

Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Maestría en Desarrollo y Operaciones

Herramientas de Automatización de Despliegues

**Actividad grupal: Instalación completa de
WordPress con Chef y pruebas unitarias y de
integración**

Trabajo de innovación presentado por	Nombre de los Alumnos
Fecha: 05/08/2023	- Carlos Eduardo Colón Rangel - Javier Gutiérrez Fausto - Elena María Elyoenia García Mejía

Contenido

Introducción.	2
Desarrollo.	3
Pruebas unitarias.	10
Pruebas de integración.	12
Conclusión.	12
Referencias.	13

Introducción.

El objetivo de esta actividad es practicar el conocimiento sobre la automatización de pruebas unitarias y de integración con Chef realizando la instalación de un servidor WordPress, incluyendo las herramientas necesarias, como lo son Apache, PHP y MySQL, utilizando como servidor destino una máquina virtual gestionada con la herramienta Vagrant, configurada para provisionar mediante Chef.

Además, de considerar casos de pruebas unitarias, de integración y de infraestructura.

El proyecto se compone de tres servicios, cada uno desplegado en una máquina virtual

1. Máquina virtual de base de datos: Se instala MySQL, que recibe peticiones por el puerto 3306.
2. Máquina virtual de Wordpress: Se instala el servidor Apache y la aplicación Wordpress es instalada para ser servida por el servidor web, que recibe peticiones por el puerto 8080, misma que se configura instalando WP-CLI para la configuración del sitio y la muestra del blog.
3. Máquina virtual proxy: Se instala un proxy Nginx, este será punto de entrada a la aplicación, recibe peticiones por el puerto 80.

A continuación, se muestra la representación virtual de la infraestructura que se consideró, así como los puertos que utilizaron.

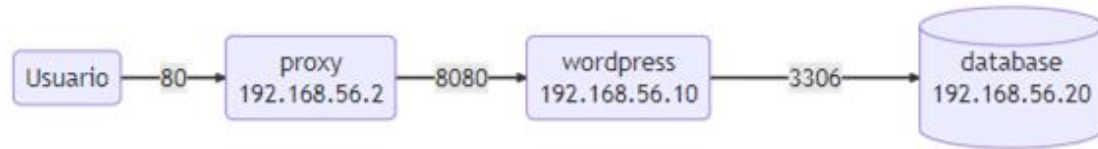


Figura 1. Infraestructura virtual del proyecto.

Desarrollo.

Antes de iniciar con la explicación del proyecto es importante mencionar que el presente proyecto se encuentra en un repositorio de Github, en donde se muestra el código fuente del mismo, así como el archise recomienda revisarlo para explorar su funcionalidad del mismo, en la siguiente liga: https://github.com/cppmx/wordpress_chef/tree/master

Configuración de las máquinas virtuales con Vagrant.

Se inicia la configuración en el Vagrantfile y se define la funcionalidad vagrant.env, lo que permite habilitar algunos valores como las IPs de las máquinas virtuales, el usuario y el password de la Base de Datos que se utilizará para configurar el servidor Wordpress.

```
Vagrant.configure("2") do |config|
  config.env.enable # Habilitamos vagrant-env(.env)

  if ENV['TESTS'] == 'true'
    config.vm.define "test" do |testing|
      config.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # Utilizamos una imagen de Ubuntu 20.04 por defecto

      testing.vm.provision "shell", inline: <<-SHELL
        # Instalar ChefDK
        wget -qO- https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk

        export CHEF_LICENSE="accept"

        # Instalar las gemas necesarias para las pruebas
        cd /vagrant/cookbooks/database && chef exec bundle install
        cd /vagrant/cookbooks/wordpress && chef exec bundle install
        cd /vagrant/cookbooks/proxy && chef exec bundle install

        chown -R vagrant:vagrant /opt/chefdk
      SHELL
    end
  end
end
```

Figura 2. Vagrantfile, configuración de máquina virtual de pruebas.

En seguida, se declaran dos bloques condicionales, mismos que se utilizan para definir dos configuraciones de máquinas virtuales, una para pruebas en caso de que la variable “TESTS” tenga un valor de “true”, en la que se instalarán las gemas necesarias para las pruebas.

En caso contrario se ejecutarán las tres máquinas virtuales que se mencionaron anteriormente, mismo que será el entorno de producción, en este caso se considera Ubuntu 20.04 para cada una de ellas.

En cada máquina virtual se asignan las configuraciones necesarias, mismas que se declaran en el archivo .env, además, se realiza lo siguiente:

- Se instala ChefDK, herramienta para desarrollar, probar y mantener infraestructuras de TI.
- Se agrega una receta de Chef “chef.add_recipe”, misma que agrega la configuración necesaria en la máquina virtual. Cada receta será tomada desde los Cookbooks, misma que hará las configuraciones adecuadas para cada máquina virtual.

```
config.vm.define "database" do |db|
  db.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # Utilizamos una imagen de Ubuntu 20.04 por defecto
  db.vm.hostname = "db.unir.mx"
  db.vm.network "private_network", ip: ENV["DB_IP"]
  db.vm.provision "chef_solo" do |chef|
    chef.install = "true"
    chef.arguments = "--chef-license accept"
    chef.add_recipe "database"
    chef.json = {
      "config" => {
        "db_ip" => "#{ENV["DB_IP"]}",
        "wp_ip" => "#{ENV["WP_IP"]}",
        "db_user" => "#{ENV["DB_USER"]}",
        "db_pwd" => "#{ENV["DB_PSWD"]}"
      }
    }
  end
end

config.vm.define "wordpress" do |sitio|
  sitio.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # Utilizamos una imagen de Ubuntu 20.04 por defecto
  sitio.vm.hostname = "wordpress.unir.mx"
  sitio.vm.network "private_network", ip: ENV["WP_IP"]

  sitio.vm.provision "chef_solo" do |chef|
    chef.install = "true"
    chef.arguments = "--chef-license accept"
    chef.add_recipe "wordpress"
    chef.json = {
      "config" => {
        "db_ip" => "#{ENV["DB_IP"]}",
        "db_user" => "#{ENV["DB_USER"]}",
        "db_pwd" => "#{ENV["DB_PSWD"]}"
      }
    }
  end
end

config.vm.define "proxy" do |proxy|
  proxy.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # Utilizamos una imagen de Ubuntu 20.04 por defecto
  proxy.vm.hostname = "wordpress.unir.mx"
  proxy.vm.network "private_network", ip: ENV["PROXY_IP"]
  proxy.vm.provision "chef_solo" do |chef|
    chef.install = "true"
    chef.arguments = "--chef-license accept"
    chef.add_recipe "proxy"
    chef.json = {
      "config" => {
        "wp_ip" => "#{ENV["WP_IP"]}"
      }
    }
  end
end
```

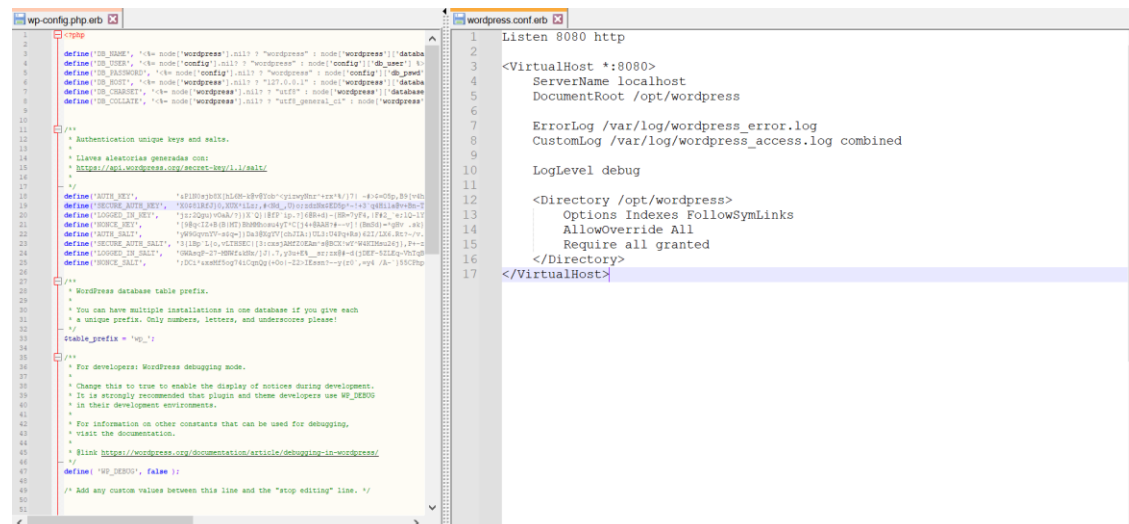
Figura 3. Vagrantfile, configuración de las máquinas virtuales del proyecto.

Wordpress:

Para hacer la instalación y configuración de Wordpress se van a requerir dos plantillas: `wodpress.conf.erb` y `wp-config.php.erb`.

En la primera se encuentra la configuración del sitio wordpress, en donde se le dirá a Apache en que ruta se encontrarán los archivos de wordpress y que mostrará el sitio cuando las peticiones lleguen al puerto 8080.

La segunda hará la sustitución de los valores predeterminados por el contenido de las variables en el nodo, mismos que se definieron anteriormente y se mandaron llamar en el Vagrantfile, los cuales son: nombre de la base de datos, el usuario de la base de datos, la contraseña del usuario de la base de datos, la dirección del servidor de la base de datos, el conjunto de caracteres de a base de datos y `db_collate`.



```
1 wp-config.php.erb
2
3 <?php
4
5 define('DB_HOST', '%{db_host}%');
6 define('DB_USER', '%{db_user}%');
7 define('DB_PASSWORD', '%{db_password}%');
8 define('DB_NAME', '%{db_name}%');
9 define('DB_CHARSET', '%{db_charset}%');
10 define('DB_COLLATE', '%{db_collate}%');
11
12 /*
13  * Authentication unique keys and salts.
14  *
15  * Llévese aleatorias generadas con:
16  * https://api.wordpress.org secret-key/1/keys/
17  */
18
19 define('AUTH_KEY', '%{auth_key}%');
20 define('SECURE_AUTH_KEY', '%{secure_auth_key}%');
21 define('LOGGED_IN_KEY', '%{logged_in_key}%');
22 define('AUTH_SALT', '%{auth_salt}%');
23 define('SECURE_AUTH_SALT', '%{secure_auth_salt}%');
24 define('LOGGED_IN_SALT', '%{logged_in_salt}%');
25
26 /*
27  * WordPress database table prefix.
28  *
29  * You can have multiple installations in one database if you give each
30  * a unique prefix. Only numbers, letters, and underscores please!
31  */
32 $table_prefix = 'wp_';
33
34 /*
35  * For developers: WordPress debugging mode.
36  *
37  * Change this to true to enable the display of notices during development.
38  * It is strongly recommended that plugin and theme developers use WP_DEBUG
39  * in their development environments.
40  *
41  * For information on other constants that can be used for debugging,
42  * visit the documentation.
43  *
44  * Link: https://wordpress.org/documentation/article/debugging-in-wordpress/
45  */
46 define('WP_DEBUG', false);
47
48 /* Add any custom values between this line and the "stop editing" line. */
49
50
51
```

```
1 wodpress.conf.erb
2
3 Listen 8080 http
4
5 <VirtualHost *:8080>
6     ServerName localhost
7     DocumentRoot /opt/wordpress
8
9     ErrorLog /var/log/wordpress_error.log
10    CustomLog /var/log/wordpress_access.log combined
11
12    LogLevel debug
13
14    <Directory /opt/wordpress>
15        Options Indexes FollowSymLinks
16        AllowOverride All
17        Require all granted
18    </Directory>
19 </VirtualHost>
```

Figura 4. Configuración de plantillas Wordpress

Creación de la receta.

Se iniciará configurando el archivo `recipes/default.rb` en el cual se podrá configura dependiendo del sistema operativo que se esté ejecutando, actualiza el sistema operativo, agrega una entrada en `/etc/hosts` para el acceso a la base de datos, instala y configura el servidor web y wordpress, incluye las recetas correspondientes a cada sistema operativo.

En el archivo `"Ubuntu_web_rb"` se instalan los paquetes: `apache2`, `php`, `php-mysql`, `php-mysqld`, `php-mysqli`, `php-json`, `unzip` y `curl`. Se agrega un archivo llamado `info.php` en la carpeta `apache2`,

php, php-mysql, php-mysqldb, php-mysqli, php-json, unzip y curl, el cual servirá para verificar que la instalación de Apache y PHP sean correctas. Finalmente se inicializa el servidor web.

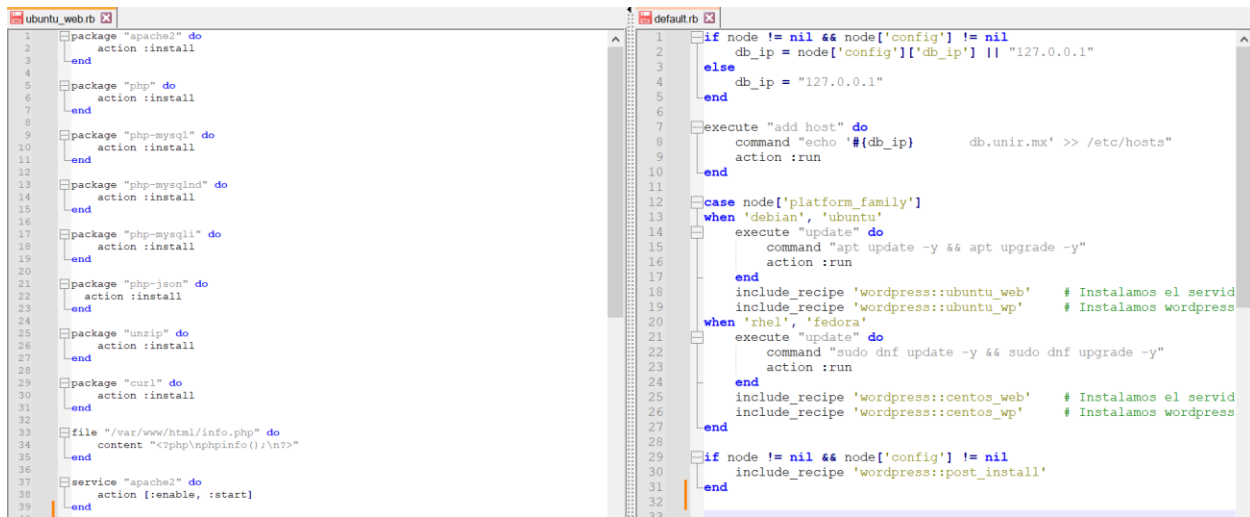


Figura 6. Recipes y Default de wordpress

Instalación de wordpress en Ubuntu.

Se crea el archivo ubuntu_wp.rb en la misma ruta de las recetas de los cookbooks, en donde se realiza lo siguiente:

- Se revisa que exista el directorio /opt que es donde se va a instalar Wordpress.
- Se descarga Wordpress del sitio oficial.
- Se extrae la carpeta en el directorio mencionado, esto creará una carpeta llamada wordpress, entonces el directorio final será /opt/wordpress.
- Se cambia el propietario de la carpeta creada en donde el dueño será www-data:www-data, esto porque en Ubuntu el servicio web corre con ese usuario y solo puede acceder a las carpetas de las que este usuario es dueño.
- Se crea el archivo wp-config.php usando una plantilla.
- Se crea el archivo /etc/apache2/sites-enabled/wordpress.conf usando una plantilla.
- Finalmente se reinicia el servidor Apache.

```

1  execute "get_wordpress" do
2      command "curl -o /tmp/wordpress.zip https://wordpress.org/latest.zip"
3      action :run
4      not_if { ::File.exist?('/tmp/wordpress.zip') }
5  end
6
7  execute "extract_wordpress" do
8      command "unzip -q /tmp/wordpress.zip -d /opt/"
9      action :run
10     notifies :run, 'execute[set_wordpress_permissions]', :immediately
11     not_if { ::File.exist?('/opt/wordpress') }
12 end
13
14 execute "set_wordpress_permissions" do
15     command "chown -R www-data:www-data /opt/wordpress"
16     action :nothing
17 end
18
19 template '/opt/wordpress/wp-config.php' do
20     source 'wp-config.php.erb'
21     owner 'www-data'
22     group 'www-data'
23     mode '0644'
24     not_if { ::File.exist?('/opt/wordpress/wp-config.php') }
25 end
26
27 template '/etc/apache2/sites-enabled/wordpress.conf' do
28     source 'wordpress.conf.erb'
29     not_if { ::File.exist?('/etc/apache2/sites-enabled/wordpress.conf') }
30 end
31
32 service "apache2" do
33     action :restart
34 end

```

Figura 7. Instalación de wordpress en Ubuntu

Post-instalación de Wordpress.

Se realiza la configuración de wordpress mediante su CLI, en el archivo post_install.rb en donde se hace lo siguiente:

- Se descarga la última versión de WP-CLI y lo guarda en el directorio temporal. Establece los permisos y propietario del archivo para que sea ejecutable por el usuario root.
- Mueve el archivo wp-cli.phar desde /tmp a /bin permitiendo que sea accesible globalmente; en caso de que ya exista el archivo no se ejecuta el comando.
- Se establecen los permisos para que el archivo sea ejecutable para todos los usuarios.
- Se instala wordpress en la ubicación /opt/wordpress/, se pasan los parámetros para la configuración del sitio, como lo son, la url, el título, el usuario, contraseña y el correo electrónico, todo esto verificando que no esté instalado, en caso de que lo esté no se ejecutará el comando “wp core install”.

```

1 # Instalar WP CLI
2 remote_file '/tmp/wp' do
3   source 'https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar'
4   owner 'root'
5   group 'root'
6   mode '0755'
7   action :create
8 end
9
10 # Mover WP CLI a /bin
11 execute 'Move WP CLI' do
12   command 'mv /tmp/wp /bin/wp'
13   not_if { !File.exist?('/bin/wp') }
14 end
15
16 # Hacer WP CLI ejecutable
17 file '/bin/wp' do
18   mode '0755'
19 end
20
21 # Instalar Wordpress y configurar
22 execute 'Finish Wordpress installation' do
23   command 'sudo -u vagrant -i -- wp core install --path=/opt/wordpress/ --url=localhost --title="UNIR Actividad Grupal 1001A" --admin_user=admin --ac'
24   not_if 'wp core is-installed', environment: { 'PATH' => '/bin:/usr/bin:/usr/local/bin' }
25 end

```

Figura 8. Instalación de WP-CLI y configuración del blog

Database:

La receta para la configuración de la base de datos involucra el archivo “default.rb” en el que se hace lo siguiente:

- Se evalúa cuál Sistema Operativo se está ejecutando; primeramente, actualiza el Sistema, sin importar cuál sea, después se carga la receta correspondiente a cada Sistema.

Para la configuración se van a requerir los valores: usuario Wordpress, contraseña del usuario wordpress, Ip de la máquina Virtual wordpress, la IP de la base de datos (solo para Ubuntu). Estas variables se definen o mandan llamar en el Vagrantfile.

Se explica el contenido de la receta de la instalación con Ubuntu:

- Se instala el paquete MySQL Server.
- Se arranca el servicio.
- Se crea la Base de Datos llamada Wordpress.
- Se agrega el usuario definido en la variable db_user con la contraseña de la variable db_pswd y se le asigna permiso de conectarse desde la IP definida en la variable wp_ip.
- Se modifica el archivo de configuración de MySQL para que MySQL escuche en la dirección IP que contenga db_ip en lugar de escuchar en el localhost.
- Se reinicia el servicio de MySQL después de modificar el archivo de configuración, de esta manera los cambios en la dirección IP de escucha se aplican sin tener que reiniciar manualmente el servicio.


```

1 case node['platform_family']
2 when 'debian', 'ubuntu'
3   execute "update" do
4     command "apt update -y && apt upgrade -y"
5     action :run
6   end
7   include_recipe 'database::ubuntu'
8 when 'rhel', 'fedora', 'centos'
9   execute "update" do
10    command "sudo dnf update -y && sudo dnf upgrade -y"
11    action :run
12  end
13  include_recipe 'database::centos'
14 end

```

Figura 9. Receta para la base de datos.

Proxy Nginx:

Esta receta se encarga de instalar y configurar el servidor proxy Nginx para que sirva de punto de entrada a la aplicación Wordpress.

- Se instala y se habilita el paquete Nginx.
- Dependiendo del Sistema Operativo se hace lo siguiente:
 - o Actualización del Sistema Operativo.
 - o Configuración de Nginx usando la plantilla correspondiente al sistema operativo.
 - o En el caso de CentOS se habilita la regla `httpd_can_network_connect` en SELinux.
 - o Y también, sólo en el caso de CentOS se abre el puerto 80 en el firewall.

Se crean las plantillas de configuración de Nginx: `default`, `centos.conf.erb` y `Ubuntu.conf.erb`, en las cuales se agrega la línea `server <%= node['config'].nil? ? "127.0.0.1" : node['config']['wp_ip'] %>:8080;` la cual es la parte donde se utiliza la IP de wordpress, está ya se definió o se mandó llamar desde el `vagrantfile`.

```

1 package "nginx" do
2   action :install
3 end
4
5 service "nginx" do
6   action [:enable, :start]
7 end
8
9 case node['platform_family']
10 when 'debian', 'ubuntu'
11   execute "update" do
12     command "apt update -y && apt upgrade -y"
13     action :run
14   end
15   template '/etc/nginx/nginx.conf' do
16     source 'ubuntu.conf.erb'
17     action :create
18     notifies :restart, 'service[nginx]', :immediately
19   end
20 when 'rhel', 'fedora'
21   execute "update" do
22     command "sudo dnf update -y && sudo dnf upgrade -y"
23     action :run
24   end
25   template '/etc/nginx/nginx.conf' do
26     source 'centos.conf.erb'
27     action :create
28     notifies :restart, 'service[nginx]', :immediately
29   end
30 end
31
32 selinux_boolean 'httpd_can_network_connect' do
33   value true
34   action :set
35 end
36
37 execute "firewall-cmd --zone=public --add-port=80/tcp --permanent" do
38   action :run
39 end
40
41 execute "firewall-cmd --reload" do
42   action :run
43 end
44 end

```

```

1 user www-data;
2 worker_processes auto;
3 pid /run/nginx.pid;
4 include /etc/nginx/modules-enabled/*.conf;
5
6 events {
7   worker_connections 768;
8   # multi_accept on;
9 }
10
11 http {
12   tcp_nopush on;
13   tcp_nodelay on;
14   keepalive_timeout 65;
15   types_hash_max_size 2048;
16
17   include /etc/nginx/mime.types;
18   default_type application/octet-stream;
19
20   ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POC0LE
21   ssl_prefer_server_ciphers on;
22
23   gzip on;
24
25   log_format custom '$remote_addr - $remote_user [$time_local] '
26     '"$request" $status $body_bytes_sent '
27     '"$http_referer" "$http_user_agent" '
28     '"$http_x_forwarded_for" $request_id ';
29
30   upstream backend {
31     server <%= node['config'].nil? ? "127.0.0.1" : node['config']['wp_ip'] %>:8080;
32   }
33
34   server {
35     server_name actividad1.unir.mx;
36     listen 80;
37
38     error_log /var/log/proxy_error.log warn;
39     access_log /var/log/proxy_access.log custom;
40
41     server_tokens off; # Don't display Nginx version
42     add_header X-XSS-Protection "1; mode=block"; # Prevent cross-site scripting exploits
43     add_header Content-Security-Policy "frame-ancestors 'self'"; # Don't allow to be embedded externally
44     add_header X-Frame-Options "SAMEORIGIN"; # Prevents clickjacking attacks by allowing/disallowi
45
46     gzip on;
47     gzip_disable "msie6";
48     gzip_types text/plain text/css application/javascript text/xml application/xml+rss
49
50     location / {
51       # don't cache it

```

Figura 10. Configuración de proxy Nginx

The image is a collage of terminal windows and a small application window. The leftmost terminal shows network connectivity tests (ping) and the installation of WordPress using 'wp-cli'. The middle terminal shows the WordPress installation progress, including database creation and user setup. The rightmost terminal shows the system status, including memory usage, network configuration, and security updates. A small window titled 'Mindblown: a blog about philosophy.' is visible in the bottom right corner.

Pruebas unitarias.

Una vez ejecutado el script podremos seleccionar la opción de ejecutar las pruebas unitarias.

```

$ sh tests.sh
Seleccione una opción:
1. Ejecutar pruebas unitarias en una VM
2. Ejecutar pruebas unitarias en un contenedor
3. Ejecutar pruebas de integración e infraestructura
4. Salir
Opción: 1

```

Después de haber seleccionado la opción número 1, se podrá ejecutar la prueba” se empezarán a ejecutar las pruebas y podremos observar los resultados.

```

database::default
Running tests for Ubuntu 20.04 platform...
  installs mysql-server
Running tests for Ubuntu 20.04 platform...
  enables and starts the mysql service
Update OS
Running tests for Ubuntu 20.04 platform...
  is expected to run execute "update"
  creates the wordpress database
Running tests for Ubuntu 20.04 platform...
  is expected to run execute "create_mysql_database"
  creates the mysql user and grants privileges
Running tests for Ubuntu 20.04 platform...
  is expected to run execute "create_mysql_user"

Finished in 10.56 seconds (files took 6.44 seconds to load)
12 examples, 0 failures

```

Figura 13. Pruebas unitarias en Ubuntu.

Como se puede observar las pruebas pasaron exitosamente, esto fue ejecutado en un Sistema Ubuntu. al ser ejecutadas en un Sistema Centos arroja el siguiente resultado:

```

wordpress::default
Running tests for CentOS 8 platform...
  installs httpd package
Running tests for CentOS 8 platform...
  installs php package
Running tests for CentOS 8 platform...
  installs php-mysqldb package
Running tests for CentOS 8 platform...
  installs php-json package
Running tests for CentOS 8 platform...
  installs unzip package
Running tests for CentOS 8 platform...
  installs curl package
Running tests for CentOS 8 platform...
  creates the info.php file
Running tests for CentOS 8 platform...
  opens port 8080 in the firewall
Running tests for CentOS 8 platform...
  opens port 80 in the firewall
Running tests for CentOS 8 platform...
  reloads the firewall
Running tests for CentOS 8 platform...
  enables and starts the httpd service
Install wordpress
Running tests for CentOS 8 platform...
  is expected to run execute "get wordpress"
Running tests for CentOS 8 platform...
  is expected to run execute "extract_wordpress"
Running tests for CentOS 8 platform...
  is expected to create template "/opt/wordpress/wp-config.php"
Running tests for CentOS 8 platform...
  is expected to create template "/etc/httpd/conf.d/wordpress.conf"

```

Figura 14. Pruebas unitarias en Centos.

Pruebas de integración.

Para las pruebas de integración se consideró, como en todo el proyecto que éstas corrieran bajo un Sistema Ubuntu y uno CentOS. Siguiendo la misma dinámica de ejecución de las pruebas con el script, seleccionamos la opción 3.

```
-----> Running postinstall for serverspec plugin
Suite path directory /tmp/verifier/suites does not exist, skipping.
Transferring files to <package-centos-8>
-----> Running serverspec test suite
-----> Installing Serverspec..
  Fetching rspec-its-1.3.0.gem
  Fetching rspec-mocks-3.12.6.gem
  Fetching diff-lcs-1.5.0.gem
  Fetching sfl-2.3.gem
  Fetching net-ssh-7.2.0.gem
  Fetching specinfra-2.86.0.gem
  Fetching rspec-expectations-3.12.3.gem
  Fetching net-telnet-0.1.1.gem
  Fetching serverspec-2.42.2.gem
  Fetching rspec-3.12.0.gem
  Fetching net-scp-4.0.0.gem
  Fetching multi_json-1.15.0.gem
-----> serverspec installed (version 2.42.2)
/opt/chef/embedded/bin/ruby -I/tmp/verifier/suites/serverspec -I/tmp/verifier/gems/gems

Service "nginx"
  is expected to be enabled

Finished in 0.11867 seconds (files took 0.2706 seconds to load)
1 example, 0 failures
```

Figura 15. Pruebas de Integración e Infraestructura en Ubuntu.

Las pruebas anteriores fueron ejecutadas en Ubuntu, de igual forma se consideró Centos y los resultados fueron los esperados, obteniendo así las pruebas de integración y de infraestructura exitosas.

```
Plugin serverspec installed (version 0.5.10)
-----> Running postinstall for serverspec plugin
Suite path directory /tmp/verifier/suites does not exist, skipping.
Transferring files to <source-ubuntu-2004>
-----> Running serverspec test suite
-----> Installing Serverspec..
  Fetching rspec-expectations-3.12.3.gem
  Fetching net-telnet-0.1.1.gem
  Fetching net-ssh-7.2.0.gem
  Fetching net-scp-4.0.0.gem
  Fetching specinfra-2.86.0.gem
  Fetching diff-lcs-1.5.0.gem
  Fetching sfl-2.3.gem
  Fetching rspec-its-1.3.0.gem
  Fetching serverspec-2.42.2.gem
  Fetching rspec-mocks-3.12.6.gem
  Fetching multi_json-1.15.0.gem
  Fetching rspec-3.12.0.gem
-----> serverspec installed (version 2.42.2)
/opt/chef/embedded/bin/ruby -I/tmp/verifier/suites/serverspec -I/tmp/verifier/gems/gems/rspec-support-3.12.1/lib:/tmp/verifier-2.2/lib /opt/chef/embedded/bin/rspec --pattern /tmp/verifier/suites/serverspec/*/*/*_spec.rb --color --format documentation --d
tes/serverspec

Package "nginx"
  is expected to be installed

Port "80"
  is expected to be listening

Finished in 0.1291 seconds (files took 1.66 seconds to load)
2 examples, 0 failures
```

Figura 16. Pruebas de integración e Infraestructura en Centos.

Conclusión.

Algunos aspectos destacados del presente proyecto y utilizar Chef son:

Facilidad de configuración: utilizar Chef permite automatizar la configuración y el aprovisionamiento de software en las máquinas virtuales, lo que facilita la instalación y puesta en marcha de las aplicaciones y servicios necesarios para el proyecto.

Flexibilidad y escalabilidad: La estructura del proyecto permite agregar o eliminar máquinas virtuales según las necesidades del desarrollo o la producción. Esto brinda la posibilidad de escalar el entorno de manera sencilla y rápida.

Automatización: La configuración y aprovisionamiento automáticos proporcionados por Chef permiten agilizar el proceso de configuración del entorno, lo que ahorra tiempo y reduce errores humanos.

Reproducibilidad del entorno: Gracias al uso de Vagrant y Chef, el proyecto garantiza que todos los miembros del equipo de desarrollo trabajen en el mismo entorno, lo que minimiza problemas relacionados con diferencias en las configuraciones y versiones de software.

Gestión eficiente de máquinas virtuales: Vagrant proporciona una capa de abstracción sobre los proveedores de máquinas virtuales, lo que permite gestionar fácilmente el ciclo de vida de las máquinas (creación, configuración, suspensión y destrucción) mediante simples comandos.

Seguridad: Al utilizar máquinas virtuales aisladas, el proyecto proporciona un entorno seguro para desarrollar y probar aplicaciones. Además, el uso de recetas Chef para la configuración asegura que la instalación de software se realice de manera segura y consistente.

Al combinar Vagrant y Chef se logra un equilibrio entre la flexibilidad, automatización y gestión de máquinas virtuales, es por ello que este proyecto demuestra una solución potente y versátil para desarrollar y desplegar aplicaciones en un entorno controlado y eficiente.

En resumen, el proyecto demostró el uso efectivo de Vagrant y Chef para automatizar la creación de infraestructuras y la configuración de componentes, lo que puede ahorrar tiempo y reducir errores en proyectos de infraestructura más complejos.

Referencias.

ProgressChef. (17 marzo 2023). Chef Documentation. <https://docs.chef.io>

ProgressChef. (9 febrero 2023). ChefSpec. <https://docs.chef.io/workstation/chefspec/>

SERVERSPEC. (s.f). RSpec tests for your servers configured by CFEngine, Puppet, Ansible, Itamae or anything else. https://serverspec.org/resource_types.html

ProgressChef. (12 octubre 2022). Test Kitchen. <https://docs.chef.io/workstation/kitchen/>