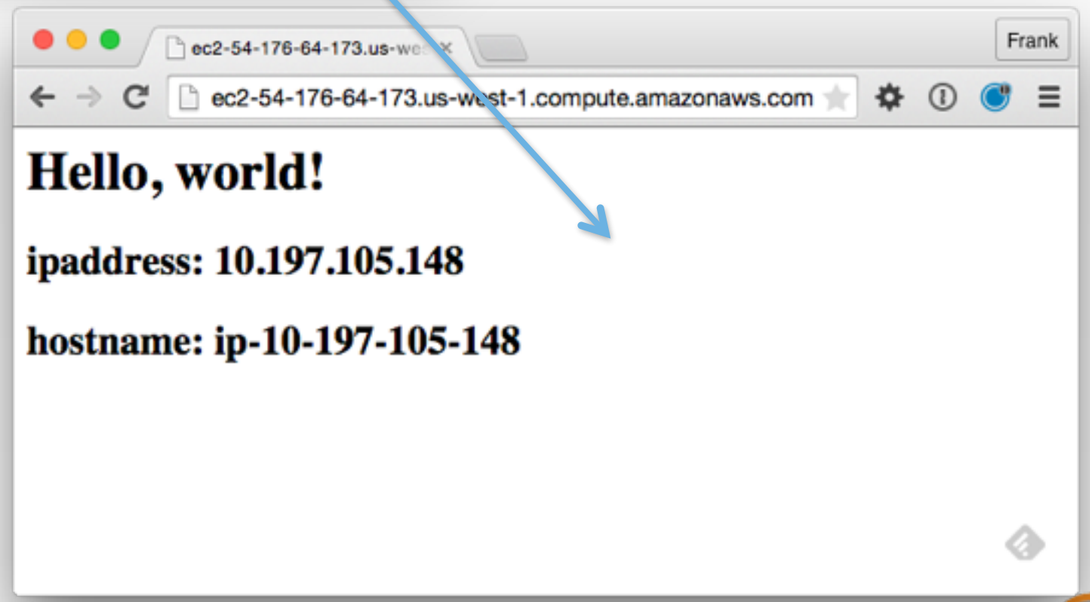

ec2-50-18-19-208.us-west
Frank
ec2-50-18-19-208.us-west-1.compute.amazonaws.com
Hello, world!
ipaddress: 10.198.51.26
hostname: ip-10-198-51-26


ec2-54-176-64-173.us-wes
Frank
ec2-54-176-64-173.us-west-1.compute.amazonaws.com
Hello, world!
ipaddress: 10.197.105.148
hostname: ip-10-197-105-148

CHEF


ec2-50-18-19-208.us-west
Frank
ec2-50-18-19-208.us-west-1.compute.amazonaws.com
Hello, world!
ipaddress: 10.197.105.148
hostname: ip-10-197-105-148


ec2-50-18-19-208.us-west
Frank
ec2-50-18-19-208.us-west-1.compute.amazonaws.com
Hello, world!
ipaddress: 10.198.51.26
hostname: ip-10-198-51-26


ec2-54-176-64-173.us-we
Frank
ec2-54-176-64-173.us-west-1.compute.amazonaws.com
Hello, world!
ipaddress: 10.197.105.148
hostname: ip-10-197-105-148

CHEF

Lab: Upload the Cookbook

- ✓ Update the major version of the myhaproxy cookbook
- ✓ Upload the cookbook
- ✓ Run chef-client on the load balancer node
- ✓ Verify that the load balancer node relays requests to both web nodes

Discussion

What happens when new web nodes are added to the organization? Removed?

What happens if you were to terminate a web node instance without removing it from the Chef Server?

CHEF™