How to Create your First Chef Cookbook

Updated Wednesday, October 7, 2020

automation

Traducciones al Español Chef cookbooks describe the *desired state* of your nodes, and allow Chef to push out the changes needed to achieve this state. In this guide you will learn how to create a cookbook that configures A LAMP stack on a Linode.

Before You Begin

1. Set up Chef with the Setting Up a Chef Server, Workstation, and Node guide. When following that guide, **choose Ubuntu 16.04 as your Linux image for the Chef node you will bootstrap and manage**. This guide will use the MySQL Chef cookbook, which does not yet support Ubuntu 18.04.
2. Once your node is bootstrapped, you can use a Chef cookbook to secure your node. Consider using the Users cookbook and the Firewall cookbook for this work. While this is not required to complete this guide, it is recommended.
3. You can also review A Beginner's Guide to Chef to receive an overview on Chef concepts.
4. The examples in this tutorial require a user account with sudo privileges. Readers who use a limited user account will need to prefix commands with sudo when issuing commands to the Chef client node and replace `-x root` with `-x username` where `username` is your limited user account.
5. Ensure that your workstation's `/etc/hosts` file contains its own IP address, the Chef server's IP address and fully qualified domain name, and the IP address and hostname for any nodes you will interact with from the workstation. For example:

File: /etc/hosts

```
127.0.0.1       localhost
192.0.2.0       workstation
192.0.1.0       www.example.com
198.51.100.0    node-hostname

```

Create the Cookbook

1. From your workstation, move to your `chef-repo/cookbooks` directory:

```
cd chef-repo/cookbooks
```

3. Create the cookbook. In this instance the cookbook is titled `lamp_stack`:

```
chef generate cookbook lamp_stack
```

5. Move to your cookbook's newly-created directory:

```
cd lamp_stack
```

If you issue the `ls` command, you should see the following files and directories:

```
Berksfile  CHANGELOG.md  chefignore  LICENSE  metadata.rb
README.md  recipes  spec  test
```

default.rb

Attributes are pieces of data that help the chef-client determine the current state of a node and any changes that have taken place on the node from one chef-client run to another. Attributes are gathered from the state of the node, cookbooks, roles and environments. Using these sources, an attribute list is created for each chef-client run and is applied to the node. If a `default.rb` file exists within a cookbook, it will be loaded first, but has the lowest attribute precedence.

The `default.rb` file in `recipes` contains the "default" recipe resources.

In this example, the `lamp_stack` cookbook's `default.rb` file is used to update the node's distribution software.

1. From within the `lamp_stack` directory, navigate to the `recipes` folder:

```
cd recipes
```

3. Open the `default.rb` file and add the following code:

File: ~/chef-repo/cookbooks/lamp_stack/recipe/default.rb

```
#
# Cookbook Name:: lamp_stack
# Recipe:: default
#
#

execute "update-upgrade" do
  command "sudo apt-get update && sudo
DEBIAN_FRONTEND=noninteractive apt-get -y -o
```

```
10 Dpkg::Options::='--force-confdef' -o Dpkg::Options::='--
   force-confold' upgrade"
     action :run
   end
```

Recipes are comprised of a series of *resources*. In this case, the *execute* resource is used, which calls for a command to be executed once. The `apt-get update && apt-get upgrade -y` commands are defined in the `command` section, and the `action` is set to `:run` the commands.

The extra variables and flags passed to the `upgrade` command are there to suppress the GRUB configuration menu, which can cause Chef to hang waiting for user input.

This is one of the simpler Chef recipes to write, and a good way to start out. Any other startup procedures that you deem important can be added to the file by mimicking the above code pattern.

4. To test the recipe, add the LAMP stack cookbook to the Chef server:

```
5. knife cookbook upload lamp_stack
```

6. Verify that the recipe has been added to the Chef server:

```
7. knife cookbook list
```

You should see a similar output:

```
Uploading lamp_stack   [0.1.0]
Uploaded 1 cookbook.
```

8. Add the recipe to your chosen node's *run list*, replacing `nodename` with your node's name:

```
9. knife node run_list add nodename "recipe[lamp_stack]"
```

10. From your workstation, apply the configurations defined in the cookbook by running the chef-client on your node. Replace `nodename` with the name of your node:

```
11.       knife ssh 'name:nodename' 'sudo chef-client' -x root
```

Your output should display a successful Chef run. If not, review your code for any errors, usually defined in the output of the `chef-client` run.

Apache

Install and Enable

1. In your Chef workstation, Create a new file under the `~/chef-repo/cookbooks/lamp_stack/recipes` directory called `apache.rb`. This will contain all of your Apache configuration information.
2. Open the file, and define the *package* resource to install Apache:

   File: ~/chef-repo/cookbooks/lamp_stack/apache.rb

   ```
   package "apache2" do
     action :install
   end
   ```

   Again, this is a very basic recipe. The *package* resource calls to a package (`apache2`). This value must be a legitimate package name. The action is *install* because Apache is being installed in this step. There is no need for additional values to run the install.

3. Set Apache to enable and start at reboot. In the same file, add the additional lines of code:

   File: ~/chef-repo/cookbooks/lamp_stack/apache.rb

   ```
   service "apache2" do
     action [:enable, :start]
   end
   ```

   This uses the *service* resource, which calls on the Apache service. The *enable* action enables it upon startup, and *start* starts Apache.

   Save and close the `apache.rb` file.

4. To test the Apache recipe, update the LAMP Stack recipe on the server:

```
knife cookbook upload lamp_stack
```

6. Add the recipe to a node's run-list, replacing `nodename` with your chosen node's name:

```
knife node run_list add nodename "recipe[lamp_stack::apache]"
```

   Because this is not the `default.rb` recipe, the recipe name, *apache*, must be appended to the recipe value.

   Note

   To view a list of all nodes managed by your Chef server, issue the following command from your workstation:

   ```
   knife node list
   ```

8. From your workstation, apply the configurations defined in the cookbook by running the chef-client on your node. Replace `nodename` with the name of your node:

```
knife ssh 'name:nodename' 'sudo chef-client' -x root
```

   If the recipe fails due to a syntax error, Chef will note it during the output.

10. After a successful `chef-client` run, check to see if Apache is running:

```
knife ssh 'name:nodename' 'systemctl status apache2' -x root
```

> **Note**
>
> Repeat steps 4-7 to upload each recipe to your Chef server as you create it. Run `chef-client` on your node as needed throughout the rest of this guide to ensure your recipes are working properly and contain no errors. When adding a new recipe, ensure you are using its correct name in the run list.
>
> This is not the recommended workflow for a production environment. You might consider creating different Chef environments for testing, staging, and production.

Configure Virtual Hosts

This configuration is based off of the How to Install a LAMP Stack on Ubuntu 16.04 guide.

1. Because multiple websites may need to be configured, use Chef's attributes feature to define certain aspects of the virtual host file(s). The ChefDK has a built-in command to generate the attributes directory and `default.rb` file within a cookbook. Replace `~/chef-repo/cookbooks/lamp_stack` with your cookbook's path:

```
chef generate attribute ~/chef-repo/cookbooks/lamp_stack default
```

3. Within the new `default.rb`, create the default values for the cookbook:

   File: ~/chef-repo/cookbooks/lamp_stack/attributes/default.rb

```
default["lamp_stack"]["sites"]["example.com"] = { "port" => 80, "servername" => "example.com", "serveradmin" => "webmaster@example.com" }
```

The prefix `default` defines that these are the normal values to be used in the `lamp_stack` where the site `example.com` will be called upon. This can be seen as a hierarchy: Under the cookbook itself are the site(s), which are then defined by their URL.

The following values in the array (defined by curly brackets ({})) are the values that will be used to configure the virtual host file. Apache will be set to listen on port `80` and use the listed values for its server name, and administrator email.

Should you have more than one available website or URL (for example, `example.org`), this syntax should be mimicked for the second URL:

File: ~/chef-repo/cookbooks/lamp_stack/attributes/default.rb

```
default["lamp_stack"]["sites"]["example.com"] = { "port" =>
80, "servername" => "example.com", "serveradmin" =>
 "webmaster@example.com" }
 default["lamp_stack"]["sites"]["example.org"] = { "port" =>
 80, "servername" => "example.org", "serveradmin" =>
 "webmaster@example.org" }
```

4. Return to your `apache.rb` file under `recipes` to call the attributes that were just defined. Do this with the node resource:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
#Install & enable Apache

package "apache2" do
  action :install
end

service "apache2" do
  action [:enable, :start]
end


# Virtual Host Files

node["lamp_stack"]["sites"].each do |sitename, data|
end
```

This calls in the values under `["lamp_stack"]["sites"]`. Code added to this block will be generated for each value, which is defined by the word `sitename`. The `data` value calls the values that are listed in the array of each `sitename` attribute.

5. Within the node resource, define a document root. This root will be used to define the public HTML files, and any log files that will be generated:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
node["lamp_stack"]["sites"].each do |sitename, data|
  document_root = "/var/www/html/#{sitename}"
end
```

6. Create the `document_root` directory. Declare a `directory` resource with a `true` recursive value so all directories leading up to the `sitename` will be created. A permissions value of `0755` allows for the file owner to have full access to the directory, while group and regular users will have read and execute privileges:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
node["lamp_stack"]["sites"].each do |sitename, data|
  document_root = "/var/www/html/#{sitename}"

  directory document_root do
    mode "0755"
    recursive true
  end

end
```

7. The template feature will be used to generate the needed virtual host files. Within the `chef-repo` directory run the `chef generate template` command with the path to your cookbook and template file name defined:

```
chef generate template ~/chef-repo/cookbooks/lamp_stack
virtualhosts
```

9. Open and edit the `virtualhosts.erb` file. Instead of writing in the true values for each VirtualHost parameter, use Ruby variables. Ruby variables are identified by the <%= @variable_name %> syntax. The variable names you use will need to be defined in the recipe file:

   File: ~/chef-repo/cookbooks/lamp_stack/templates/virtualhosts.erb

```
<VirtualHost *:<%= @port %>>
        ServerAdmin <%= @serveradmin %>
        ServerName <%= @servername %>
        ServerAlias www.<%= @servername %>
        DocumentRoot <%= @document_root %>/public_html
        ErrorLog <%= @document_root %>/logs/error.log
        <Directory <%= @document_root %>/public_html>
                Require all granted
```

```
        </Directory>
</VirtualHost>
```

Some variables should look familiar. They were created in Step 2, when naming default attributes.

10. Return to the `apache.rb` recipe. In the space after the `directory` resource, use the `template` resource to call upon the template file just created:

    File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
# [...]

#Virtual Host Files

node["lamp_stack"]["sites"].each do |sitename, data|
  document_root = "/var/www/html/#{sitename}"

  directory document_root do
    mode "0755"
    recursive true
  end

  template "/etc/apache2/sites-available/#{sitename}.conf" do
    source "virtualhosts.erb"
    mode "0644"
    variables(
      :document_root => document_root,
      :port => data["port"],
      :serveradmin => data["serveradmin"],
      :servername => data["servername"]
    )
  end

end
```

    The name of the template resource should be the location where the virtual host file is placed on the nodes. The `source` is the name of the template file. Mode `0644` gives the file owner read and write privileges, and everyone else read privileges. The values defined in the `variables` section are taken from the attributes file, and they are the same values that are called upon in the template.

11. The sites need to be enabled in Apache, and the server restarted. This should *only* occur if there are changes to the virtual hosts, so the `notifies` value should be added to the

`template` resource. `notifies` tells Chef when things have changed, and **only then** runs the commands:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
template "/etc/apache2/sites-available/#{sitename}.conf" do
  source "virtualhosts.erb"
  mode "0644"
  variables(
    :document_root => document_root,
    :port => data["port"],
    :serveradmin => data["serveradmin"],
    :servername => data["servername"]
  )
  notifies :restart, "service[apache2]"
end
```

The `notifies` command names the `:action` to be committed, then the resource, and resource name in square brackets.

12. `notifies` can also call on `execute` commands, which will run `a2ensite`and enable the sites that have corresponding virtual host files. Add the following `execute` command **above** the `template` resource code to create the `a2ensite` script:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
# [...]

directory document_root do
  mode "0755"
  recursive true
end

execute "enable-sites" do
  command "a2ensite #{sitename}"
  action :nothing
end

template "/etc/apache2/sites-available/#{sitename}.conf" do

# [...]
```

The `action :nothing` directive means the resource will wait to be called on. Add a new `notifies` line above the previous `notifies` line to the `template` resource code to use it:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
# [...]

template "/etc/apache2/sites-available/#{sitename}.conf" do
  # [...]
  notifies :run, "execute[enable-sites]"
  notifies :restart, "service[apache2]"
end

# [...]
```

13. The paths referenced in the virtual host files need to be created. Once more, this is done with the `directory` resource, and should be added before the final `end` tag:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
# [...]

node["lamp_stack"]["sites"].each do |sitename, data|
  # [...]

  directory "/var/www/html/#{sitename}/public_html" do
    action :create
  end

  directory "/var/www/html/#{sitename}/logs" do
    action :create
  end
end
```

Apache Configuration

With the virtual host files configured and your website enabled, configure Apache to efficiently run on your servers. Do this by enabling and configuring a multi-processing module (MPM), and editing `apache2.conf`.

The MPMs are all located in the `mods_available` directory of Apache. In this example the `prefork` MPM will be used, located at `/etc/apache2/mods-available/mpm_prefork.conf`. If we were planning on deploying to nodes of varying size we would create a template file to replace the original, which would allow for more customization of specific variables. In this instance, a *cookbook file* will be used to edit the file.

Cookbook files are static documents that are run against the document in the same locale on your servers. If any changes are made, the cookbook file makes a backup of the original file and replaces it with the new one.

1. To create a cookbook file navigate to `files/default` from your cookbook's main directory. If the directories do not already exist, create them:

```
mkdir -p ~/chef-repo/cookbooks/lamp_stack/files/default/
cd ~/chef-repo/cookbooks/lamp_stack/files/default/
```

4. Create a file called `mpm_prefork.conf` and copy the MPM event configuration into it, changing any needed values:

File: ~/chef-repo/cookbooks/lamp_stack/files/default/mpm_prefork.conf

```
<IfModule mpm_prefork_module>
        StartServers             4
        MinSpareServers          3
        MaxSpareServers          40
        MaxRequestWorkers        200
        MaxConnectionsPerChild   10000
</IfModule>
```

5. Return to `apache.rb`, and use the `cookbook_file` resource to call the file we just created. Because the MPM will need to be enabled, we'll use the `notifies` command again, this time to execute `a2enmod mpm_event`. Add the `execute` and `cookbook_file` resources to the `apache.rb` file prior to the final `end` tag:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
# [...]

node["lamp_stack"]["sites"].each do |sitename, data|
  # [...]

  execute "enable-prefork" do
    command "a2enmod mpm_prefork"
    action :nothing
  end

  cookbook_file "/etc/apache2/mods-
available/mpm_prefork.conf" do
    source "mpm_prefork.conf"
    mode "0644"
    notifies :run, "execute[enable-prefork]"
  end
end
```

6. Within the `apache2.conf` the `KeepAlive` value should be set to `off`, which is the only change made within the file. This can be altered through templates or cookbook files, although in this instance a simple `sed` command will be used, paired with the `execute` resource. Update `apache.rb` with the new `execute` resource:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/apache.rb

```
# [...]

directory "/var/www/html/#{sitename}/logs" do
  action :create
end

execute "keepalive" do
  command "sed -i 's/KeepAlive On/KeepAlive Off/g'
/etc/apache2/apache2.conf"
  action :run
end

execute "enable-prefork" do

# [...]
```

   Your `apache.rb` is now complete. An example of the final file is located here.

MySQL

Download the MySQL Library

1. The Chef Supermarket has an OpsCode-maintained MySQL cookbook that sets up MySQL *lightweight resources/providers* (LWRPs) to be used. From the workstation, download and install the cookbook:

```
2. knife cookbook site install mysql
```

   This will also install any and all dependencies required to use the cookbook. These dependencies include the `smf` and `yum-mysql-community` cookbooks, which in turn depend on the `rbac` and `yum` cookbooks.

3. From the main directory of your LAMP stack cookbook, open the `metadata.rb` file and add a dependency to the MySQL cookbook:

   File: ~/chef-repo/cookbooks/lamp_stack/metadata.rb

```
depends          'mysql', '~> 8.6.0'
```

Note

Check the MySQL Cookbook's Supermarket page to ensure this is the latest version of the cookbook. The MySQL Cookbook does not yet support Ubuntu 18.04.

4. Upload these cookbooks to the server:

```
5. knife cookbook upload mysql --include-dependencies
```

Create and Encrypt Your MySQL Password

Chef contains a feature known as *data bags*. Data bags store information, and can be encrypted to store passwords, and other sensitive data.

1. On the workstation, generate a secret key:

```
2. openssl rand -base64 512 > ~/chef-
   repo/.chef/encrypted_data_bag_secret
```

3. Upload this key to your node's `/etc/chef` directory, either manually by `scp` from the node (an example can be found in the Setting Up Chef guide), or through the use of a recipe and cookbook file.
4. On the workstation, create a `mysql` data bag that will contain the file `rtpass.json` for the root password:

```
5. knife data bag create mysql rtpass.json --secret-file ~/chef-
   repo/.chef/encrypted_data_bag_secret
```

Note

Some knife commands require that information be edited as JSON data using a text editor. Your `config.rb` file should contain a configuration for the text editor to use for such commands. If your `config.rb` file does not already contain this configuration, add `knife[:editor] = "/usr/bin/vim"` to the bottom of the file to set vim as the default text editor.

You will be asked to edit the `rtpass.json` file:

File: ~/chef-repo/data_bags/mysql/rtpass.json

```
1 {
2   "id": "rtpass.json",
3   "password": "password123"
4 }
```

Replace `password123` with a secure password.

6. Confirm that the `rtpass.json` file was created:

```
7. knife data bag show mysql
```

It should output `rtpass.json`. To ensure that is it encrypted, run:

```
knife data bag show mysql rtpass.json
```

The output will be unreadable due to encryption, and should resemble:

```
WARNING: Encrypted data bag detected, but no secret provided
for decoding.  Displaying encrypted data.
    id:       rtpass.json
    password:
      cipher:         aes-256-cbc
      encrypted_data:
wpEAb7TGUqBmdB1TJA/5vyiAo2qaRSIF1dRAc+vkBhQ=

      iv:             E5TbF+9thH9amU3QmGxWmw==

      version:        1
    user:
      cipher:         aes-256-cbc
      encrypted_data:
VLA00Wrnh9DrZqDcytvo0HQUG0oqI6+6BkQjHXp6c0c=

      iv:             6V+3ROpW9RG+/honbf/RUw==

      version:        1
```

Set Up MySQL

With the MySQL library downloaded and an encrypted root password prepared, you can now set up the recipe to download and configure MySQL.

1. Open a new file in `recipes` called `mysql.rb` and define the data bag that will be used:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/mysql.rb

```
1 mysqlpass = data_bag_item("mysql", "rtpass.json")
```

2. Thanks to the LWRPs provided through the MySQL cookbook, the initial installation and database creation for MySQL can be done in one resource:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/mysql.rb

```
mysqlpass = data_bag_item("mysql", "rtpass.json")

mysql_service "mysqldefault" do
  version '5.7'
  initial_root_password mysqlpass["password"]
  action [:create, :start]
end
```

`mysqldefault` is the name of the MySQL service for this container. The `inital_root_password` calls to the value defined in the text above, while the action creates the database and starts the MySQL service.

3. The version of MySQL the `mysql` cookbook installation creates uses a sock file at a non-standard location, so you must declare this location in order to interact with MySQL from the command line. To do this, create a cookbook file called `my.cnf` with the following configuration:

   File: ~/chef-repo/cookbooks/lamp_stack/files/default/my.cnf

   ```
   [client]
   socket=/run/mysql-mysqldefault/mysqld.sock
   ```

4. Open `mysql.rb` again, and add the following lines to the end of the file:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/mysql.rb

   ```
   # [...]

   cookbook_file "/etc/my.cnf" do
     source "my.cnf"
     mode "0644"
   end
   ```

PHP

1. Under the recipes directory, create a new `php.rb` file. The commands below install PHP and all the required packages for working with Apache and MySQL:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/php.rb

   ```
   package "php" do
     action :install
   end

   ```

```
 5  package "php-pear" do
 6    action :install
 7  end
 8
 9  package "php-mysql" do
10    action :install
11  end
12
13  package "libapache2-mod-php" do
14    action :install
15  end
```

2. For easy configuration, the `php.ini` file will be created and used as a cookbook file, much like the MPM module above. You can either:
   - Add the PHP recipe, run `chef-client` and copy the file from a node (located in `/etc/php/7.0/cli/php.ini`), or:
   - Copy it from this chef-php.ini sample. The file should be moved to the `chef-repo/cookbooks/lamp_stack/files/default/` directory. This can also be turned into a template, if that better suits your configuration.
3. `php.ini` is a large file. Search and edit the following values to best suit your Linodes. The values suggested below are for 2GB Linodes:

   File: ~/chef-repo/cookbooks/lamp_stack/files/default/php.ini

```
max_execution_time = 30
memory_limit = 128M
error_reporting =
E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR
display_errors = Off
log_errors = On
error_log = /var/log/php/error.log
max_input_time = 30
```

4. Return to `php.rb` and append the `cookbook_file` resource to the end of the recipe:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/php.rb

```
cookbook_file "/etc/php/7.0/cli/php.ini" do
  source "php.ini"
  mode "0644"
  notifies :restart, "service[apache2]"
end
```

5. Because of the changes made to `php.ini`, a `/var/log/php` directory needs to be made and its ownership set to the Apache user. This is done through a `notifies` command and *execute* resource, as done previously. Append these resources to the end of `php.rb`:

   File: ~/chef-repo/cookbooks/lamp_stack/recipes/php.rb

```
execute "chownlog" do
  command "chown www-data /var/log/php"
  action :nothing
end

directory "/var/log/php" do
  action :create
  notifies :run, "execute[chownlog]"
end
```

   The PHP recipe is now done! View an example of the php.rb file here.

6. Ensure that your Chef server contains the updated cookbook, and that your node's run list is up-to-date. Replace nodename with your Chef node's name:

```
knife cookbook upload lamp_stack
knife node run_list add nodename
"recipe[lamp_stack],recipe[lamp_stack::apache],recipe[lamp_stac
k::mysql],recipe[lamp_stack::php]"
```

Testing Your Installation

1. To ensure that the Apache service has been successfully installed and running, you can execute the following command, substituting node_name for the name of your node:

```
knife ssh 'name:node_name' 'systemctl status apache2' -x root
```

   Additionally, you can visit your server's domain name in your browser. If it is working, you should see a Chef server page that will instruct you on how to set up the Management Console (as you have not uploaded any files to your server yet.)

3. To check on the status of PHP, you'll need to upload a file to your server to make sure it's being rendered correctly. A simple PHP file that you can create is a PHP info file. Create a file called `info.php` in the same directory as the other cookbook files you've created:

   File: ~/chef-repo/cookbooks/lamp_stack/files/default/info.php

```
<?php phpinfo(); ?>
```

Modify your `php.rb` file and add the following to the end of the file, replacing `example.com` your website's domain name:

File: ~/chef-repo/cookbooks/lamp_stack/recipes/php.rb

```
cookbook_file "/var/www/html/example.com/public_html/info.php"
do
  source "info.php"
end
```

Upload your cookbook to your Chef server, and then run `chef-client` on your node, replacing node_name with the name of your node:

```
knife cookbook upload lamp_stack
knife ssh 'name:node_name' 'sudo chef-client' -x root
```

Visit `example.com/info.php` in your browser. You should see a page that houses information about your PHP settings.

4. To test your MySQL installation, you can check on the status of MySQL using `systmectl`. Issue the following command to ensure the MySQL service is running correctly:

```
5. knife ssh 'name:node_name' 'systemctl status mysql-
   mysqldefault' -u root
```

`chef-client` is not designed to accept user input, and as such using commands like `mysqladmin status` that require a password can cause Chef to hang. If you need to be able to interact with MySQL client directly, consider logging in to your server directly.

You have just created a LAMP Stack cookbook. Through this guide, you should have learned to use the execute, package, service, node, directory, template, cookbook_file, and mysql_service resources within a recipe, as well as download and use LWRPs, create encrypted data bags, upload/update your cookbooks to the server, and use attributes, templates, and cookbook files. This gives you a strong basis in Chef and cookbook creation for future projects.