**Managing multiple nodes**

**Heads up!** To follow along with the examples in this chapter you'll need the Hosted Chef account and set up described in the last two chapters.

In this chapter we'll take our example WordPress hosting set up to the enterprise level by using Chef to configure multiple nodes.

**Too much traffic?**

Having too much traffic for a single web server to handle is a nice problem to have! As our imaginary WordPress website empire grows and we get more traffic, we will need to increase our capacity beyond our little web server. We have two options: vertical or horizontal scaling.

Vertical scaling is adding more power to the machine that runs our website or application. It is often the easiest option as code only written to run on a single machine - which is usually most code - can be moved easily to a bigger machine, especially if you already have automated server setup with Chef.

At enterprise level you should probably have a server on standby for redundancy in case of hardware or software failure anyway so you may as well use it to handle traffic.

Horizontal scaling is adding more servers and sharing the load between them. It can also have the advantage of adding redundancy. If we have two web servers handling traffic and one breaks our websites will remain online. If we can separate various parts of our website or application into different services we can have even more redundancy.

Our **phpapp** default recipe already supports vertical scaling - we can use it to build a new server - but we need to make some changes to it before we can scale horizontally. Let's split up our recipe in two. We'll have a recipe that sets up the MySQL server, database and acesss details and another to set up our web server and site code.

**Two recipes**

Let's create a recipe called **mysql\_server** and **web** inside the **phpapp** cookbook.

On your workstation, create *cookbooks/phpapp/recipes/mysql\_server.rb* in your text editor.

#

# Cookbook Name:: phpapp

# Recipe:: mysql\_server

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "mysql::server"

include\_recipe "mysql::ruby"

sites = data\_bag("wp-sites")

sites.each do |site|

opts = data\_bag\_item("wp-sites", site)

mysql\_database opts["database"] do

connection ({:host => 'localhost', :username => 'root', :password => node['mysql']['server\_root\_password']})

action :create

end

mysql\_database\_user opts["db\_username"] do

connection ({:host => 'localhost', :username => 'root', :password => node['mysql']['server\_root\_password']})

password opts["db\_password"]

database\_name opts["database"]

privileges [:select,:update,:insert,:create,:delete]

action :grant

end

end

Add the code above. We've included just the parts that are needed to setup the MySQL server.

Now save the file.

Again on your workstation, create *cookbooks/phpapp/recipes/web.rb* in your text editor.

#

# Cookbook Name:: phpapp

# Recipe:: web

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

include\_recipe "mysql::client"

include\_recipe "php"

include\_recipe "php::module\_mysql"

include\_recipe "apache2::mod\_php5"

apache\_site "default" do

enable false

end

sites = data\_bag("wp-sites")

sites.each do |site|

opts = data\_bag\_item("wp-sites", site)

wordpress\_site opts["host"] do

path "/var/www/" + opts["host"]

database opts["database"]

db\_username opts["db\_username"]

db\_password opts["db\_password"]

template "site.conf.erb"

end

end

Replace the contents of the file with the above. Save the file.

Finally we'll edit the default recipe and simply make it call both new recipes. This will allow us to continue to use the recipe on a single server.

On your workstation, open *cookbooks/phpapp/recipes/default.rb* in your text editor.

#

# Cookbook Name:: phpapp

# Recipe:: default

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "phpapp::mysql\_server"

include\_recipe "phpapp::web"

Replace what's there with the code above. Save the file.

$ knife cookbook upload phpapp

Uploading phpapp [0.1.0]

Uploaded 1 cookbook.

Our server instance from the last chapter should still be running, if so let's just check our new split cookbooks still work. On your node type the following.

$ chef-client

..

Chef Client finished, 7 resources updated

Our recipe still works on a single instance. Now let's try to create an environment with more than one instance. We don't need this instance anymore so feel free to remove it.

**Multiple nodes**

We now need to create two separate roles. One for web instances and one for database instances. Firstly let's create the role for web servers.

$ knife role create phpapp-web

{

"name": "phpapp-web",

"description": "Role for web nodes",

"json\_class": "Chef::Role",

"default\_attributes": {

},

"override\_attributes": {

},

"chef\_type": "role",

"run\_list": [

"recipe[apt]", "recipe[phpapp::web]"

],

"env\_run\_lists": {

}

}

Add the code in green and save the file.

Created role[phpapp-web]

Let's download the role from the Chef server so we have it locally in a Chef repository.

$ knife role show phpapp-web -d -Fjson > roles/phpapp-web.json

Okay, let's create a role for our database server.

$ knife role create phpapp-mysql

{

"name": "phpapp-mysql",

"description": "Role for database server",

"json\_class": "Chef::Role",

"default\_attributes": {

},

"override\_attributes": {

},

"chef\_type": "role",

"run\_list": [

"recipe[apt]", "recipe[phpapp::mysql\_server]"

],

"env\_run\_lists": {

}

}

Add the code in green and save the file.

Created role[phpapp-mysql]

Again, let's download the role from the Chef server so we have it locally in a Chef repository.

$ knife role show phpapp-mysql -d -Fjson > roles/phpapp-mysql.json

So, let's create a MySQL server node. We'll bootstrap the node using knife. A couple of example commands are listed below but refer to the [Introducing Chef Server](http://gettingstartedwithchef.com/introducing-chef-server.html) chapter for more details.

* [Bootstrap via SSH](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd1-bootstrap)
* [Brightbox](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd1-brightbox)
* [Rackspace](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd1-rackspace)
* [Amazon](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd1-amazon)

$ knife bootstrap --run-list "role[phpapp-mysql]" --sudo hostname

Replace hostname with the name or IP of the machine.

Once your Chef bootstrap is complete you'll get the following message.

..

xxx.xxx.xxx.xxx Chef Client finished, 28 resources updated

**MySQL host**

So that's our MySQL server set up. Now we need to create a instance to serve web traffic. First we need to change our cookbook so it supports using a MySQL server on a host other than **localhost**.

First, we'll edit our **wp-config.php.erb** template.

Open *cookbooks/phpapp/templates/default/wp-config.php.erb* in your text editor.

<?php

define('DB\_NAME', '<%= @database %>');

define('DB\_USER', '<%= @user %>');

define('DB\_PASSWORD', '<%= @password %>');

define('DB\_HOST', '<%= @host %>');

define('DB\_CHARSET', 'utf8');

define('DB\_COLLATE', '');

<%= @wp\_secrets %>

$table\_prefix = 'wp\_';

define('WPLANG', '');

define('WP\_DEBUG', false);

if ( !defined('ABSPATH') )

define('ABSPATH', dirname(\_\_FILE\_\_) . '/');

require\_once(ABSPATH . 'wp-settings.php');

Replace **localhost** with the code in green. Save the file.

Now open *cookbooks/phpapp/definitions/wordpress\_site.rb* in your text editor.

define :wordpress\_site, :path => "/var/www/phpapp", :database => "phpapp", :db\_username => "phpapp", :db\_password => "phpapp", :db\_host => "localhost", :template => "site.conf.erb" do

wordpress\_latest = Chef::Config[:file\_cache\_path] + '/wordpress-latest.tar.gz'

remote\_file wordpress\_latest do

source 'http://wordpress.org/latest.tar.gz'

mode 0644

end

directory params[:path] do

owner 'root'

group 'root'

mode 0755

action :create

recursive true

end

execute 'untar-wordpress' do

cwd params[:path]

command 'tar --strip-components 1 -xzf ' + wordpress\_latest

creates params[:path] + '/wp-settings.php'

end

wp\_secrets = Chef::Config[:file\_cache\_path] + '/wp-secrets.php'

remote\_file wp\_secrets do

source 'https://api.wordpress.org/secret-key/1.1/salt/'

action :create\_if\_missing

mode 0644

end

salt\_data = ''

ruby\_block 'fetch-salt-data' do

block do

salt\_data = File.read(wp\_secrets)

end

action :create

end

template params[:path] + '/wp-config.php' do

source 'wp-config.php.erb'

mode 0755

owner 'root'

group 'root'

variables(

:database => params[:database],

:user => params[:db\_username],

:password => params[:db\_password],

:host => params[:db\_host],

:wp\_secrets => salt\_data)

end

docroot = params[:path]

server\_name = params[:name]

web\_app server\_name do

template "site.conf.erb"

docroot docroot

server\_name server\_name

end

end

Add the **two bits of code** in green. Save the file.

So we can now specify the host. But how do we find it out what the database host is? We can search for a node with the **phpapp-mysql** role.

Let's go back to the **chef-shell** and learn how to search for our role.

$ chef-shell -c .chef/knife.rb

loading configuration: none (standalone session)

Session type: standalone

Loading...done.

This is the chef-shell.

Chef Version: 11.x.x

http://www.opscode.com/chef

http://wiki.opscode.com/display/chef/Home

run `help' for help, `exit' or ^D to quit.

Ohai2u andy@computer!

chef > nodes.search('role:phpapp-mysql')

=> [node[servername-11001]]

Type the command in bold, and you'll receive a response similar to the one above. It returns a list of nodes. The list is called an array. To view the first item of the array we need to use **[0]**.

chef > nodes.search('role:phpapp-mysql')[0]

=> node[servername-11001]

We want to find the IP address of our MySQL server node. We can view all the attributes available with **y** command which will print the data out in YAML format. The attribute you need might be different on platforms.

chef > y nodes.search('role:phpapp-mysql')[0]

...

That will return all the attributes. You'll need to hunt through and find the IP address of the node. You should use the private IP where there is one.

A few examples of where that IP might be:

chef > nodes.search('role:phpapp-mysql')[0].cloud.private\_ips[0]

=> 10.x.x.x

chef > y nodes.search('role:phpapp-mysql')[0].ipaddress

=> 10.x.x.x

Make a note of where the private IP address can be found. Press Ctrl-D to quit chef-shell. Now let's update our recipe to do the search.

Open *cookbooks/phpapp/recipes/web.rb* in your text editor.

#

# Cookbook Name:: phpapp

# Recipe:: web

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

include\_recipe "mysql::client"

include\_recipe "php"

include\_recipe "php::module\_mysql"

include\_recipe "apache2::mod\_php5"

apache\_site "default" do

enable false

end

db\_hosts = search(:node, 'role:phpapp-mysql')

if db\_hosts.empty?

db\_host = 'localhost'

else

begin

db\_host = db\_hosts[0].cloud.private\_ips[0]

rescue NoMethodError => e

db\_host = db\_hosts[0].ipaddress

end

end

sites = data\_bag("wp-sites")

sites.each do |site|

opts = data\_bag\_item("wp-sites", site)

wordpress\_site opts["host"] do

path "/var/www/" + opts["host"]

database opts["database"]

db\_host db\_host

db\_username opts["db\_username"]

db\_password opts["db\_password"]

template "site.conf.erb"

end

end

Add the code in green. You may need to alter it if you found the private IP in a different place but the code we've provided should work for most. Our changes default to "localhost" so the recipe still works when running on a single server.

Now we upload the latest version of our cookbook!

$ knife cookbook upload phpapp

Uploading phpapp [0.1.0]

Uploaded 1 cookbook.

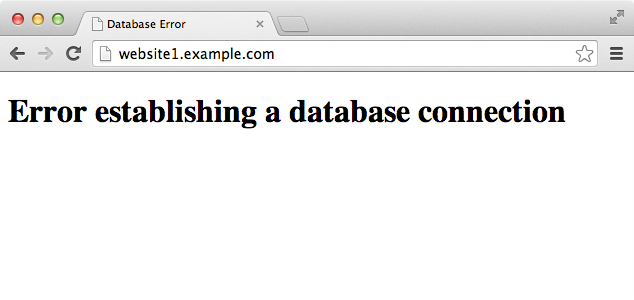
Let's create a web server.

* [Bootstrap via SSH](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd2-bootstrap)
* [Brightbox](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd2-brightbox)
* [Rackspace](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd2-rackspace)
* [Amazon](http://gettingstartedwithchef.com/managing-multiple-nodes.html#cmd2-amazon)

$ knife bootstrap --run-list "role[phpapp-web]" --sudo hostname

Replace hostname with the name or IP of the machine.

Once your node is bootstrapped, update your hosts file entries for the \*.example.com hosts we created in the last chapter and open [website1.example.com](http://website1.example.com) in your web browser.



Oh dear! Why has that happened? The problem is in our mysql\_server recipe.

#

# Cookbook Name:: phpapp

# Recipe:: mysql\_server

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "mysql::server"

include\_recipe "mysql::ruby"

sites = data\_bag("wp-sites")

sites.each do |site|

opts = data\_bag\_item("wp-sites", site)

mysql\_database opts["database"] do

connection ({:host => 'localhost', :username => 'root', :password => node['mysql']['server\_root\_password']})

action :create

end

mysql\_database\_user opts["db\_username"] do

connection ({:host => 'localhost', :username => 'root', :password => node['mysql']['server\_root\_password']})

password opts["db\_password"]

database\_name opts["database"]

privileges [:select,:update,:insert,:create,:delete]

action :grant

end

end

Our issue is that our call to the **mysql\_database\_user** resource does not specify at host for the user. By default the MySQL user that the resource creates is only allowed to connect from **localhost**. This presents an interesting issue. We could search for nodes with the **phpapp-web** role and create a user that can connect from each node's IP address but that means any new web server we create will have to wait for **chef-client** to run again on the MySQL server before it can connect. This isn't a great solution because if we are creating additional web servers it's probably because we want to handle more traffic! Waiting isn't an option.

For now, we'll create a MySQL user that can connect from the private IP range 10.x.x.x. Many cloud providers give you a virtual LAN for your account which keeps the traffic between your nodes private so this isn't an issue but in a production environment you should be sure that you lock this down properly.

First of all, let's have an attribute that allows us to specify a default host for our MySQL users.

Open *cookbooks/phpapp/attributes/default.rb* in your text editor.

default["phpapp"]["database"] = "phpapp"

default["phpapp"]["db\_username"] = "phpapp"

default['phpapp']['path'] = '/var/www/phpapp'

default['phpapp']['server\_name'] = 'phpapp'

default["phpapp"]["db\_user\_host"] = "localhost"

Add the code in green. Save the file.

Open *cookbooks/phpapp/recipes/mysql\_server.rb* in your text editor.

#

# Cookbook Name:: phpapp

# Recipe:: mysql\_server

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "mysql::server"

include\_recipe "mysql::ruby"

sites = data\_bag("wp-sites")

sites.each do |site|

opts = data\_bag\_item("wp-sites", site)

mysql\_database opts["database"] do

connection ({:host => 'localhost', :username => 'root', :password => node['mysql']['server\_root\_password']})

action :create

end

mysql\_database\_user opts["db\_username"] do

connection ({:host => 'localhost', :username => 'root', :password => node['mysql']['server\_root\_password']})

password opts["db\_password"]

database\_name opts["database"]

privileges [:select,:update,:insert,:create,:delete]

host node['phpapp']['db\_user\_host']

action :grant

end

end

Add the code in green. Save the file.

We'll now upload our cookbook changes.

$ knife cookbook upload phpapp

Uploading phpapp [0.1.0]

Uploaded 1 cookbook.

Now we'll edit our **phpapp-mysql** role and override the default attribute with our value that is specific to our set up.

$ knife role edit phpapp-mysql

{

"name": "phpapp-mysql",

"description": "Role for database server",

"json\_class": "Chef::Role",

"default\_attributes": {

phpapp": {

"db\_user\_host": "10.%"

}

},

"override\_attributes": {

},

"chef\_type": "role",

"run\_list": [

"recipe[apt]",

"recipe[phpapp::mysql\_server]"

],

"env\_run\_lists": {

}

}

Save your changes to update the role.

Let's download the role from the Chef server so we have it locally in a Chef repository.

$ knife role show phpapp-mysql -d -Fjson > roles/phpapp-mysql.json

We're good to go. Let's connect to the node with the role **phpapp-mysql** and run **chef-client**.

# chef-client

..

Chef Client finished, 4 resources updated

If you visit [website1.example.com](http://website1.example.com) it might now work and you'll get the WordPress installation screen again. However you might find you are still unable to connect. That's what happened to me. I'll show you now how I debugged the issue.

First things first, let's see if we can connect from the command line on the web server node. SSH to that node.

Let's remind ourselves of the details that WordPress is using to connect to the database.

$ cat /var/www/website1.example.com/wp-config.php

<?php

define('DB\_NAME', 'website1');

define('DB\_USER', 'website1');

define('DB\_PASSWORD', '212b09752d173876a84d374333ae1ffe');

define('DB\_HOST', '10.226.16.22');

define('DB\_CHARSET', 'utf8');

define('DB\_COLLATE', '');

We'll use those details now to attempt to connect.

$ mysql -u website1 -h 10.226.16.22 -p212b09752d173876a84d374333ae1ffe website1

ERROR 1129 (HY000): Host 'srv-9mxex.gb1.brightbox.com' is blocked because of many connection errors; unblock with 'mysqladmin flush-hosts'

This means the MySQL server has stopped our web server node connecting because we've been using incorrect access details. Disconnect from the web server node.

Back on your workstation run the following command to find out the root password for the MySQL server.

$ knife search role:phpapp-mysql --attribute mysql.server\_root\_password

1 items found

gswc-mysql:

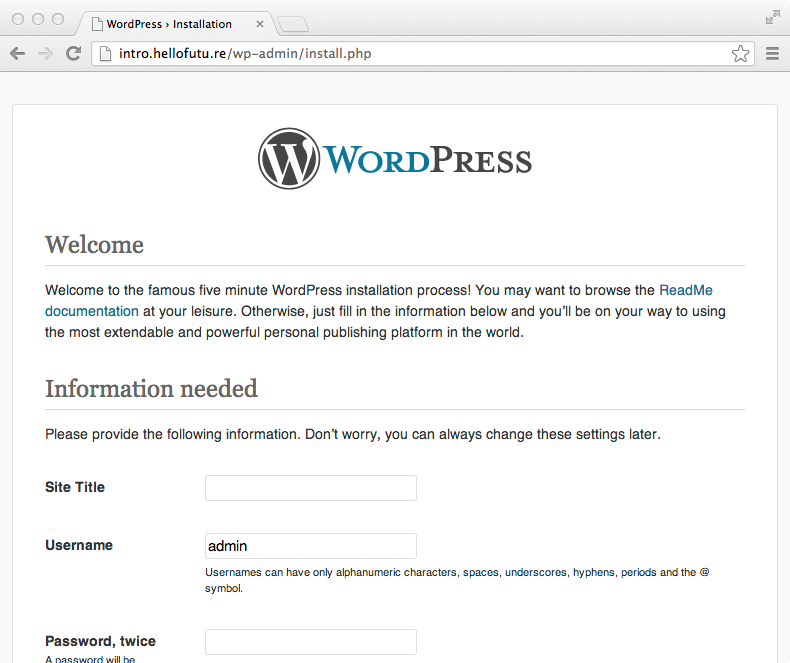
mysql.server\_root\_password: IARHKfG0I\_fiujJ5uBfd

Neat, huh?

Now SSH to the MySQL server node. On that node run the following command.

# mysqladmin flush-hosts -uroot -p

Paste in the password and you should now be able to view the WordPress installation page.



Success! We now have the basis of system that allows us to add more web nodes to handle traffic increases. These more complicated configurations often require a bit more thought however.

For example, by default WordPress saves uploads to the local disk. These uploads would then not be accessable on the other node. There are WordPress plugins that instead store these files to Amazon S3 or alternatives. Issues with multiple node configurations can often benefit from features of individual cloud providers like this.

That's the end of this chapter. In the next chapter we'll look at ways we can speed up our development of cookbooks and improve our workflow.

Don't forget to delete those two nodes we created.