**What is Chef:-**

Chef is a configuration management tool that is written in Ruby. It's capable of managing both your on premise and cloud servers with ease.

You can easily manage up to 10000 nodes using chef. Replicating the infrastructure components is easy once we have them automated via chef.

Chef works with three core components: The Chef server, workstations, and nodes. The Chef server is the hub of Chef Operations, where changes are stored for use. Workstations are static computers or virtual servers where all code is created or changed. There can been as many workstations as needed, whether this be one per person or otherwise. Finally, nodes are the servers that need to be managed by Chef – these are the machines that changes are being pushed to, generally a fleet of multiple machines that require the benefits of an automation program.

**The Chef Server:-**

The Chef server is the primary mode of communication between the workstations where your infrastructure is coded, and the nodes where it is deployed. All configuration files, cookbooks, metadata, and other information are stored on the server. The Chef server also keeps information regarding the state of all nodes at the time of the last [chef-client](https://www.linode.com/docs/applications/chef/beginners-guide-chef#chef-client) run.

Any changes made must pass through the Chef server to be deployed. Prior to accepting or pushing changes, it verifies that the nodes and workstations are paired with the server through the use of authorization keys, and then allows for communication between the workstations and nodes.

## Workstations:-

Workstations are where users create, test, and maintain cookbooks and policies that will be pushed to nodes. Cookbooks created on workstations can be used privately by one organization, or uploaded to the Chef Supermarket for others to use. Similarly, workstations can be used to download cookbooks created by other Chef users and found in the Supermarket.

Workstations are set up to use the *Chef Development Kit* (ChefDK), and can be located on virtual servers or on physical workstation computers. Workstations are set to interact with only one Chef server, and most work will be done in the chef-repo directory located on the workstation.

### Chef-repo:-

The chef-repo directory is the specific area of the workstation where cookbooks are authored and maintained. The chef-repo is always version-controlled, most often through the use of Git, and stores information and history that will be used on nodes, such as cookbooks, environments, roles, and data bags. Chef is able to communicate with the server from the chef-repo and push any changes via the use of the knife command, which is included in the ChefSDK.

Originally the chef-repo had to be pulled from GitHub using git commands, but that action is now integrated into Chef through the use of the chef generate repo chef-repo command.

### Knife:-

The knife command communicates between the chef-repo located on a workstation and the Chef server. knife is configured with the knife.rb file, and is used from the workstation:

~/chef-repo/.chef/knife.rb

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | log\_level :info  log\_location STDOUT  node\_name 'username'  client\_key '~/chef-repo/.chef/username.pem'  validation\_client\_name 'shortname-validator'  validation\_key '~/chef-repo/.chef/shortname.pem'  chef\_server\_url 'https://123.45.67.89/organizations/shortname'  syntax\_check\_cache\_path '~/chef-repo/.chef/syntax\_check\_cache'  cookbook\_path [ '~/chef-repo/cookbooks' ] |

## Nodes:-

A *node* is a system configured to run the chef-client. This can be any system, as long as it is being maintained by Chef.

Nodes are validated through the validator.pem and client.pem certificates that are created on the node when it is bootstrapped. All nodes must be bootstrapped over SSH as either the root user or a user with elevated privileges.

Nodes are kept up-to-date through the use of the chef-client, which runs a convergence between the node and the Chef server. What cookbooks and roles the node will take on depends on the run list and environment set for the node in question.

### chef-client

The chef-client checks the current configuration of the node against the recipes and policies stored in the Chef server and bring the node up to match. The process begins with the chef-client checking the node’s [run list](https://www.linode.com/docs/applications/chef/beginners-guide-chef#run-lists), loading the cookbooks required, then checking and syncing the cookbooks with the current configuration of the node.

The chef-client must be run with elevated privileges in order to properly configure the node, and should be run periodically to ensure that the server is always up to date – often this is achieved through a cron job or by setting up the chef-client to run as a service.

### Run Lists

Run lists define what cookbooks a node will use. The run list is an ordered list of all cookbooks and recipes that the chef-client needs to pull from the Chef server to run on a node. Run lists are also used to define [roles](http://docs.chef.io/server_manage_roles.html), which are used to define patterns and attributes across nodes.

### Ohai

Ohai collects information regarding nodes for the Chef server. It is required to be present on every node, and is installed as part of the bootstrap process.

The information gathered includes network and memory usage, CPU data, kernel data, hostnames, FQDNs, and other automatic attributes that need to remain unchanged during the chef-client run.

## Cookbooks

Cookbooks are the main component of configuring nodes on a Chef infrastructure. Cookbooks contain values and information about the *desired state* of a node, not how to get to that desired state – Chef does all the work for that, through their extensive libraries.

Cookbooks are comprised of recipes, metadata, attributes, resources, templates, libraries, and anything else that assists in creating a functioning system, with attributes and recipes being the two core parts of creating a cookbook. Components of a cookbook should be modular, keeping recipes small and related.

Cookbooks can and should be version controlled. Versions can help when using environments and allow for the easier tracking of changes that have been made to the cookbook.

### Recipes

Recipes are the fundamental part of cookbooks. Recipes are written in Ruby and contain information in regards to everything that needs to be run, changed, or created on a node. Recipes work as a collection of *resources* that determine the configuration or policy of a node, with resources being a configuration element of the recipe. For a node to run a recipe, it must be on that node’s run list.

### Attributes

Attributes define specific values about a node and its configuration. These values are used to override default settings, and are loaded in the order cookbooks are listed in the run list. Often attributes are used in conjunction with templates and recipes to define settings.

### Files

These are static files that can be uploaded to nodes. Files can be configuration and set-up files, scripts, website files – anything that does not been to have different values on different nodes.

### Providers and Resources

Providers and resources are also used to define new functionality to use in Chef recipes. A *resource* defines a set of actions and attributes, whereas *provider* informs the chef-client how to commit each action.

### Templates

Templates are embedded Ruby files (.erb) that allows for content based on the node itself and other variables generated when the chef-client is run and the template is used to create or update a file.

Also gone through below chef Reference links

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Tutorial+Introduction>

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Tutorial+1>

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Tutorial+2>

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Current+Environment+Details+at+DR>

<http://confluence.mpls.digitalriver.com/display/serv/Git+Commands+for+use+with+Chef+Development>

<https://www.linode.com/docs/applications/chef/beginners-guide-chef>

<https://www.linode.com/docs/applications/chef/deploy-a-chef-server-workstation-and-node-on-ubuntu-14-04/>

<https://www.linode.com/docs/applications/chef/creating-your-first-chef-cookbook>

**Installation of Chef:-**

Chef has three major components:

1. Workstation
2. Server
3. Nodes

Following are the steps to install Chef:

* 1. Install Chef DK (Development Kit) on Chef Workstation
  2. Setup a Chef Server
  3. Create a Recipe or a Cookbook / download a Cookbook from Chef Supermarket in Workstation
  4. Upload a Cookbook on the Chef Server
  5. Connect A Node To The Chef Server
  6. Deploy the Cookbook from the Server to the Node

## ****Install Chef DK (Development Kit):-****

## In my Chef Workstation I will install Chef DK. Chef DK is a package that contains all the development tools that you will need when coding Chef.

## Download the chef development kit.

## #wget <https://packages.chef.io/stable/el/6/chefdk-1.0.3-1.el6.x86_64.rpm>

## The package is now downloaded. Now install this package using rpm.

## #rpm -ivh chefdk-1.0.3-1.el6.x86\_64.rpm

## ****Setup Chef Server:-****

## I will use the hosted version of Chef Server on the cloud but you can use a physical machine as well. This Chef-Server is present at [manage.chef.io](http://manage.chef.io/)

## Over here, create an account if you don’t have one. Once you have created an account, sign-in with your login credentials.

## If you are signing in for the first time, the very first thing that you will be doing is creating an organization. Organization is basically a group of Machines that you will be managing with the Chef Server.

## First, I will go to the administration tab. Over there, I have already created an organization called edu. So I need to download the starter kit in my Workstation. This starter kit will help you to push files from the Workstation to the Chef Server. Click on the settings icon on the right hand side and click on Starter Kit.

## Now you will get an option to download the Starter Kit. Just click on it to download the Starter Kit zip file.

## Move this file to your root directory. Now unzip this file by using unzip command in your terminal. You will notice that it includes a directory called chef-repo.

## #unzip chef-starter.zip

## Now move this starter kit to the cookbook directory in chef-repo directory.

## Execute this:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **#mv starter /root/chef-repo/cookbooks** Download A Cookbook From Chef Supermarket In Workstation:- Chef Cookbooks are available in the Cookbook Supermarket, we can go to the Chef Supermarket. Download the required Cookbooks from[supermarket.chef.io](http://supermarket.chef.io/). I’m downloading one of the Cookbook to install Apache from there.  Execute this:   |  |  | | --- | --- | |  | **#cd chef-repo**  **#knife cookbook site download learn\_chef\_httpd** |  There is Tar ball downloaded for the Apache Cookbook. Now, I will extract the contents from this downloaded Tar file. For that, I will use tar command. Execute this:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | **#tar -xvf learn\_chef\_httpd-0.2.0.tar.gz**  All the required files are automatically created under this Cookbook. There is no need to make any modifications. Let’s check the Recipe description inside my recipe folder.  Now, I will just upload this cookbook to my Chef Server as it looks perfect to me  Upload A Cookbook In The Chef Server  **#cd /root/chef-repo/learn\_chef\_httpd/recipes**  **#cat default.rb**  In order to upload the Apache Cookbook that I have downloaded, first move this learn\_chef\_httpd file to the Cookbooks folder in the chef-repo. Then change your directory to cookbooks.  Execute this:   |  |  | | --- | --- | |  | **#mv /root/chef-repo/learn\_chef\_httpd /root/chef-repo/cookbooks**  **#cd /root/chef-repo/cookbooks** |   Now in this directory, execute the below command to upload the Apache Cookbook:  Execute this:   |  |  | | --- | --- | |  | **#knife cookbook upload learn\_chef\_httpd** |   Now, our final step is to add Chef Node. We’ve setup a Workstation, a Chef Server and we need to add our Nodes to the Chef Server for automation.  **5. Connect A Node To The Chef Server:-**  The terminal color of my Node machine is different from the Workstation so that you will be able to differentiate between both.  I just need the IP address of my Node for that I will execute the below command in my Node machine.  Execute this:   |  |  | | --- | --- | |  | **#ifconfig** |   I will add my Chef Node to the Server by executing Knife Bootstrap command in which I will specify the IP address of The Chef Node and its name. Execute the command shown below:  **Execute this:**  **#knife bootstrap 192.168.56.102 --ssh-user root --ssh-password edureka --node-name chefNode**  This command will also initialize the installation of the Chef-Client in the Chef Node. You can verify it from the CLI on the Workstation using the knife command,as shown below:  Execute this:   |  |  | | --- | --- | |  | **#Knife node list** |  6. Deploy The Cookbook From The Server To The Node Let’s see how we can add a Cookbook to the Node and manage its Run list from the Chef Server. Run List describes the order in which the Cookbooks should be executed. As you can see in the screenshot below, click the Actions tab and select the Edit Run list option to manage the Run list.  In the Available Recipes,  you can see our learn\_chef\_httpd Recipe, you can drag that from the available packages to the current Run List and save the Run list.  Now login to your Node and just run chef-client to execute the Run List.  **#chef-client**  After successfully run the chef client cookbooks and recipes have applied to node. | |

## 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/" \t "_blank), [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" \t "_blank) 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" \t "_blank).

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

### 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" \t "_blank) 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.