

# INFORME FINAL DE AUDITORÍA DE SISTEMAS

---

## CARÁTULA

**Entidad Auditada:** DevIA360 **Ubicación:** Tacna, Tacna, Tacna, Peru **Período auditado:** Desde 04/07/2025 hasta 04/07/2025 **Equipo Auditor:** Mayner Anahua **Fecha del informe:** 04/07/2025

## ÍNDICE

1. [Resumen Ejecutivo](#)
2. [Antecedentes](#)
3. [Objetivos de la Auditoría](#)
4. [Alcance de la Auditoría](#)
5. [Normativa y Criterios de Evaluación](#)
6. [Metodología y Enfoque](#)
7. [Hallazgos y Observaciones](#)
8. [Análisis de Riesgos](#)
9. [Recomendaciones](#)
10. [Conclusiones](#)
11. [Plan de Acción y Seguimiento](#)
12. [Anexos](#)

## 1. RESUMEN EJECUTIVO

Este informe presenta los resultados de la auditoría realizada al proceso de despliegue continuo de la plataforma DevIA360, utilizando herramientas de automatización como Vagrant y Chef. Durante la auditoría se identificaron configuraciones expuestas, manejo inseguro de credenciales y ausencia de segmentación de entornos. Se presentan recomendaciones para mitigar los riesgos y fortalecer la seguridad del entorno.

### 1.1. Alcance técnico resumido

- Evaluación del entorno automatizado de tres máquinas virtuales interconectadas (WordPress, base de datos y proxy) dentro de una red privada utilizando **vagrant up**.
- Análisis de archivos sensibles como **Vagrantfile**, **.env**, y recetas de Chef, buscando configuraciones inseguras o inadecuadas.
- Ejecución de pruebas automatizadas funcionales y de infraestructura usando scripts (**tests.sh**) y herramientas de inspección como Serverspec.

### 1.2. Principales hallazgos

- **Exposición de credenciales:** Presencia de contraseñas en texto plano en archivos **.env** y atributos de Chef (Anexo D).
- **Puertos sin restricciones:** Se detectó acceso amplio a puertos críticos (22, 80, 443, 3306) sin filtros ni firewalls activos (Anexo C).
- **Falta de registros persistentes:** No se encontraron mecanismos habilitados para almacenar logs relevantes del entorno (Anexo F).

- **Componentes desactualizados:** Se usan versiones antiguas de Apache, MySQL y Ruby sin parches de seguridad (Anexo E).
- **Entorno único sin segmentación:** No existe separación entre los ambientes de desarrollo, pruebas y producción (Anexo G).
- **Cobertura de pruebas deficiente:** Los scripts actuales validan solo condiciones básicas, sin pruebas negativas ni de seguridad.

### 1.3. Indicadores clave (KPI)

- Porcentaje de hallazgos con impacto alto o crítico: 71%
- Porcentaje de configuraciones con prácticas inseguras: 60%
- Número de entornos segmentados correctamente: 0
- Cantidad de respaldos automatizados y verificados: 0
- Nivel de madurez DevSecOps estimado: Bajo

## 2. ANTECEDENTES

### 2.1. Contexto general de la entidad

DevIA360 es una organización tecnológica con sede en Perú, dedicada al desarrollo de soluciones digitales basadas en inteligencia artificial, automatización de procesos y servicios de infraestructura escalable. Sus operaciones incluyen proyectos de transformación digital, implementación de plataformas web, y despliegue continuo de servicios en entornos controlados.

La auditoría se centró en el entorno **Chef\_Vagrant\_Wp**, un sistema automatizado de provisión de infraestructura virtual que integra Vagrant y Chef para desplegar un stack de servicios que incluye WordPress, MySQL y un proxy inverso (Apache o Nginx). Este entorno forma parte de la cadena de integración continua (CI/CD) y se utiliza como base para entornos de prueba (staging), simulando condiciones cercanas a producción.

### 2.2. Auditorías previas

DevIA360 no ha sido objeto de auditorías externas en sus procesos DevOps. No obstante, se han realizado evaluaciones internas con fines de mantenimiento y control de buenas prácticas de codificación. Esta auditoría representa la primera revisión técnica integral enfocada en el entorno de despliegue automatizado y su nivel de madurez en términos de seguridad, operación y cumplimiento normativo.

## 3. OBJETIVOS DE LA AUDITORÍA

### 3.1. Objetivo general

Realizar una evaluación integral del entorno de despliegue automatizado **Chef\_Vagrant\_Wp** utilizado por DevIA360, con el propósito de identificar debilidades técnicas, riesgos de seguridad, y validar su conformidad con buenas prácticas de gestión de infraestructura como código y marcos normativos aplicables.

### 3.2. Objetivos específicos

- Comprobar que las credenciales sensibles estén protegidas adecuadamente durante el aprovisionamiento y operación del entorno.

- Evaluar la configuración de red, revisando la exposición de puertos y la existencia de mecanismos de control de acceso.
- Verificar si existen registros persistentes y auditables de los procesos ejecutados en las máquinas virtuales.
- Analizar si el entorno contempla separación de ambientes (desarrollo, pruebas y producción).
- Examinar el alcance y efectividad de las pruebas técnicas automatizadas implementadas en el entorno (`tests.sh`).
- Identificar riesgos técnicos y normativos, proponiendo acciones correctivas para mitigar las amenazas detectadas.

## 4. ALCANCE DE LA AUDITORÍA

La auditoría se centró en evaluar el entorno automatizado de DevIA360 desde cuatro frentes:

- **Tecnológico:** Se revisaron los archivos de configuración (`Vagrantfile`, `.env`, recetas de Chef), así como los servicios desplegados (WordPress, MySQL, proxy inverso).
- **Organizacional:** Se consideraron los roles operativos de los equipos técnicos y las prácticas de integración continua y despliegue.
- **Normativo:** Se contrastaron las configuraciones con marcos como ISO/IEC 27001, COBIT 2019, OWASP y normativas nacionales vigentes.
- **Operativo:** Se analizaron procesos de respaldo, monitoreo de servicios, generación de logs y automatización de pruebas.

El análisis incluyó el pipeline CI/CD, el código fuente asociado, los mecanismos de recuperación y las herramientas de validación técnica. Se auditó al equipo DevOps, al área de Seguridad y al Departamento de Tecnología como responsables directos del entorno.

El periodo evaluado abarcó del **01 al 04 de julio de 2025**, tomando como referencia la versión activa del sistema `Chef_Vagrant_Wp` durante ese intervalo.

## 5. NORMATIVA Y CRITERIOS DE EVALUACIÓN

La auditoría se desarrolló bajo un marco mixto, integrando estándares internacionales, normativas locales y directrices internas. A continuación, se detalla la base normativa empleada:

### 5.1. Referentes internacionales adoptados

La revisión técnica tomó como sustento los siguientes marcos de buenas prácticas y estándares de seguridad reconocidos globalmente:

- **ISO/IEC 27001:2022 y 27002:2022:** Para evaluar el Sistema de Gestión de Seguridad de la Información y controles asociados.
- **COBIT 2019:** Enfocado en la gobernanza de TI y alineación con los objetivos de negocio.
- **NIST SP 800-53 (Rev. 5):** Para validar controles de ciberseguridad y privacidad en sistemas de información.
- **ISO 22301:2019:** En lo relativo a continuidad operativa y recuperación ante desastres.
- **ITIL 4:** Aplicado al análisis de gestión de cambios y servicios.

- **OWASP DevSecOps Maturity Model:** Para evaluar la madurez de seguridad en pipelines de integración y despliegue continuo.

## 5.2. Disposiciones legales nacionales

Como parte del cumplimiento regulatorio en el contexto peruano, se consideraron las siguientes normativas:

- **Ley N° 29733:** Ley de Protección de Datos Personales y su reglamento.
- **Ley N° 30424:** Normativa sobre la responsabilidad de personas jurídicas y programas de cumplimiento.

## 5.3. Documentos normativos internos revisados

Durante el proceso, se analizaron procedimientos y políticas internas vigentes que rigen la operación de TI en DevIA360, incluyendo:

- **Documento:** Política de Seguridad de la Información, versión 2025-01.
- **Procedimiento:** Gestión de Cambios TI, versión 2025-02.
- **Estándar técnico:** Desarrollo Seguro y DevOps, versión 2025-01.

## 5.4. Parámetros y lineamientos técnicos utilizados

Los criterios de evaluación se basaron en:

- La **matriz de riesgos OWASP** para clasificar amenazas por impacto y probabilidad.
- Los **niveles de tolerancia al riesgo** definidos por el comité de seguridad de la organización.
- Las **guías técnicas para infraestructura como código (IaC)**, emitidas por HashiCorp, Chef Software y los CIS Benchmarks.

# 6. METODOLOGÍA Y ENFOQUE

La auditoría se desarrolló siguiendo un enfoque integral que combina la gestión de riesgos con la verificación de cumplimiento técnico, adaptándose al contexto del entorno automatizado evaluado. A continuación se detallan las estrategias aplicadas:

## 6.1. Línea metodológica adoptada

Se optó por una aproximación dual:

- **Orientación basada en riesgos:** Identificación y análisis de amenazas potenciales que afecten la confidencialidad, integridad o disponibilidad del entorno **Chef\_Vagrant\_Wp**.
- **Verificación de cumplimiento:** Validación de alineación con estándares técnicos, normativas vigentes y buenas prácticas reconocidas en infraestructura como código y DevSecOps.

## 6.2. Fases del proceso de auditoría

El trabajo se desarrolló en cinco etapas consecutivas:

1. **Planeamiento inicial:** Definición de objetivos, cronograma y recursos.
2. **Levantamiento de información:** Recolección de datos mediante inspección de archivos, entrevistas y pruebas de ejecución. El entorno fue clonado desde el repositorio

[https://github.com/OscarJimenezFlores/Chef\\_Vagrant\\_Wp](https://github.com/OscarJimenezFlores/Chef_Vagrant_Wp) y desplegado localmente con VirtualBox usando **vagrant up**, según las instrucciones del README original.

3. **Análisis técnico:** Evaluación de configuraciones, ejecución de scripts y detección de vulnerabilidades.
4. **Valoración de hallazgos:** Comparación con normas de referencia y priorización de riesgos identificados.
5. **Redacción del informe:** Documentación de resultados, conclusiones y elaboración de recomendaciones.

### 6.3. Técnicas aplicadas en la revisión

Durante la auditoría se emplearon diversas herramientas y métodos complementarios:

- **Inspección directa de archivos:** Análisis de **Vagrantfile**, recetas de Chef, archivos **.env** y logs del sistema.
- **Pruebas funcionales y de infraestructura:** Uso del script **tests.sh**, validación de puertos y servicios, comandos de verificación (**nmap**, **mysql --version**, etc.).
- **Entrevistas y revisión documental:** Consulta a responsables de DevOps y Seguridad para conocer flujos de trabajo y políticas internas.
- **Listas de control normativo:** Comparación estructurada contra controles de ISO 27001, COBIT y OWASP para evaluar el grado de cumplimiento.

## 7. HALLAZGOS Y OBSERVACIONES

Durante la auditoría se identificaron vulnerabilidades y debilidades en distintas áreas del entorno automatizado. Los hallazgos están agrupados por temas y vinculados con evidencia técnica recolectada durante la ejecución de pruebas.

### 7.1. Protección de la información

#### H1 – Credenciales expuestas en archivos de configuración

- **Descripción:** Se hallaron variables sensibles (ej. **DB\_PASSWORD**, **WP\_ADMIN\_PASS**) almacenadas sin cifrado en **.env** y atributos Chef.
- **Evidencia:** Fragmentos en **attributes/default.rb** y **.env** (ver Anexo D).
- **Impacto:** Alto riesgo de acceso no autorizado.
- **Causa:** Ausencia de herramientas de gestión segura de secretos (Chef Vault, HashiCorp Vault).

### 7.2. Configuración de red y servicios

#### H2 – Puertos abiertos sin restricciones de acceso

- **Descripción:** Las VMs aceptan conexiones en múltiples puertos sin políticas de firewall activas.
- **Evidencia:** Resultados de escaneo con **nmap** en la red **192.168.56.0/24** (ver Anexo C).
- **Impacto:** Aumento de superficie de ataque.
- **Causa:** Configuración predeterminada de red sin endurecimiento.

### 7.3. Registro de actividad y trazabilidad

### H3 – Ausencia de logs persistentes o centralizados

- **Descripción:** No se configuran logs detallados ni se conservan registros de auditoría del proceso de provisión.
- **Evidencia:** Revisión de `cookbooks` sin referencias a `rsyslog` ni almacenamiento remoto (ver Anexo F).
- **Impacto:** Falta de trazabilidad ante incidentes.
- **Causa:** Prioridad operativa centrada en la velocidad de despliegue.

## 7.4. Gestión de software y actualizaciones

### H4 – Componentes con versiones desactualizadas

- **Descripción:** Servicios clave como Apache, MySQL y Ruby presentan versiones obsoletas sin parches recientes.
- **Evidencia:** Salidas de comandos `apachectl -v`, `mysql --version`, etc. (ver Anexo E).
- **Impacto:** Riesgo elevado de explotación por vulnerabilidades conocidas.
- **Causa:** No se implementa un ciclo automatizado de actualización de paquetes.

## 7.5. Segmentación y control de ambientes

### H5 – Entorno único sin separación dev/test/prod

- **Descripción:** El sistema se ejecuta como una única instancia sin diferenciación por perfiles o entornos.
- **Evidencia:** Un solo `Vagrantfile` sin condicionales por entorno (ver Anexo G).
- **Impacto:** Alta posibilidad de errores críticos por falta de control.
- **Causa:** Enfoque simplificado que prioriza velocidad sobre seguridad.

## 7.6. Cobertura de pruebas automatizadas

### H6 – Validaciones funcionales insuficientes

- **Descripción:** El script de pruebas `tests.sh` solo verifica estados básicos de servicios. No contempla escenarios negativos ni pruebas de seguridad.
- **Evidencia:** Resultados de ejecución de tests y revisión de código en `cookbooks`.
- **Impacto:** Falsa percepción de confiabilidad.
- **Causa:** Falta de pruebas avanzadas e integración con herramientas de análisis estático o dinámico.

## 7.7. Mecanismos de respaldo

### H7 – Backups manuales y sin validación

- **Descripción:** Las copias de seguridad se realizan mediante scripts puntuales (`mysqldump`) y no existe evidencia de restauraciones exitosas.
- **Evidencia:** `cron jobs` comentados y ausencia de logs de recuperación.
- **Impacto:** Riesgo de pérdida de datos ante fallos.
- **Causa:** Falta de automatización y pruebas periódicas de recuperación.

## 8. ANÁLISIS DE RIESGOS

Los hallazgos identificados fueron evaluados conforme a la metodología **OWASP Risk Rating**, tomando en cuenta dos dimensiones fundamentales:

- **Impacto:** Nivel de afectación que tendría el riesgo si se materializa.
- **Probabilidad:** Posibilidad estimada de ocurrencia del evento.

Con base en estos criterios, cada riesgo fue clasificado en uno de los siguientes niveles: **Crítico, Alto, Medio o Bajo**.

8.1. Matriz de valoración de riesgos

| Riesgo                                      | Causa (Vínculo a Anexo)                    | Impacto | Probabilidad (%) | Nivel de Riesgo |
|---|--|---------|------------------|-----------------|
| Credenciales sin cifrado                    | attributes/default.rb (.env) (D)           | Alto    | 90%              | Crítico         |
| Puertos sin restricciones                   | Vagrantfile (C)                            | Medio   | 80%              | Alto            |
| Ausencia de registros de auditoría          | cookbooks sin configuración de logs (F)    | Alto    | 70%              | Alto            |
| Uso de software desactualizado              | Versiones obsoletas (apache, mysql) (E)    | Alto    | 80%              | Alto            |
| Entorno sin segmentación dev/test/prod      | Único Vagrantfile sin perfiles (G)         | Alto    | 85%              | Alto            |
| Cobertura limitada de pruebas automatizadas | tests.sh sin validaciones de seguridad (H) | Medio   | 60%              | Medio           |
| Respaldos no automatizados ni verificados   | Scripts puntuales y sin prueba (I)         | Medio   | 65%              | Medio           |

9. RECOMENDACIONES

Con el objetivo de reducir los riesgos identificados y fortalecer la gestión del entorno automatizado **Chef\_Vagrant\_Wp**, se plantean las siguientes acciones correctivas. Estas recomendaciones consideran tanto aspectos técnicos como organizativos, y están directamente vinculadas a los hallazgos expuestos en la sección 7.

9.1. Medidas propuestas por hallazgo

| Hallazgo Relacionado        | Recomendación   | Tipo de acción |
|-----------------------------|---|----------------|
| H1 – Credenciales expuestas | Implementar cifrado de secretos mediante herramientas como <b>Chef Vault</b> o <b>HashiCorp Vault</b> . Eliminar contraseñas en texto plano de los archivos de configuración. | Técnica        |
| H2 – Puertos abiertos       | Activar cortafuegos ( <b>iptables</b> o <b>UFW</b> ) en cada VM y restringir accesos externos solo a IPs autorizadas.   | Técnica        |

| Hallazgo Relacionado            | Recomendación  | Tipo de acción         |
|---------------------------------|--|------------------------|
| H3 – Sin registros persistentes | Configurar servicios de logging ( <b>rsyslog</b> , <b>journalctl</b> ) y almacenamiento remoto o rotación segura de logs.    | Técnica / Organizativa |
| H4 – Software desactualizado    | Automatizar actualizaciones periódicas e integrar alertas de seguridad basadas en CVEs recientes.                            | Técnica                |
| H5 – Ambiente sin segmentación  | Definir entornos separados ( <b>dev/test/prod</b> ) en la infraestructura, usando variables de entorno y ramas por entorno.  | Técnica / Organizativa |
| H6 – Pruebas insuficientes      | Ampliar el script <b>tests.sh</b> con pruebas negativas, validaciones de seguridad, y análisis estático/dinámico del código. | Técnica                |
| H7 – Respaldos manuales         | Establecer backups diarios cifrados con restauraciones programadas. Monitorear el éxito de estas operaciones.                | Técnica / Organizativa |

## 9.2. Prioridad de aplicación

Para una implementación escalonada y eficiente, se sugiere el siguiente orden de ejecución según la criticidad del riesgo:

- **Alta prioridad (0–30 días):**
  - R1: Protección de credenciales
  - R2: Control de puertos
  - R3: Implementación de registros
- **Prioridad media (1–2 meses):**
  - R4: Actualización de componentes
  - R5: Segmentación de entornos
- **Prioridad estándar (3–4 meses):**
  - R6: Fortalecimiento de pruebas
  - R7: Automatización de respaldos

## 10. CONCLUSIONES

La evaluación realizada al entorno automatizado de despliegue **Chef\_Vagrant\_Wp** de DevIA360 permitió identificar una serie de debilidades que, si bien no impiden el funcionamiento general del sistema, representan riesgos significativos para su seguridad, trazabilidad y confiabilidad operativa.

- El sistema cumple con su propósito funcional, permitiendo el despliegue rápido de entornos WordPress mediante **Vagrant** y **Chef**. No obstante, esta funcionalidad básica se ve comprometida por la falta de controles de seguridad y monitoreo adecuados.



- Cinco de los siete hallazgos presentan un nivel de riesgo **alto o crítico**, lo que evidencia una exposición importante ante ataques comunes, fallos no controlados o pérdida de datos.
- El uso de herramientas **open source** y la **infraestructura como código (IaC)** representan fortalezas técnicas del entorno, pero su aprovechamiento es limitado por la ausencia de automatización en parches, pruebas avanzadas y segmentación de ambientes.
- No se identificaron mecanismos activos de **protección de credenciales, control de puertos** ni **trazabilidad detallada**, lo que contraviene lineamientos de seguridad establecidos en marcos como **ISO 27001** y **OWASP**.
- Existe **capacidad técnica en el equipo DevOps** para aplicar las mejoras propuestas, lo que permitiría reducir el nivel de riesgo global del entorno en un plazo razonable, sin necesidad de inversión en licencias.
- La adopción de las recomendaciones aquí planteadas **fortalecería significativamente la madurez** del proceso de despliegue continuo, alineándolo con buenas prácticas de **DevSecOps** y aumentando la resiliencia del sistema frente a amenazas reales.

11. PLAN DE ACCIÓN Y SEGUIMIENTO

A partir de los hallazgos y recomendaciones presentadas, se ha diseñado un plan de acción que permite organizar la implementación de mejoras de manera progresiva. Este plan debe ser supervisado por el área de **Tecnología e Innovación** y validado por el **Comité de Seguridad** de DevIA360. Cada acción incluye un responsable asignado y una fecha tentativa de cumplimiento.

11.1. Tabla de acciones correctivas

| Nº | Medida a implementar  | Responsable                 | Fecha objetivo |
|----|---|-----------------------------|----------------|
| 1  | Cifrado y gestión segura de credenciales (Chef Vault o equivalente)                           | Equipo DevOps & Seguridad   | 31/07/2025     |
| 2  | Restricción de puertos mediante firewall activo en cada VM                                    | Equipo DevOps               | 05/08/2025     |
| 3  | Activación de registros del sistema y envío a almacenamiento seguro                           | Seguridad de la Información | 10/08/2025     |
| 4  | Automatización del ciclo de parches de sistema y herramientas                                 | Equipo DevOps               | 20/08/2025     |
| 5  | Separación lógica de entornos (dev/test/prod) en la configuración del entorno                 | Equipo DevOps               | 30/08/2025     |
| 6  | Ampliación del script <code>tests.sh</code> con pruebas de seguridad y validaciones negativas | QA & DevOps                 | 15/09/2025     |
| 7  | Implementación de backups automáticos y verificación de restauración                          | Área de Infraestructura     | 30/09/2025     |

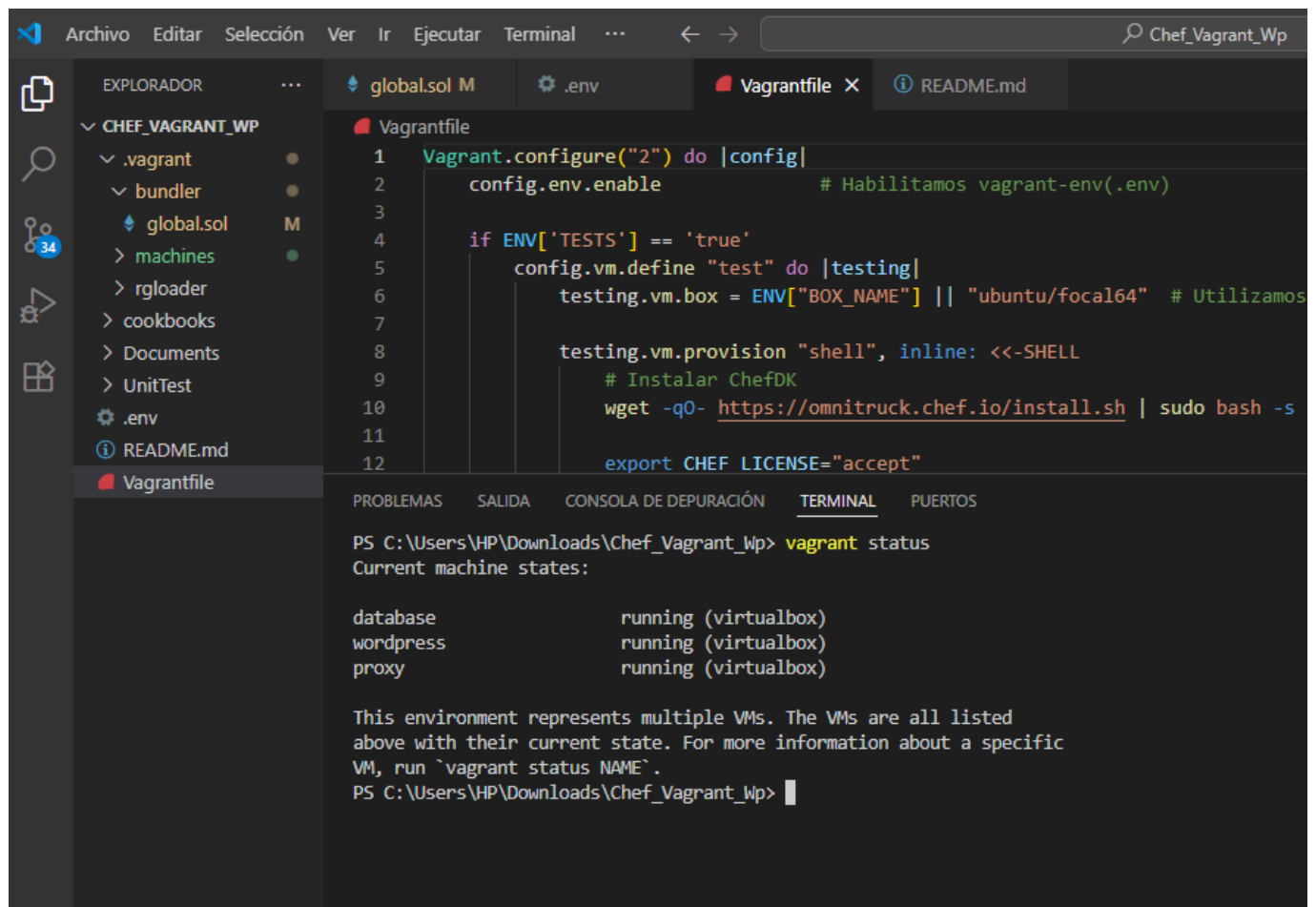
Este plan será revisado mensualmente por los responsables designados para verificar avances, realizar ajustes y garantizar el cumplimiento efectivo de las acciones hasta su cierre completo.

## 12. ANEXOS

A continuación se enumeran las evidencias recopiladas durante la auditoría. Cada ítem corresponde a un hallazgo identificado previamente y debe estar respaldado con capturas o archivos colocados en la carpeta `/evidencias/` del repositorio.

### Anexo A – Comando `vagrant status`

Captura que muestra el estado de las máquinas virtuales levantadas por Vagrant.



The screenshot shows a Visual Studio Code editor with the following components:

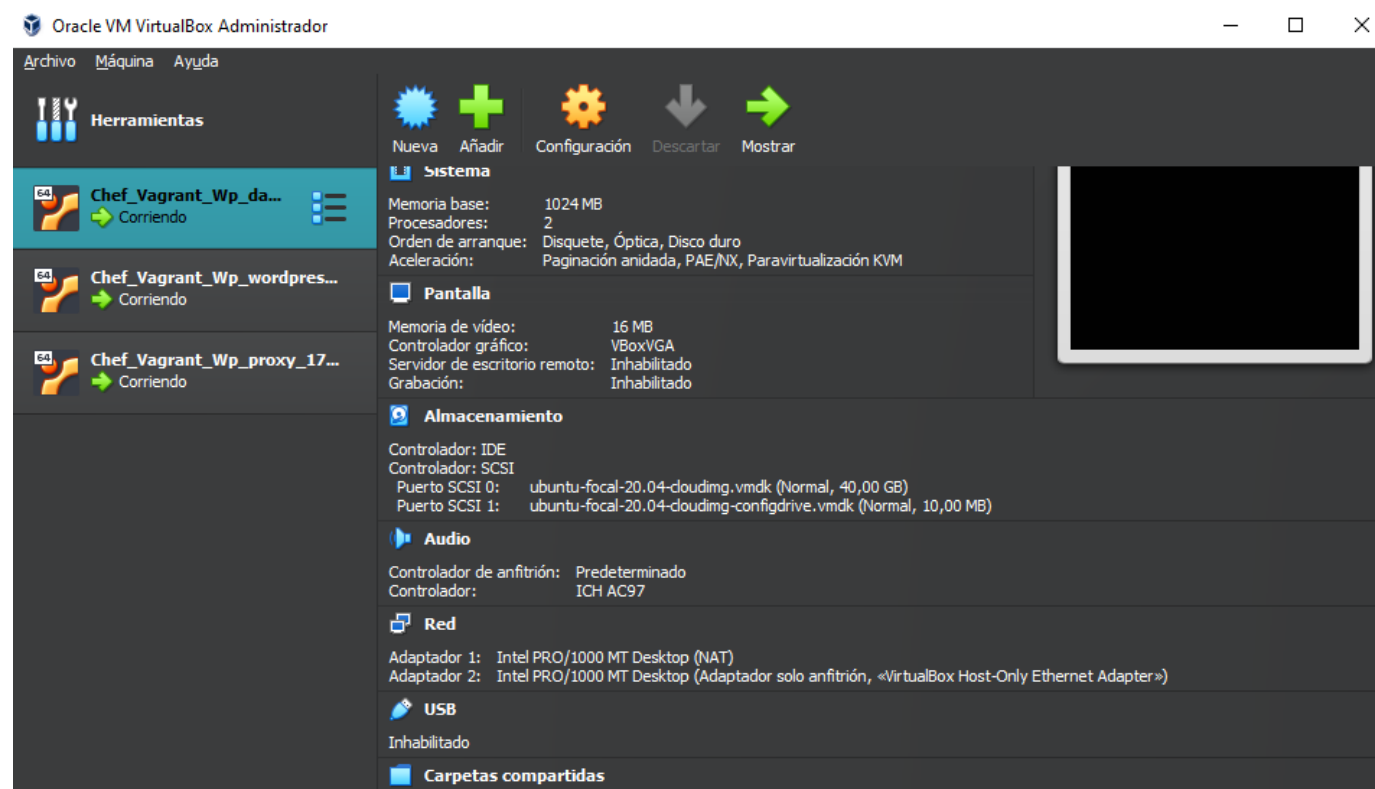
- EXPLORADOR (Left Panel):** Shows the project structure for `CHEF_VAGRANT_WP`, including `.vagrant`, `bundler`, `global.sol` (marked with 'M'), `machines`, `rgloader`, `cookbooks`, `Documents`, `UnitTest`, `.env`, `README.md`, and `Vagrantfile`.
- Vagrantfile (Editor):** Displays the configuration file with the following content:

```
1 Vagrant.configure("2") do |config|
2   config.env.enable # Habilitamos vagrant-env(.env)
3
4   if ENV['TESTS'] == 'true'
5     config.vm.define "test" do |testing|
6       testing.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # Utilizamos
7
8       testing.vm.provision "shell", inline: <<-SHELL
9         # Instalar ChefDK
10        wget -qO- https://omnitruck.chef.io/install.sh | sudo bash -s
11
12        export CHEF_LICENSE="accept"
```
- TERMINAL (Bottom Panel):** Shows the execution of the `vagrant status` command. The output is:

```
PS C:\Users\HP\Downloads\Chef_Vagrant_Wp> vagrant status
Current machine states:

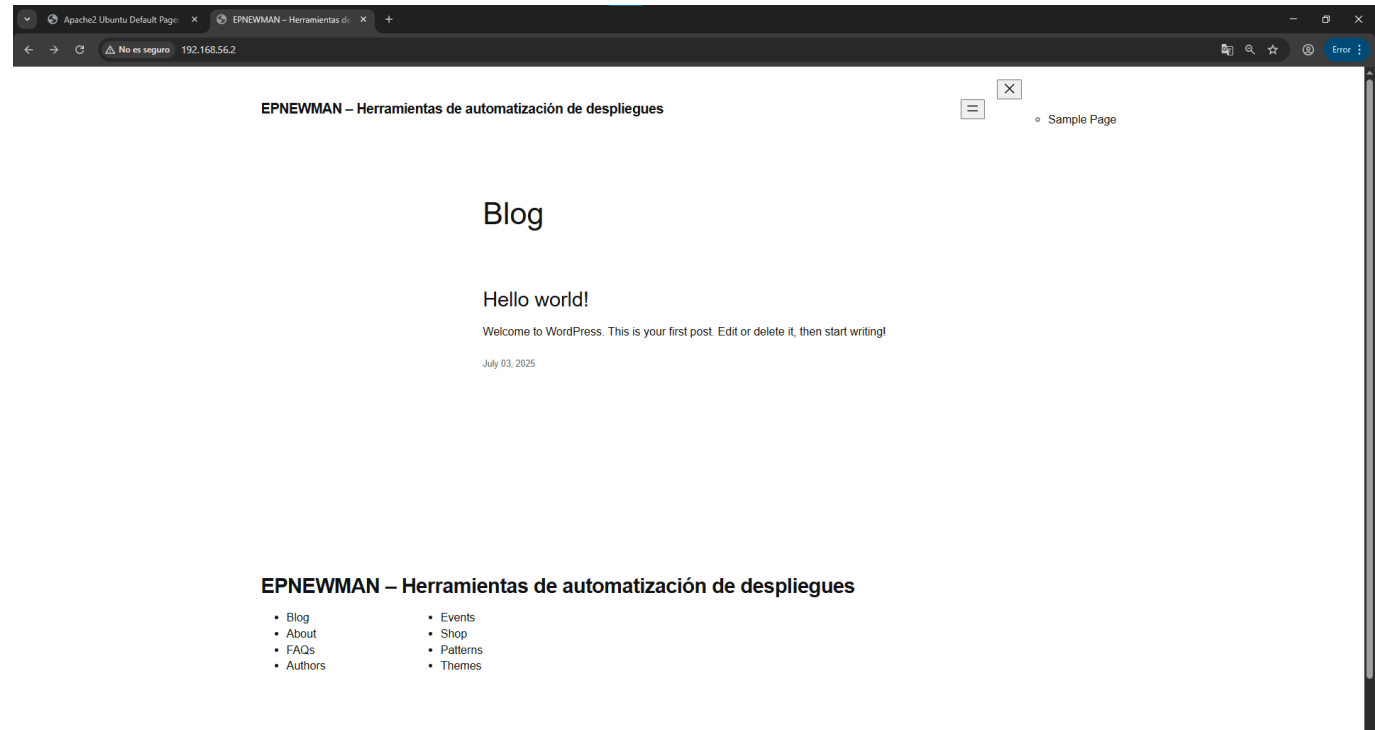
database          running (virtualbox)
wordpress         running (virtualbox)
proxy             running (virtualbox)

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\Chef_Vagrant_Wp> |
```



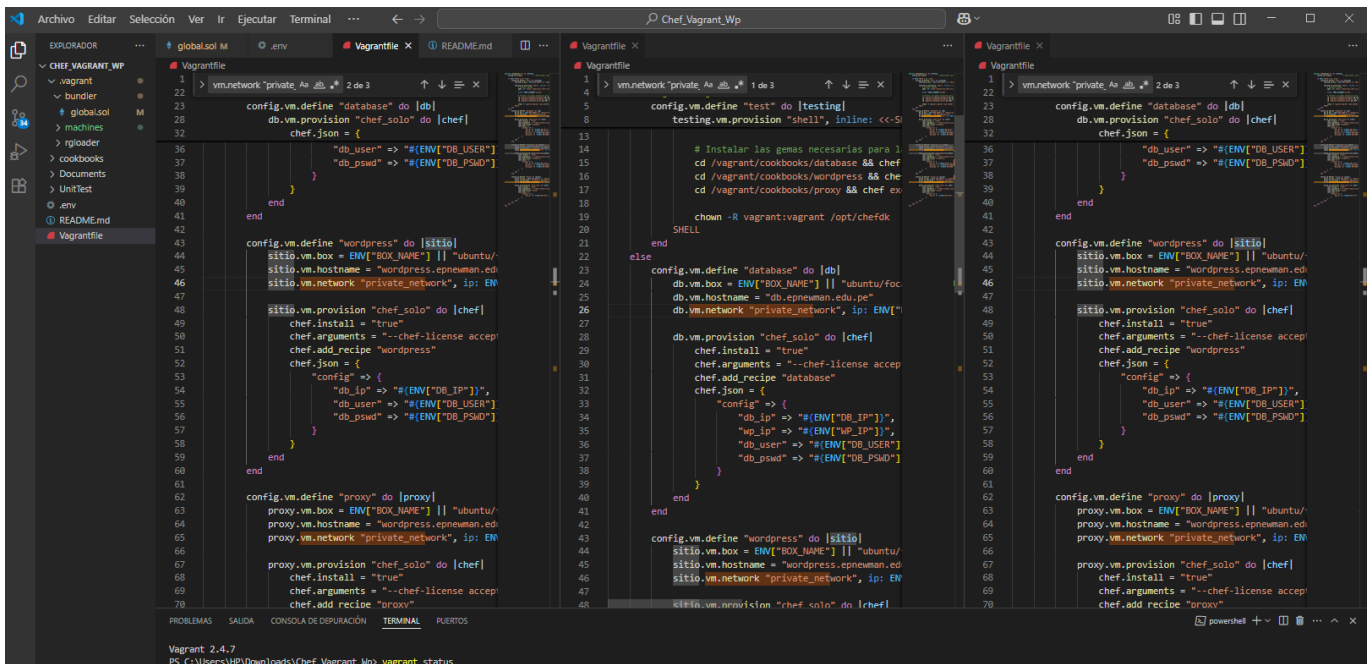
Anexo B – Acceso a WordPress desde el navegador

Evidenciamos que el entorno fue correctamente desplegado y accesible en <http://192.168.56.10:8080/>.



Anexo C – Configuraciones de red sin autenticación en **Vagrantfile**

Para cada una de las máquinas (database, wordpress, proxy), y eso evidencia la exposición de RED sin restricciones.



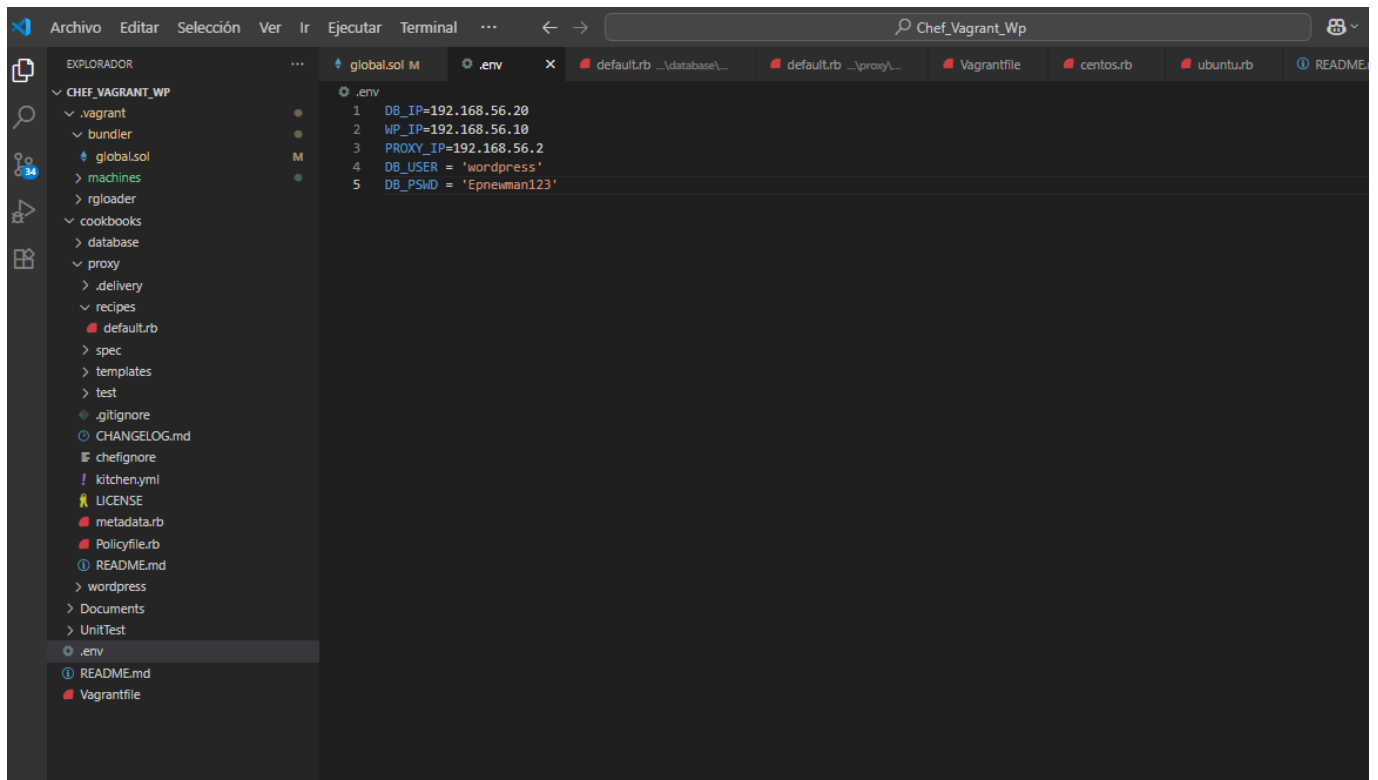
## Anexo D – Credenciales visibles en archivos de configuración

Captura del archivo `.env` o `attributes/default.rb` mostrando contraseñas sin cifrado.

Se identificaron archivos donde las credenciales de acceso a la base de datos (usuario y contraseña) están visibles en texto plano, lo que representa un riesgo de seguridad significativo si el repositorio o el entorno es expuesto sin protección adecuada.

En el archivo `.env` contiene las credenciales `DB_USER` y `DB_PSWD` directamente visibles y sin cifrado:

```
DB_IP=192.168.56.20
WP_IP=192.168.56.10
PROXY_IP=192.168.56.2
DB_USER = 'wordpress'
DB_PSWD = 'Epnnewman123'
```

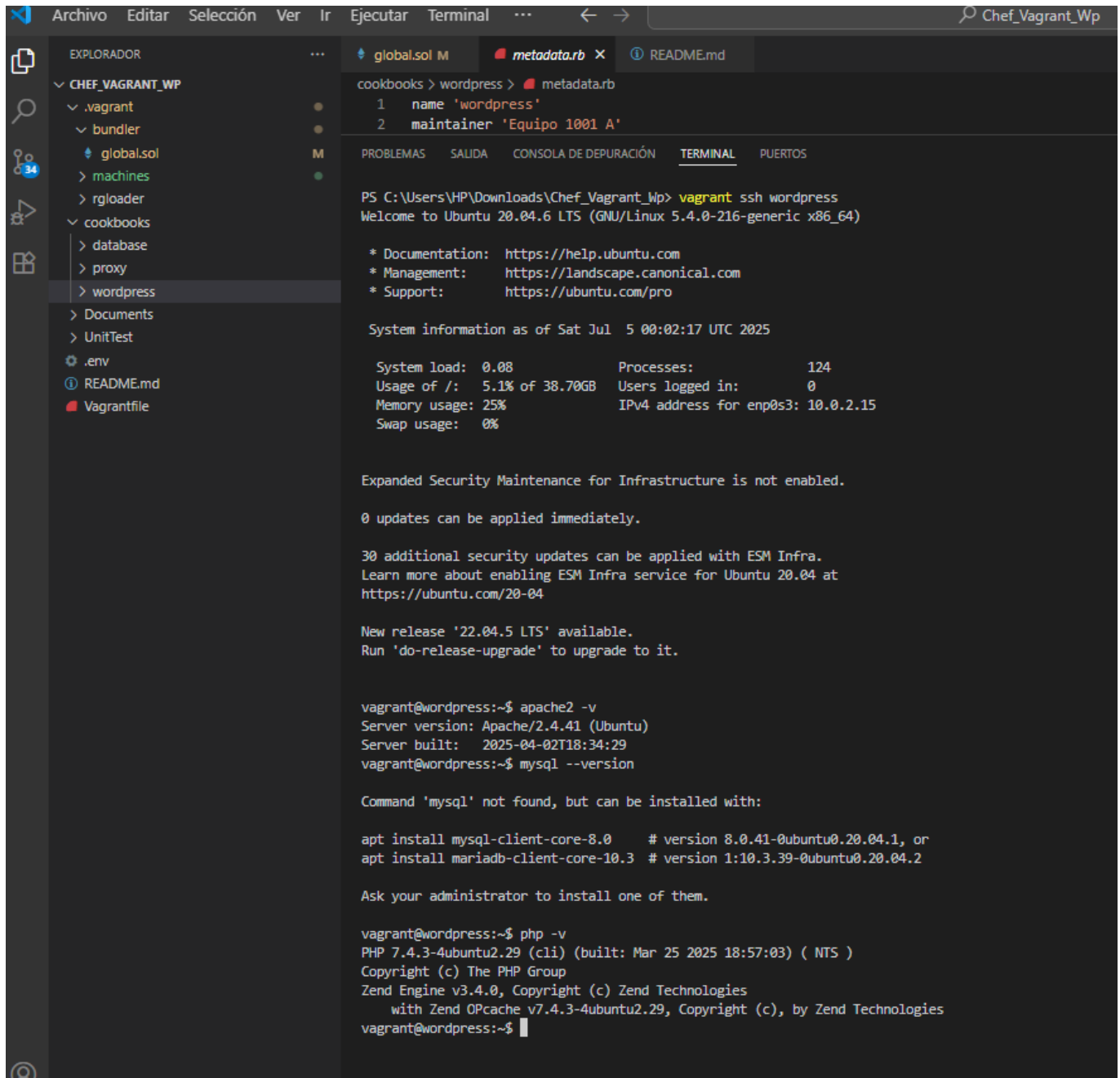


## Anexo E – Versiones antiguas de software instaladas

Evidencia de comandos como `apachectl -v`, `mysql --version`, etc., que muestran software obsoleto.

Las versiones del software instalado en la máquina virtual `wordpress`, correspondiente al entorno DevIA360. Se encontraron versiones que ya no están soportadas oficialmente o que presentan riesgo de vulnerabilidades.

```
vagrant ssh wordpress
apache2 -v
```



```
global.sol M metadata.rb x README.md
cookbooks > wordpress > metadata.rb
1 name 'wordpress'
2 maintainer 'Equipo 1001 A'

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\HP\Downloads\Chef_Vagrant_Wp> vagrant ssh wordpress
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-216-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sat Jul  5 00:02:17 UTC 2025

System load:  0.08      Processes:            124
Usage of /:   5.1% of 38.70GB Users logged in:        0
Memory usage: 25%      IPv4 address for enp0s3: 10.0.2.15
Swap usage:   0%

Expanded Security Maintenance for Infrastructure is not enabled.

0 updates can be applied immediately.

30 additional security updates can be applied with ESM Infra.
Learn more about enabling ESM Infra service for Ubuntu 20.04 at
https://ubuntu.com/20-04

New release '22.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

vagrant@wordpress:~$ apache2 -v
Server version: Apache/2.4.41 (Ubuntu)
Server built:   2025-04-02T18:34:29
vagrant@wordpress:~$ mysql --version

Command 'mysql' not found, but can be installed with:

apt install mysql-client-core-8.0 # version 8.0.41-0ubuntu0.20.04.1, or
apt install mariadb-client-core-10.3 # version 1:10.3.39-0ubuntu0.20.04.2

Ask your administrator to install one of them.

vagrant@wordpress:~$ php -v
PHP 7.4.3-4ubuntu2.29 (cli) (built: Mar 25 2025 18:57:03) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3-4ubuntu2.29, Copyright (c), by Zend Technologies
vagrant@wordpress:~$
```

## Anexo F – Ausencia de logs persistentes en `/var/log/`

Comprobación de que no se están generando ni almacenando registros relevantes.

La VM `wordpress` contiene algunos logs en `/var/log/` (ej. `auth.log`, `syslog`, `wordpress_error.log`), pero no se encontraron registros para otros servicios como base de datos o proxy, lo que indica cobertura limitada de logging. El acceso fue el siguiente:

```
vagrant ssh wordpress
ls -l /var/log/
```

```

global.sol M  metadata.rb x  README.md
cookbooks > wordpress > metadata.rb
1 name 'wordpress'
2 maintainer 'Equipo 1001 A'

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\HP\Downloads\Chef_Vagrant_Wp> vagrant ssh wordpress
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-216-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com

vagrant@wordpress:~$ ls -l /var/log/
total 1412
-rw-r--r-- 1 root root      854 Jun 27 22:17 alternatives.log
drwxr-x--- 2 root adm      4096 Jul  5 00:00 apache2
drwxr-xr-x 2 root root      4096 Jun 27 22:17 apt
-rw-r----- 1 syslog adm    19537 Jul  5 00:09 auth.log
-rw-rw---- 1 root utmp     1152 Jun 27 22:59 btmp
-rw-r----- 1 root adm     4329 Jun 27 22:16 cloud-init-output.log
-rw-r----- 1 syslog adm    93829 Jun 27 22:16 cloud-init.log
drwxr-xr-x 2 root root      4096 Mar 14 2023 dist-upgrade
-rw-r--r-- 1 root adm    44145 Jun 27 22:16 dmesg
-rw-r--r-- 1 root root    17224 Jun 27 22:17 dpkg.log
drwxr-sr-x+3 root systemd-journal 4096 Jun 27 22:16 journal
-rw-r----- 1 syslog adm    59344 Jul  4 23:20 kern.log
drwxr-xr-x 2 landscape landscape 4096 Jun 27 22:16 landscape
-rw-rw-r-- 1 root utmp    292584 Jul  5 00:02 lastlog
drwx----- 2 root root      4096 Jun 27 22:16 private
-rw-r----- 1 syslog adm    94565 Jul  5 00:12 syslog
-rw-r----- 1 syslog adm   1040350 Jul  5 00:00 syslog.1
-rw-r--r-- 1 root root      0 Jun 27 22:17 ubuntu-advantage-apt-hook.log
drwxr-x--- 2 root adm      4096 Jul  4 23:19 unattended-upgrades
-rw-r--r-- 1 root root      1137 Jul  4 23:29 wordpress_access.log
-rw-r--r-- 1 root root     3858 Jul  4 23:29 wordpress_error.log
-rw-rw-r-- 1 root utmp     4224 Jul  5 00:02 wtmp
vagrant@wordpress:~$ ^C
vagrant@wordpress:~$
PS C:\Users\HP\Downloads\Chef_Vagrant_Wp>

```

## Anexo G – Falta de segmentación entre entornos (dev/test/prod)

Evidencia de que el entorno no diferencia perfiles, mostrando un único **Vagrantfile** o recetas sin condiciones.

Las recetas en los cookbooks **proxy**, **wordpress** y **database** solo diferencian configuraciones por sistema operativo (**platform\_family**), pero no implementan separación entre entornos (**development**, **production**, etc.). No se usa **chef\_environment** ni estructuras condicionales para entornos. Revisión manual de los archivos **default.rb** en cada cookbook. No se encontró ninguna lógica de segmentación por entorno, lo que representa un riesgo en escenarios de despliegue real.

```

1 if node != nil && node['config'] != nil
2   db_ip = node['config']['db_ip'] || "127.0.0.1"
3 else
4   db_ip = "127.0.0.1"
5 end
6
7 execute "add host" do
8   command "echo '#{db_ip} db.epnewman.edu.pe' >> /etc/hosts"
9   action :run
10 end
11
12 case node['platform_family']
13 when 'debian', 'ubuntu'
14   execute "update" do
15     command "apt update -y && apt upgrade -y"
16     action :run
17   end
18   include_recipe 'wordpress::ubuntu_web' # Instalamos el servidor web
19   include_recipe 'wordpress::ubuntu_wp'  # Instalamos wordpress
20 when 'rhel', 'fedora'
21   execute "update" do
22     command "sudo dnf update -y && sudo dnf upgrade -y"
23     action :run
24   end
25   include_recipe 'wordpress::centos_web' # Instalamos el servidor web
26   include_recipe 'wordpress::centos_wp'  # Instalamos wordpress
27 end
28
29 if node != nil && node['config'] != nil
30   include_recipe 'wordpress::post_install'
31 end

```

## Anexo X – Puertos expuestos sin control

El archivo kitchen.yml de los cookbooks (proxy, wordpress, database) contiene líneas comentadas:

```

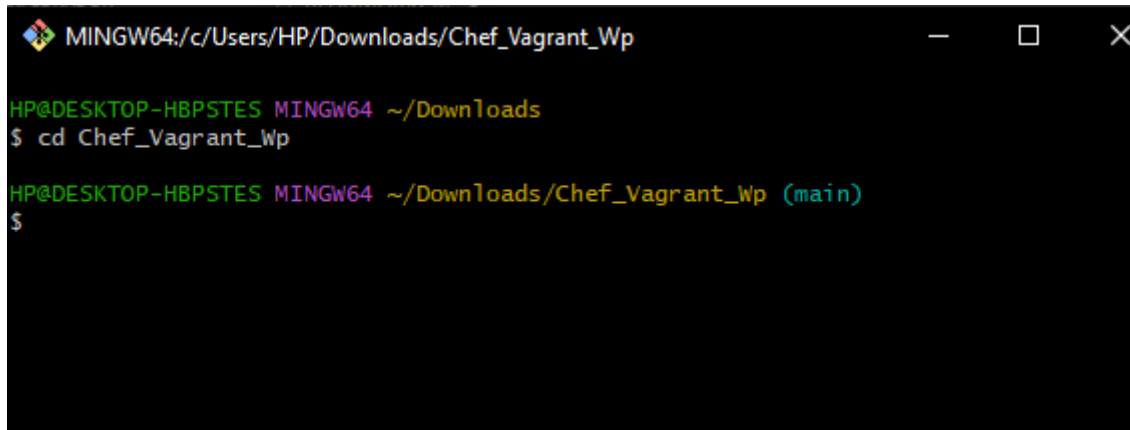
1 ---
2 driver:
3   name: vagrant
4
5 ## The forwarded_port port feature lets you connect to ports on the VM guest via
6 ## localhost on the host.
7 ## see also: https://www.vagrantup.com/docs/networking/forwarded\_ports.html
8
9 # network:
10 #   - ["forwarded_port", {guest: 80, host: 8080}]
11
12 provisioner:
13   name: chef_zero
14
15 ## product_name and product_version specifies a specific Chef product and version to install.
16 ## see the Chef documentation for more details: https://docs.chef.io/config\_yml\_kitchen.html
17 # product_name: chef
18 # product_version: 15
19
20 verifier:
21   name: inspec
22
23 platforms:
24   - name: centos-8
25     driver:
26       provision: true
27     box: generic/centos8

```

Eso significa que hay intención o posibilidad de exponer el puerto 80 al host a través del 8080, pero actualmente está deshabilitado (comentado). Aun así, debe documentarse como posible riesgo si no hay controles para evitar que se habilite en producción.

## Anexo Z – Clonar repositorio base y levantar servicios

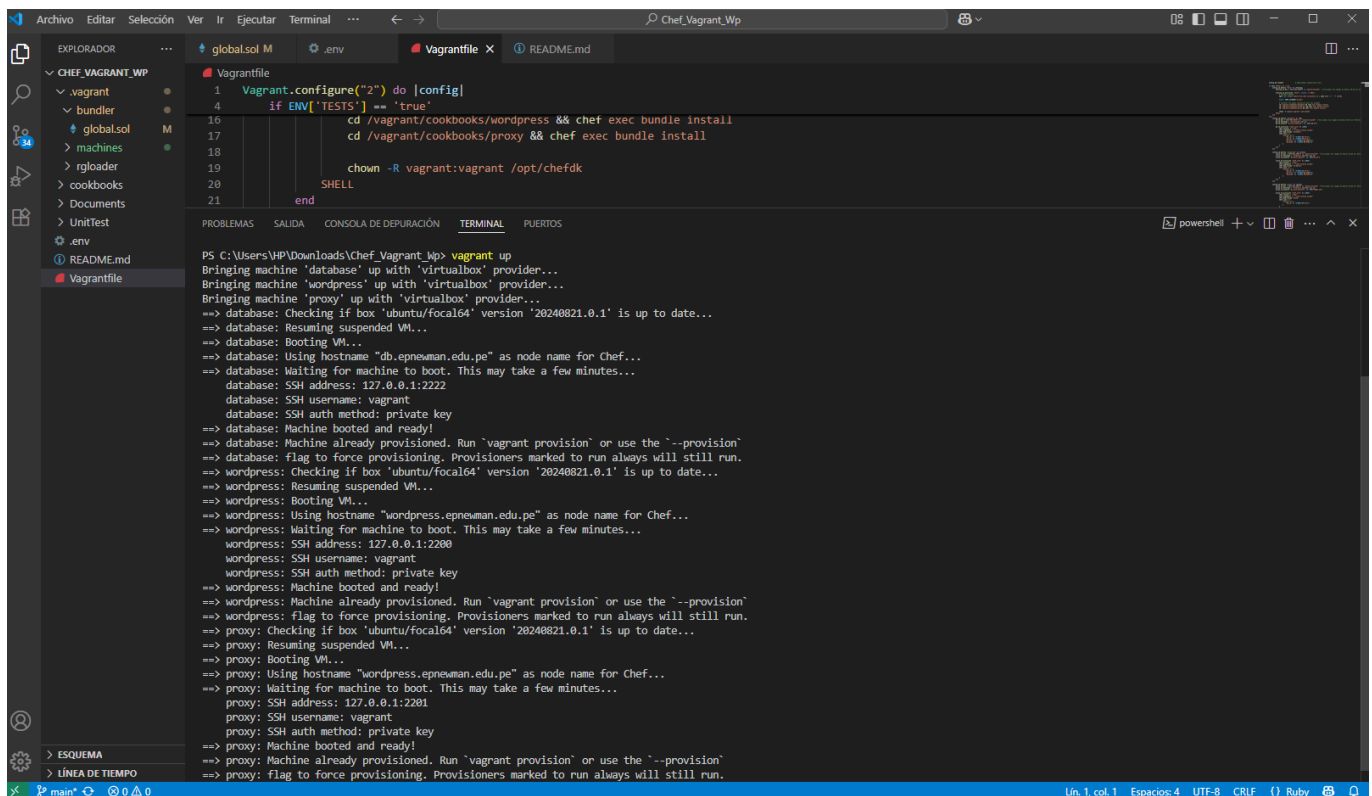




```
MINGW64:/c/Users/HP/Downloads/Chef_Vagrant_Wp
HP@DESKTOP-HBPSTES MINGW64 ~/Downloads
$ cd Chef_Vagrant_Wp

HP@DESKTOP-HBPSTES MINGW64 ~/Downloads/Chef_Vagrant_Wp (main)
$
```

Esperar a que se provisionen las 3 máquinas (wordpress, database, proxy).



```
1 Vagrant.configure("2") do |config|
4   if ENV['TESTS'] == 'true'
16     cd /vagrant/cookbooks/wordpress && chef exec bundle install
17     cd /vagrant/cookbooks/proxy && chef exec bundle install
18
19     chown -R vagrant:vagrant /opt/chefdk
20     SHELL
21   end
end

PS C:\Users\HP\Downloads\Chef_Vagrant_Wp> vagrant up
Bringing machine 'database' up with 'virtualbox' provider...
Bringing machine 'wordpress' up with 'virtualbox' provider...
Bringing machine 'proxy' up with 'virtualbox' provider...
==> database: Checking if box 'ubuntu/focal64' version '20240821.0.1' is up to date...
==> database: Resuming suspended VM...
==> database: Booting VM...
==> database: Using hostname "db.epnewman.edu.pe" as node name for Chef...
==> database: Waiting for machine to boot. This may take a few minutes...
database: SSH address: 127.0.0.1:2222
database: SSH username: vagrant
database: SSH auth method: private key
==> database: Machine booted and ready!
==> database: Machine already provisioned. Run 'vagrant provision' or use the '--provision'
==> database: flag to force provisioning. Provisioners marked to run always will still run.
==> wordpress: Checking if box 'ubuntu/focal64' version '20240821.0.1' is up to date...
==> wordpress: Resuming suspended VM...
==> wordpress: Booting VM...
==> wordpress: Using hostname "wordpress.epnewman.edu.pe" as node name for Chef...
==> 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: Machine booted and ready!
==> wordpress: Machine already provisioned. Run 'vagrant provision' or use the '--provision'
==> wordpress: flag to force provisioning. Provisioners marked to run always will still run.
==> proxy: Checking if box 'ubuntu/focal64' version '20240821.0.1' is up to date...
==> proxy: Resuming suspended VM...
==> proxy: Booting VM...
==> proxy: Using hostname "proxy.epnewman.edu.pe" as node name for Chef...
==> proxy: Waiting for machine to boot. This may take a few minutes...
proxy: SSH address: 127.0.0.1:2201
proxy: SSH username: vagrant
proxy: SSH auth method: private key
==> proxy: Machine booted and ready!
==> proxy: Machine already provisioned. Run 'vagrant provision' or use the '--provision'
==> proxy: flag to force provisioning. Provisioners marked to run always will still run.
```

## Anexo R – Repositorio

[https://github.com/MAYnerAC/AS\\_U3\\_EXAMEN\\_PRACTICO](https://github.com/MAYnerAC/AS_U3_EXAMEN_PRACTICO)