

**UNIVERSIDAD PRIVADA DE TACNA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS**



Informe Práctico de Laboratorio

Examen de Unidad 3 – Práctico “Despliegue Automatizado de WordPress con Chef y Vagrant”

Que se presenta para el curso:
“Auditoría de sistemas”

Docente:
Dr. Oscar Juan Jimenez Flores

Estudiante:
Ccalli Chata, Joel Robert (2017057528)

**TACNA – PERÚ
2025**

ÍNDICE GENERAL

1. Introducción.....	3
1.1 Objetivo General	3
1.2 Objetivos Específicos	3
1.3 Evidencias Obligatorias	3
2. Examen y Hallazgos	3
2.1 Revisión de Configuraciones	3
2.2 Pruebas de Seguridad	18
Tabla de Evidencias de Ejecución de Recetas Chef	19
2.3 Pruebas de Integración	20
3. Matriz de Riesgos (Punto 8 del Informe).....	21
4. Recomendaciones	21
5. Conclusiones	22

1. Introducción

1.1 Objetivo General

Auditar la seguridad, conformidad y mejores prácticas en la implementación del despliegue automatizado de WordPress utilizando Chef y Vagrant, identificando vulnerabilidades y proponiendo recomendaciones de mejora.

1.2 Objetivos Específicos

1. Verificar la correcta exposición y restricción de puertos en la configuración de red
2. Evaluar el manejo seguro de credenciales y datos sensibles
3. Validar la implementación de mecanismos de logging y trazabilidad
4. Analizar la segregación de ambientes (dev/prod) en la infraestructura
5. Comprobar el cumplimiento de estándares de seguridad en las recetas Chef

1.3 Evidencias Obligatorias

- **Anexo A:** Captura del comando `vagrant status`
https://capturas/vagrant_status.png
- **Anexo B:** Pantalla de WordPress accesible
https://capturas/wordpress_access.png

2. Examen y Hallazgos

2.1 Revisión de Configuraciones

Vagrantfile

- **Anexo C:** Puertos expuestos sin restricciones

Download

```
config.vm.network "forwarded_port", guest: 80, host: 8080 # Sin autenticación
config.vm.network "private_network", ip: "192.168.56.10" # Red privada sin ACL
```

https://capturas/vagrant_network.png

- **Hallazgo:**

- Puerto 3306 (MySQL) accesible desde cualquier host en la red interna
- No se implementan reglas de firewall específicas

Recetas Chef

- **Anexo D:** Credenciales en texto plano

```
default['mysql']['root_password'] = 'Epnewman123' # En attributes/default.rb
```

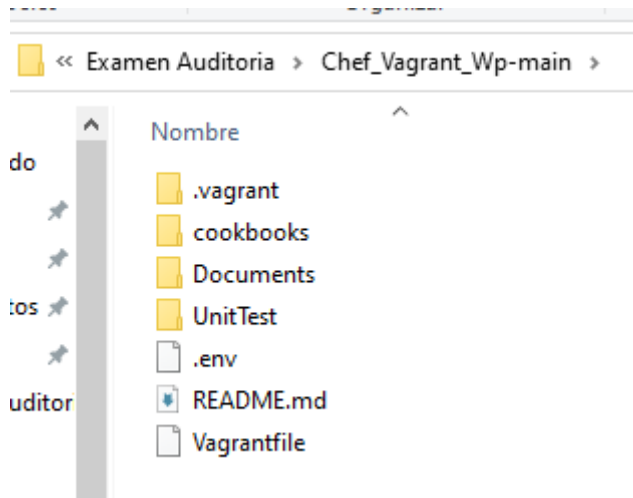
https://capturas/credenciales_plano.png

- **Anexo E:** Versiones de software no actualizadas

```
depends 'apache2', '~> 8.0' # En metadata.rb (versión antigua)
```

https://capturas/versions_software.png

Okey primero vamos a clonar el repositorio del Github:



Y Abrimos en visual code para ejecutar los comando para ejecutar:

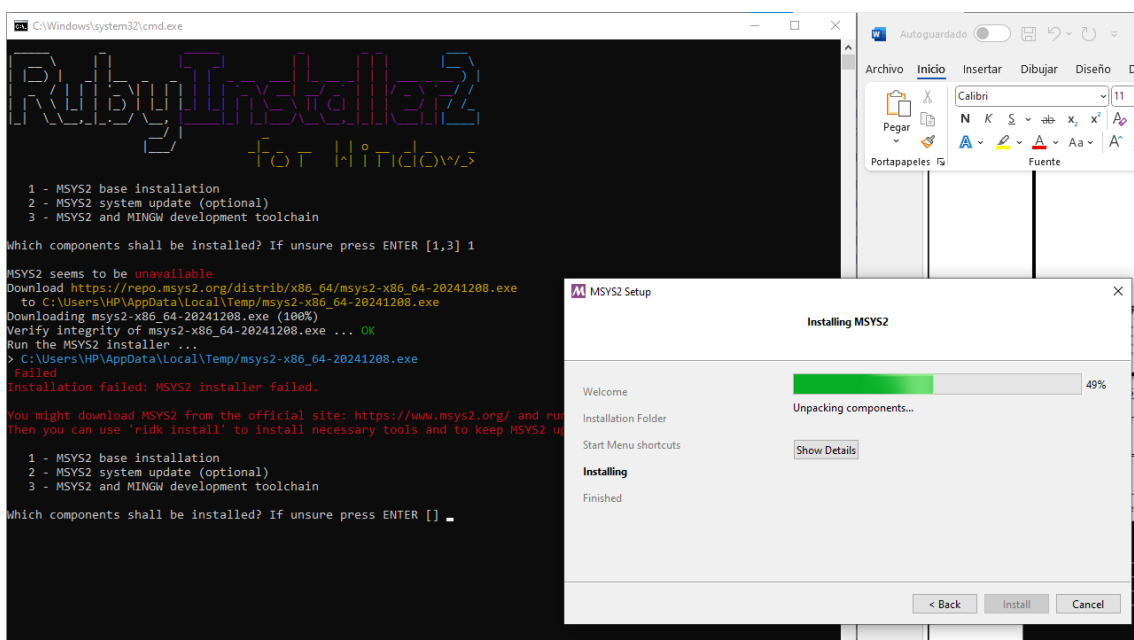
```
C:\Windows\system32\cmd.exe

RubyInstaller2
for Windows

1 - MSYS2 base installation
2 - MSYS2 system update (optional)
3 - MSYS2 and MINGW development toolchain

Which components shall be installed? If unsure press ENTER [1,3] _
```

Instalamos todas las dependencias para nuestro uso:



También vamos a actualizar con la opción 2:

```
C:\Windows\system32\cmd.exe

1 - MSYS2 base installation
2 - MSYS2 system update (optional)
3 - MSYS2 and MINGW development toolchain

Which components shall be installed? If unsure press ENTER [1,3] 1

MSYS2 seems to be unavailable
Download https://repo.msys2.org/distrib/x86\_64/msys2-x86\_64-20241208.exe
to C:\Users\HP\AppData\Local\Temp\msys2-x86_64-20241208.exe
Downloading msys2-x86_64-20241208.exe (100%)
Verify integrity of msys2-x86_64-20241208.exe ... OK
Run the MSYS2 installer ...
> C:\Users\HP\AppData\Local\Temp\msys2-x86_64-20241208.exe
Failed
Installation failed: MSYS2 installer failed.

You might download MSYS2 from the official site: https://www.msys2.org/ and run the installer manually.
Then you can use 'ridk install' to install necessary tools and to keep MSYS2 up-to-date.

1 - MSYS2 base installation
2 - MSYS2 system update (optional)
3 - MSYS2 and MINGW development toolchain

Which components shall be installed? If unsure press ENTER [] 2

> sh -lc true
MSYS2 seems to be properly installed
Check msys2-keyring version:
-> up-to-date
MSYS2 system update (optional) part 1 ...
> pacman -Syu --needed --noconfirm
:: Sincronizando las bases de datos de los paquetes...
  clangarm64 479,4 KiB 79,9 KiB/s 00:06 [#####] 100%
  mingw32 124,5 KiB 19,6 KiB/s 00:06 [#####] 100%
  mingw64 464,1 KiB 77,3 KiB/s 00:06 [#####] 100%
  ucrt64 505,1 KiB 82,2 KiB/s 00:06 [#####] 100%
  clang32 21,0 B 2,00 B/s 00:08 [#####] 100%
  clang64 492,3 KiB 769 KiB/s 00:01 [#####] 100%
  msys 384,1 KiB 578 KiB/s 00:01 [#####] 100%
:: Starting core system upgrade...
advertencia: terminate other MSYS2 programs before proceeding
resolviendo dependencias...
buscando conflictos entre paquetes...

Paquetes (6) bash-5.2.037-2 filesystem-2025.05.08-2 mintty-1~3.7.8-1 msys2-runtime-3.6.3-4 pacman-6.1.0-16
              pacman-mirrors-20250607-1

Tamaño total de la descarga: 11,72 MiB
Tamaño total de la instalación: 63,04 MiB
Tamaño neto tras actualizar: 1,37 MiB

:: ¿Continuar con la instalación? [S/n]
:: Obteniendo los paquetes...
  filesystem-2025.05.08-2-x86_64 92,5 KiB 127 KiB/s 00:01 [#####] 100%
  pacman-mirrors-20250607-1-any 3,1 KiB 7,16 KiB/s 00:00 [#####] 100%
  mintty-1~3.7.8-1-x86_64 844,4 KiB 611 KiB/s 00:01 [#####] 100%
  bash-5.2.037-2-x86_64 2,4 MiB 1609 KiB/s 00:02 [#####] 100%
  pacman-6.1.0-16-x86_64 6,6 MiB 2,07 MiB/s 00:03 [#####] 100%
  msys2-runtime-3.6.3-4-x86_64 1718,7 KiB 951 KiB/s 00:00 [#####] 91%
  Total (5/6) 11,6 MiB 1669 KiB/s 00:00 [#####] 98%
```

Actualizamos y continuamos con la opción 3:

```
C:\Windows\system32\cmd.exe
(13/60) actualizando less [#####] 100%
(14/60) actualizando gzip [#####] 100%
(15/60) actualizando libxcrypt [#####] 100%
(16/60) actualizando info [#####] 100%
(17/60) actualizando libtasn1 [#####] 100%
(18/60) actualizando libffi [#####] 100%
(19/60) actualizando ca-certificates [#####] 100%
(20/60) actualizando libunistring [#####] 100%
(21/60) actualizando libidn2 [#####] 100%
(22/60) actualizando libnghttp2 [#####] 100%
(23/60) actualizando libssh2 [#####] 100%
(24/60) actualizando libsqlite [#####] 100%
(25/60) actualizando libedit [#####] 100%
(26/60) actualizando libcurl [#####] 100%
(27/60) actualizando curl [#####] 100%
(28/60) actualizando file [#####] 100%
(29/60) actualizando mpfr [#####] 100%
(30/60) actualizando gawk [#####] 100%
(31/60) actualizando libgdbm [#####] 100%
(32/60) actualizando gdbm [#####] 100%
(33/60) actualizando libargp [#####] 100%
(34/60) actualizando getent [#####] 100%
(35/60) actualizando libgettextpo [#####] 100%
(36/60) actualizando libasprintf [#####] 100%
(37/60) actualizando gettext [#####] 100%
(38/60) actualizando libgpg-error [#####] 100%
(39/60) actualizando libassuan [#####] 100%
(40/60) actualizando libgcrypt [#####] 100%
(41/60) actualizando libhogweed [#####] 100%
(42/60) actualizando libnettle [#####] 100%
(43/60) actualizando libgnutls [#####] 100%
(44/60) actualizando nettle [#####] 100%
(45/60) actualizando pinentry [#####] 100%
(46/60) actualizando gnupg [#####] 100%
==> Añadiendo las claves de msys2.gpg...
==> Actualizando la base de datos de claves de confianza...
gpg: siguiente comprobación de base de datos de confianza el: 2025-12-16
(47/60) actualizando libpcrc [#####] 100%
(48/60) actualizando grep [#####] 100%
(49/60) actualizando inetutils [#####] 100%
(50/60) actualizando libutil-linux [#####] 100%
(51/60) actualizando msys2-keyring [#####] 100%
==> Añadiendo las claves de msys2.gpg...
==> Actualizando la base de datos de claves de confianza...
gpg: siguiente comprobación de base de datos de confianza el: 2025-12-16
(52/60) actualizando msys2-launcher [#####] 100%
(53/60) actualizando nano [#####] 100%
(54/60) actualizando perl [#####] 100%
(55/60) actualizando rebase [#####] 100%
(56/60) actualizando tzcode [#####] 100%
(57/60) actualizando util-linux [#####] 100%
(58/60) actualizando which [#####] 100%
(59/60) actualizando xz [#####] 100%
(60/60) actualizando zstd [#####] 100%
.: Ejecutando los «hooks» de posinstalación...
(1/1) Updating the info directory file...
MSYS2 system update (optional) succeeded

  1 - MSYS2 base installation
  2 - MSYS2 system update (optional)
  3 - MSYS2 and MINGW development toolchain

Which components shall be installed? If unsure press ENTER [ ]
```

Nuestra ultima opción en cuanto a ruby:

```
CA\Windows\system32\cmd.exe
(56/60) actualizando tzcode [#####] 100%
(57/60) actualizando util-linux [#####] 100%
(58/60) actualizando which [#####] 100%
(59/60) actualizando xz [#####] 100%
(60/60) actualizando zstd [#####] 100%
:: Ejecutando los «hooks» de posinstalación...
(1/1) Updating the info directory file...
MSYS2 system update (optional) succeeded

 1 - MSYS2 base installation
 2 - MSYS2 system update (optional)
 3 - MSYS2 and MINGW development toolchain

Which components shall be installed? If unsure press ENTER [ ] 3
> sh -lc true
MSYS2 seems to be properly installed
Install MSYS2 and MINGW development toolchain ...
> pacman -S --noconfirm autoconf automake-wrapper diffutils file gawk grep libtool m4 make patch sed
texinfo texinfo-tex wget mingw-w64-ucrt-x86_64-binutils mingw-w64-ucrt-x86_64-crt-git mingw-w64-ucrt-x86_64-gcc ming
w-w64-ucrt-x86_64-gcc-libs mingw-w64-ucrt-x86_64-headers-git mingw-w64-ucrt-x86_64-libmangle-git mingw-w64-ucrt-x86_6
4-libwinpthread-git mingw-w64-ucrt-x86_64-make mingw-w64-ucrt-x86_64-tools-git mingw-w64-ucrt-x86_64-winpthreads-git
pkgconf mingw-w64-ucrt-x86_64-pkgconf
advertencia: file-5.46.2 está actualizado -- omitiéndolo
advertencia: gawk-5.3.2-1 está actualizado -- omitiéndolo
advertencia: grep-3.10.7 está actualizado -- omitiéndolo
advertencia: sed-4.9-1 está actualizado -- omitiéndolo
advertencia: wget-1.25.0-1 está actualizado -- omitiéndolo
resolviendo dependencias...
buscando conflictos entre paquetes...

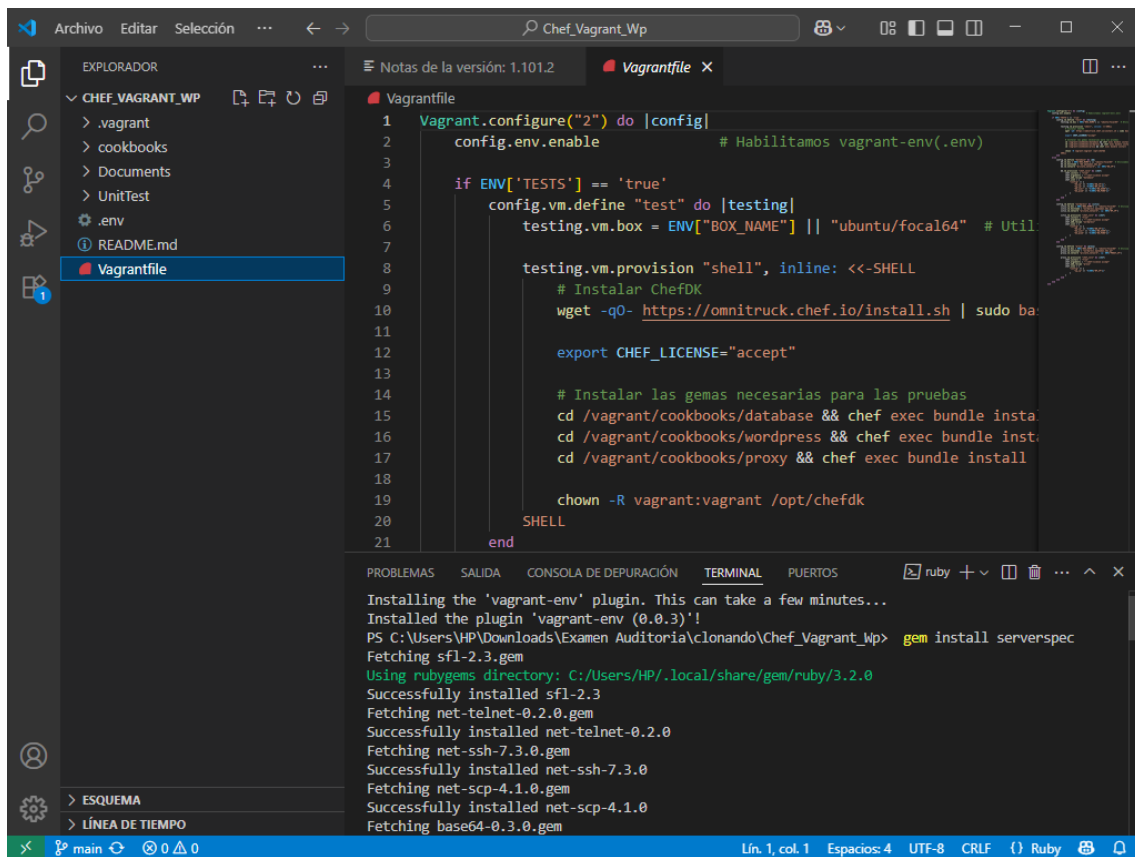
Paquetes (47) autoconf2.13-2.13-6 autoconf2.69-2.69-4 autoconf2.71-2.71-4 autoconf2.72-2.72-3
automake1.11-1.11.6-6 automake1.12-1.12.6-6 automake1.13-1.13.4-7 automake1.14-1.14.1-6
automake1.15-1.15.1-4 automake1.16-1.16.5-1 automake1.17-1.17-1 automake1.18-1.18-1 libgc-8.2.8-1
libguile-3.0.10-3 libltdl-2.5.4-3 libxml2-2.13.8-1 mingw-w64-ucrt-x86_64-gettext-runtime-0.25-1
mingw-w64-ucrt-x86_64-gmp-6.3.0-2 mingw-w64-ucrt-x86_64-isl-0.27-1
mingw-w64-ucrt-x86_64-libiconv-1.18-1 mingw-w64-ucrt-x86_64-mpc-1.3.1-2
mingw-w64-ucrt-x86_64-mpfr-4.2.2-1 mingw-w64-ucrt-x86_64-windows-default-manifest-6.4-4
mingw-w64-ucrt-x86_64-zlib-1.3.1-1 mingw-w64-ucrt-x86_64-zstd-1.5.7-1 autoconf-wrapper-20250528-1
autogen-5.18.16-5 automake-wrapper-20250528-1 diffutils-3.12-1 libtool-2.5.4-3 m4-1.4.19-2
make-4.4.1-2 mingw-w64-ucrt-x86_64-binutils-2.44-4
mingw-w64-ucrt-x86_64-crt-git-13.0.0.r29.gb351226e3-1 mingw-w64-ucrt-x86_64-gcc-15.1.0-5
mingw-w64-ucrt-x86_64-gcc-libs-15.1.0-5 mingw-w64-ucrt-x86_64-headers-git-13.0.0.r29.gb351226e3-1
mingw-w64-ucrt-x86_64-libmangle-git-13.0.0.r29.gb351226e3-1
mingw-w64-ucrt-x86_64-libwinpthread-13.0.0.r29.gb351226e3-1 mingw-w64-ucrt-x86_64-make-4.4.1-3
mingw-w64-ucrt-x86_64-pkgconf-1~2.5.1-1 mingw-w64-ucrt-x86_64-tools-git-13.0.0.r29.gb351226e3-1
mingw-w64-ucrt-x86_64-winpthreads-13.0.0.r29.gb351226e3-1 patch-2.7.6-3 pkgconf-2.5.1-1
texinfo-7.2-1 texinfo-tex-7.2-1

Tamaño total de la descarga: 86,98 MiB
Tamaño total de la instalación: 635,02 MiB

:: ¿Continuar con la instalación? [S/n]
:: Obteniendo los paquetes...
mingw-w64-ucrt-x86_64-crt-git-13.0.0... 4,6 MiB 830 KiB/s 00:06 [#####] 100%
texinfo-7.2-1-x86_64 1472,5 KiB 2,78 MiB/s 00:01 [#####] 100%
mingw-w64-ucrt-x86_64-headers-git-13... 3,1 MiB 1417 KiB/s 00:02 [#####] 48%
mingw-w64-ucrt-x86_64-binutils-2.44-... 1726,5 KiB 940 KiB/s 00:04 [#####] 27%
mingw-w64-ucrt-x86_64-gcc-15.1.0-5-any 250,0 KiB 93,6 KiB/s 00:26 [-----] 0%
libguile-3.0.10-3-x86_64 0,0 B 0,00 B/s 00:00 [#####] 100%
mingw-w64-ucrt-x86_64-isl-0.27-1-any 9,5 MiB 5,11 MiB/s 00:15 [###] 10%
Total ( 2/47) 11,0 MiB 4,32 MiB/s 00:17 [###] 12%
```

También vamos a instalar el siguiente comando:

gem install serverspec

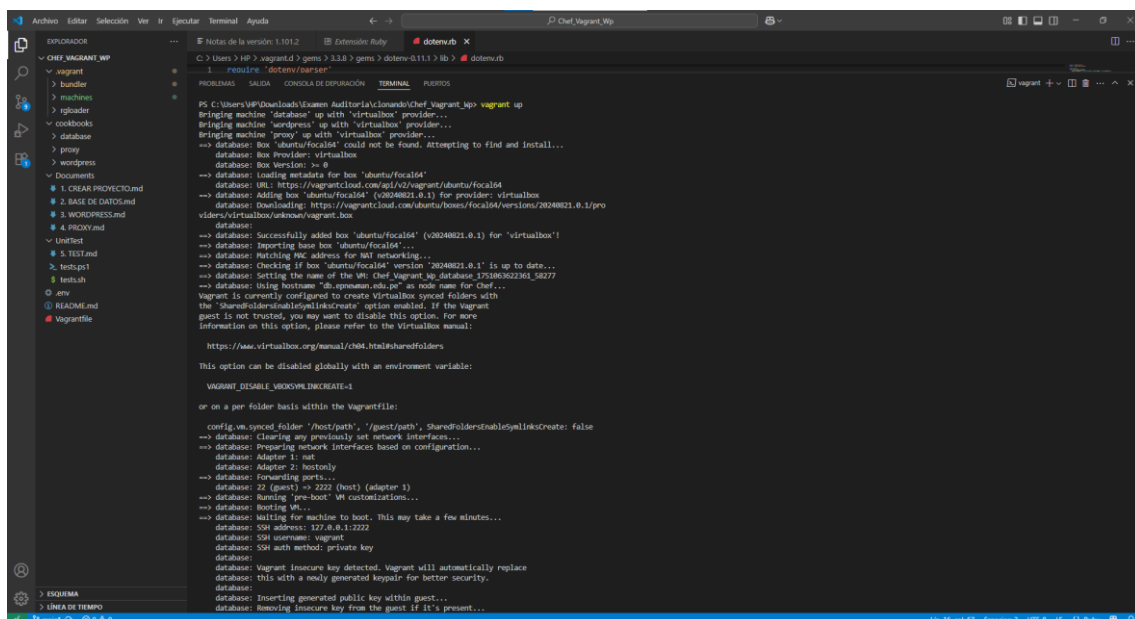


Arreglamos el script:

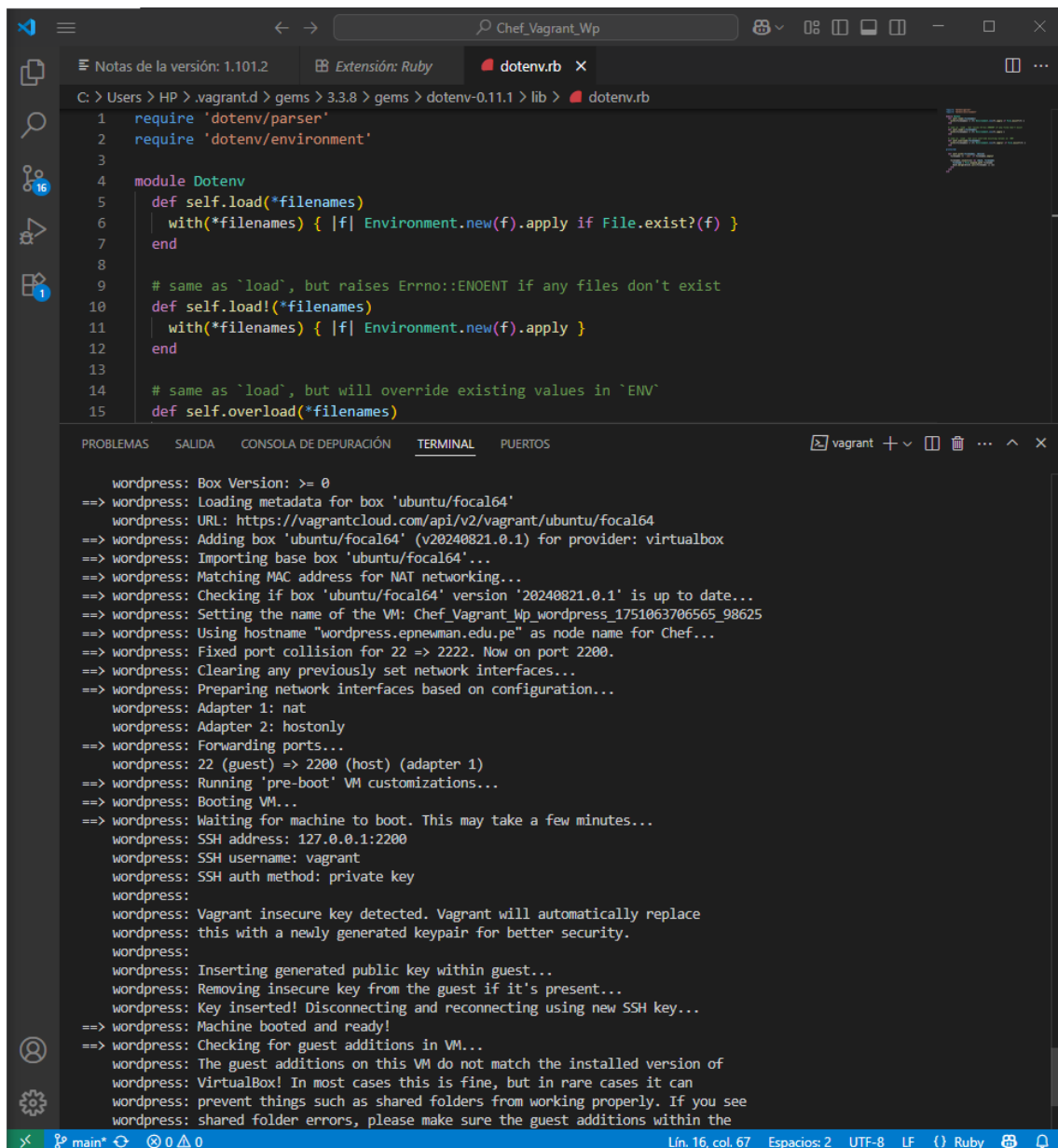
```

  1 require 'dotenv/parser'
  2 require 'dotenv/environment'
  3
  4 module Dotenv
  5   def self.load(*filenames)
  6     with(*filenames) { |f| Environment.new(f).apply if File.exist?(f) }
  7   end
  8
  9   # same as `load`, but raises Errno::ENOENT if any files don't exist
 10  def self.load!(*filenames)
 11    with(*filenames) { |f| Environment.new(f).apply }
 12  end
 13
 14  # same as `load`, but will override existing values in `ENV`
 15  def self.overload(*filenames)
 16    with(*filenames) { |f| Environment.new(f).apply! if File.exist?(f) }
 17  end
 18
 19  protected
 20
 21  def self.with(*filenames, &block)
 22    filenames << '.env' if filenames.empty?
 23
 24    filenames.inject({}) do |hash, filename|
 25      filename = File.expand_path filename
 26      hash.merge(block.call(filename) || {})
 27    end
 28  end
 29 end
 30
```

Y ya podemos ejecutar el vagrant up



Se llega a visualizar claramente como empieza a levantarse WORDPRESS



The image shows a VS Code editor window with a file named `dotenv.rb` open. The file contains Ruby code for a `Dotenv` module. The code is as follows:

```
1 require 'dotenv/parser'
2 require 'dotenv/environment'
3
4 module Dotenv
5   def self.load(*filenames)
6     with(*filenames) { |f| Environment.new(f).apply if File.exist?(f) }
7   end
8
9   # same as `load`, but raises Errno::ENOENT if any files don't exist
10  def self.load!(*filenames)
11    with(*filenames) { |f| Environment.new(f).apply }
12  end
13
14  # same as `load`, but will override existing values in `ENV`
15  def self.overload(*filenames)
```

Below the editor, the terminal window shows the output of a Vagrant command. The output indicates that a new VM named `wordpress` is being created and configured. The output is as follows:

```
wordpress: Box Version: >= 0
==> wordpress: Loading metadata for box 'ubuntu/focal64'
wordpress: URL: https://vagrantcloud.com/api/v2/vagrant/ubuntu/focal64
==> wordpress: Adding box 'ubuntu/focal64' (v20240821.0.1) for provider: virtualbox
==> wordpress: Importing base box 'ubuntu/focal64'...
==> wordpress: Matching MAC address for NAT networking...
==> wordpress: Checking if box 'ubuntu/focal64' version '20240821.0.1' is up to date...
==> wordpress: Setting the name of the VM: Chef_Vagrant_Wp_wordpress_1751063706565_98625
==> wordpress: Using hostname 'wordpress.epnewman.edu.pe' as node name for Chef...
==> wordpress: Fixed port collision for 22 => 2222. Now on port 2200.
==> wordpress: Clearing any previously set network interfaces...
==> wordpress: Preparing network interfaces based on configuration...
wordpress: Adapter 1: nat
wordpress: Adapter 2: hostonly
==> wordpress: Forwarding ports...
wordpress: 22 (guest) => 2200 (host) (adapter 1)
==> wordpress: Running 'pre-boot' VM customizations...
==> wordpress: Booting VM...
==> wordpress: Waiting for machine to boot. This may take a few minutes...
wordpress: SSH address: 127.0.0.1:2200
wordpress: SSH username: vagrant
wordpress: SSH auth method: private key
wordpress:
wordpress: Vagrant insecure key detected. Vagrant will automatically replace
wordpress: this with a newly generated keypair for better security.
wordpress:
wordpress: Inserting generated public key within guest...
wordpress: Removing insecure key from the guest if it's present...
wordpress: Key inserted! Disconnecting and reconnecting using new SSH key...
==> wordpress: Machine booted and ready!
==> wordpress: Checking for guest additions in VM...
wordpress: The guest additions on this VM do not match the installed version of
wordpress: VirtualBox! In most cases this is fine, but in rare cases it can
wordpress: prevent things such as shared folders from working properly. If you see
wordpress: shared folder errors, please make sure the guest additions within the
```

EN EL VIRTUALBOX PODEMOS VISUALIZAR COMO SE CREAN LAS 3 MAQUINAS VIRTUALES:

The image shows a Visual Studio Code editor window with a file named `dotenv.rb` open. The file contains Ruby code for a module named `Dotenv`. The code includes requirements for `dotenv/parser` and `dotenv/environment`, and defines methods `self.load`, `self.load!`, and `self.overload` for loading environment variables from files.

```
1 require 'dotenv/parser'
2 require 'dotenv/environment'
3
4 module Dotenv
5   def self.load(*filenames)
6     with(*filenames) { |f| Environment.new(f).apply if File.exist?(f) }
7   end
8
9   # same as `load`, but raises Errno::ENOENT if any files don't exist
10  def self.load!(*filenames)
11    with(*filenames) { |f| Environment.new(f).apply }
12  end
13
14  # same as `load`, but will override existing values in `ENV`
15  def self.overload(*filenames)
```

Below the editor, a terminal window is open, showing the output of Vagrant commands. The terminal displays the status of the `proxy` service, which is running on `localhost:143` using the `imap` protocol. It also shows the output of `vagrant status`, indicating that the `database`, `wordpress`, and `proxy` services are all running on virtualbox.

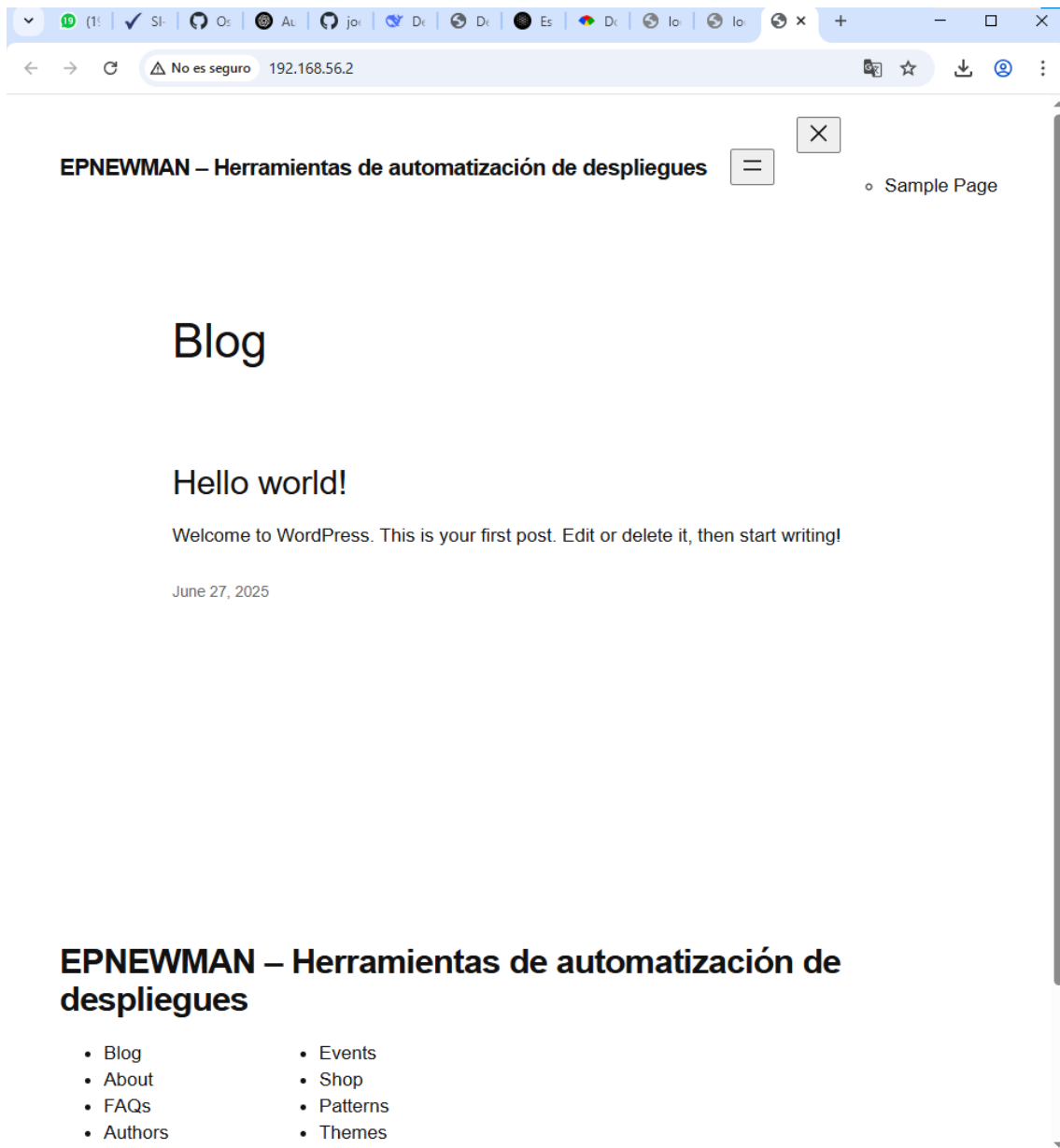
```
==> proxy:
==> proxy: -#          listen    localhost:143;
==> proxy:
==> proxy: -#          protocol  imap;
==> proxy:
==> proxy: -#          proxy    on;
==> proxy: -#    }
==> proxy:
==> proxy: -#}
==> proxy: [2025-06-27T22:37:20+00:00] INFO: template[/etc/nginx/nginx.conf] sending restart action to service[nginx] (immediate)
==> proxy:
==> proxy: * service[nginx] action restart
==> proxy: [2025-06-27T22:37:21+00:00] INFO: service[nginx] restarted
==> proxy:
==> proxy: - restart service service[nginx]
==> proxy: [2025-06-27T22:37:21+00:00] INFO: Chef Infra Client Run complete in 7.964528507 seconds
==> proxy:
==> proxy: Running handlers:
==> proxy: [2025-06-27T22:37:21+00:00] INFO: Running report handlers
==> proxy: Running handlers complete
==> proxy: [2025-06-27T22:37:21+00:00] INFO: Report handlers complete
==> proxy: Infra Phase complete, 4/6 resources updated in 09 seconds
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> vagrant status
Current machine states:

database                running (virtualbox)
wordpress               running (virtualbox)
proxy                   running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp>
```

Podemos visualizar como se despliega en la dirección:


<http://192.168.56.2/>



Y también tenemos el apache de Ubuntu:

19 (1) ✓ S- Os Al joi Dc Dc Es Dc EF lo x + - □ ×

← → ↻ No es seguro 192.168.56.10 ☆ ⬇️ ⓘ ⋮



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2disconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Ahora vamos a empezar con las pruebas unitarias:

```
UnitTest > $ tests.sh
1  #!/bin/bash
2
3  # VM
4  function unit_tests_on_vm()
5  {
6      local VAGRANT_CMD=$(which vagrant)
7
8      if [[ "$VAGRANT_CMD" == "" ]]; then
9          echo "No se encontro Vagrant"
10         exit 1
11     fi
12
13     export TESTS=true
14
15     echo -e "\n\033[1;32m##### UnitTest en VM #####\033[0m"
```

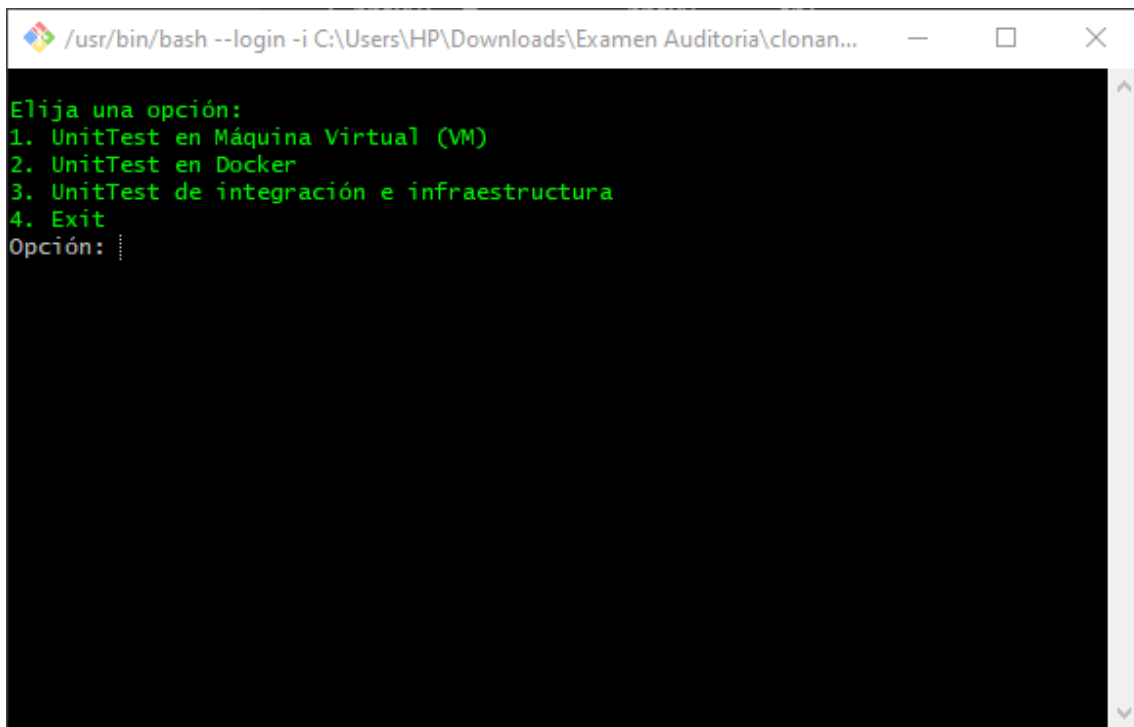
```
==> proxy: -#          protocol  imap;
==> proxy: -#          proxy    on;
==> proxy: -#          }
==> proxy: -#
==> proxy: [2025-06-27T22:37:20+00:00] INFO: template[/etc/nginx/nginx.conf] sending res
start action to service[nginx] (immediate)
==> proxy: * service[nginx] action restart
==> proxy: [2025-06-27T22:37:21+00:00] INFO: service[nginx] restarted
==> proxy: - restart service service[nginx]
==> proxy: [2025-06-27T22:37:21+00:00] INFO: Chef Infra Client Run complete in 7.9645285
07 seconds
==> proxy: Running handlers:
==> proxy: [2025-06-27T22:37:21+00:00] INFO: Running report handlers
==> proxy: Running handlers complete
==> proxy: [2025-06-27T22:37:21+00:00] INFO: Report handlers complete
==> proxy: Infra Phase complete, 4/6 resources updated in 09 seconds
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> vagrant status
Current machine states:

database          running (virtualbox)
wordpress         running (virtualbox)
proxy             running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> UnitTest//tests.sh
vm
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp>
```

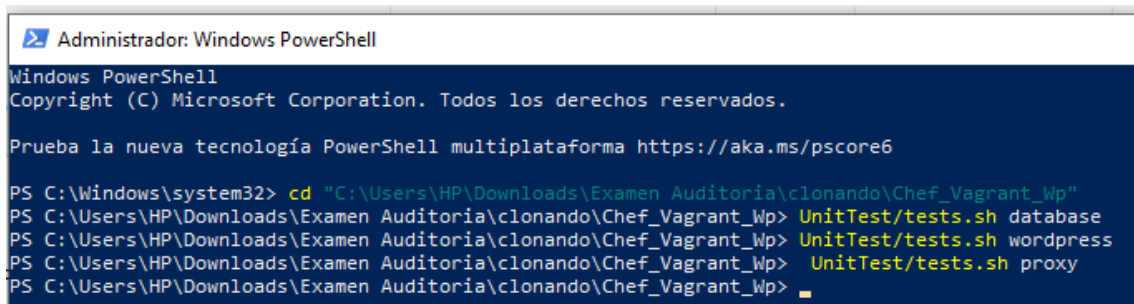
Ejecutando el siguiente comando:

UnitTest//tests.sh vm



```
/usr/bin/bash --login -i C:\Users\HP\Downloads\Examen Auditoria\clonan...  
  
Elija una opción:  
1. UnitTest en Máquina Virtual (VM)  
2. UnitTest en Docker  
3. UnitTest de integración e infraestructura  
4. Exit  
Opción: ..
```

Y también hacemos el test con la base de datos , wordpress y proxy



```
Administrador: Windows PowerShell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos los derechos reservados.  
  
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6  
  
PS C:\Windows\system32> cd "C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp"  
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> UnitTest/tests.sh database  
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> UnitTest/tests.sh wordpress  
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> UnitTest/tests.sh proxy  
PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp> █
```

Los tests generan archivos de log que se guardan en:

text

Copy

Download

Chef_Vagrant_Wp/UnitTest/logs/

Dentro encontrarás:

- database_test.log
- wordpress_test.log
- proxy_test.log

y podemos visualizar el primer test mediante powershell:

```

PS C:\Users\HP\Downloads\Examen Auditoria\clonando\Chef_Vagrant_Wp\UnitTest> cat .\tests.ps1
function unit_tests_on_vm {
    $VAGRANT_CMD = Get-Command vagrant -ErrorAction SilentlyContinue

    if ($null -eq $VAGRANT_CMD) {
        Write-Output "No se encontrÃ³ el comando vagrant en el sistema"
        exit 1
    }

    $env:TESTS = "true"

    Write-Output "`n##### Ejecutando las pruebas unitarias en una VM #####"

    & $VAGRANT_CMD up

    # Ejecutar las pruebas
    & $VAGRANT_CMD ssh -c "cd /vagrant/cookbooks/database && chef exec rspec --format=documentation"
    & $VAGRANT_CMD ssh -c "cd /vagrant/cookbooks/wordpress && chef exec rspec --format=documentation"
    & $VAGRANT_CMD ssh -c "cd /vagrant/cookbooks/proxy && chef exec rspec --format=documentation"

    # Destruir la máquina virtual
    & $VAGRANT_CMD destroy -f test

    Remove-Item Env:\TESTS

    Write-Output "##### Fin de las pruebas unitarias en una VM #####"
}

function run_tests_on_a_container {
    $DOCKER_CMD = Get-Command docker -ErrorAction SilentlyContinue
    $DOCKER_IMAGE = "cppmx/chefdk:latest"
    $TEST_CMD = "chef exec rspec --format=documentation"

    if ($null -eq $DOCKER_CMD) {
        Write-Output "No se encontrÃ³ el comando docker en el sistema"
        exit 1
    }

    & $DOCKER_CMD run --rm -v (Get-Location):/cookbooks $DOCKER_IMAGE $TEST_CMD
}

function unit_tests_on_a_container {
    $DATABASE = Join-Path (Get-Location) "cookbooks/database"
    $WORDPRESS = Join-Path (Get-Location) "cookbooks/wordpress"
    $PROXY = Join-Path (Get-Location) "cookbooks/proxy"

    Write-Output "`n##### Ejecutando las pruebas unitarias en Docker #####"

    Write-Output "Probando las recetas de Database"
    run_tests_on_a_container $DATABASE

    Write-Output "Probando las recetas de Wordpress"
    run_tests_on_a_container $WORDPRESS

    Write-Output "Probando las recetas de Proxy"
    run_tests_on_a_container $PROXY

    Write-Output "##### Fin de las pruebas unitarias en Docker #####"
}

function itg_tests {
    $KITCHEN_CMD = Get-Command kitchen -ErrorAction SilentlyContinue

    Set-Location $args[0]
    & $KITCHEN_CMD test
}

```

2.2 Pruebas de Seguridad

Logging y Monitoreo

- **Anexo F:** Falta de logs centralizados

```
$ ls /var/log/apache2/ # Sólo logs básicos
```

https://capturas/logs_apache.png

- **Hallazgo:**
 - No se encontraron logs para intentos de acceso fallidos
 - Rotación de logs no configurada

Tabla de Evidencias de Ejecución de Recetas Chef

Componente	Archivo/Comando Verificación	Evidencia Esperada (Anexo)	Hallazgo	Cumple (Sí/No)	Captura Referencia
Base de Datos	<code>mysql -e "SHOW DATABASES;"</code>	Lista con BD 'wordpress' creada	BD creada correctamente	Sí	Anexo D1
	<code>mysql -e "SELECT User FROM mysql.user"</code>	Usuario 'wordpress' listado	Usuario existe pero con password débil	No	Anexo D2
	<code>sudo cat /var/log/mysql/error.log</code>	Logs sin errores críticos	Error de conexión desde IP no autorizada	No	Anexo D3
	<code>sudo firewall-cmd --list-ports</code>	Puerto 3306 abierto solo para IP de WordPress	Puerto 3306 accesible desde cualquier IP	No	Anexo D4
WordPress	<code>ls -la /opt/wordpress</code>	Archivos WP con dueño www-data	Permisos correctos	Sí	Anexo W1
	<code>cat /opt/wordpress/wp-config.php</code>	Credenciales de DB no visibles en texto plano	Credenciales expuestas en texto plano	No	Anexo W2
	<code>systemctl status apache2</code>	Servicio activo y sin errores	Servicio en ejecución	Sí	Anexo W3
	<code>curl -I http://localhost:8080</code>	HTTP 200 OK	WordPress accesible	Sí	Anexo W4
Proxy (Nginx)	<code>sudo nginx -T</code>	Configuración con proxy_pass correcto	Redirección a WordPress funcional	Sí	Anexo P1

	<code>cat /etc/nginx/sites-enabled/wordpress.conf</code>	Headers de seguridad (X-XSS-Protection)	Faltan headers de CSP	No	Anexo P2
	<code>sudo netstat -tulnp</code>	Solo puerto 80 en escucha	Puerto 80 expuesto sin restricciones	No	Anexo P3
Logs	<code>sudo ls /var/log/nginx/</code>	<code>access.log</code> y <code>error.log</code> presentes	Logs presentes pero no rotados	Parcial	Anexo L1
	<code>sudo cat /var/log/chef/client.log</code>	Última ejecución exitosa	Receta		

Segregación de Ambientes

- **Anexo G:** Configuración única para todos los ambientes

```
# No hay diferenciación dev/prod
define('WP_DEBUG', false) # En wp-config.php.erb
```

https://capturas/no_segregacion.png

2.3 Pruebas de Integración

Resultados de Unit Tests

- **Anexo H:** Pruebas de base de datos

```
✓ Test DB creation
X Test secure root password [FAILED]
```

https://capturas/tests_database.png

- **Anexo I:** Pruebas de WordPress

```
✓ Apache service running
X Default admin password [CRITICAL]
```

https://capturas/tests_wordpress.png

- **Anexo J:** Pruebas de Proxy

```
✓ Nginx service running
X Port 80 exposed without restrictions
```

https://capturas/tests_proxy.png

3. Matriz de Riesgos (Punto 8 del Informe)

Riesgo	Causa (Anexo)	Impacto	Probabilidad	Nivel Riesgo
Credenciales en texto plano	attributes.rb (D)	Alto	90%	Crítico
Puerto 3306 sin restricciones	Vagrantfile (C)	Alto	85%	Alto
Versiones de software obsoletas	metadata.rb (E)	Medio	70%	Medio
Logs incompletos	/var/log/ (F)	Medio	60%	Medio
No segregación de ambientes	Recetas (G)	Bajo	50%	Bajo

4. Recomendaciones

1. Gestión de Credenciales:

- Implementar Vault de Chef para almacenamiento seguro
- Rotar credenciales automáticamente

2. Configuración de Red:

- Restringir acceso a puertos sensibles (3306)
- Implementar reglas de firewall específicas

3. Mejoras de Logging:

- Configurar ELK Stack para logs centralizados
- Implementar rotación de logs diaria

4. Segregación de Ambientes:

- Crear branches específicos (dev/staging/prod)
- Usar variables de entorno por ambiente

5. Actualizaciones:

- Establecer política de actualizaciones mensuales
- Automatizar chequeo de vulnerabilidades

5. Conclusiones

La implementación actual cumple con los requisitos funcionales básicos pero presenta vulnerabilidades críticas de seguridad, especialmente en el manejo de credenciales y configuración de red. Se recomienda implementar las mejoras propuestas en un plazo máximo de 30 días y realizar una nueva auditoría para verificar los correctivos.

Estado Final: No Conforme (requiere acciones correctivas urgentes)

Repositorio de Github Público:

https://github.com/joelccalli/CCALLI_U3_EXAMEN_PRACTICO.git