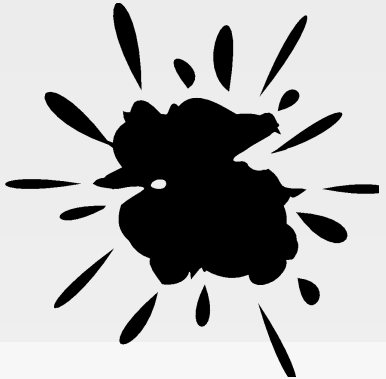


SSH Woes

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

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
```


CHEF

Run chef-client on all nodes

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

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```


CHEF

Verify the port and identity file

```
$ vagrant ssh-config load-balancer
```

```
Host load-balancer
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/USER/chef-repo/.vagrant/machines/load-balancer/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```


CHEF

Run chef-client on a Vagrant instance

```
$ knife ssh localhost "sudo chef-client" --manual-list --ssh-port PORT
--ssh-user vagrant --identity-file /PATH/TO/KEY
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```


CHEF

Run chef-client on Vagrant - short options

```
$ knife ssh localhost "sudo chef-client" -m -p PORT -x vagrant -i /PATH/TO/KEY
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```

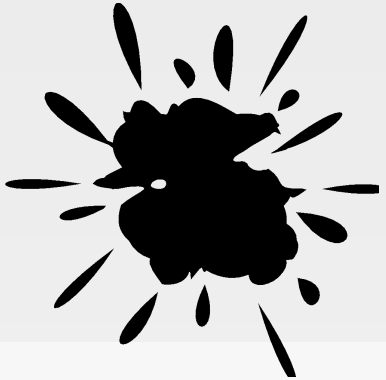

CHEF

Example - Run chef-client on Vagrant

```
$ knife ssh localhost "sudo chef-client" -m -p 2222 -x vagrant \
-i /Users/technotrainer/chef-repo/.vagrant/machines/load-balancer/virtualbox/private_key
```

```
localhost Starting Chef Client, version 12.17.44
localhost resolving cookbooks for run list: ["myhaproxy"]
localhost Synchronizing Cookbooks:
localhost   - myhaproxy (0.2.0)
localhost   - haproxy (2.0.0)
localhost   - build-essential (7.0.3)
localhost   - seven_zip (2.0.2)
localhost   - windows (2.1.1)
localhost   - ohai (4.2.3)
localhost   - compat_resource (12.16.3)
localhost   - mingw (1.2.4)
localhost   - cpu (1.0.0)
....
```


CHEF



SSH Woes

When using a Cloud Provider like Amazon EC2, knife ssh is a great way to issue a remote command to a set of nodes, not just a single node.

Example - Converge All Nodes

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

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-166.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list:
["workstation","apache"]
ec2-54-210-86-164.compute-1.amazonaws.com  resolving cookbooks for run list:
["workstation","apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["myhaproxy"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - workstation
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
....
```


CHEF

Example - Converge All Webserver Nodes

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

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list: ["apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
ec2-54-175-46-24.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com    - apache
```

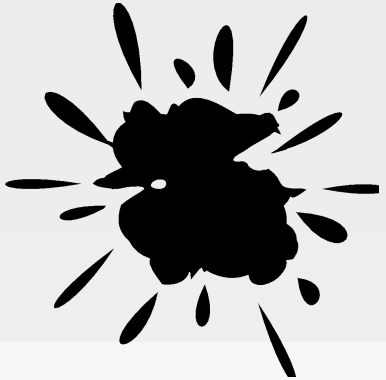

CHEF

Example - Converge All Proxy Nodes

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

```
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list:
["myhaproxy"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list:
["myhaproxy"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
  - myhaproxy (0.2.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - ohai (4.2.3)
  - compat_resource (12.16.3)
  - mingw (1.2.4)
```


CHEF



SSH Woes

When using a Cloud Provider like Amazon EC2, knife ssh allows us to specify what node attribute to use as the “ipaddress”

This allows us to ssh in using the node[‘cloud’] attribute, which contains the public ipaddress.

NOTE: cloud web1's Public Host Name and IP

```
$ knife node show web1 -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
```


CHEF

Example - Converge All Nodes

```
$ knife ssh "*:*" -x USER -P PWD "sudo chef-client" -a cloud.public_ipv4
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-166.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list:
["workstation","apache"]
ec2-54-210-86-164.compute-1.amazonaws.com  resolving cookbooks for run list:
["workstation","apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["myhaproxy"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - workstation
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
....
```


CHEF

Example - Converge All Webserver Nodes

```
$ knife ssh "role:web" -x USER -P PWD "sudo chef-client" -a cloud.public_ipv4
```

```
ec2-54-175-46-24.compute-1.amazonaws.com  Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com Starting Chef Client, version 12.3.0
ec2-54-210-86-164.compute-1.amazonaws.com resolving cookbooks for run list: ["apache"]
ec2-54-175-46-24.compute-1.amazonaws.com  resolving cookbooks for run list: ["apache"]
ec2-54-210-86-164.compute-1.amazonaws.com Synchronizing Cookbooks:
ec2-54-210-86-164.compute-1.amazonaws.com   - apache
ec2-54-210-86-164.compute-1.amazonaws.com Compiling Cookbooks...
ec2-54-210-86-164.compute-1.amazonaws.com Converging 3 resources
ec2-54-210-86-164.compute-1.amazonaws.com Recipe: apache::server
ec2-54-175-46-24.compute-1.amazonaws.com  Synchronizing Cookbooks:
ec2-54-175-46-24.compute-1.amazonaws.com    - apache
```


CHEF

Search

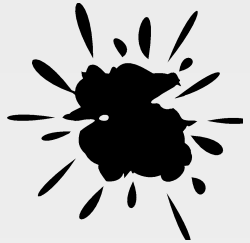
Update a Cookbook to Dynamically Use Nodes with the Web Role

CHEF

Objectives

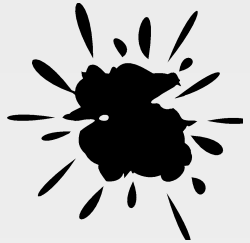
After completing this module, you should be able to

- 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



How Do We See Detail Across All Nodes?

`knife node show` only works with individual nodes so how do I see this info for multiple nodes?



How Do We See Detail Across All Nodes?

`knife node show` only works with individual nodes so 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.

Chef Server

Web Nodes	Proxy Nodes
Node 1	Node 2
Node 3	Node 5
Node 8	Node 6

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

GL: View Information for All Nodes

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

```
3 items found

Node Name:   web1
Environment: _default
FQDN:        web1
IP:          192.168.10.43
Run List:    role[web]
Roles:       web
Recipes:     workstation, workstation::default, apache, apache::default, workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:

Node Name:   load-balancer
Environment: _default
FQDN:        load-balancer
IP:          10.0.2.15
Run List:    role[load-balancer]
....
```


CHEF

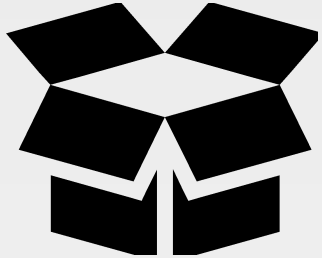
GL: Narrow the Search

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

```
2 items found

web1:
 ipaddress : 192.168.10.43

load-balancer:
 ipaddress : 10.0.2.15

web2:
 ipaddress : 192.168.10.44
```


CHEF

CONCEPT

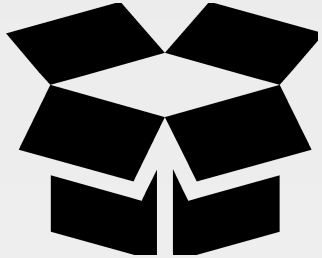


Search

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.

CONCEPT



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.

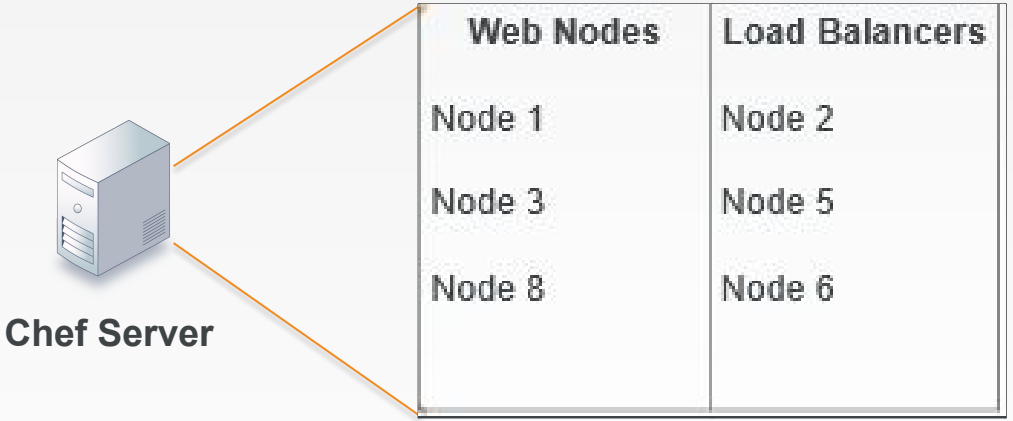
The Chef Server and Search

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.

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

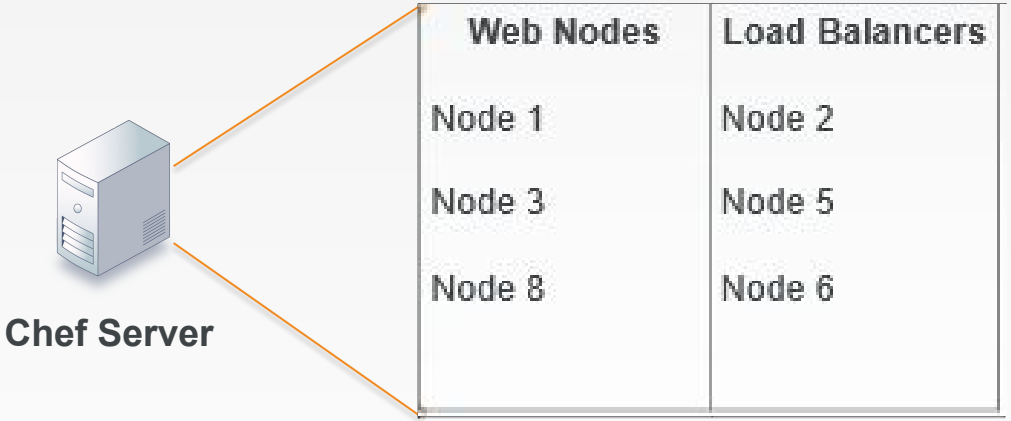
https://docs.chef.io/chef_search.html#search-indexes



CHEF

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`.



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

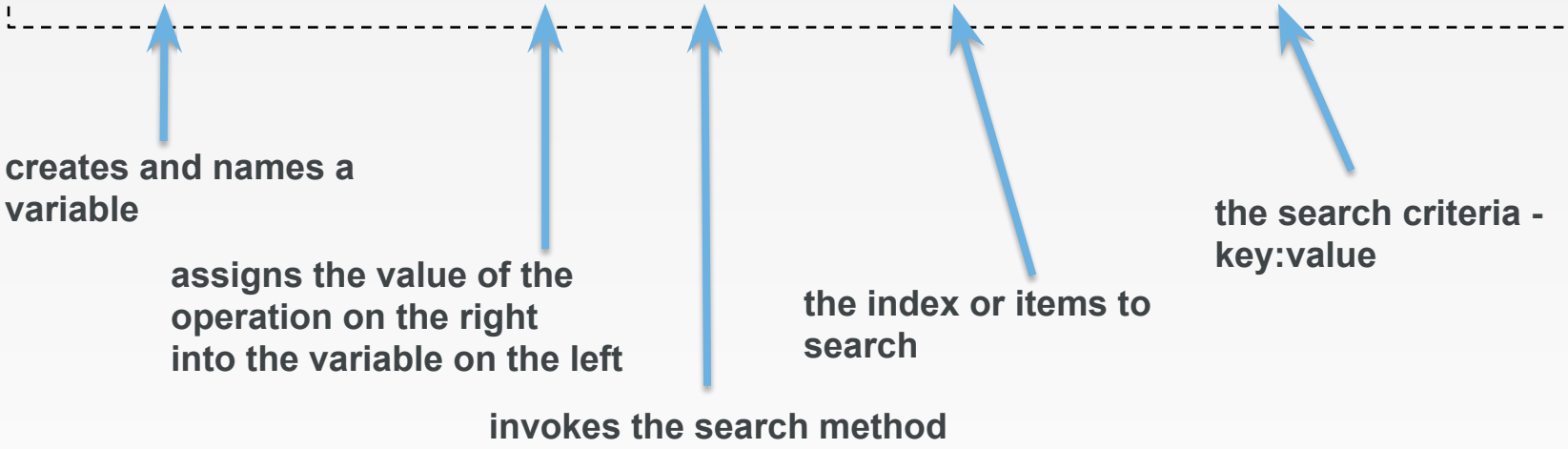
A search query is comprised of two parts: the key and the search pattern. A search query has the following syntax:

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')
```



Search Syntax within a Recipe

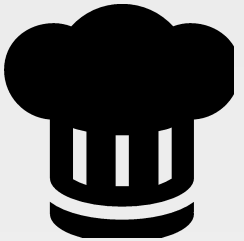
```
all_web_nodes = search('node','role:web')
```

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'  => 'web1',
    'ipaddress' => '192.168.10.43',
    'port' => 80,
    'ssl_port' => 80
  },
  {
    'hostname'  => 'web2',
    'ipaddress' => '192.168.10.44',
    'port' => 80,
    'ssl_port' => 80
  }
]
include_recipe 'haproxy::manual'
```



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

Showing web1 Attributes

```
$ knife node show web1 -a ipaddress
```

```
web1:
  ipaddress: 192.168.10.43
```


CHEF

Showing web2 Attributes

```
$ knife node show web2 -a ipaddress
```

```
web2:
  ipaddress: 192.168.10.44
```


CHEF

NOTE: Showing web1 CLOUD Attributes

```
$ knife node show web1 -a cloud
```

```
web1:
  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
```


CHEF

NOTE: Showing web1 CLOUD Attributes

```
$ knife node show web2 -a cloud
```

```
web2:
  cloud:
    local_hostname:  ip-10-197-105-149.us-west-1.compute.internal
    local_ipv4:      10.197.105.149
    private_ips:     10.197.105.149
    provider:        ec2
    public_hostname: ec2-54-176-64-174.us-west-1.compute.amazonaws.com
    public_ips:      54.176.64.174
    public_ipv4:     54.176.64.174
```


CHEF

GL: Remove the Hard-coded Members

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

```
node.default['haproxy']['members'] = [{
    'hostname'  => 'web1',
    'ipaddress' => '192.168.10.43',
    'port' => 80,
    'ssl_port' => 80
  },
  {
    'hostname'  => 'web2',
    'ipaddress' => '192.168.10.44',
    'port' => 80,
    'ssl_port' => 80
  }
]
include_recipe 'haproxy::manual'
```

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::manual'
```

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::manual'
```

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['hostname'],
    'ipaddress' => web_node[‘ipaddress'],
    '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['hostname'],
    'ipaddress' => web_node[`ipaddress'],
    'port' => 80,
    'ssl_port' => 80
  }
  members.push(member)
end

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

include_recipe 'haproxy::manual'
```

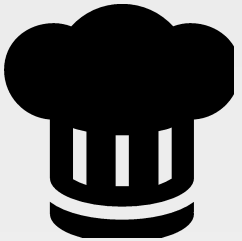
NOTE: The Final Recipe - Cloud Instances

~/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::manual'
```

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



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:

- ✓ Refactor the myhaproxy 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', '= 2.0.0'
```

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 (2.0.0)
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/ORGNAME'
Uploaded cpu (0.2.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
Uploaded haproxy (2.0.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
Uploaded myhaproxy (1.0.0) to: 'https://api.opscode.com:443/organizations/ORGNAME'
PS C:\Users\USER\chef-repo\cookbooks\myhaproxy>
```

Login to Load Balancer

```
$ vagrant ssh load-balancer
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@load-balancer ~]$
```

Converge the Load Balancer

```
[vagrant@load-balancer ~]$ sudo chef-client
```

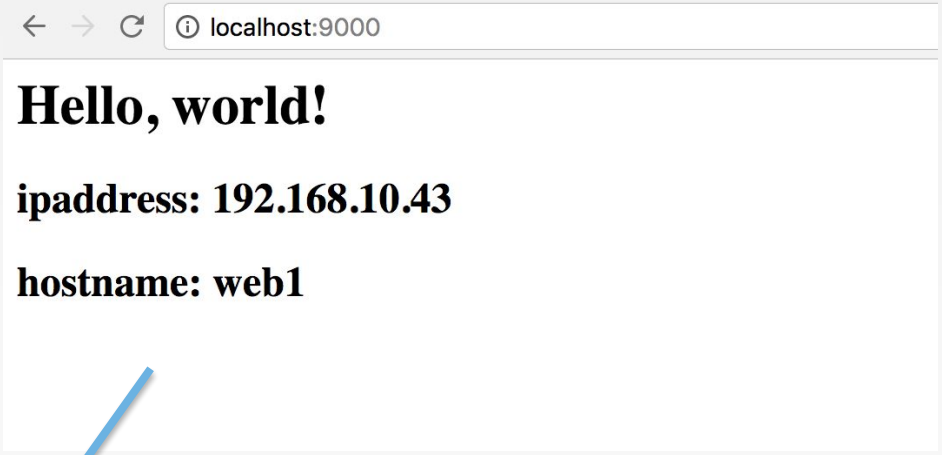
```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["myhaproxy"]
Synchronizing Cookbooks:
  - myhaproxy (0.1.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - windows (2.1.1)
  - ohai (4.2.3)
....
```

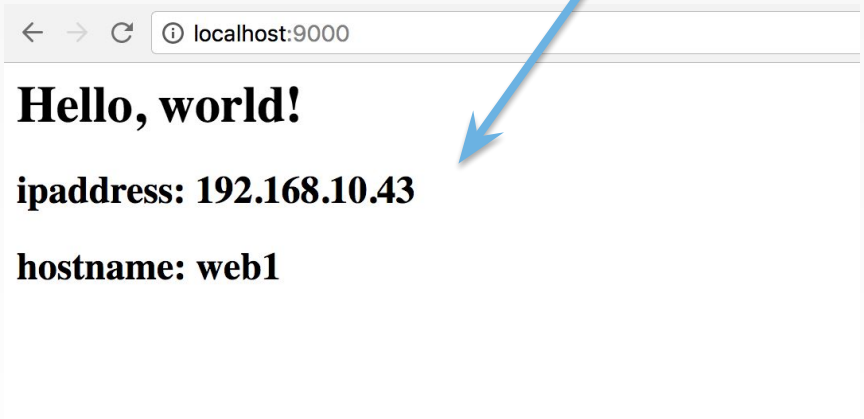
Return to your Workstation

```
[vagrant@load-balancer ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

The load balancer should still function





The load balancer should still function

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

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?

Environments

Using Environments to Reflect Organization Patterns and Workflow

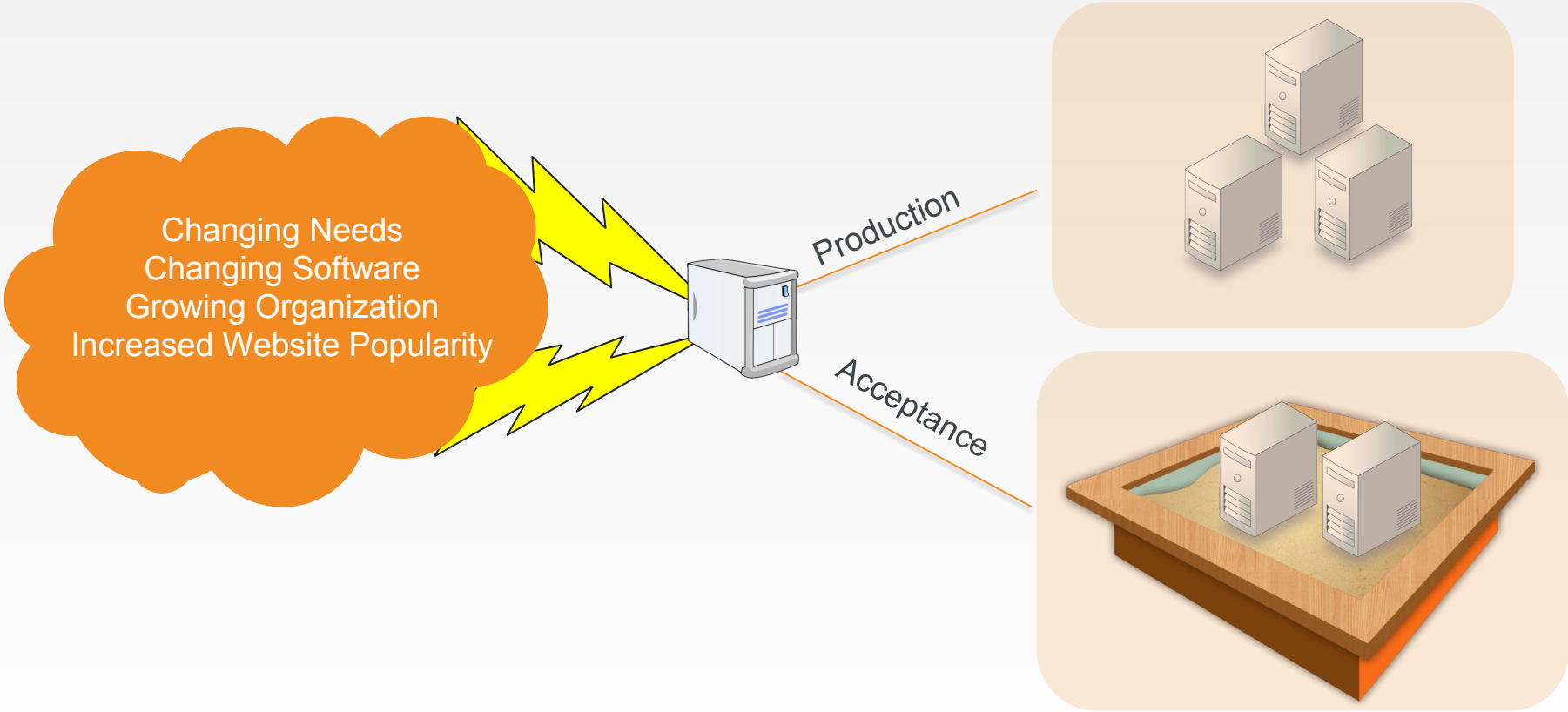
CHEF

Objectives

After completing this module, you should be able to

- Create a production environment
- Deploy web1 and load-balancer nodes to the production environment

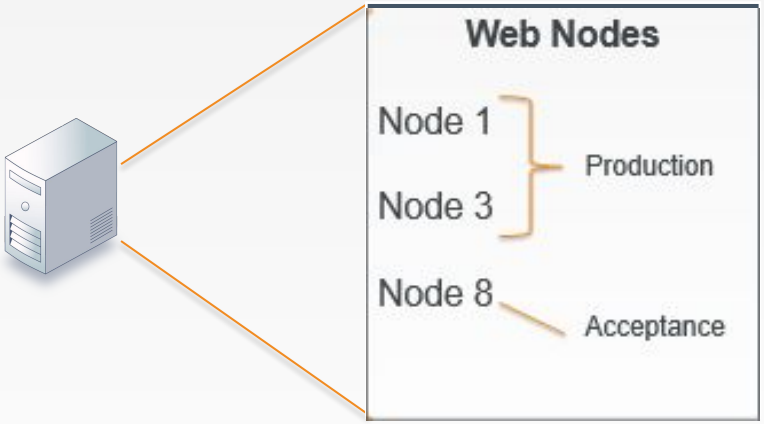
Keeping Your Infrastructure Current



CHEF

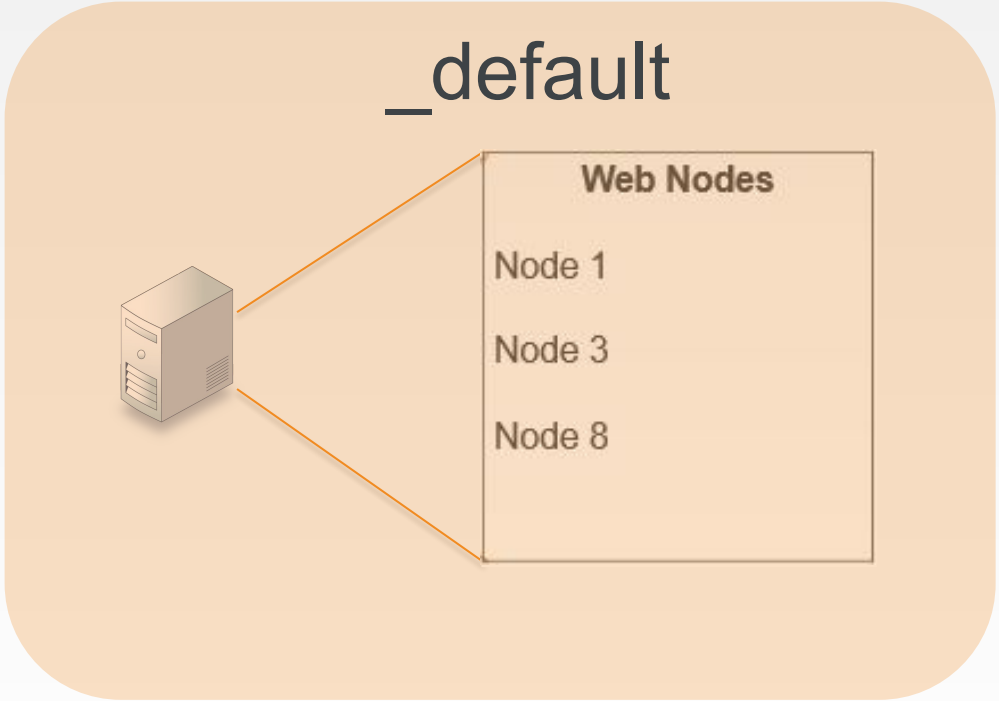
Environments

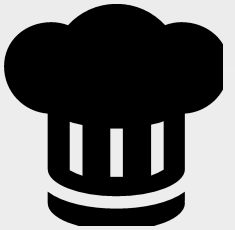
Environments can define different functions of nodes that live on the same system.



Environments

Every organization or infrastructure starts with the _default environment.





Production

Let's create a reliable environment for our nodes.

Objective:

- ❑ Deploy Our Site to Production

Using 'knife environment --help'

```
$ cd ~/chef-repo
$ knife environment --help
```

```
** ENVIRONMENT COMMANDS **
knife environment compare [ENVIRONMENT..] (options)
knife environment create ENVIRONMENT (options)
knife environment delete ENVIRONMENT (options)
knife environment edit ENVIRONMENT (options)
knife environment from file FILE [FILE..] (options)
knife environment list (options)
knife environment show ENVIRONMENT (options)
```

View List of Defined Environments

```
$ knife environment list
```

```
_default
```

Viewing the _default Environment

```
$ knife environment show _default
```

```
chef_type:            environment
cookbook_versions:
default_attributes:
description:          The default Chef environment
json_class:           Chef::Environment
name:                 _default
override_attributes:
```


CHEF

Searching All of Our Nodes

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

```
3 items found

Node Name:   web1
Environment: _default
FQDN:        web1
IP:          192.168.10.43
Run List:    role[web]
Roles:       web
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Create an environments Directory

```
$ mkdir environments
```

Create a New Environment File

~/chef-repo/environments/production.rb

```
name 'production'
description 'Where we run production code'

cookbook 'apache', '= 0.2.1'
cookbook 'myhaproxy', '= 1.0.0'
```

Upload the production.rb File

```
$ knife environment from file production.rb
```

```
Updated Environment production
```

View the List of Environments

```
$ knife environment list
```

```
_default
production
```

View the Production Environment

```
$ knife environment show production
```

```
chef_type:            environment
cookbook_versions:
  apache:     = 0.2.1
  myhaproxy:  = 1.0.0
default_attributes:
description:          Where we run production code
json_class:           Chef::Environment
name:                 production
override_attributes:
```


CHEF

Viewing 'knife node --help'

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

```
** NODE COMMANDS **
knife node bulk delete REGEX (options)
knife node create NODE (options)
knife node delete NODE (options)
knife node edit NODE (options)
knife node environment set NODE ENVIRONMENT
knife node from file FILE (options)
knife node list (options)
knife node run_list add [NODE] [ENTRY[,ENTRY]] (options)
knife node run_list remove [NODE] [ENTRY[,ENTRY]] (options)
knife node run_list set NODE ENTRIES (options)
knife node show NODE (options)
```

Viewing 'knife node set --help'

```
$ knife node environment set --help
```

```
knife node environment set NODE ENVIRONMENT
    -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, --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
```

Using 'knife environment node set'

```
$ knife node environment set web1 production
```

```
web1:
  chef_environment: production
```

Lab: Verify the New Node

```
$ knife node show web1
```

```
Node Name:   web1
Environment: production
FQDN:        web1
IP:          192.168.10.43
Run List:    role[web]
Roles:       web
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Login to web1

```
$ vagrant ssh web1
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web1 ~]$
```

Run chef-client to converge web1

```
[vagrant@web1 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["workstation", "apache"]
Synchronizing Cookbooks:
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

Return to your Workstation

```
[vagrant@web1 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Set proxy node's Environment to Production

```
$ knife node environment set load-balancer production
```

```
load-balancer:
  chef_environment: production
```

Verify the Run List

```
$ knife node show load-balancer
```

```
Node Name:   load-balancer
Environment: production
FQDN:        load-balancer
IP:          10.0.2.15
Run List:    role[load-balancer]
Roles:       load-balancer
Recipes:     myhaproxy, myhaproxy::default, haproxy::default, haproxy::install_package
Platform:    centos 7.2
Tags:
```

Login to Load Balancer

```
$ vagrant ssh load-balancer
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@load-balancer ~]$
```

Converge the Load Balancer

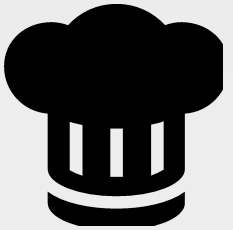
```
[vagrant@load-balancer ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["myhaproxy"]
Synchronizing Cookbooks:
  - myhaproxy (0.1.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - windows (2.1.1)
  - ohai (4.2.3)
....
```

Return to your Workstation

```
[vagrant@load-balancer ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```



Production

Let's create a reliable environment for our nodes.

Objective:

- ✓ Deploy our site to Production

Lab: Acceptance Environment

- ❑ Create an environment named "acceptance" that has no cookbook restrictions.
- ❑ Upload acceptance.rb to the Chef Server
- ❑ Move web2 into the acceptance environment
- ❑ Run chef-client on the web2 node

Lab: Create a New Environment File

~/chef-repo/environments/acceptance.rb

```
name 'acceptance'
description 'Where code and apps are tested'
# No Cookbook Restrictions
```

Lab: Upload the .rb File

```
$ knife environment from file acceptance.rb
```

```
Updated Environment acceptance
```

Lab: Verify that the Environment was Set

```
$ knife environment list
```

```
_default
production
acceptance
```


CHEF

Lab: Verify the Contents of the Environment

```
$ knife environment show acceptance
```

```
chef_type:            environment
cookbook_versions:
default_attributes:
description:          Where code and applications are tested
json_class:           Chef::Environment
name:                 acceptance
override_attributes:
```


CHEF

Viewing 'knife node set --help'

```
$ knife node environment set --help
```

```
knife node environment set NODE ENVIRONMENT
    -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, --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
```

Using 'knife environment node set'

```
$ knife node environment set web2 acceptance
```

```
web2:
  chef_environment: acceptance
```

Lab: Verify the web2 Node

```
$ knife node show web2
```

```
Node Name:   web2
Environment: acceptance
FQDN:        web2
IP:          192.168.10.44
Run List:    role[web]
Roles:       web
Recipes:     workstation, workstation::default, apache, apache::default,
workstation::setup, workstation::vagrant, apache::server
Platform:    centos 7.2.1511
Tags:
```


CHEF

Login to web2

```
$ vagrant ssh web2
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web2 ~]$
```

Run chef-client to converge web2

```
[vagrant@web2 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["workstation", "apache"]
Synchronizing Cookbooks:
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

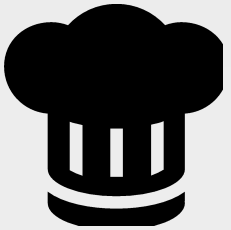
Return to your Workstation

```
[vagrant@web2 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Lab: Acceptance Environment

- ✓ Create an environment named "acceptance" that has no cookbook restrictions.
- ✓ Upload acceptance.rb to the Chef Server
- ✓ Move web2 into the acceptance environment
- ✓ Run chef-client on the web2 node



Separating Environments

Objective:

- ❑ Use Search to separate out the environments
- ❑ Update myhaproxy cookbook's version number

CHEF

DISCUSSION

Expected Situation

What do we expect to happen when we set a web node to a specific environment?

DISCUSSION

Balancing Nodes

Which cookbook handles balancing the requests between web nodes?

Which recipe within that cookbook sets up the request balancing between the two nodes?

DISCUSSION

Search Criteria

How are we currently searching for web nodes?

How can we further refine our search results?

CHEF

Search Criteria

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

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

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

members = []

#...
```

GL: Modify the myhaproxy default.rb

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

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

all_web_nodes = search('node',"role:web AND chef_environment:#{node.chef_environment}")

members = []

#...
```

GL: Version the myhaproxy metadata.rb

~/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           '1.0.1'

depends 'haproxy', '= 2.0.0'
```

GL: Run 'berks install'

```
$ cd 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)
Installing haproxy (2.0.0)
Using cpu (0.2.0)
Using myhaproxy (1.0.1) from source at .
```

GL: Run 'berks upload'

```
$ berks upload
```

```
Skipping build-essential (2.2.3) (frozen)
Skipping cpu (0.2.0) (frozen)
Skipping haproxy (2.0.0) (frozen)
Uploaded myhaproxy (1.0.1) to:
'https://api.opscode.com:443/organizations/vogue'
```


CHEF

DISCUSSION

A Brief Recap

We restricted the production environment to specific cookbook version.

We created an acceptance environment with no cookbook restrictions.

We set web1 and load-balancer to the production environment, and web2 to the acceptance environment.

We updated the myhaproxy's default recipe to include environment search criteria.

And we changed the version number in the myhaproxy metadata.rb file.

CHEF

Update Production Environment

- Update the environment named production:

```
'myhaproxy' cookbook version equal to
                    '1.0.1'
```

Lab: Update production.rb

~/chef-repo/environments/production.rb

```
name 'production'
description 'Where we run production code'

cookbook 'apache', '= 0.2.1'
cookbook 'myhaproxy', '= 1.0.1'
```

Lab: cd and Run 'knife environment...'

```
$ cd ~/chef-repo
$ knife environment from file production.rb
```

```
Updated Environment production
```


CHEF

Lab: Verify the Version Number

```
$ knife environment show production
```

```
chef_type:            environment
cookbook_versions:
  apache:     = 0.2.1
  myhaproxy: = 1.0.1
default_attributes:
description:          Where we run production code
json_class:           Chef::Environment
name:                 production
override_attributes:
```


CHEF

Login to Load Balancer

```
$ vagrant ssh load-balancer
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@load-balancer ~]$
```

Converge the Load Balancer

```
[vagrant@load-balancer ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: ["myhaproxy"]
Synchronizing Cookbooks:
  - myhaproxy (0.1.0)
  - haproxy (2.0.0)
  - build-essential (7.0.3)
  - seven_zip (2.0.2)
  - windows (2.1.1)
  - ohai (4.2.3)
....
```

Return to your Workstation

```
[vagrant@load-balancer ~]$ exit
```

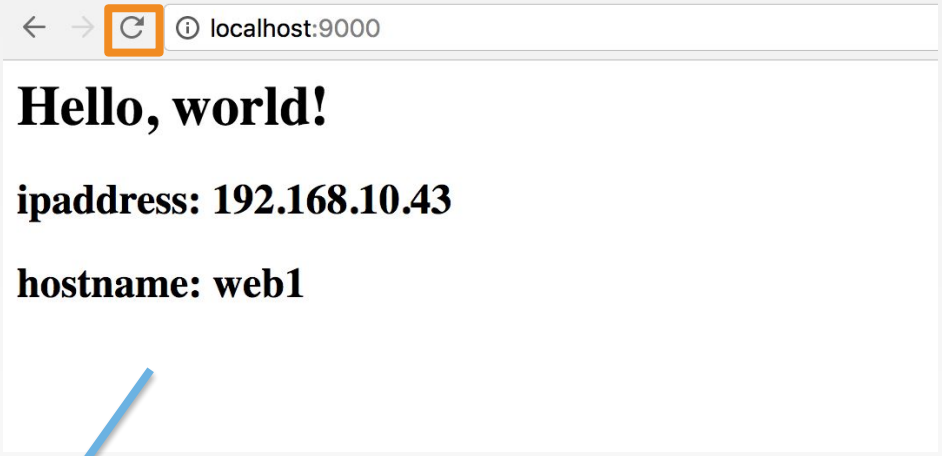
```
logout
Connection to 127.0.0.1 closed.
```

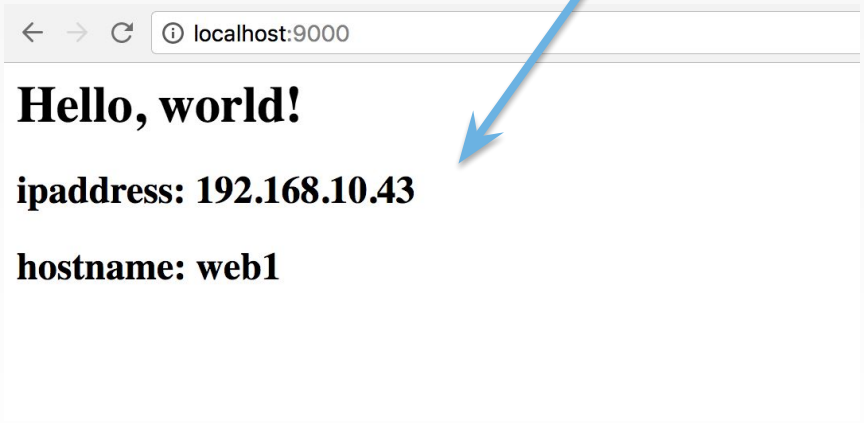
Lab: Update Production

✓ Update the environment named production:

```
'myhaproxy' cookbook version equal to
'1.0.1'
```


CHEF





The load balancer will only send traffic to web1, since they are both assigned to the production environment.

CHEF

Discussion

What is the benefit of constraining cookbooks to a particular environment?

What are the benefits of **not** constraining cookbooks to a particular environment?

Data Bags

Working with Custom Data Sets

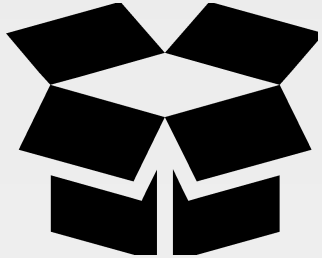
CHEF

Objectives

After completing this module, you should be able to

- Understand how to use and manage a Data Bag
- Create a Data Bag
- Upload Data Bag to Chef Server

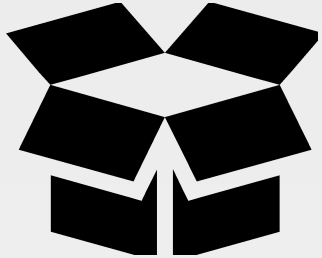
CONCEPT



What about user data?

Where should we store data that each node might need access to?

CONCEPT

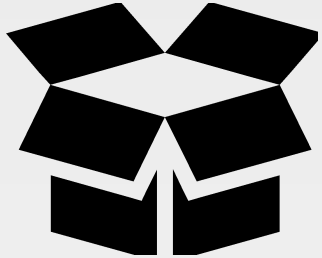


What about user data?

We could start by storing information about users as Node Attributes…

But this would duplicate a lot of information - every user in the company would be stored in every Node object!

CONCEPT

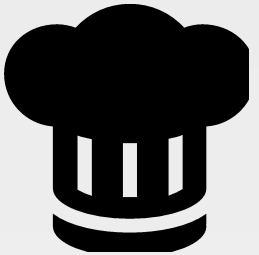


Data Bags

A data bag is a container for items that represent information about your infrastructure that is not tied to a single node.

Examples:

- Users
- Groups
- Application Release Information
- Passwords (in an encrypted data bag)



Custom Data Sets

We can store sets of JSON data on our Chef Server, accessible by a node with search

Objective:

- ❑ Create “users” and “groups” data bags
- ❑ Upload data bags to Chef Server
- ❑ Create a cookbook to manage users and groups

What Can 'knife data bag' Do?

```
$ cd ~/chef-repo
$ knife data bag --help
```

```
** DATA BAG COMMANDS **
knife data bag create BAG [ITEM] (options)
knife data bag delete BAG [ITEM] (options)
knife data bag edit BAG ITEM (options)
knife data bag from file BAG FILE|FOLDER [FILE|FOLDER..] (options)
knife data bag list (options)
knife data bag show BAG [ITEM] (options)
```

Run 'knife data bag list'

```
$ knife data bag list
```

Create a data_bags Directory

```
$ mkdir data_bags
```

Create a data_bags/users Directory

```
$ mkdir data_bags/users
```

Create users Data Bag on Chef Server

```
$ knife data bag create users
```

```
Created data_bag[users]
```

Create user1.json

~/chef-repo/data_bags/users/user1.json

```
{
   "id": "user1",
   "comment": "I am user1",
   "uid": 100,
   "gid": 1,
   "home": "/home/user1",
   "shell": "/bin/bash",
   "platform": "centos"
}
```

Create user2.json

~/chef-repo/data_bags/users/user2.json

```
{
   "id": "user2",
   "comment": "I am user2",
   "uid": 101,
   "gid": 1,
   "home": "/home/user2",
   "shell": "/bin/bash",
   "platform": "centos"
}
```

Upload data bag item to Chef Server

```
$ knife data bag from file users data_bags/users/user1.json data_bags/users/user2.json
```

```
Updated data_bag_item[users::user1]
Updated data_bag_item[users::user2]
```

Validate Chef Server Received It

```
$ knife data bag list
```

```
users
```

View Details of users Data Bag

```
$ knife data bag show users
```

```
user1
user2
```

View Details of user1

```
$ knife data bag show users/user1
```

```
comment
gid
home
id
platform
shell
uid
```

Search the “users” index

```
$ knife search users "*:*"
```

```
2 items found

chef_type: data_bag_item
Comment:   I am user2
gid:       1
home:      /home/user2
id:        user2
Platform:  centos
shell:     /bin/bash
uid:       100
....
```

Return users with “platform:centos”

```
$ knife search users “platform:centos*”
```

```
2 items found

chef_type: data_bag_item
Comment:   I am user2
gid:       1
home:      /home/user2
id:        user2
Platform:  centos
shell:     /bin/bash
uid:       100
....
```

Create a data_bags/groups Directory

```
$ mkdir data_bags/groups
```

Create groups Data Bag on Chef Server

```
$ knife data bag create groups
```

```
Created data_bag[groups]
```

Create group1.json

~/chef-repo/data_bags/groups/group1.json

```
{
   "id": "group1",
   "gid": 2000,
   "members": ["user1","user2"],
   "platform": "centos"
}
```

Upload data bag item to Chef Server

```
$ knife data bag from file groups data_bags/groups/group1.json
```

```
Updated data_bag_item[groups::group1]
```

Validate Chef Server Received It

```
$ knife data bag list
```

```
users
groups
```

View Details of users Data Bag

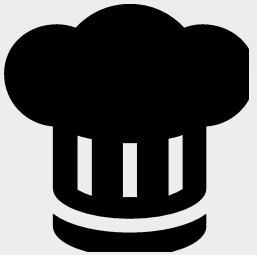
```
$ knife data bag show groups
```

```
group1
```

View Details of user1

```
$ knife data bag show groups group1
```

```
gid:      2000
id:       group1
members:
  user1
  user2
```



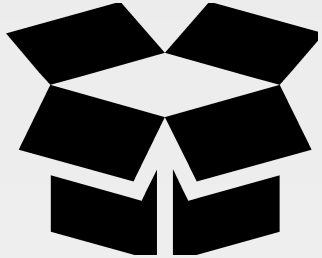
Create Users from a Data Bag

Dynamically search through the Chef Server under the 'users' index to create users

Objective:

- ❑ Generate the 'myusers' cookbook
- ❑ Search for all users belonging to group1
- ❑ Create Users and Groups based on data bag contents
- ❑ Upload 'myusers' cookbook to Chef Server
- ❑ Update 'web' role
- ❑ Converge web1 node

CONCEPT



Where are the users?

Since our users and groups Data Bags are now indexed on the Chef Server, we have a centralized source of truth for this information.

We can search through this information inside of our recipes.

cd and Generate the myusers Cookbook

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

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


CHEF

Create default recipe

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

```
search("users", "platform:centos").each do |user_data|
  user user_data['id'] do
    comment user_data['comment']
    uid user_data['uid']
    gid user_data['gid']
    home user_data['home']
    shell user_data['shell']
  end
end
```

Change to the cookbooks/myusers Directory

```
$ cd cookbooks/myusers
```

Run berks install

```
$ berks install
```

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

Upload the Cookbook to the Chef Server

```
$ berks upload
```

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

Display Cookbooks within Your Org

```
$ knife cookbook list
```

```
apache                  0.2.1
build-essential         7.0.2
compat_resource         12.16.2
cpu                     1.0.0
haproxy                 2.0.0
mingw                   1.2.4
myhaproxy               0.1.0
myusers                 0.1.0
ohai                    4.2.3
seven_zip               2.0.2
windows                 2.1.1
workstation             0.2.1
```

Update the web.rb Role to include myusers

~/chef-repo/roles/web.rb

```
name 'web'
description 'Web Server Role'
run_list 'recipe[myusers]','recipe[workstation]','recipe[apache]'
```

Upload it to the Chef Server

```
$ knife role from file roles/web.rb
```

```
Updated Role web!
```

View Details of the Role

```
$ knife role show web
```

```
chef_type:            role
default_attributes:
description:          Web Server Role
env_run_lists:
json_class:           Chef::Role
name:                 web
override_attributes:
run_list:
  recipe[myusers]
  recipe[workstation]
  recipe[apache]
```

Login to web1

```
$ vagrant ssh web1
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web1 ~]$
```

Run chef-client to converge web1

```
[vagrant@web1 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: [“myusers”, "workstation", "apache"]
Synchronizing Cookbooks:
  - myusers (0.1.0)
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

Verify the new users has been created

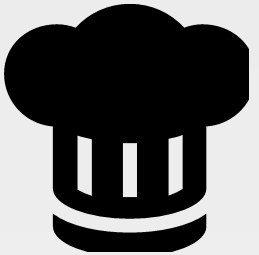
```
[vagrant@web1 ~]$ cat /etc/passwd
```

```
....
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
user2:x:101:1:I am user2:/home/user2/:/bin/bash
user1:x:100:1:I am user1:/home/user1/:/bin/bash
```

Return to your Workstation

```
[vagrant@web1 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```



Create Users from Data bag

Dynamically search through the Chef Server under the 'users' index to create users

Objective:

- ✓ Generate the 'myusers' cookbook
- ✓ Search for all users belonging to group1
- ✓ Create Users and Groups based on data bag contents
- ✓ Upload 'myusers' cookbook to Chef Server
- ✓ Update 'web' role
- ✓ Converge web1 node

Lab: Manage Users and Groups

- ❑ Create 3 users and upload them to the ‘users’ data bag
- ❑ Create and upload a ‘groups’ data bag with your users as members
- ❑ Update the ‘myusers’ cookbook to add your users to the group using search
- ❑ Verify ‘web’ role contains ‘myusers’, Converge your web server
- ❑ Verify the new users in the /etc/passwd file
- ❑ Verify the new group in the /etc/group file

Create anthony.json

~/chef-repo/data_bags/users/anthony.json

```
{
  "id": "anthony",
  "comment": "anthony bourdain",
  "uid": 200,
  "gid": 0,
  "home": "/home/anthony",
  "shell": "/bin/bash",
  "platform": "centos",
  "tvshow": "food network"
}
```

Create gordon.json

~/chef-repo/data_bags/users/gordon.json

```
{
  "id": "gordon",
  "comment": "gordon ramsay",
  "uid": 201,
  "gid": 0,
  "home": "/home/gordon",
  "shell": "/bin/bash",
  "platform": "centos",
  "tvshow": "MasterChef"
}
```

Create julia.json

~/chef-repo/data_bags/users/julia.json

```
{
  "id": "julia",
  "comment": "julia child",
  "uid": 202,
  "gid": 0,
  "home": "/home/julia",
  "shell": "/bin/bash",
  "platform": "centos",
  "tvshow": "baking with julia"
}
```

Upload data bag item to Chef Server

```
$ knife data bag from file users data_bags/users/anthony.json data_bags/users/gordon.json \
  data_bags/users/julia.json
```

```
Updated data_bag_item[users::anthony]
Updated data_bag_item[users::gordon]
Updated data_bag_item[users::julia]
```

Validate Chef Server Received It

```
$ knife data bag list
```

```
users
```

View Details of users Data Bag

```
$ knife data bag show users
```

```
anthony
gordon
julia
```

Search the “users” index

```
$ knife search users "*:*"
```

```
3 items found

chef_type: data_bag_item
Comment:   julia child
gid:       0
home:      /home/julia
id:        julia
Platform:  centos
shell:     /bin/bash
uid:       202
....
```


CHEF

Create chefs.json

~/chef-repo/data_bags/groups/chefs.json

```
{
  "id": "chefs",
  "gid": "42",
  "members": ["anthony","gordon","julia"],
  "platform": "centos"
}
```

Upload data bag item to Chef Server

```
$ knife data bag from file groups data_bags/groups/chefs.json
```

```
Updated data_bag_item[groups::chefs]
```

Return groups with “platform:centos”

```
$ knife search groups "platform:centos*"
```

```
1 items found

chef_type: data_bag_item
data_bag:   groups
gid:        42
id:         chefs
members:
  julia
  anthony
  gordon
platform:   centos
```


CHEF

Create a ‘groups’ recipe

~/chef-repo/cookbooks/myusers/recipes/groups.rb

```
search(“groups”, “platform:centos”).each do |group_data|
  group group_data[‘id’] do
    gid group_data[‘gid’]
    members group_data[‘members’]
  end
end
```

Update the default recipe

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

```
search("users", "platform:centos").each do |user_data|
  user user_data['id'] do
    comment user_data['comment']
    uid user_data['uid']
    gid user_data['gid']
    home user_data['home']
    shell user_data['shell']
  end
end

include_recipe 'myusers::groups'
```

Version the myusers metadata.rb

~/chef-repo/cookbooks/myusers/metadata.rb

```
name              'myusers'
maintainer        'The Authors'
maintainer_email  'you@example.com'
license           'all_rights'
description       'Installs/Configures myusers'
long_description  'Installs/Configures myusers'
version           '0.2.0'
```

Change to the cookbooks/myusers Directory

```
$ cd cookbooks/myusers
```

Run berks install

```
$ berks install
```

```
Resolving cookbook dependencies...
Fetching 'myusers' from source at .
Fetching cookbook index from https://supermarket.chef.io...
Using myusers (0.2.0) from source at .
```

Upload the Cookbook to the Chef Server

```
$ berks upload
```

```
Uploaded myusers (0.2.0) to: 'https://api.opscode.com:443/organizations/ORG_NAME'
```


CHEF

Display Cookbooks within Your Org

```
$ knife cookbook list
```

```
apache                  0.2.1
build-essential         7.0.2
compat_resource         12.16.2
cpu                     1.0.0
haproxy                 2.0.0
mingw                   1.2.4
myhaproxy               0.1.0
myusers                 0.2.0
ohai                    4.2.3
seven_zip               2.0.2
windows                 2.1.1
workstation             0.2.1
```

Create new ‘base’ Role

~/chef-repo/roles/base.rb

```
name 'base'
description 'Base Role For All Servers'
run_list 'recipe[myusers]'
```

Update the ‘web’ Role

~/chef-repo/roles/web.rb

```
name 'web'
description 'Web Server Role'
run_list 'role[base]', 'recipe[workstation]', 'recipe[apache]'
```

Update the ‘load-balancer’ Role

~/chef-repo/roles/load-balancer.rb

```
name 'load-balancer'
description 'Load Balancer Role'
run_list 'role[base]', 'recipe[myhaproxy]'
```

Upload Roles to the Chef Server

```
$ knife role from file roles/base.rb roles/web.rb roles/load-balancer.rb
```

```
Updated Role base!
Updated Role web!
Updated Role load-balancer!
```

View Details of the Role

```
$ knife role show base
```

```
chef_type:            role
default_attributes:
description:          Base Role For All Servers
env_run_lists:
json_class:           Chef::Role
name:                 base
override_attributes:
run_list:
  recipe[myusers]
```


CHEF

Login to web1

```
$ vagrant ssh web1
```

```
Last login: Sat Dec 31 02:59:27 2016 from 10.0.2.2
[vagrant@web1 ~]$
```

Run chef-client to converge web1

```
[vagrant@web1 ~]$ sudo chef-client
```

```
Starting Chef Client, version 12.17.44
resolving cookbooks for run list: [“myusers”, "workstation", "apache"]
Synchronizing Cookbooks:
  - myusers (0.2.0)
  - apache (0.2.1)
  - workstation (0.2.1)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 8 resources....
```

Verify the new user has been created

```
[vagrant@web1 ~]$ cat /etc/passwd
```

```
....
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
user2:x:101:1:I am user1:/home/user1/:/bin/bash
julia:x:202:0:julia child:/home/julia/:/bin/bash
anthony:x:200:0:anthony bourdain:/home/anthony/:/bin/bash
gordon:x:201:0:gordon ramsay:/home/gordon/:/bin/bash
user1:x:100:1:I am user1:/home/user2/:/bin/bash
```

Verify the new group has been created

```
[vagrant@web1 ~]$ cat /etc/group
```

```
....
apache:x:48:
chefs:x:42:julia,anthony,gordon
```

Return to your Workstation

```
[vagrant@web1 ~]$ exit
```

```
logout
Connection to 127.0.0.1 closed.
```

Lab: Manage Users and Groups

- ✓ Create 3 users and upload them to the ‘users’ data bag
- ✓ Create and upload a ‘groups’ data bag with your users as members
- ✓ Update the ‘myusers’ cookbook to add your users to the group using search
- ✓ Verify ‘web’ role contains ‘myusers’, Converge your web server
- ✓ Verify the new users in the /etc/passwd file
- ✓ Verify the new group in the /etc/group file

Further Resources

Other Places to Talk About, Practice, and Learn Chef

MOTIVATION

Going Forward

There are many Chef resources available to you outside this class. During this module we will talk about just a few of those resources.

But…remember what we said at the beginning of this class:

The best way to learn Chef is to use Chef

CHEF

Practice Chef

First, let's talk about stuff you can read to help you learn Chef.

learn.chef.io

Interactive learning for those new to Chef.

Beyond Essentials

What happens during a knife bootstrap?
What happens during a chef-client run?
What is the security model used by chef-client
Explains Attribute Precedence

https://learn.chef.io/skills

Community Resources

Example Cookbooks, Articles, Podcasts, and More

https://github.com/obazoud/awesome-chef

CHEF

Resources You Can Read

A lot of people in the Chef community have written about Chef.

Here are just a few of those resources.

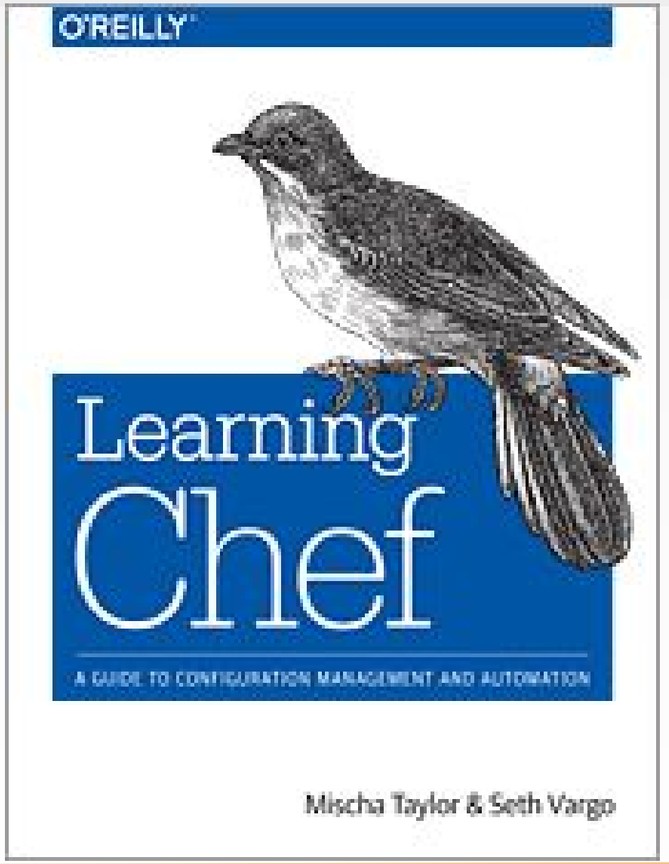
docs.chef.io

Docs are available to you, 24 hours a day, 7 days a week.

Any question you have, you probably will find the answer for on our Docs site.

Learning Chef

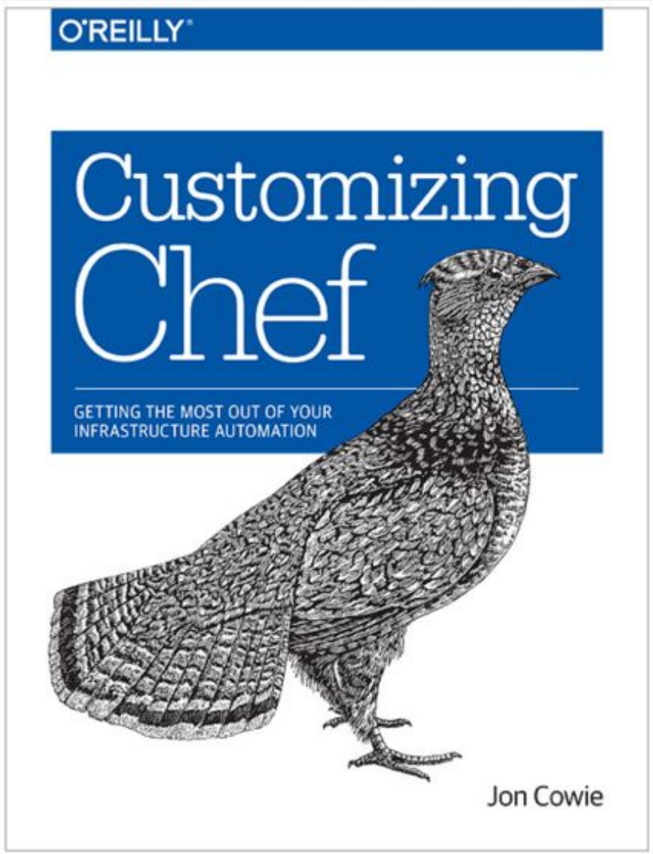
A Guide to Configuration Management and Automation



CHEF

Customizing Chef

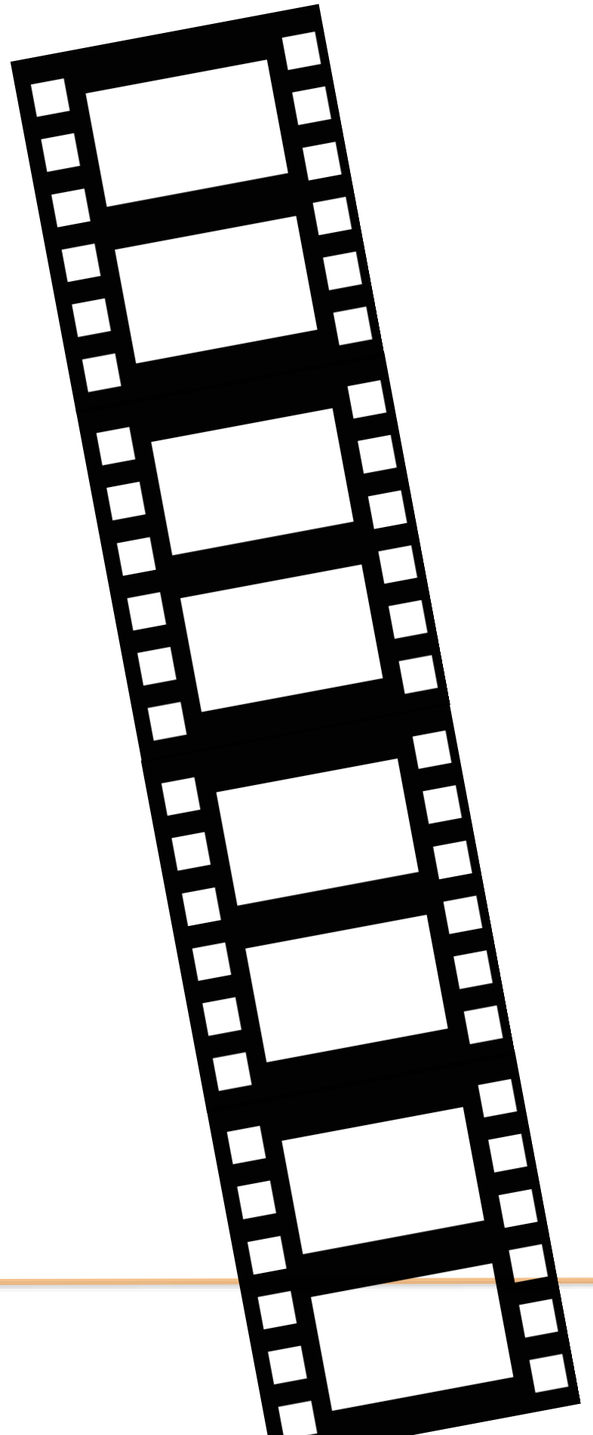
Getting the Most Out of Your Infrastructure Automation



CHEF

YouTube Channel

- ChefConf Talks
- Training Videos



https://www.youtube.com/user/getchef/playlists

foodfightshow.org

Chef Developers' IRC Meeting

https://github.com/chef/chef-community-irc-meetings

Chef Product Feedback Forum

Create your product feature ideas for the Chef engineering teams. As a registered user, you'll be able to vote on your features and the features proposed by others...

https://feedback.chef.io

Austin, TX • May 22 – 24

https://www.chef.io/chefconf/

CHEF COMMUNITY SUMMIT
SAVE THE DATE!

LONDON SUMMIT

SEATTLE SUMMIT

https://www.chef.io/summit/

CHEF™

Thank You!