**Getting started with Chef**

**1. First steps with Chef**

In the first chapter we'll introduce you to Chef and we'll be working over a SSH connection. There are better ways to work with Chef and we'll get to them in later chapters but things will be kept as simple as possible to start with.

The simplest way to use Chef is [chef-solo](http://docs.opscode.com/chef_solo.html). It allows you to install, configure and manage the packages required by your application without the complication of any client and server configuration. We'll start with the common scenario that you have to setup a website designer with a WordPress environment.

Every time you do this you have to setup a web server, remember lots of installation commands, edit configuration files, fetch a copy of WordPress and do lots of setup. Generally you always forget one step and it is an unnecessary distraction from what you should be doing. Wouldn't it be nice if this was all entirely automated? With Chef we can define our infrastructure as code and automate tasks just like this.

So before we start, we need somewhere test our code. We'll introduce you to tools that help you manage your Chef development and testing later in the book but for now we'll just need root access to a fresh install of **Ubuntu 12.04**. Don't just run Chef on your Ubuntu or Mac desktop, we need somewhere we can play around and re-image later.

In order to get the most out of this book, you should sign up with a cloud server provider. [Brightbox](http://brightbox.com/), [Amazon EC2](http://aws.amazon.com/ec2/) or [Rackspace](http://www.rackspace.com/cloud/servers/) are supported throughout the guide. Alternatively, you can install Ubuntu in a virtual machine. If you do, create a snapshot once Ubuntu is installed so we can re-use the fresh installation later on.

**Installing Chef**

SSH to your vanilla Ubuntu box and run the following command to install Chef. This used to be a more involved process but thanks to the new omnibus installer it couldn't be simpler.

root@intro:~# cd ~

root@intro:~# curl -L https://www.opscode.com/chef/install.sh | bash

Thank you for installing Chef!

Confirm Chef has successfully installed.

root@intro:~# chef-solo -v

...

Chef: 11.4.0

Of course, your version number may be different.

**Our first Chef cookbook**

So, what do we need to do to get our web server up and running?

* Install and configure Apache
* Install and configure MySQL
* Install and configure PHP
* Deploy our website code to the site

How do we do that? We write our first Chef cookbook. But before we do that we should setup a file structure that will help us organise our various Chef files. Opscode, the makers of Chef provide one. They call it simply the Chef Repository.

root@intro:~# wget http://github.com/opscode/chef-repo/tarball/master

root@intro:~# tar -zxf master

root@intro:~# mv chef-chef-repo\* chef-repo

root@intro:~# rm master

If we look inside the chef-repo directory we can see the following:

root@intro:~# cd chef-repo/

root@intro:~/chef-repo# ls

certificates chefignore config cookbooks data\_bags environments LICENSE Rakefile

README.md roles

Our Chef cookbook should unsurprisingly reside within the cookbooks directory. We're going to call it "phpapp". We can use the command [knife](http://docs.opscode.com/knife.html) to help us manage our cookbooks. First we should tell knife where to find our cookbooks directory.

root@intro:~/chef-repo# mkdir .chef

root@intro:~/chef-repo# echo "cookbook\_path [ '/root/chef-repo/cookbooks' ]" > .chef/knife.rb

That creates a simple configuration file for knife which tells knife to use the cookbook directory inside our Chef Repository. Now we'll ask knife to create our "phpapp" cookbook.

root@intro:~/chef-repo# knife cookbook create phpapp

\*\* Creating cookbook phpapp

\*\* Creating README for cookbook: phpapp

\*\* Creating CHANGELOG for cookbook: phpapp

\*\* Creating metadata for cookbook: phpapp

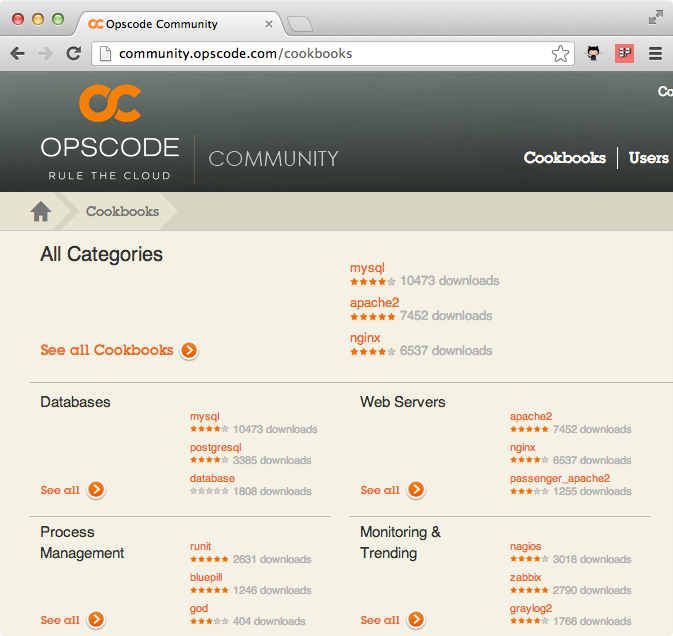
So let's look at what knife has created.

root@intro:~/chef-repo# cd cookbooks/phpapp

root@intro:~/chef-repo/cookbooks/phpapp# ls

attributes CHANGELOG.md definitions files libraries metadata.rb providers README.md  
recipes resources templates

So now we'd need write our cookbook to install and configure Apache, MySQL and PHP. How do we do that? Well, thanks to the open source nature of Chef, we don't have to. Welcome to the [Opscode Community](http://community.opscode.com/cookbooks) cookbook site.



Here you'll find lots of well crafted, tested and battle hardened cookbooks that will do most of the work for you. Think of them as libraries you will use inside your code. We'll start with the [apache2](http://community.opscode.com/cookbooks/apache2) cookbook. There's no need to manually download it from the community site, knife has this functionality built in. We'll also install the [apt](http://community.opscode.com/cookbooks/apt) cookbook. This will help us ensure chef-solo does an apt-get update before we install any packages.

root@intro:~/chef-repo/cookbooks/phpapp# cd ..

root@intro:~/chef-repo/cookbooks# knife cookbook site download apache2

Downloading apache2 from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/apache2-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/apache2-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf apache2\*

root@intro:~/chef-repo/cookbooks# rm apache2\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download apt

Downloading apt from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/apt-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/apt-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf apt\*

root@intro:~/chef-repo/cookbooks# rm apt\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download iptables

Downloading iptables from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/iptables-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/iptables-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf iptables\*

root@intro:~/chef-repo/cookbooks# rm iptables\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download logrotate

Downloading logrotate from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/logrotate-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/logrotate-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf logrotate\*

root@intro:~/chef-repo/cookbooks# rm logrotate\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download pacman

Downloading pacman from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/pacman-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/logrotate-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf pacman\*

root@intro:~/chef-repo/cookbooks# rm pacman\*.tar.gz

Let's go back into our cookbook.

root@intro:~/chef-repo/cookbooks# cd phpapp

Open metadata.rb in your favourite text editor. Vim or nano are both available by default on Ubuntu. We suggest using nano if you're not used to Vim. So type "nano metadata.rb".

name 'phpapp'

maintainer 'YOUR\_COMPANY\_NAME'

maintainer\_email 'YOUR\_EMAIL'

license 'All rights reserved'

description 'Installs/Configures phpapp'

long\_description IO.read(File.join(File.dirname(\_\_FILE\_\_), 'README.md'))

version '0.1.0'

**depends "apache2"**

Add the line in green. This tells Chef our cookbook relies on the apache2 cookbook to function. Save the file. Then open 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 "apache2"**

**apache\_site "default" do**

**enable true**

**end**

Again add the lines in green. This includes the default recipe from the apache2 cookbook in our recipe. The default apache2 recipe (which can be found in cookbooks/apache2/recipes/default.rb) installs and configures Apache for us.

Okay. Let's see if what we've got so far works! Go back to the chef-repo directory.

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

Create a new file called solo.rb in your text editor.

**file\_cache\_path "/root/chef-solo"**

**cookbook\_path "/root/chef-repo/cookbooks"**

Add the lines in green. This file configures chef-solo, telling it where to keep its cache of files and where our cookbooks are. Save the file. Now create a file called web.json.

**{**

**"run\_list": [ "recipe[apt]", "recipe[phpapp]" ]**

**}**

Add the lines in green. We tell chef to run the apt cookbook followed by our phpapp cookbook. Why have we not included the apt cookbook inside our recipe as we did with the apache2 cookbook? It's because our PHP application requires Apache to function but we don't necessarily want to tie our cookbook to platforms that only support apt.

**Our first Chef run**

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

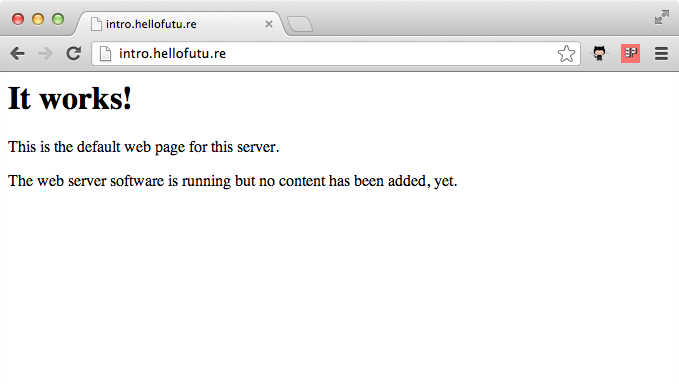
Starting Chef Client, version 11.4.0

...

Chef Client finished, 14 resources updated

Chef gives you comprehensive information about exactly what it's done. By default, the actions it's taken are displayed in green and when it updates a template it shows you what's changed.

Now you can visit your new Apache server.



Next we'll setup MySQL. As the community site has a [cookbook for MySQL](http://community.opscode.com/cookbooks/mysql), the process is similar to Apache. Again we'll ask knife to fetch the cookbook from the community site for us.

root@intro:~/chef-repo# cd cookbooks

root@intro:~/chef-repo/cookbooks# knife cookbook site download mysql 4.1.2

Downloading mysql from the cookbooks site at version 4.1.2 to /root/chef-repo/cookbooks/mysql-4.1.2.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/mysql-4.1.2.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf mysql\*

root@intro:~/chef-repo/cookbooks# rm mysql-\*.tar.gz

You may notice that we've specified **4.1.2** after the cookbook name. That's because we want to use that version rather than the latest version as version 5 introduces some great changes but will break our recipe.

So let's install MySQL then. We want to install both the MySQL client and the server as we'll be running our application on a single instance for now. Look inside the MySQL cookbook and see exactly what we need to include in our recipe.

root@intro:~/chef-repo/cookbooks# cd mysql/recipes/

root@intro:~/chef-repo/cookbooks/mysql/recipes# ls

client.rb default.rb ruby.rb server\_ec2.rb server.rb

There's a client recipe and a server recipe. We'll need to include both then. Go back to our cookbook.

root@intro:~/chef-repo/cookbooks/mysql/recipes# cd ../../phpapp

Open metadata.rb in your text editor.

name 'phpapp'

maintainer 'YOUR\_COMPANY\_NAME'

maintainer\_email 'YOUR\_EMAIL'

license 'All rights reserved'

description 'Installs/Configures phpapp'

long\_description IO.read(File.join(File.dirname(\_\_FILE\_\_), 'README.md'))

version '0.1.0'

depends "apache2"

**depends "mysql", "4.1.2"**

Add the line in green and save the file. You'll notice we include a specific version for the mysql cookbook here as well. Now edit recipes/default.rb.

#

# Cookbook Name:: phpapp

# Recipe:: default

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

**include\_recipe "mysql::client"**

**include\_recipe "mysql::server"**

apache\_site "default" do

enable true

end

Add the two lines in green and save the file. Now let's run chef-solo again!

**Our second Chef run**

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

Compiling Cookbooks...

[2013-02-11T21:47:33+00:00] ERROR: Running exception handlers

[2013-02-11T21:47:33+00:00] ERROR: Exception handlers complete

Chef Client failed. 0 resources updated

[2013-02-11T21:47:33+00:00] FATAL: Stacktrace dumped to /root/chef-solo/chef-stacktrace.out

[2013-02-11T21:47:33+00:00] FATAL: Chef::Exceptions::CookbookNotFound: Cookbook build-essential not found. If you're loading build-essential from another cookbook, make sure you configure the dependency in your metadata

Oh that's not good! We've got an error. The cookbook build-essential is not found. We haven't included it in our cookbook so it's probably required by the mysql cookbook we've just added. We know we have to specify the cookbook dependencies for our cookbook in metadata.rb, so we'll look at the metadata.rb file in the mysql cookbook.

Open cookbooks/mysql/metadata.rb in your editor.

name "mysql"

maintainer "Opscode, Inc."

maintainer\_email "cookbooks@opscode.com"

license "Apache 2.0"

description "Installs and configures mysql for client or server"

long\_description IO.read(File.join(File.dirname(\_\_FILE\_\_), 'README.md'))

version "2.1.2"

recipe "mysql", "Includes the client recipe to configure a client"

recipe "mysql::client", "Installs packages required for mysql clients using run\_action magic"

recipe "mysql::server", "Installs packages required for mysql servers w/o manual intervention"

recipe "mysql::server\_ec2", "Performs EC2-specific mountpoint manipulation"

%w{ debian ubuntu centos suse fedora redhat scientific amazon freebsd windows mac\_os\_x }.each do |os|

supports os

end

**depends "openssl"**

**depends "build-essential"**

suggests "homebrew"

suggests "windows"

There's more below but let's just concentrate on depends and suggests. The suggests entries tell Chef that some optional functionality may depend on that cookbook. We already know the depends entries specify a hard requirement. So we need to download [build-essential](http://community.opscode.com/cookbooks/build-essential) and [openssl](http://community.opscode.com/cookbooks/openssl).

There are some good tools available to help you manage cookbooks and dependences which we'll cover in later chapters but we'll just download the required cookbooks using knife for now.

root@intro:~/chef-repo# cd cookbooks

root@intro:~/chef-repo/cookbooks# knife cookbook site download openssl

Downloading openssl from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/openssl-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/openssl-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf openssl\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm openssl\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download build-essential

Downloading build-essential from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/build-essential-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/build-essential-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf build-essential-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm build-essential-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download homebrew

Downloading homebrew from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/homebrew-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/homebrew-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf homebrew-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm homebrew-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download windows

Downloading windows from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/windows-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/windows-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf windows-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm windows-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download chef\_handler

Downloading chef\_handler from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/chef\_handler-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/chef\_handler-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf chef\_handler-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm chef\_handler-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download chef-sugar

Downloading chef-sugar from the cookbooks site at version x.x.x to /root/chef-repo/cookbooks/chef-sugar-x.x.x.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/chef-sugar-x.x.x.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf chef-sugar-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm chef-sugar-\*.tar.gz

And now we've fulfilled those dependencies let's try and re-run chef-solo!

root@intro:~/chef-repo/cookbooks# cd ..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

Compiling Cookbooks...

[2013-02-12T18:44:18+00:00] WARN: Cloning resource attributes for service[apache2] from prior resource (CHEF-3694)

[2013-02-12T18:44:18+00:00] WARN: Previous service[apache2]: /root/chef-repo/cookbooks/apache2/recipes/default.rb:24:in `from\_file'

[2013-02-12T18:44:18+00:00] WARN: Current service[apache2]: /root/chef-repo/cookbooks/apache2/recipes/default.rb:221:in `from\_file'

[2013-02-12T18:44:18+00:00] FATAL: You must set node['mysql']['server\_debian\_password'], node['mysql']['server\_root\_password'], node['mysql']['server\_repl\_password'] in chef-solo mode. For more information, see https://github.com/opscode-cookbooks/mysql#chef-solo-note

Again we have an error. But that's okay, the MySQL cookbook has told us how to fix the error. We need to define a root password for MySQL. This is an [attribute](http://docs.opscode.com/chef_overview_attributes.html). In Chef, attributes are values which we use to configure our applications or platform. An attribute could be a port number for Apache. Often a sensible default is specified inside a cookbook. Such a default for a web server port would be 80. There's no sensible default MySQL password, so we need to specify one. Open web.json.

{

"mysql": {"server\_root\_password": "808052769e2c6d909027a2905b224bad", "server\_debian\_password": "569d1ed2d46870cc020fa87be83af98d", "server\_repl\_password": "476911180ee92a2ee5a471f33340f6f4"},

"run\_list": [ "recipe[apt]", "recipe[phpapp]" ]

}

Add the code in green. Let's re-run chef-solo.

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 14 resources updated

Much more successful, we now have MySQL. Now to install PHP. You guessed it, there's a community cookbook for PHP.

root@intro:~/chef-repo# cd cookbooks/

root@intro:~/chef-repo/cookbooks# knife cookbook site download php

Downloading php from the cookbooks site at version 1.1.8 to /root/chef-repo/cookbooks/php-1.1.8.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/php-1.1.8.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf php\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm php\*.tar.gz

The php cookbook depends on the **xml**, **yum-epel**, **windows**, and **iis** cookbooks, so we'll need those even though we won't be using all of them. We'll also have to install sub-dependencies **yum** (a dependency of yum-epel), **chef\_handler**, and **powershell** (dependencies of windows).

root@intro:~/chef-repo/cookbooks# knife cookbook site download xml

Downloading xml from the cookbooks site at version 1.2.0 to /root/chef-repo/cookbooks/xml-1.2.0.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/xml-1.2.0.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf xml-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download yum

Downloading yum from the cookbooks site at version 3.0.4 to /root/chef-repo/cookbooks/yum-3.0.4.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/yum-3.0.4.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf yum-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download yum-epel

Downloading yum-epel from the cookbooks site at version 0.2.0 to /root/chef-repo/cookbooks/yum-epel-0.2.0.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/yum-epel-0.2.0.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf yum-epel-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download powershell

Downloading powershell from the cookbooks site at version 1.1.2 to /root/chef-repo/cookbooks/powershell-1.1.2.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/powershell-1.1.2.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf powershell-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download iis

Downloading iis from the cookbooks site at version 1.6.6 to /root/chef-repo/cookbooks/iis-1.6.6.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/iis-1.6.6.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf iis-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm \*.tar.gz

Let's use the php cookbook in our cookbook.

root@intro:~/chef-repo/cookbooks# cd phpapp

Next we add the new php cookbook as a dependency for our cookbook. Open metadata.rb.

name 'phpapp'

maintainer 'YOUR\_COMPANY\_NAME'

maintainer\_email 'YOUR\_EMAIL'

license 'All rights reserved'

description 'Installs/Configures phpapp'

long\_description IO.read(File.join(File.dirname(\_\_FILE\_\_), 'README.md'))

version '0.1.0'

depends "apache2"

depends "mysql"

depends "php"

Add the code in green and save the file. And now lets include the php recipe to our cookbook. Open recipes/default.rb.

#

# Cookbook Name:: phpapp

# Recipe:: default

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

include\_recipe "mysql::client"

include\_recipe "mysql::server"

include\_recipe "php"

include\_recipe "php::module\_mysql"

include\_recipe "apache2::mod\_php5"

apache\_site "default" do

enable true

end

Add the code in green. Here we add the PHP default recipe, one to install the PHP MySQL extension and one to enable the Apache PHP module mod\_php. We also enable the default site so we can check our installation has worked. Save the file and we're good to run chef-solo again to install all of those things.

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

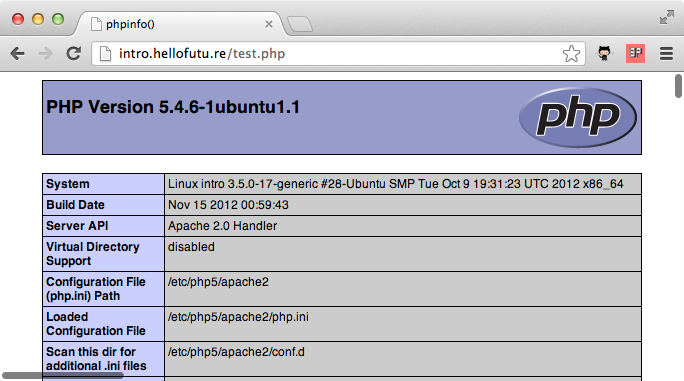
...

Chef Client finished, 8 resources updated

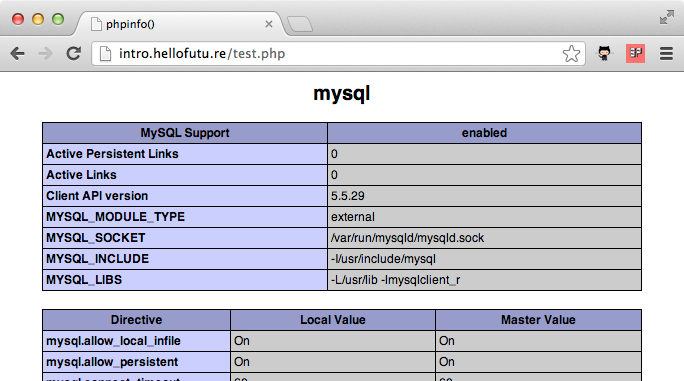
So that's PHP installed. Let's confirm that by creating a test page. Open /var/www/test.php in your editor.

<?php phpinfo(); ?>

Add the code in green and save the file. Now goto http://yourserver/test.php



Scroll down and make sure the MySQL extension is installed.



We can see MySQL is installed. Let's delete the test file.

root@intro:~/chef-repo# rm /var/www/test.php

**Idempotence**

At this point you might well be thinking all that was a lot of effort for not much reward. You could have installed that with a line similar to "apt-get install mysql-server apache2 php5 libapache2-mod-php5". What's happened behind the scenes though is that as well as installing our packages, Chef has updated all of the configuration files for each package with consistent defaults. If we bring up another server and run the same cookbooks, the same things will be installed. We can repeat the chef-solo command repeatedly and we'll end up with the same result as we did the first time we ran it. The process is idempotent; it will always produce the same result no matter how many times it is run.

Now if you were setting up a server by hand, this is where you'd manually setup a database, copy the web application code over, create a MySQL user for the website and configure virtual hosts. Instead, we'll use Chef to automate the setup of our application. This allows us to set up multiple servers and know we will always get the same results.

Before we setup our database we need to fetch a final few cookbooks. The **database** cookbook provides resources that allow us to easily manage databases and database users. The database cookbook depends on the **postgresql**, **mariadb**, **xfs** and **aws** cookbooks so we'll need those as well even though we won't be using them. We'll also fetch the **mysql2\_chef\_gem** cookbook which the **database** cookbook requires when used with MySQL.

root@intro:~/chef-repo/cookbooks# knife cookbook site download database

Downloading database from the cookbooks site at version 1.3.12 to /root/chef-repo/cookbooks/database-1.3.12.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/database-1.3.12.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf database-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download postgresql

Downloading postgresql from the cookbooks site at version 2.2.2 to /root/chef-repo/cookbooks/postgresql-2.2.2.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/postgresql-2.2.2.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf postgresql-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download xfs

Downloading xfs from the cookbooks site at version 1.1.0 to /root/chef-repo/cookbooks/xfs-1.1.0.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/xfs-1.1.0.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf xfs-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download mariadb

Downloading mariadb from the cookbooks site at version 1.1.0 to /root/chef-repo/cookbooks/mariadb-1.1.0.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/mariadb-1.1.0.tar.gz

root@intro:~/chef-repo/cookbooks# tar mariadb xfs-\*.tar.gz

root@intro:~/chef-repo/cookbooks# knife cookbook site download aws

Downloading aws from the cookbooks site at version 0.100.6 to /root/chef-repo/cookbooks/aws-0.100.6.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/aws-0.100.6.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf aws-\*.tar.gz

@intro:~/chef-repo/cookbooks# knife cookbook site download mysql2\_chef\_gem 0.1.0

Downloading aws from the cookbooks site at version 0.1.0 to /root/chef-repo/cookbooks/mysql2\_chef\_gem-0.1.0.tar.gz

Cookbook saved: /root/chef-repo/cookbooks/mysql2\_chef\_gem-0.1.0.tar.gz

root@intro:~/chef-repo/cookbooks# tar zxf mysql2\_chef\_gem-\*.tar.gz

root@intro:~/chef-repo/cookbooks# rm \*.tar.gz

root@intro:~/chef-repo/cookbooks# cd phpapp

We'll now add the new dependencies to our cookbook. Open metadata.rb.

name 'phpapp'

maintainer 'YOUR\_COMPANY\_NAME'

maintainer\_email 'YOUR\_EMAIL'

license 'All rights reserved'

description 'Installs/Configures phpapp'

long\_description IO.read(File.join(File.dirname(\_\_FILE\_\_), 'README.md'))

version '0.1.0'

depends "apache2"

depends "mysql"

depends "php"

depends "database"

depends "mysql2\_chef\_gem"

Add the code in green and save the file. It's now time to write our first Chef recipe. To setup our web application we need to:

1. Create a MySQL database for our application
2. Create a MySQL database user for our application
3. Fetch the code for our web application
4. Create a configuration file with database details and other configuration options for our web application
5. Create an Apache VirtualHost for our web application

**Resources**

A [resource](http://docs.opscode.com/resource.html) is an action that your recipe can perform. The [template](http://docs.opscode.com/resource_template.html) resource creates a file by expanding variables in a template. The user resource can be used to manage users. The database cookbook provides the resource **mysql\_database** which we will now use to perform the first step.

Open recipes/default.rb

#

# Cookbook Name:: phpapp

# Recipe:: default

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

include\_recipe "mysql::client"

include\_recipe "mysql::server"

include\_recipe "php"

include\_recipe "php::module\_mysql"

include\_recipe "apache2::mod\_php5"

include\_recipe "mysql::ruby"

include\_recipe "mysql2\_chef\_gem"

apache\_site "default" do

enable true

end

mysql\_database 'phpapp' do

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

action :create

end

Add the code in green but hang on a minute. We've hard-coded the name of the database name to be "phpapp". That could prevent our recipe from being reusable. Really the database name should be an attribute. Let's change our code so that's the case.

We've also included the **mysql2\_chef\_gem** recipe which installs a Ruby library (called a "gem") that allows Chef to interact with a MySQL server.

mysql\_database node['phpapp']['database'] do

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

action :create

end

Replace 'phpapp' with node['phpapp']['database']. Save the file. Let's see run chef-solo again and see if our changes are successful.

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

And that's failed. Let's look at the key parts of the message we received; highlighted bellow in yellow.

NoMethodError

-------------

undefined method `[]' for nil:NilClass

Cookbook Trace:

---------------

/root/chef-repo/cookbooks/phpapp/recipes/default.rb:17:in `from\_file'

Relevant File Content:

----------------------

/root/chef-repo/cookbooks/phpapp/recipes/default.rb:

16: include\_recipe "mysql::ruby"

17: include\_recipe "mysql2\_chef\_gem"

18:

19: apache\_site "default" do

20: enable true

21: end

22:

23>> mysql\_database node['phpapp']['database'] do

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

25: action :create

26: end

27:

The message **undefined method `[]' for nil:NilClass** is Ruby telling us that the attribute **node['phpapp']['database']** doesn't exist. We need to define it.

**Attributes**

The missing database attribute can be defined in a few places. We could define the attribute in web.json like we did with the MySQL ones but that makes our cookbook unnecessarily difficult to use. We want to provide the option to use a database called something other than **phpapp** but we should really provide a default.

root@intro:~/chef-repo# cd cookbooks/phpapp

We now create called attributes/default.rb. Open it in your editor.

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

Add the code in green and save the file.

root@intro:~/chef-rep/cookbooks/phpapp# cd ../..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 7 resources updated

We've got a database! Let's create the database user.

root@intro:~/chef-repo# cd cookbooks/phpapp

Open recipes/default.rb.

#

# Cookbook Name:: phpapp

# Recipe:: default

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

include\_recipe "mysql::client"

include\_recipe "mysql::server"

include\_recipe "php"

include\_recipe "php::module\_mysql"

include\_recipe "apache2::mod\_php5"

include\_recipe "mysql::ruby"

include\_recipe "mysql2\_chef\_gem"

apache\_site "default" do

enable true

end

mysql\_database node['phpapp']['database'] do

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

action :create

end

mysql\_database\_user node['phpapp']['db\_username'] do

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

password node['phpapp']['db\_password']

database\_name node['phpapp']['database']

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

action :grant

end

Add the code in green and save the file. We'll create a default value for our web application's mysql username but not for the password. That will need to be specified in web.json. Open attributes/default.rb

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

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

Add the code in green and save the file.

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

Open web.json.

{

"mysql": {"server\_root\_password": "808052769e2c6d909027a2905b224bad", "server\_debian\_password": "569d1ed2d46870cc020fa87be83af98d", "server\_repl\_password": "476911180ee92a2ee5a471f33340f6f4"},

"phpapp": {"db\_password": "212b09752d173876a84d374333ae1ffe"},

"run\_list": [ "recipe[apt]", "recipe[phpapp]" ]

}

Add the code in green and save the file. Let's check our recipe still works.

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 2 resources updated

Now we've setup our database we need to fetch a fresh copy of WordPress.

**Fetching WordPress**

We want to setup each new WordPress build with the latest version so we'll fetch it from WordPress.org. But how? Fortunately enough, Chef comes with the resource [remote\_file](http://docs.opscode.com/resource_remote_file.html) which will do just that for us.

root@intro:~/chef-repo# cd cookbooks/phpapp

Open up recipes/default.rb and add the following to the end of the recipe.

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 node["phpapp"]["path"] do

owner "root"

group "root"

mode "0755"

action :create

recursive true

end

execute "untar-wordpress" do

cwd node['phpapp']['path']

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

creates node['phpapp']['path'] + "/wp-settings.php"

end

You can see we've also used another new resource. The [execute](http://docs.opscode.com/resource_execute.html) resource will run a shell command for us. Here we're asking it to untar the file we've downloaded from wordpress.org. We're ensuring our recipe is idempotent by telling **execute** that the command it's to run creates the file wp-settings.php. If it finds that file it will not run the command specified.

Save the file.

We need to tell our recipe where to put the WordPress code so we'll add a default attribute. Open attributes/default.rb.

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

Add the line in green to the end of the list of attributes and save the file. Now we'll see if WordPress is downloaded.

root@intro:~/chef-rep/cookbooks/phpapp# cd ../..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 9 resources updated

Let's look at the output from chef-solo to ensure our changes have had the desired effect.

\* remote\_file[/root/chef-solo/wordpress-latest.tar.gz] action create

- copy file downloaded from [] into /root/chef-solo/wordpress-latest.tar.gz

Binary files /tmp/chef-tempfile20130316-22001-svs56r and /tmp/chef-rest20130316-22001-9ohdk differ

- change mode from '' to '0644'

\* directory[/var/www/phpapp] action create

- create new directory /var/www/phpapp

- change mode from '' to '0755'

- change owner from '' to 'root'

- change group from '' to 'root'

\* execute[untar-wordpress] action run

- execute tar --strip-components 1 -xzf /root/chef-solo/wordpress-latest.tar.gz

Let's see if the files are there.

root@intro:~/chef-repo# ls /var/www/phpapp

index.php wp-admin wp-content wp-load.php wp-signup.php

license.txt wp-blog-header.php wp-cron.php wp-login.php wp-trackback.php

readme.html wp-comments-post.php wp-includes wp-mail.php xmlrpc.php

wp-activate.php wp-config-sample.php wp-links-opml.php wp-settings.php

Good so we can see that works. But is our recipe idempotent? Let's re-run chef-solo and see.

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 6 resources updated

Let's check the output of chef-solo.

\* remote\_file[/root/chef-solo/wordpress-latest.tar.gz] action create (up to date)

\* directory[/var/www/phpapp] action create (up to date)

\* execute[untar-wordpress] action run (up to date)

No actions were performed by the code we've added (there is nothing in green), so our recipe is idempotent! Now we need to configure WordPress.

**Templates**

WordPress keeps its configuration in a file called wp-config.php. We need to create that file and put database names and user access details inside it. Chef provides a resource called [template](http://docs.opscode.com/resource_template.html) that can do just that.

root@intro:~/chef-repo# cd cookbooks/phpapp

WordPress wp-config.php files should have some random salt strings inside them for security. We can use a service provided by WordPress to generate a file containing them.

Open recipes/default.rb and add the code in green to the end.

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

if File.exist?(wp\_secrets)

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

else

require 'open-uri'

salt\_data = open('https://api.wordpress.org/secret-key/1.1/salt/').read

open(wp\_secrets, 'wb') do |file|

file << salt\_data

end

end

template node['phpapp']['path'] + '/wp-config.php' do

source 'wp-config.php.erb'

mode 0755

owner 'root'

group 'root'

variables(

:database => node['phpapp']['database'],

:user => node['phpapp']['db\_username'],

:password => node['phpapp']['db\_password'],

:wp\_secrets => salt\_data)

end

That will fetch us some unique salt strings once. Ideal for our purposes.

Save the file.

WordPress comes with an example configuration file which we'd usually alter to create our template but for brevity we'll just specify a cut down version below. Create templates/default/wp-config.php.erb.

<?php

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

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

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

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

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');

?>

Variables that should be inserted into the template are done so with **<%= @database %>**. You can also include attributes inside templates using **<%= node['phpapp']['database'] %>** although that can prevent your template from being easily reused elsewhere so is not considered best practice.

Save the file.

Let's run chef-solo again and check our recipe.

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 8 resources updated

**Creating an Apache VirtualHost**

We need to define a template for the Apache VirtualHost that will run WordPress. We'll create that now.

root@intro:~/chef-repo# cd cookbooks/phpapp

Create templates/default/site.conf.erb

# Auto generated by Chef. Changes will be overwritten.

<VirtualHost \*:80>

ServerName <%= @params[:server\_name] %>

DocumentRoot <%= @params[:docroot] %>

<Directory <%= @params[:docroot] %>>

Options FollowSymLinks

AllowOverride FileInfo Options

AllowOverride All

Order allow,deny

Allow from all

</Directory>

<Directory />

Options FollowSymLinks

AllowOverride None

</Directory>

</VirtualHost>

Add the code in green and save the file.

We'll now use a new resource which is provided by the **apache2** cookbook called **web\_app** to create an Apache VirtualHost using our template site.conf.erb.

Open recipes/default.rb.

#

# Cookbook Name:: phpapp

# Recipe:: default

#

# Copyright 2013, YOUR\_COMPANY\_NAME

#

# All rights reserved - Do Not Redistribute

#

include\_recipe "apache2"

include\_recipe "mysql::client"

include\_recipe "mysql::server"

include\_recipe "php"

include\_recipe "php::module\_mysql"

include\_recipe "apache2::mod\_php5"

include\_recipe "mysql::ruby"

include\_recipe "mysql2\_chef\_gem"

apache\_site "default" do

enable false

end

mysql\_database node['phpapp']['database'] do

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

action :create

end

mysql\_database\_user node['phpapp']['db\_username'] do

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

password node['phpapp']['db\_password']

database\_name node['phpapp']['database']

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

action :grant

end

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 node["phpapp"]["path"] do

owner "root"

group "root"

mode "0755"

action :create

recursive true

end

execute "untar-wordpress" do

cwd node['phpapp']['path']

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

creates node['phpapp']['path'] + "/wp-settings.php"

end

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

if File.exist?(wp\_secrets)

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

else

require 'open-uri'

salt\_data = open('https://api.wordpress.org/secret-key/1.1/salt/').read

open(wp\_secrets, 'wb') do |file|

file << salt\_data

end

end

template node['phpapp']['path'] + '/wp-config.php' do

source 'wp-config.php.erb'

mode 0755

owner 'root'

group 'root'

variables(

:database => node['phpapp']['database'],

:user => node['phpapp']['db\_username'],

:password => node['phpapp']['db\_username'],

:wp\_secrets => salt\_data

)

end

web\_app 'phpapp' do

template 'site.conf.erb'

docroot node['phpapp']['path']

server\_name node['phpapp']['server\_name']

end

Disable the default apache site which is highlighted in blue just after the include\_recipe lines and then, of course, add the code in green. Your final recipe should be as above.

Save the file. You may have noticed we have defined a new attribute **node['phpapp']['server\_name']**. Let's create a default for that attribute.

Open attributes/default.rb.

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

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

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

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

Add the new attribute and save the file.

root@intro:~/chef-repo/cookbooks/phpapp# cd ../..

**Overriding a default attribute**

If your server has a hostname setup in DNS we should override our default attribute and specify the actual name in web.json. If there is no proper hostname defined ignore this step.

Open web.json.

{

"mysql": {"server\_root\_password": "808052769e2c6d909027a2905b224bad", "server\_debian\_password": "569d1ed2d46870cc020fa87be83af98d", "server\_repl\_password": "476911180ee92a2ee5a471f33340f6f4"},

"phpapp": {"db\_password": "212b09752d173876a84d374333ae1ffe", "server\_name": "intro.hellofutu.re"},

"run\_list": [ "recipe[apt]", "recipe[phpapp]" ]

}

Add the code in green. Editing **server\_name** to something you've setup for the box. Save the file.

Finally, let's run chef-solo.

**A working WordPress installation!**

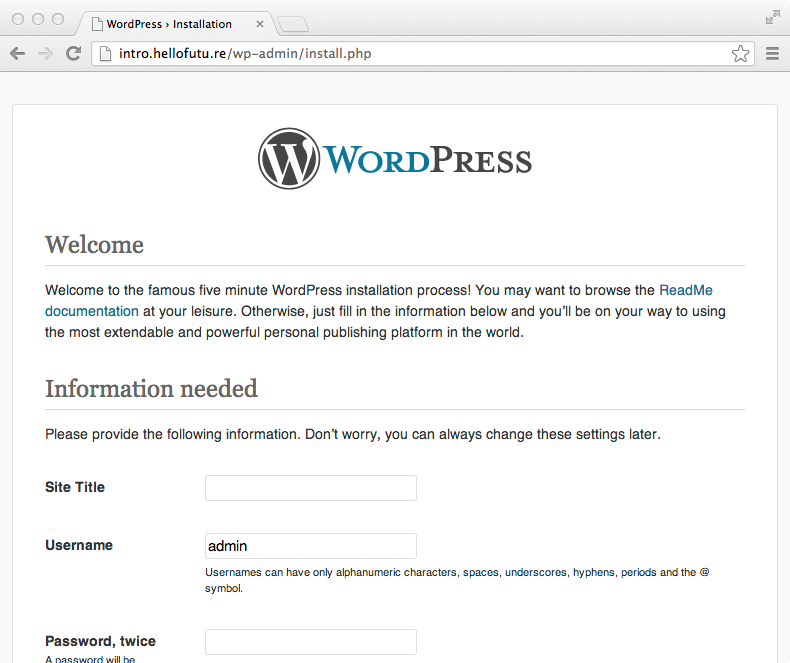
root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 11 resources updated

Let's visit our web server and see if that's worked.



Excellent it's all ready to be setup! Congratulations on your first working Chef setup! So what do we do now? We're going to destroy it! That's right! The only way to know our cookbook works properly is to test it again from scratch. We'll re-image the server and run it all again!

Unless you're particullarly attached to it, there's no need to backup your Chef Repository because we provide a URL to download it all from.

**A new server**

Now use your cloud control panel or your virtual machine software to create a new Ubuntu box. You can delete the old one.

SSH into your box. We now only need to go through a few steps to create a server.

1. Install Chef
2. Copy our Chef Repository over
3. Run chef-solo.

root@intro:~# curl -L https://www.opscode.com/chef/install.sh | bash

root@intro:~# wget http://gettingstartedwithchef.com/downloads/cp1.tgz

root@intro:~# tar zxf cp1.tgz

root@intro:~# cd chef-repo

Add this point edit web.json and enter your server name again if required. Now run chef-solo.

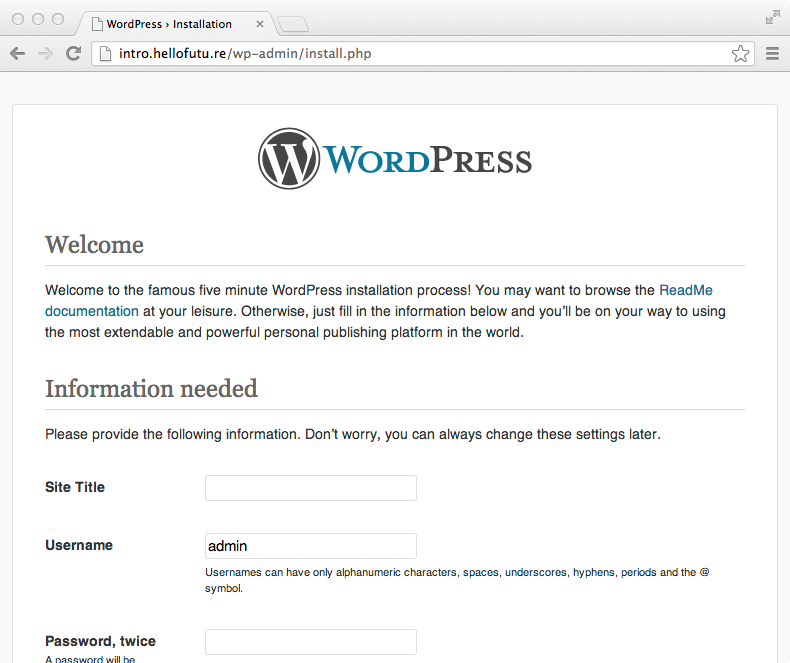
root@intro:~/chef-repo# chef-solo -c solo.rb -j web.json

Starting Chef Client, version 11.4.0

...

Chef Client finished, 57 resources updated

Exciting, let's check to see if we can access our WordPress site!



So that's worked. We can now setup a WordPress install with just a few commands! In the next chapter we'll show you how to provision new cloud servers with a single command and introduce better ways of working with Chef.

* [← Introduction](http://gettingstartedwithchef.com/index.html)
* [Introducing Chef Server →](http://gettingstartedwithchef.com/introducing-chef-server.html)

[Sponsored by Brightbox](http://brightbox.com/)

© [Hello Future Ltd](http://hellofutu.re). Registered in England and Wales. Company No: 08378526  
Created with [Twitter Bootstrap](http://twitter.github.com/bootstrap/) and [Bootswatch Cerulean.](http://bootswatch.com/cerulean/)