Templates, Variables & Search

Creating Dynamic Content

CHEF

Objectives

After completing this module, you should be able to

- Set the runlist while bootstrapping a node
- Identify EC2 specific node attributes
- Execute a search using knife and within a recipe

Manage multiple nodes

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

CHEF

Lab: Lets bootstrap a new node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node2 -r 'recipe[apache]'
```

```
...
54.84.233.7     +    <h2>ipaddress: 172.31.29.219</h2>
54.84.233.7     +    <h2>hostname: ip-172-31-29-219</h2>
54.84.233.7     +</body>
54.84.233.7     +</html>
54.84.233.7   * service[httpd] action enable
54.84.233.7     - enable service service[httpd]
54.84.233.7   * service[httpd] action start
54.84.233.7     - start service service[httpd]
54.84.233.7
54.84.233.7 Running handlers:
54.84.233.7 Running handlers complete
54.84.233.7 Chef Client finished, 4/4 resources updated in 24.447046971 seconds
```

Lab: Lets bootstrap a new node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node2 -r 'recipe[apache]'
```

```
...
54.84.233.7     +    <h2>ipaddress: 172.31.29.219</h2>
54.84.233.7     +    <h2>hostname: ip-172-31-29-219</h2>
54.84.233.7     +</body>
54.84.233.7     +</html>
54.84.233.7   * service[httpd] action enable
54.84.233.7     - enable service service[httpd]
54.84.233.7   * service[httpd] action start
54.84.233.7     - start service service[httpd]
54.84.233.7
54.84.233.7 Running handlers:
54.84.233.7 Running handlers complete
54.84.233.7 Chef Client finished, 4/4 resources updated in 24.447046971 seconds
```

DISCUSSION

Test the Webservers

To test these web servers we need to find their FQDN or public IP Address

CHEF

Lab: List all our nodes

```
$ knife node list
```

```
node1
node2
```

Lab: View More Information About Nodes

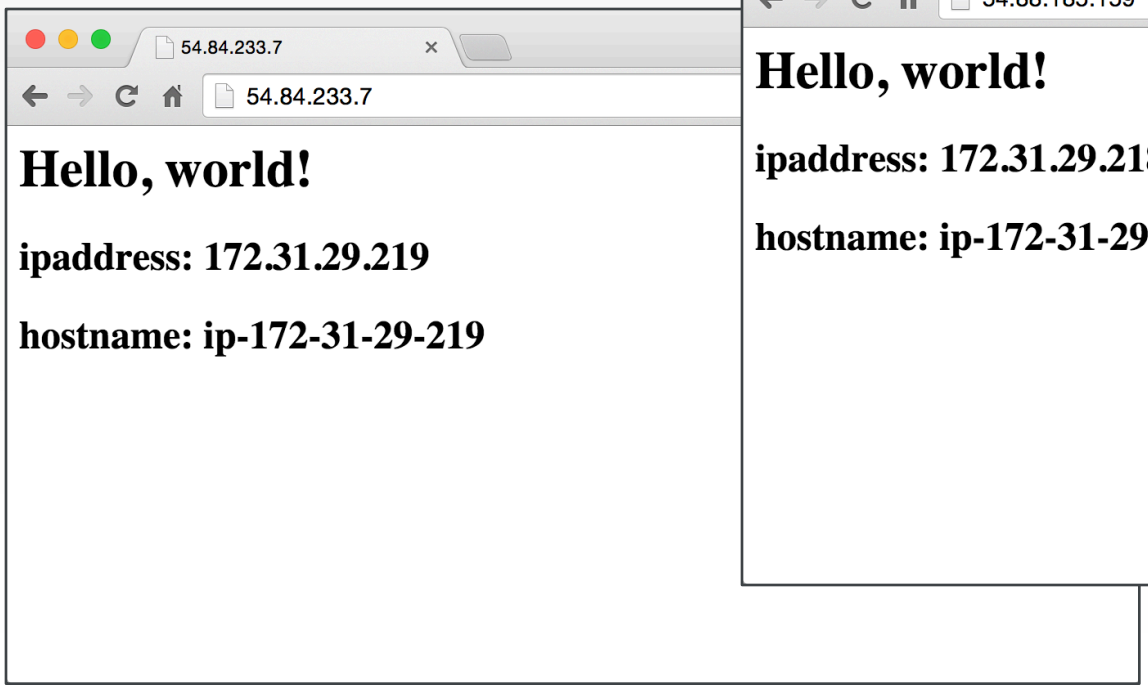
```
$ knife node show node1
```

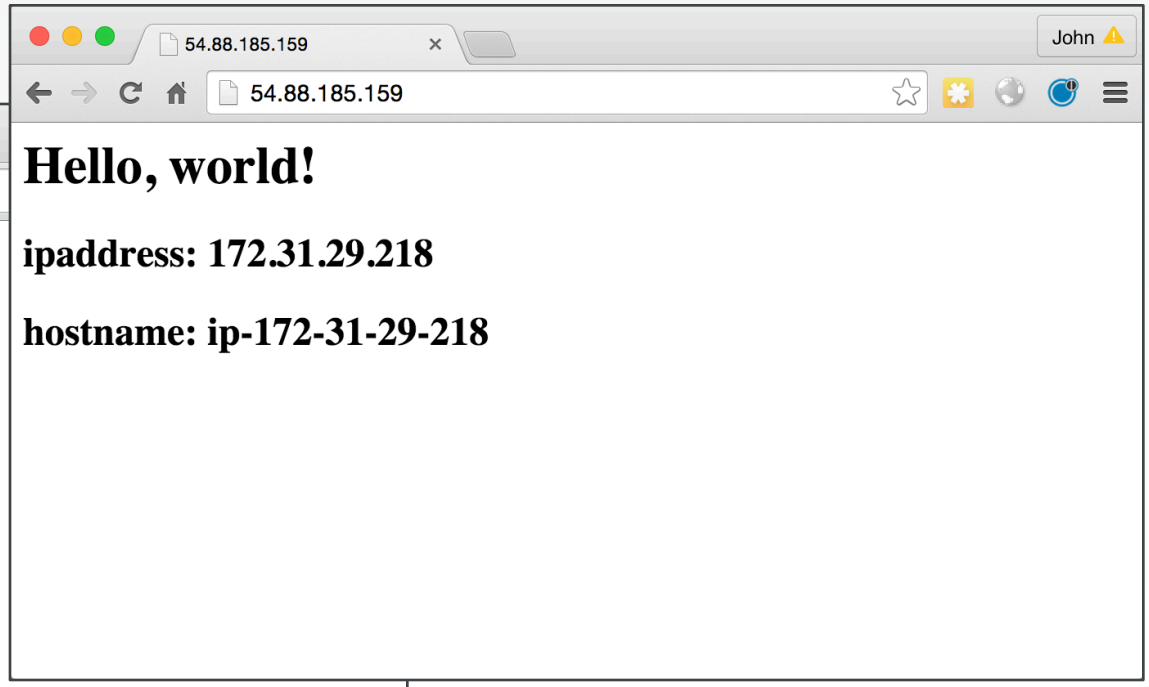
```
Node Name:   node2
Environment: _default
FQDN:        ip-172-31-29-218.ec2.internal
IP:          54.88.185.159
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:
```

```
$ knife node show node2
```

```
Node Name:   node2
Environment: _default
FQDN:        ip-172-31-29-219.ec2.internal
IP:          54.84.233.7
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:
```

Lab: Testing our websites





CHEF

How do we see detail across all nodes?

Ugh. `knife node show` only works with individual nodes – how do I see this info for multiple nodes

CHEF

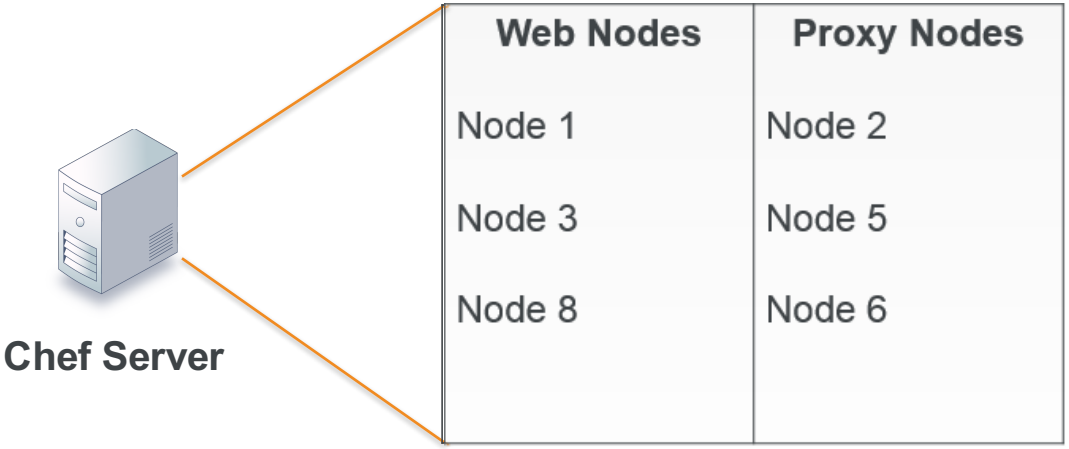
How do we see detail across all nodes?

Ugh. `knife node show` only works with individual nodes – how do I see this info for multiple nodes

Search allows you to retrieve information across multiple nodes

The Chef Server and Search

Chef Server maintains a searchable index of all nodes within our infrastructure



What is search?

Use search to query data indexed on the chef server

```
$ knife search INDEX SEARCH_QUERY
```

The search runs on the server and is invoked from within a recipe or using knife

`INDEX` can be 'client', 'environment', 'node', 'role', (or the name of a data bag)

`SEARCH_QUERY` is of the format "attribute:value"

Querying **`*:*`** returns everything

CHEF

Lab: View information for all nodes

```
$ knife search node "*:*"
```

```
2 items found

Node Name:   node1
Environment: _default
FQDN:        ip-172-31-29-218.ec2.internal
IP:          54.88.185.159
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:

Node Name:   node2
Environment: _default
FQDN:        ip-172-31-29-219.ec2.internal
IP:          54.84.233.7
Run List:    recipe[apache]
Roles:
Recipes:     apache::default, apache::server
Platform:    centos 6.7
Tags:
```

Lab: Narrow the search

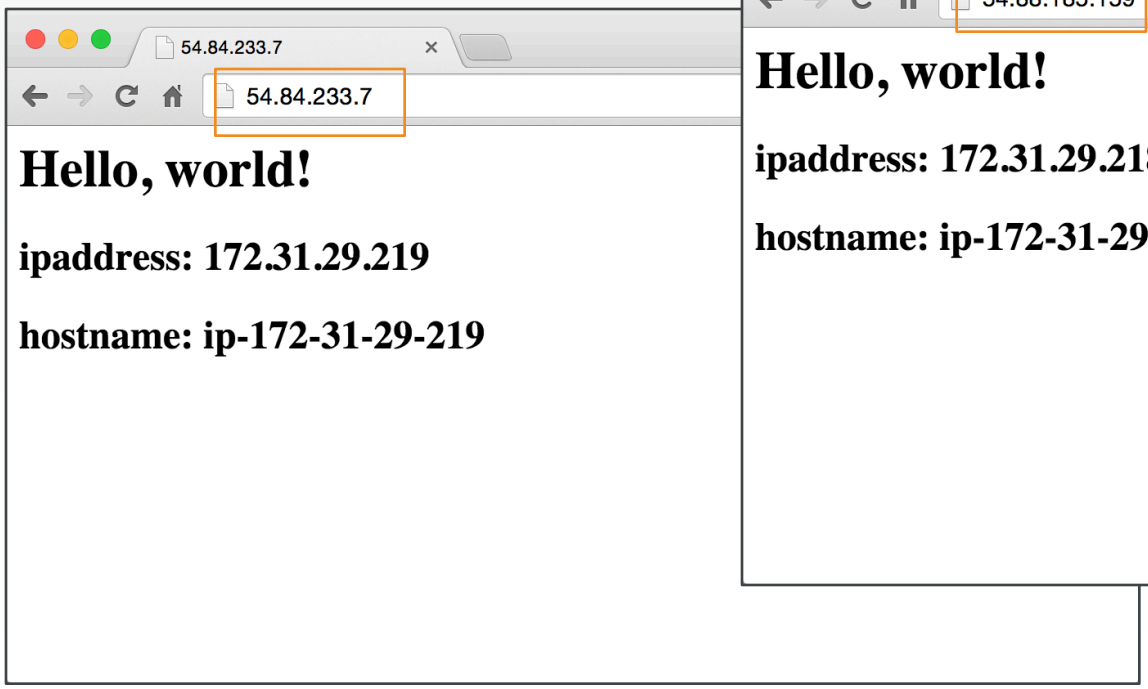
```
$ knife search node "*:*" -a ipaddress
```

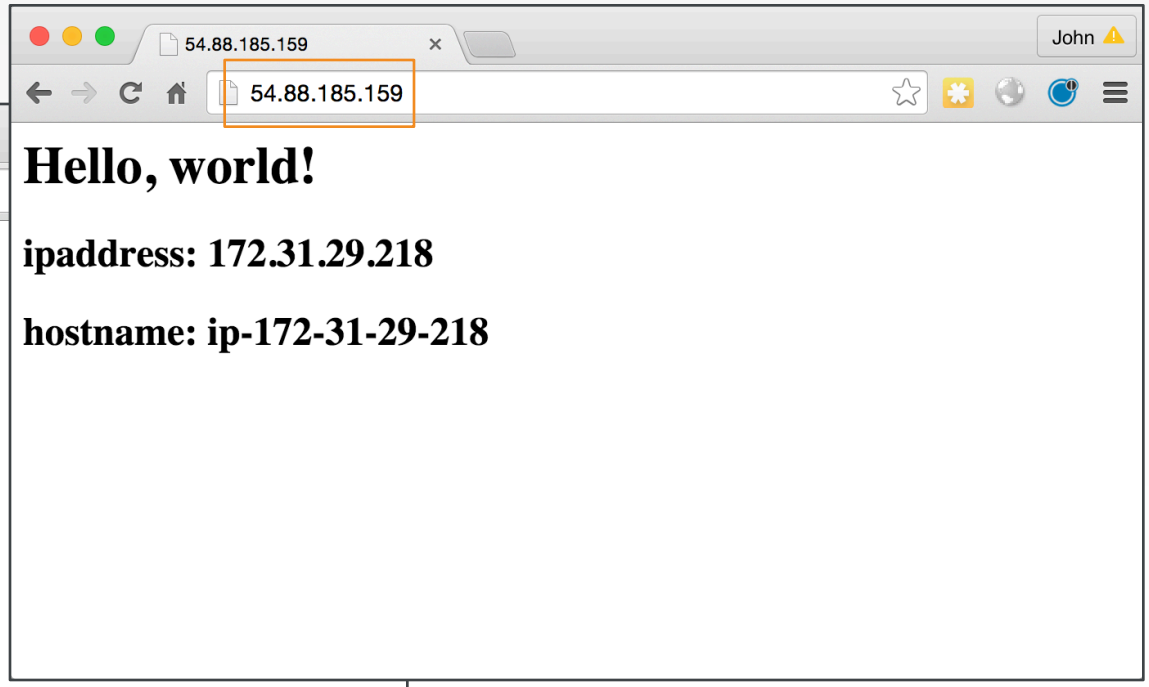
```
2 items found

node1:
  name: 172.31.13.2

node2:
  name: 172.31.6.173
```

Back to our websites…

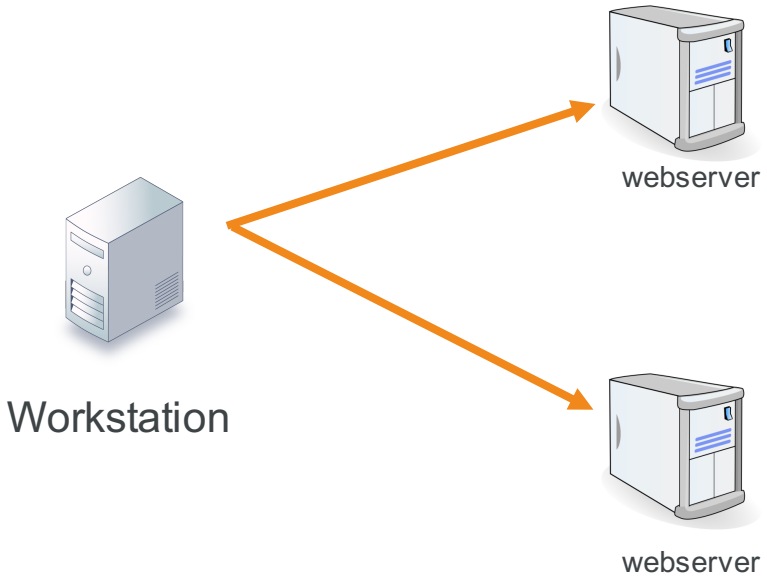




We have to browse to each site individually, ugh

Two webservers, one browser

So we have scaled out our webservers, but we still need to hit each webserver individually

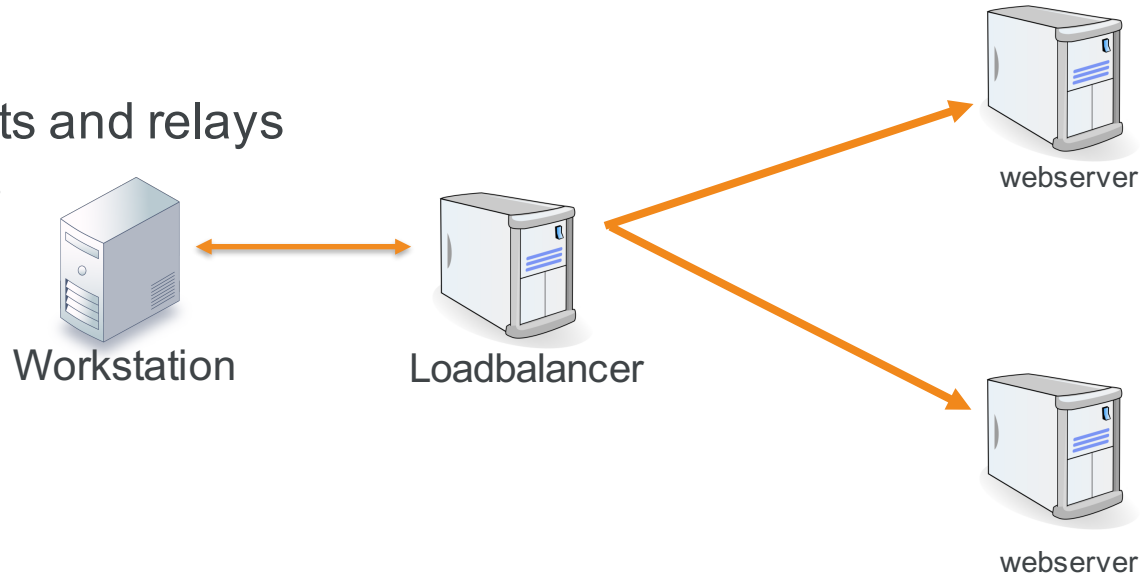


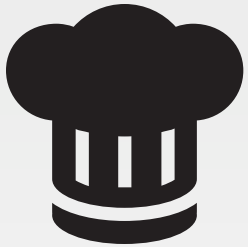
CHEF

Load balancer

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

The LB receives requests and relays them to the web servers





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

Lab: 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
```

Lab: Edit haproxy cookbook's default recipe

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

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2015 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
```

Lab: Edit haproxy cookbook's default recipe

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

```
#
# Cookbook Name:: myhaproxy
# Recipe:: default
#
# Copyright (c) 2015 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
```

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

CHEF

Lab: Generate the Template

```
$ cd chef-repo
$ 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)
```

Lab: Configuring haproxy.cfg.erb

cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...
frontend  main *:5000
    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
```

If you are feeling hardcore, type it http://bit.ly/1Xoai9R

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

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

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

Search!

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
```

Values from

```
knife search node "recipes:apache\:\:default" -a ipaddress
```

CHEF

Heuston, we have a problem!

```
$ knife search node "recipes:apache\:\:default" -a ipaddress
```

```
2 items found

node1:
  ipaddress: 172.31.29.218

node2:
  ipaddress: 172.31.29.219
```

- These IP Addresses are not accessible from the outside network

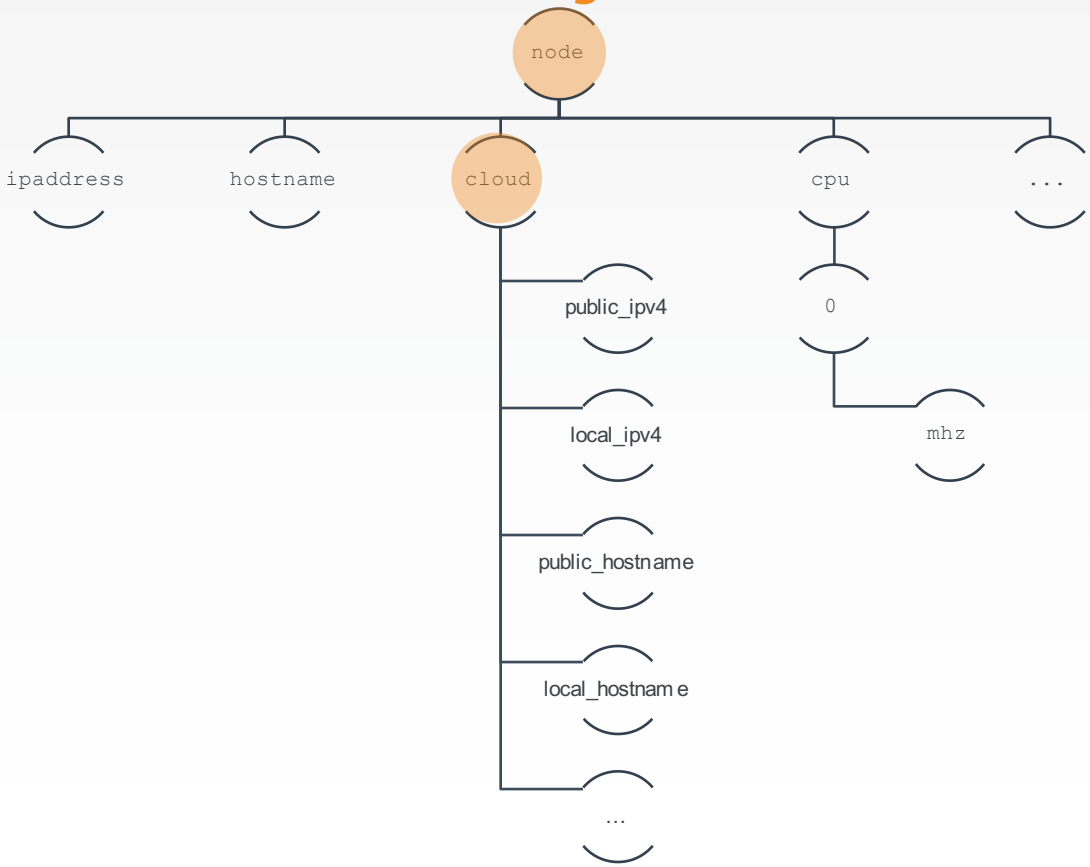
Amazon EC2 Instances

We can't use the `ipaddress` attribute within our recipes – they're on the private (internal) network & 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 Public IP Address

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

```
node1:
  cloud.public_ipv4: 54.88.185.159
```

View EC2 information for all nodes

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

```
2 items found

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

node2:
  cloud:
    local_hostname:  ip-172-31-29-219.ec2.internal
    local_ipv4:      172.31.29.219
...
```

View Public IP for all nodes

```
$ knife search node "*:*" -a cloud.public_ipv4
```

```
2 items found

node1:
  cloud.public_ipv4: 54.88.185.159

node2:
  cloud.public_ipv4: 54.84.233.7
```

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
```

Values from

```
knife search node "recipes:apache\:\:default" -a cloud.public_ipv4
```

CHEF

Lab: Edit haproxy cookbook's default recipe

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

```
...
package 'haproxy'

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  variables(
    :webservers => webservers
  )
 notifies :restart, 'service[haproxy]'
end

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

Lab: Edit haproxy cookbook's default recipe

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

```
...
package 'haproxy'

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  variables(
    :webservers => webservers
  )
 notifies :restart, 'service[haproxy]'
end

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

Invoke a search in the recipe and save the results in the variable 'webservers'

Lab: Edit haproxy cookbook's default recipe

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

```
...
package 'haproxy'

webservers = search('node', 'recipes:apache\:\:default')

template '/etc/haproxy/haproxy.cfg' do
  source 'haproxy.cfg.erb'
  variables(
    :webservers => webservers
  )
 notifies :restart, 'service[haproxy]'
end

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

Pass the variable 'webservers' into the template

Lab: Configuring haproxy.cfg.erb

cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...
    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
    <% @webservers.each_with_index do |web, n| -%>
      server <%= "app#{n}" %> <%= web['cloud']['public_ipv4'] %>:80 weight 1 maxconn 100 check
    <% end -%>
```

Remove line in template with hardcoded IP placeholder

Lab: Configuring haproxy.cfg.erb

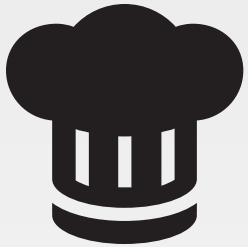
cookbooks/haproxy/templates/default/haproxy.cfg.erb

```
...
    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
    <% @webservers.each_with_index do |web, n| -%>
      server <%= "app#{n}" %> <%= web['cloud']['public_ipv4'] %>:80 weight 1 maxconn 100 check
    <% end -%>
```

Iterate over the 'webservers' and add a line for each



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

Lab: Upload the Cookbook

- Upload the haproxy cookbook to the Chef Server

Lab: Upload the Cookbook

```
$ cd chef-repo/cookbooks/haproxy
```


CHEF

Lab: Upload the Cookbook

```
$ berks install
```

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

Lab: Upload the Cookbook

```
$ berks upload
```

```
Uploaded haproxy (0.1.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
```

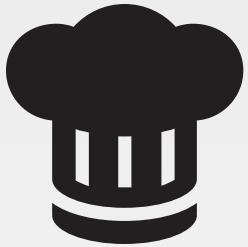

CHEF

Lab: Verify the Cookbook Upload

```
$ knife cookbook list
```

```
apache              0.2.1
haproxy             0.1.0
workstation         0.2.1
```


CHEF



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

Lab: Bootstrap a Load Balancer Node

```
$ knife bootstrap FQDN -x USER -P PWD --sudo -N node3 -r 'recipe[haproxy]'
```

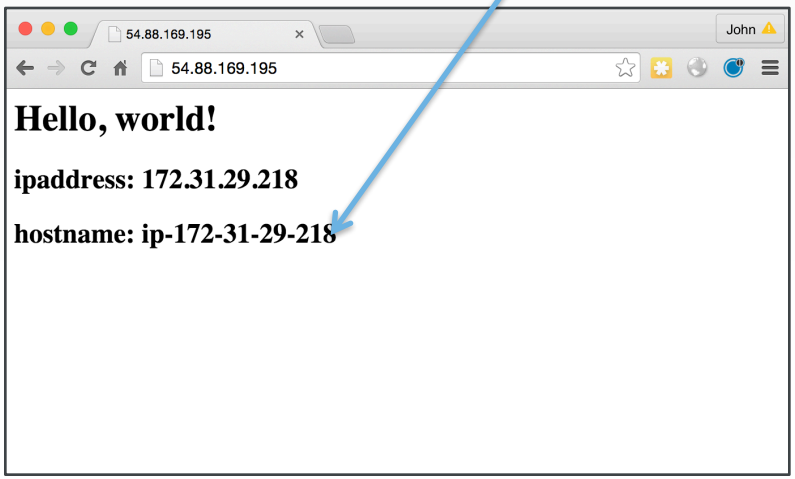
```
...
54.88.169.195     -    server  app3 127.0.0.1:5003 check
54.88.169.195     -    server  app4 127.0.0.1:5004 check
54.88.169.195     -
54.88.169.195     +    server app0 54.88.185.159:80 weight 1 maxconn 100 check
54.88.169.195     +    server app1 54.84.233.7:80 weight 1 maxconn 100 check
54.88.169.195   * service[haproxy] action start
54.88.169.195     - start service service[haproxy]
54.88.169.195   * service[haproxy] action enable
54.88.169.195     - enable service service[haproxy]
54.88.169.195
54.88.169.195 Running handlers:
54.88.169.195 Running handlers complete
54.88.169.195 Chef Client finished, 4/4 resources updated in 11.370356638 seconds
```

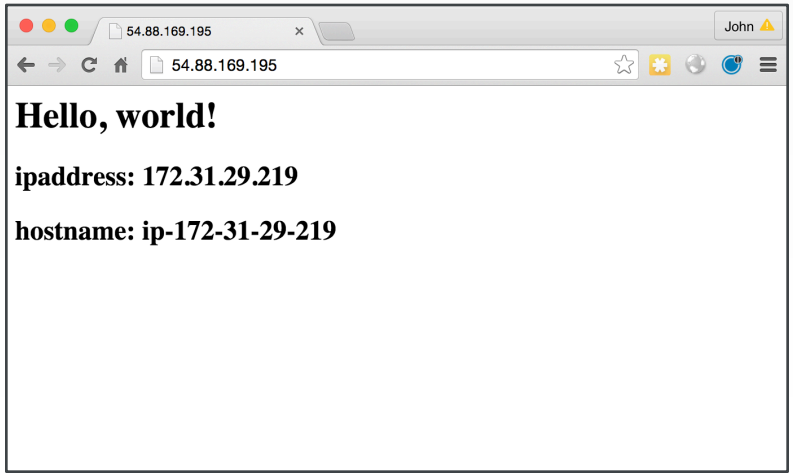
Lab: Validate the New Node

```
$ knife node show node3
```

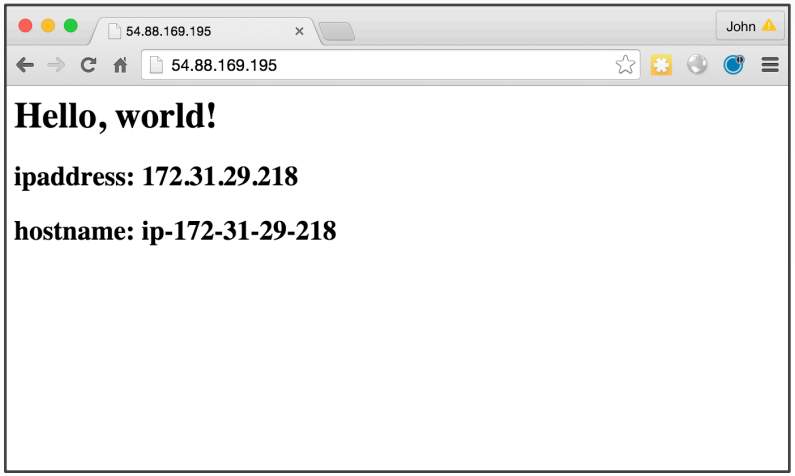
```
Node Name:   node3
Environment: _default
FQDN:        ip-172-31-29-217.ec2.internal
IP:          54.88.169.195
Run List:    recipe[haproxy]
Roles:
Recipes:     haproxy::default
Platform:    centos 6.7
Tags:
```

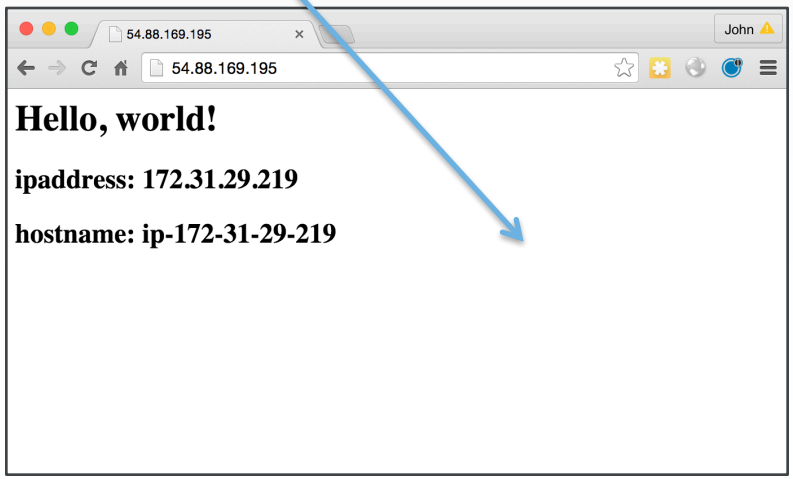

54.88.169.195
John
54.88.169.195
Hello, world!
ipaddress: 172.31.29.218
hostname: ip-172-31-29-218


54.88.169.195
John
54.88.169.195
Hello, world!
ipaddress: 172.31.29.218
hostname: ip-172-31-29-218


54.88.169.195
John
54.88.169.195
Hello, world!
ipaddress: 172.31.29.219
hostname: ip-172-31-29-219


54.88.169.195
John
54.88.169.195
Hello, world!
ipaddress: 172.31.29.219
hostname: ip-172-31-29-219


54.88.169.195
John
54.88.169.195
Hello, world!
ipaddress: 172.31.29.218
hostname: ip-172-31-29-218


54.88.169.195
John
54.88.169.195
Hello, world!
ipaddress: 172.31.29.219
hostname: ip-172-31-29-219

DISCUSSION

Discussion

How might Search work in the context of a webserver & database?

Where else might you use dynamic content in files?

CHEF

DISCUSSION

Q&A

What questions can we help you answer?

- Search
- Passing variables into templates

CHEF

CHEF™