

INFORME FINAL DE AUDITORÍA DE SISTEMAS

Entidad Auditada: DevIA360

Ubicación: Tacna, Tacna, Tacna

Período auditado: 27/06/2025 hasta 27/06/2025

Equipo Auditor: Jhonny Rivera Mendoza **Fecha del informe:** 27/06/2025

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 apartado se desarrolla al final de la auditoría realizada y es una descripción breve y concisa del propósito de la auditoría, principales hallazgos, conclusiones y recomendaciones más relevantes.

2. ANTECEDENTES

DevIA360 es una empresa especializada en soluciones tecnológicas avanzadas, ofreciendo servicios de despliegue y gestión automatizada de aplicaciones web para sus clientes. Para agilizar sus operaciones, DevIA360 utiliza la solución Chef_Vagrant_Wp (https://github.com/OscarJimenezFlores/Chef_Vagrant_Wp.git), una herramienta que automatiza el despliegue de entornos de Wordpress mediante el uso de Vagrant y Chef. Este sistema de despliegue continuo es crítico para asegurar que los entornos sean confiables, homogéneos y seguros en la provisión de servicios. A través de este enfoque, DevIA360 busca mejorar la eficiencia operativa y garantizar la coherencia en la configuración de sus entornos de desarrollo, prueba y producción.

Anteriormente se le hizo una auditoria pero esta no fue completada. debido a que no se llego a terminar el informe final.

3. OBJETIVOS DE LA AUDITORÍA

Objetivo general

Evaluar los controles de seguridad, configuración y cumplimiento implementados en la infraestructura de desarrollo y despliegue automatizado basada en Vagrant y Chef de la empresa DevIA360 S.A.C., identificando vulnerabilidades, deficiencias de configuración y riesgos asociados a la confidencialidad, integridad y disponibilidad de la información.

Objetivos específicos

- Verificar la implementación de mecanismos de protección de datos sensibles (credenciales y configuraciones).
- Evaluar el cumplimiento de buenas prácticas en la gestión de configuraciones y aprovisionamiento automatizado.
- Analizar la seguridad en la infraestructura de red y segmentación del entorno virtual.
- Determinar la adecuación del entorno al marco normativo y políticas internas.
- Identificar riesgos de continuidad operativa y trazabilidad en procesos CI/CD.

4. ALCANCE DE LA AUDITORÍA

La auditoría comprenderá el análisis técnico y documental de los sistemas de desarrollo y despliegue automático utilizados por DevIA360 S.A.C., focalizado en el entorno de virtualización Vagrant y las configuraciones empleadas mediante ChefDK.

Componentes revisados:

- Archivos de configuración Vagrantfile y cookbooks asociados.

- Control de versiones y manejo de entornos (producción/pruebas).
- Variables de entorno, credenciales y protección de secretos.
- Proceso de aprovisionamiento e instalación automatizada de software.
- Topología de red virtual y configuración de acceso.

Ámbitos evaluados

- Tecnológico (configuración de sistemas, redes, automatización)
- Organizacional (gestión de configuración, control de versiones)
- Normativo (cumplimiento de estándares de seguridad de la información)

Unidades auditadas

- Área de desarrollo
- Área de infraestructura y operaciones
- Periodo auditado
- Primer semestre del año 2025 (enero-junio)

5. NORMATIVA Y CRITERIOS DE EVALUACIÓN

- ISO/IEC 27001:2022 - Gestión de Seguridad de la Información
- ISO/IEC 27002:2022 - Controles de seguridad de la información
- NIST SP 800-53 Rev.5 - Controles de seguridad y privacidad
- OWASP DevSecOps Maturity Model
- Ley N° 29733 - Ley de Protección de Datos Personales (Perú)
- Políticas internas de TI de DevIA360 S.A.C.

6. METODOLOGÍA Y ENFOQUE

Descripción del enfoque utilizado (basado en riesgos, cumplimiento, mixto) y métodos aplicados:

Enfoque

Basado en riesgos y cumplimiento

Métodos aplicados

- Revisión de configuraciones y código (Vagrantfile, Chef recipes)
- Inspección de documentos y políticas internas
- Pruebas técnicas de configuración, seguridad de red y aprovisionamiento

- Análisis de dependencias y binarios utilizados en instalación automatizada
- Aplicación de listas de verificación (checklists basados en ISO 27001/NIST)

7. HALLAZGOS Y OBSERVACIONES

Hallazgo 1: Exposición de credenciales en texto plano

Anexo: [Hallazgo 1: Exposición de credenciales en texto plano](#)

Elemento	Descripción
Evidencia	Uso directo de variables DB_USER, DB_PSWD en archivos JSON de Chef.
Criticidad	Alta
Criterio vulnerado	ISO 27002 - 8.2.3 (Gestión de información confidencial)
Causa	Falta de mecanismo seguro para manejo de secretos
Efecto	Riesgo de acceso no autorizado a la base de datos

Hallazgo 2: Instalación de ChefDK sin validación de integridad

Anexo: [Hallazgo 2: Instalación de ChefDK sin validación de integridad](#)

Elemento	Descripción
Evidencia	Uso de script remoto sin validación (wget
Criticidad	Alta
Criterio vulnerado	NIST SI-7 (Protección contra modificaciones maliciosas)
Causa	Práctica de instalación insegura
Efecto	Riesgo de ataque tipo supply-chain

Hallazgo 3: Uso de herramienta obsoleta (ChefDK)

Anexo: [Hallazgo 3: Uso de herramienta obsoleta \(ChefDK\)](#)

Elemento	Descripción
Evidencia	ChefDK sin soporte oficial ni actualizaciones desde 2020
Criticidad	Media
Criterio vulnerado	ISO 27002 - 12.1.2 (Gestión de vulnerabilidades)
Causa	Dependencia de herramientas sin mantenimiento
Efecto	Exposición a vulnerabilidades sin parches

Hallazgo 4: Segmentación de red insuficiente

Anexo: [Hallazgo 4: Segmentación de red insuficiente](#)

Elemento	Descripción
Evidencia	Red compartida entre servicios críticos sin reglas de control
Criticidad	Media
Criterio vulnerado	ISO 27002 - 13.1.3 (Segmentación de red)
Causa	Falta de segmentación lógica
Efecto	Riesgo de movimiento lateral entre servicios

Hallazgo 5: Ambigüedad en hostname de máquinas virtuales

Anexo: [Hallazgo 5: Ambigüedad en hostname de máquinas virtuales](#)

Elemento	Descripción
Evidencia	Doble asignación del FQDN "wordpress.epnewman.edu.pe"
Criticidad	Baja
Criterio vulnerado	ISO 27002 - 8.1.1 (Identificación de activos)
Causa	Error de configuración
Efecto	Conflictos DNS o validación TLS

8. ANÁLISIS DE RIESGOS

Evaluación del impacto y probabilidad de los riesgos identificados, asociados a los hallazgos encontrados.

Riesgo	Causa (Vínculo a Hallazgo)	Impacto	Probabilidad (%)	Nivel de Riesgo
Exposición de credenciales sensibles	Hallazgo 1: Falta de manejo seguro de secretos	Alto	90%	Crítico
Instalación sin validación de integridad	Hallazgo 2: Práctica de instalación insegura	Alto	80%	Alto
Dependencia de herramienta obsoleta	Hallazgo 3: Uso de ChefDK sin soporte	Medio	70%	Medio
Falta de segmentación de red	Hallazgo 4: Red compartida sin aislamiento lógico	Medio	80%	Medio
Ambigüedad en hostname	Hallazgo 5: FQDN duplicado en entorno virtual	Bajo	50%	Bajo

9. RECOMENDACIONES

Se emite la tabla de recomendaciones asociada al responsable de cada area.

Hallazgo	Recomendación	Responsable
1	Implementar mecanismos de gestión de secretos como Chef Encrypted Data Bags o HashiCorp Vault.	Área de Seguridad de la Información
2	Descargar y verificar binarios mediante SHA-256 o GPG antes de instalación.	Área de Infraestructura o Sistemas
3	Migrar a Chef Workstation o herramientas modernas soportadas.	Área de Desarrollo o DevOps
4	Crear subredes separadas para cada rol (DB, Web, Proxy) e implementar firewalls internos.	Área de Redes y Comunicaciones
5	Asignar nombres de host únicos y consistentes por cada VM.	Área de Infraestructura o DevOps

10. CONCLUSIONES

La configuración actual del entorno de despliegue presenta deficiencias importantes en el manejo de credenciales, validación de software instalado y segmentación de red. Los controles de seguridad son limitados y no están alineados con las buenas prácticas y normas vigentes, lo que expone a la organización a riesgos operacionales y de seguridad significativos. Es imperativo fortalecer los mecanismos de seguridad y revisión periódica del entorno.

11. PLAN DE ACCIÓN Y SEGUIMIENTO

Hallazgo	Recomendación	Responsable	Fecha Comprometida
1	Gestión segura de secretos	DevSecOps Lead	15/07/2025
2	Validar binarios antes de instalar	Administrador de sistemas	15/07/2025
3	Migrar a Chef Workstation	Equipo de Desarrollo	31/07/2025
4	Aplicar segmentación de red	Infraestructura y Redes	31/07/2025
5	Corregir nombres de host duplicados	Equipo DevOps	10/07/2025

12. ANEXOS

Evidencia de instalacion y dificultades

Problemas al instalar

Problema de configuracion en un texto Nos da un error en la configuración para eso vamos a ir al archivo

```
PS C:\Users\HP\Desktop\vagrant\Chef_Vagrant_Wp-main> vagrant up

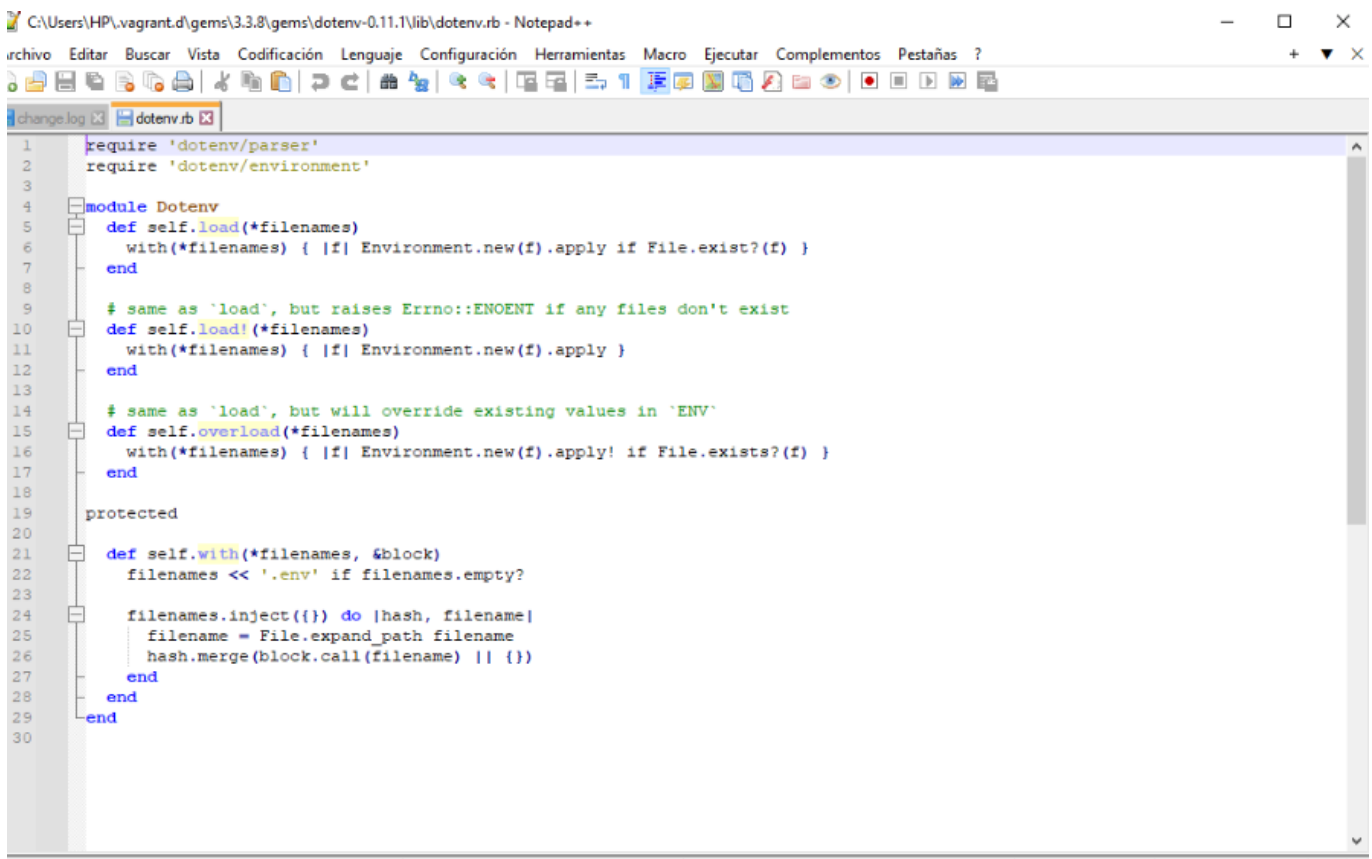
There was an error loading a Vagrantfile. The file being loaded
and the error message are shown below. This is usually caused by
an invalid or undefined variable.

Path: C:/Users/HP/.vagrant.d/gems/3.3.8/gems/dotenv-0.11.1/lib/dotenv.rb
Line number: 0
Message: undefined method `exists?'

```

Y debemos de eliminar la

"s" sobrante de "exists"



```

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.exists?(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

```

Visualizacion de servicio wordpress

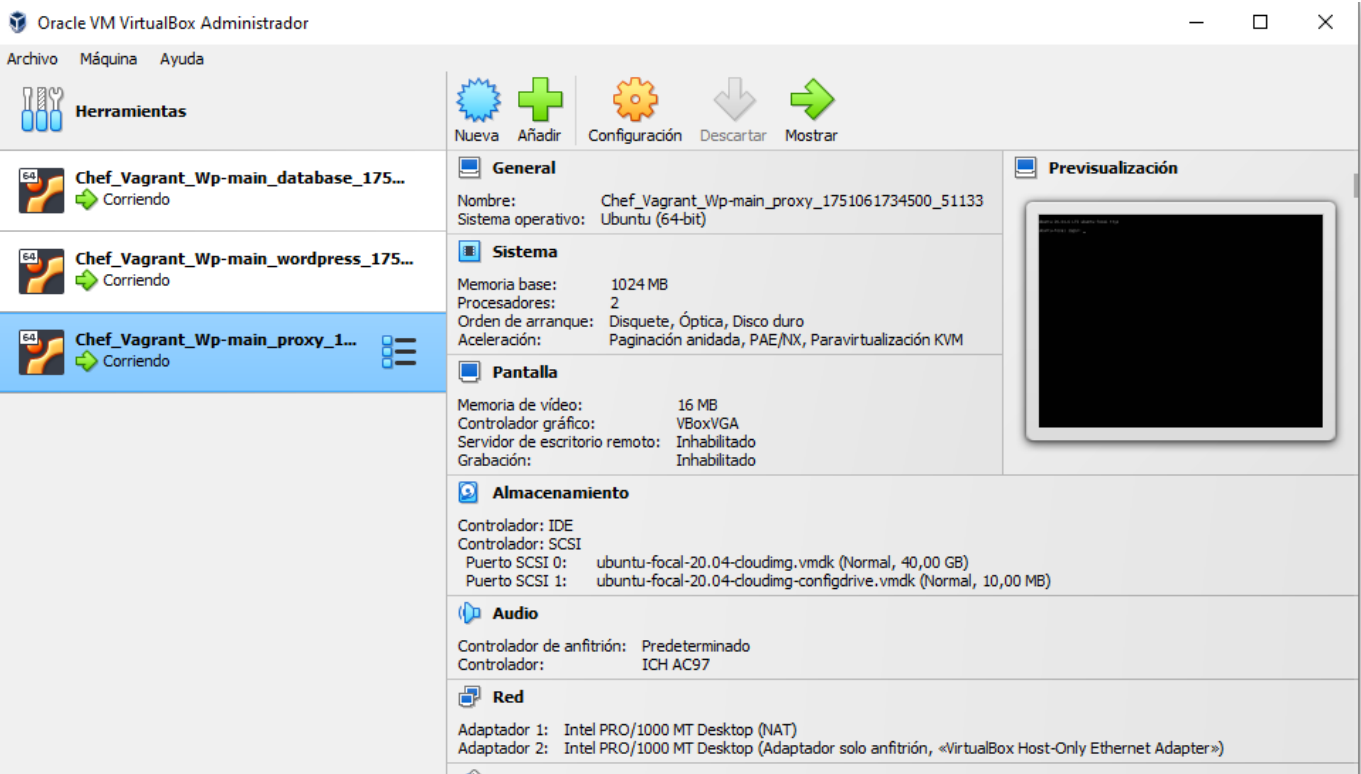


Ejecucion de vagrant up

Se ejecuta el comando vagrant up

```
=> wordpress: Reading state information...
=> wordpress: Calculating upgrade...
=> wordpress: The following security updates require Ubuntu Pro with 'esm-infra' enabled:
=> wordpress:   linux-headers-generic libblockdev-swap2 libpython3.8-minimal libsystemd0
=> wordpress:   python3-urllib3 libpython3.8 python3.8 libblockdev-crypto2 udev
=> wordpress:   libblockdev-loop2 libblockdev-fs2 libblockdev-part2 python3-requests
=> wordpress:   libudev1 systemd-timesyncd udisks2 linux-virtual python3.8-minimal
=> wordpress:   systemd-sysv libblockdev2 libpam-systemd systemd libblockdev-utils2
=> wordpress:   libnss-systemd linux-headers-virtual libblockdev-part-err2
=> wordpress:   libpython3.8-stdlib libudisks2-0 linux-image-virtual libxslt1.1
=> wordpress:   Learn more about Ubuntu Pro at https://ubuntu.com/pro
=> wordpress:   0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
=> wordpress: [2025-06-27T22:01:33+00:00] INFO: execute[update] ran successfully
=> wordpress:   - execute apt update -y && apt upgrade -y
=> wordpress: Recipe: wordpress::ubuntu_web
=> wordpress:
=> wordpress: * apt_package[apache2] action install
=> wordpress: [2025-06-27T22:01:39+00:00] INFO: apt_package[apache2] installed apache2 at 2.4.41-4ubuntu3.23
=> wordpress:   - install version 2.4.41-4ubuntu3.23 of package apache2
=> wordpress:
=> wordpress: * apt_package[php] action install
=> wordpress: [2025-06-27T22:01:47+00:00] INFO: apt_package[php] installed php at 2:7.4+75
=> wordpress:   - install version 2:7.4+75 of package php
=> wordpress: * apt_package[php-mysql] action install
=> wordpress: [2025-06-27T22:01:50+00:00] INFO: apt_package[php-mysql] installed php-mysql at 2:7.4+75
=> wordpress:   - install version 2:7.4+75 of package php-mysql
=> wordpress: * apt_package[php-mysqldb] action install
=> wordpress: [2025-06-27T22:01:51+00:00] INFO: apt_package[php-mysqldb] is a virtual package, actually acting on package[php7.4-mysql]
=> wordpress: (up to date)
=> wordpress: * apt_package[php-mysqli] action install
=> wordpress: [2025-06-27T22:01:51+00:00] INFO: apt_package[php-mysqli] is a virtual package, actually acting on package[php7.4-mysql]
=> wordpress: (up to date)
=> wordpress: * apt_package[php-json] action install
```

Se observa que las maquinas fueron levantadas



Evidencia de hallazgos

Hallazgo 1: Evidencia Exposicion de credenciales en texto plano

Las credenciales se obtienen desde un ENV, es buena practica pero no se encuentra cifrado, esta expuesto como variable de entorno.

```
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.epnewman.edu.pe"
  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_pswd" => "#{ENV["DB_PSWD"]}"
      }
    }
  end
end
```

Hallazgo 2: Evidencia Instalación de ChefDK sin validación de integridad

El script de instalación de ChefDK se ejecuta directamente desde Internet sin verificación de firma ni hash. Esto podría permitir la ejecución de código malicioso si el recurso remoto es interceptado o comprometido.

```

if ENV['TESTS'] == 'true'
  config.vm.define "test" do |testing|
    testing.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
else

```

Hallazgo 3: Evidencia Uso de herramienta obsoleta (ChefDK)

Se utiliza ChefDK, una herramienta sin soporte desde 2020, lo que implica exposición continua a vulnerabilidades sin parchear. Esto podría facilitar la explotación de fallas conocidas en entornos críticos.

```

if ENV['TESTS'] == 'true'
  config.vm.define "test" do |testing|
    testing.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

```

Hallazgo 4: Evidencia Segmentación de red insuficiente

Las máquinas virtuales definidas (base de datos, frontend y proxy) comparten la misma red privada sin segmentación lógica ni restricciones de tráfico entre ellas. Esto permite que, en caso de comprometer una de las VMs (como el servidor WordPress), un atacante pueda desplazarse lateralmente a otros servicios críticos

con facilidad.

```
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_pswd" => "#{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.epnewman.edu.pe"
  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_pswd" => "#{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.epnewman.edu.pe"
  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" => {
        "db_ip" => "#{ENV["DB_IP"]}",
        "db_user" => "#{ENV["DB_USER"]}",
        "db_pswd" => "#{ENV["DB_PSWD"]}"
      }
    }
  end
end
```

Hallazgo 5: Evidencia Ambigüedad en hostname de máquinas virtuales

El Vagrantfile asigna el mismo nombre de host `wordpress.epnewman.edu.pe` a dos máquinas distintas (wordpress y proxy), generando conflictos en la resolución de nombres. Esta duplicación puede provocar fallos en la validación TLS, errores de conectividad y debilitar la autenticación basada en certificados en entornos reales.

```
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.epnewman.edu.pe"
  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_pswd" => "#{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.epnewman.edu.pe"
  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
```