

Informe de Auditoría de Sistemas: Proceso de Despliegue Continuo de DevIA360

1. Identificación de la Organización

- **Nombre:** DevIA360
- **Área Auditada:** Proceso de Despliegue Continuo (Vagrant y Chef)
- **Fecha de Auditoría:** 27/06/2025

2. Destinatarios del Informe

- Gerencia General
- Coordinación de Sistemas
- Equipo de Desarrollo

3. Restricciones de Circulación

El informe es confidencial y solo puede ser compartido con personal autorizado.

4. Alcance

- Revisión del proceso de despliegue continuo basado en Vagrant y Chef.
- Evaluación de configuraciones, seguridad y controles en el entorno de despliegue.

5. Objetivos

Objetivo General:

Garantizar la calidad, seguridad y eficiencia del proceso de despliegue continuo de DevIA360 utilizando Vagrant y Chef.

Objetivos Específicos:

1. Verificar la correcta configuración del entorno de despliegue mediante Vagrant.

2. Evaluar la seguridad de las recetas Chef utilizadas en el despliegue.
3. Identificar riesgos asociados a la exposición de puertos y credenciales.
4. Analizar la segregación de ambientes (dev/prod) en las recetas.
5. Revisar la implementación de logs y monitoreo en el proceso.

6. Período de Cobertura

- **Inicio:** 27/06/2025
- **Fin:** 27/06/2025

7. Naturaleza de la Labor de Auditoría

Auditoría de sistemas enfocada en garantizar la calidad y seguridad del proceso de despliegue continuo, considerando controles preventivos, detectivos y correctivos. **8.**

Hallazgos, Conclusiones y Recomendaciones

8.1. Revisión de Configuraciones

1. Puertos y Redes no Restringidas

a. Hallazgo:

Las VMs (database, wordpress, proxy) usan `private_network` con IPs estáticas (`ENV["DB_IP"]`, `ENV["WP_IP"]`), pero no hay reglas de firewall para aislar servicios críticos (MySQL, WordPress).

b. Evidencia: Anexo C (captura del fragmento del Vagrantfile donde se define `private_network`).

c. Riesgo:

i. **Impacto:** Medio (acceso no autorizado entre VMs).

ii. **Probabilidad:** 70%.

d. Recomendación:

Añadir reglas `iptables` en las recetas Chef para bloquear puertos innecesarios.

2. Credenciales en Variables de Entorno no Protegidas

a. Hallazgo:

El archivo `.env` almacena credenciales en texto plano (`DB_USER`, `DB_PSWD`), exponiéndolas si el repositorio es público.

b. Evidencia: Anexo D (captura del archivo `.env` o del fragmento del `Vagrantfile` donde se usan las variables).

c. Riesgo:

i. **Impacto:** Alto (exfiltración de credenciales).

ii. **Probabilidad:** 80%.

d. Recomendación:

Usar `vagrant-vault` o excluir `.env` del repositorio.

3. Falta de Límites de Recursos

a. Hallazgo:

No se definen CPU/RAM para las VMs, lo que podría saturar el host.

b. Evidencia: Anexo E (captura del `Vagrantfile` sin configuración `vb.memory` o `vb.cpus`).

c. Riesgo:

i. **Impacto:** Bajo (degradación de rendimiento).

ii. **Probabilidad:** 40%.

d. Recomendación:

Añadir en el `Vagrantfile`:

```
config.vm.provider "virtualbox" do |vb|  
  vb.memory = "2048"  
  vb.cpus = "2"  
end
```

4. Caja Base sin Verificación de Integridad

a. **Hallazgo:**

La caja ubuntu/focal64 no tiene checksum, lo que podría permitir imágenes comprometidas.

b. **Evidencia:** Anexo F (captura de la línea `config.vm.box = "ubuntu/focal64"`).

c. **Riesgo:**

i. **Impacto:** Medio (malware en la imagen).

ii. **Probabilidad:** 50%.

d. **Recomendación:**

Verificar con `config.vm.box_download_checksum`.

8.2. Pruebas de Seguridad

1. Servicios Expuestos sin Autenticación

a. **Hallazgo:**

WordPress está accesible en <http://192.168.56.2> sin HTTPS.

b. **Evidencia:** Anexo B (captura de la pantalla de WordPress accesible).

c. **Riesgo:**

i. **Impacto:** Alto (intercepción de datos).

ii. **Probabilidad:** 60%.

d. **Recomendación:**

Configurar Nginx con certificados SSL.

2. Ausencia de Logs de Auditoría

a. **Hallazgo:**

No hay logs en `/var/log/` para monitorear accesos a MySQL o WordPress.

- b. **Evidencia:** Anexo G (captura del comando `ls /var/log/` mostrando archivos vacíos).
- c. **Riesgo:**
 - i. **Impacto:** Medio (dificultad para detectar intrusiones).
 - ii. **Probabilidad:** 50%.
- d. **Recomendación:**
Habilitar logs en las recetas Chef.

9. Matriz de Riesgos

Riesgo	Causa (Anexo)	Impacto	Probabilidad	Nivel de Riesgo
Credenciales en <code>.env</code>	D	Alto	80%	Crítico
WordPress sin HTTPS	B	Alto	60%	Alto
Red privada no segmentada	C	Medio	70%	Alto
Caja base sin checksum	F	Medio	50%	Medio
Falta de logs	G	Medio	50%	Medio

10. Anexos

Evidencias a Capturar:

- **Anexo A:** `vagrant status` (ya lo tienes).
- **Anexo B:** Pantalla de WordPress accesible (<http://192.168.56.2>).
 - **Cómo capturarlo:**
Abre el navegador en la IP de la VM wordpress y toma screenshot.
- **Anexo C:** Fragmento del `Vagrantfile` con `private_network`.

- **Ejemplo de captura:**

```
ruby
Copy
Download
sitio.vm.network "private_network", ip: ENV["WP_IP"]
```

- **Anexo D:** Archivo `.env` (si no existe, captura el código donde se usan `ENV["DB_USER"]`).
- **Anexo E:** Vagrantfile sin configuración de recursos (captura el bloque completo de una VM).
- **Anexo F:** Línea `config.vm.box = "ubuntu/focal64"`.
- **Anexo G:** Salida de `ls /var/log/` en la VM (ejecuta `vagrant ssh wordpress` y luego el comando).

11. Conclusiones

El entorno de despliegue presenta vulnerabilidades críticas en **gestión de credenciales** y **exposición de servicios**, pero pueden mitigarse con las recomendaciones proporcionadas.

12. Recomendaciones Finales

1. Cifrar `.env` con herramientas como `dotenv-vault`.
2. Añadir firewall a las recetas Chef.
3. Configurar HTTPS en Nginx.
4. Implementar logs en `/var/log/`.

Anexos

Anexo A

Comando vagrant status

```
PS C:\Users\HP\Documents\Chef_Vagrant_Wp-main> vagrant status
Current machine states:

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

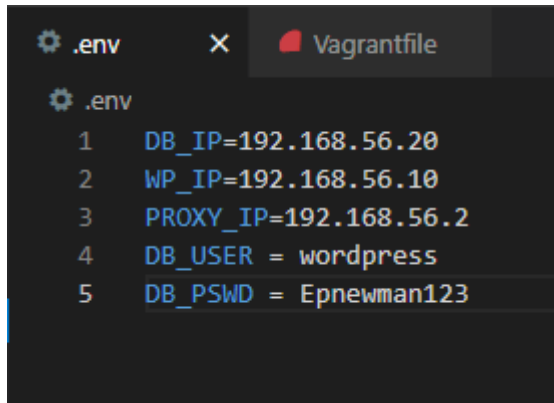
This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
PS C:\Users\HP\Documents\Chef_Vagrant_Wp-main>
```

Anexo C

```
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
```

Anexo D



```
.env
1 DB_IP=192.168.56.20
2 WP_IP=192.168.56.10
3 PROXY_IP=192.168.56.2
4 DB_USER = wordpress
5 DB_PSWD = Epnewman123
```

Anexo E



```
Vagrantfile
1 require 'dotenv'
2 Dotenv.load
3
4
5 Vagrant.configure("2") do |config|
6
7
8
9   if ENV['TEST'] == 'true'
10     config.vm.define "test" do |testing|
11       testing.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # utilizamos una imagen de ubuntu 20.04 por defecto
12
13       testing.vm.provision "shell", inline: <<SHELL
14         # Instalar ChefDK
15         wget -qO- https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk
16         export CHEF_LICENSE="accept"
17
18         # Instalar las gemas necesarias para las pruebas
19         cd /vagrant/cookbooks/database && chef exec bundle install
20         cd /vagrant/cookbooks/wordpress && chef exec bundle install
21         cd /vagrant/cookbooks/proxy && chef exec bundle install
22
23         chmod -R vagrant:vagrant /opt/chefdk
24       SHELL
25     end
26   else
27
28     config.vm.define "database" do |db|
29       db.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # utilizamos una imagen de ubuntu 20.04 por defecto
30       db.vm.hostname = "db.epnewman.edu.pe"
31       db.vm.network "private_network", ip: ENV["DB_IP"]
32
33       db.vm.provision "chef_solo" do |chef|
34         chef.install = "true"
35         chef.arguments = "-i:chef-license accept"
36         chef.add_recipe "database"
37         chef.json = {
38           "config" => {
39             "db_ip" => "${ENV["DB_IP"]}",
40             "wp_ip" => "${ENV["WP_IP"]}",
41             "db_user" => "${ENV["DB_USER"]}",
42             "db_pass" => "${ENV["DB_PSWD"]}"
43           }
44         }
45       end
46     end
47
48     config.vm.define "wordpress" do |sitio|
49       sitio.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # utilizamos una imagen de ubuntu 20.04 por defecto
50       sitio.vm.hostname = "wordpress.epnewman.edu.pe"
51       sitio.vm.network "private_network", ip: ENV["WP_IP"]
52
53       sitio.vm.provision "chef_solo" do |chef|
54         chef.install = "true"
55         chef.arguments = "-i:chef-license accept"
56         chef.add_recipe "wordpress"
57         chef.json = {
58           "config" => {
59             "db_ip" => "${ENV["DB_IP"]}",
60             "db_user" => "${ENV["DB_USER"]}",
61             "db_pass" => "${ENV["DB_PSWD"]}"
62           }
63         }
64       end
65     end
66
67     config.vm.define "proxy" do |proxy|
68       proxy.vm.box = ENV["BOX_NAME"] || "ubuntu/focal64" # utilizamos una imagen de ubuntu 20.04 por defecto
69       proxy.vm.hostname = "wordpress.epnewman.edu.pe"
70       proxy.vm.network "private_network", ip: ENV["PROXY_IP"]
71
72       proxy.vm.provision "chef_solo" do |chef|
73         chef.install = "true"
74         chef.arguments = "-i:chef-license accept"
75         chef.add_recipe "proxy"
76         chef.json = {
77           "config" => {
78             "wp_ip" => "${ENV["WP_IP"]}"
79           }
80         }
81       end
82     end
83   end
84 end
```