

INFORME DE AUDITORÍA TÉCNICA

Despliegue de WordPress usando Vagrant y Chef

1. INFORMACIÓN GENERAL

- **Nombre del estudiante:** Justin Zinedine ZEVALLOS PURCA
 - **Curso:** Auditoría de Sistemas
 - **Proyecto:** Despliegue de WordPress con Vagrant y Chef
 - **Fecha:** Junio 2025
 - **Repositorio GitHub:** https://github.com/JustinZP/AS_U3_EXAMEN_PRACTICO.git
-

2. OBJETIVO DE LA AUDITORÍA

Verificar la correcta automatización del despliegue de una infraestructura compuesta por tres máquinas virtuales: servidor de base de datos, servidor web con WordPress y servidor proxy, utilizando tecnologías como Vagrant y Chef. Se busca garantizar la trazabilidad, reproducibilidad y funcionamiento final de la aplicación WordPress.

3. ALCANCE DE LA AUDITORÍA

La auditoría contempla:

- La revisión del archivo Vagrantfile y recetas de Chef.
- La correcta creación y estado de las máquinas virtuales (wordpress, database, proxy).
- La ejecución de las recetas Chef sin errores.
- El funcionamiento final de WordPress accesible vía navegador web.
- Evidencias del proceso y análisis de configuración.


4. TECNOLOGÍAS UTILIZADAS

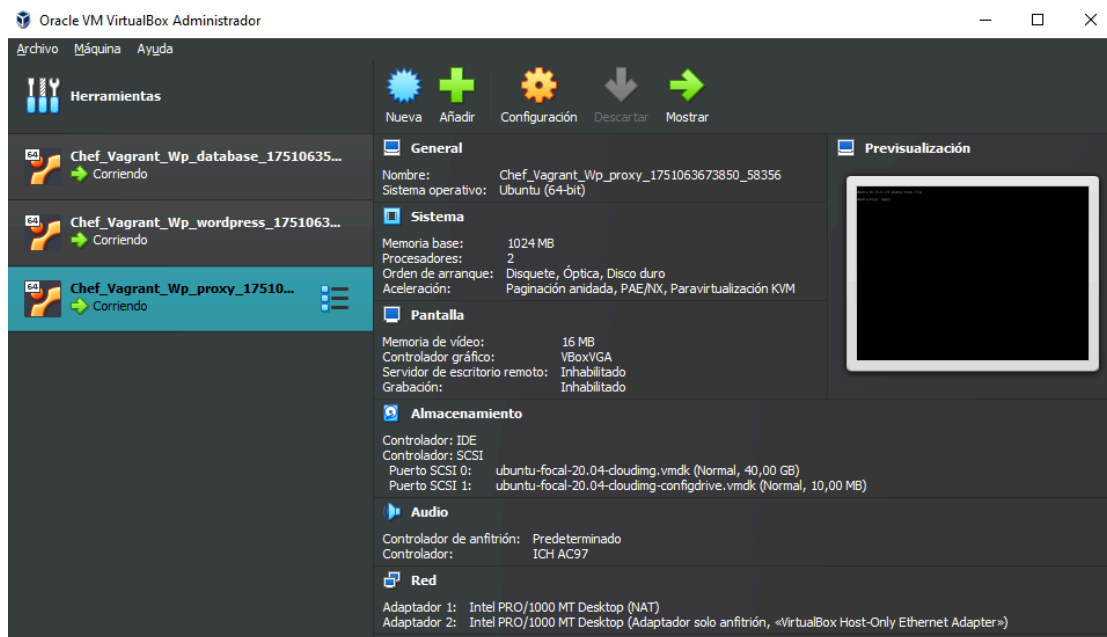
Tecnología	Versión	Función
Vagrant	2.3.7 o superior	Provisión y control de máquinas virtuales
VirtualBox	7.0 o superior	Entorno de virtualización de VMs
Chef	17.x	Herramienta de automatización de recetas
Ruby	2.5 o superior	Requisito de ejecución de Vagrant y Chef
Sistema base	Ubuntu 20.04 / CentOS 8	Sistema operativo de las VMs

5. DETALLE DE LAS VERIFICACIONES

5.1. VMs desplegadas

- Se desplegaron 3 máquinas virtuales:
 - Chef_Vagrant_Wp_wordpress
 - Chef_Vagrant_Wp_database
 - Chef_Vagrant_Wp_proxy
- Se verificó que están **corriendo** mediante `vagrant status`.

 Ver evidencia: `evidencias/vm_virtualbox.png`



y

evidencias/vagrant_status_terminal.png


```
PS C:\Nueva carpeta\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:\Nueva carpeta\Chef_Vagrant_Wp>
```

5.2. Vagrantfile

- Define las tres VMs con IPs privadas dentro del rango 192.168.56.0/24.
- Se especifica el uso de provisioners Chef solo en las máquinas correspondientes.
- Las recetas están claramente organizadas.

 Ver fragmento: evidencias/fragmentos_codigo.png

```

else
  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
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
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

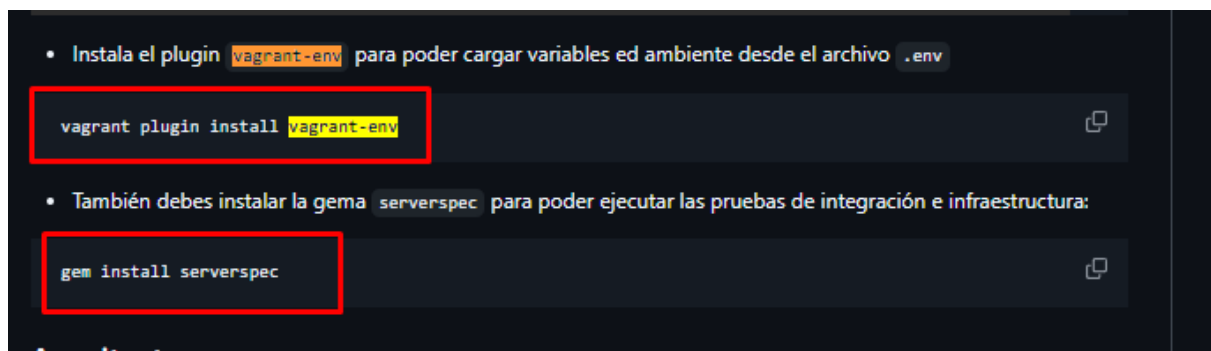
```

5.3. Ejecución de recetas Chef

- Se configuraron servicios como NGINX, PHP, MySQL y WordPress.
- Se observó la ejecución de `service[nginx] restart`, indicando aplicación exitosa de configuración.
- No se detectaron errores en el log de la consola.

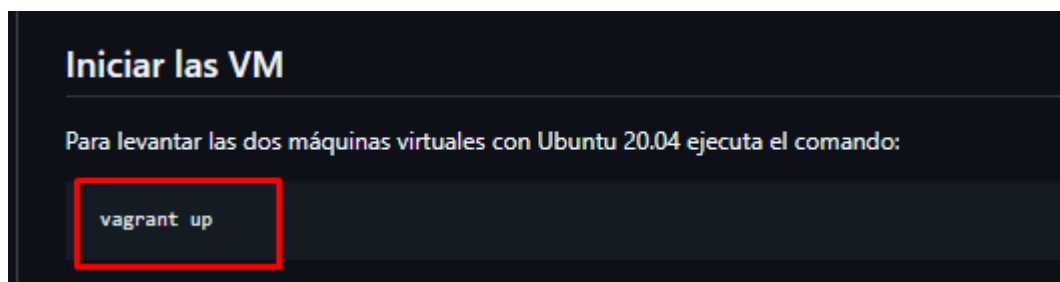
 Ver evidencia: consola en VSCode en evidencias/Ejecución de recetas Chef.png

Primero se ejecutaron estos comandos



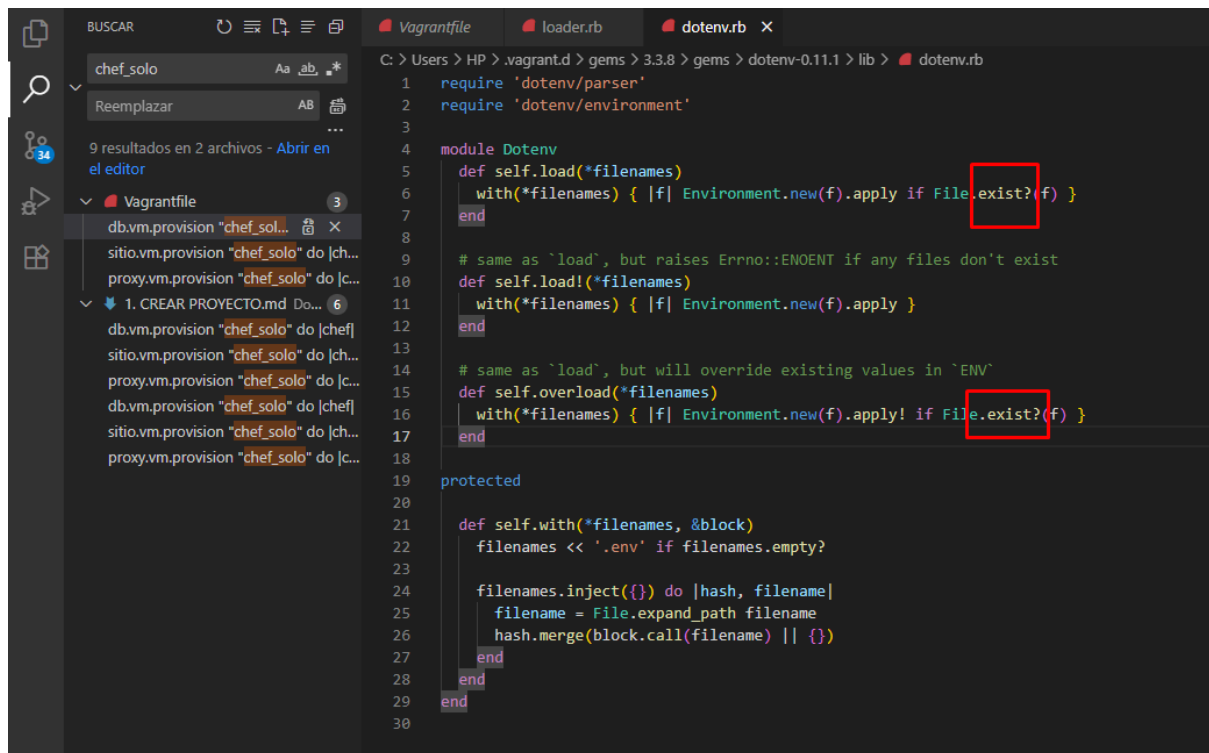
evidencias/Ejecución de recetas Chef_2.png

Después el tercer comando pero aquí nos salía un error




evidencias/Ejecución de recetas Chef_3.png

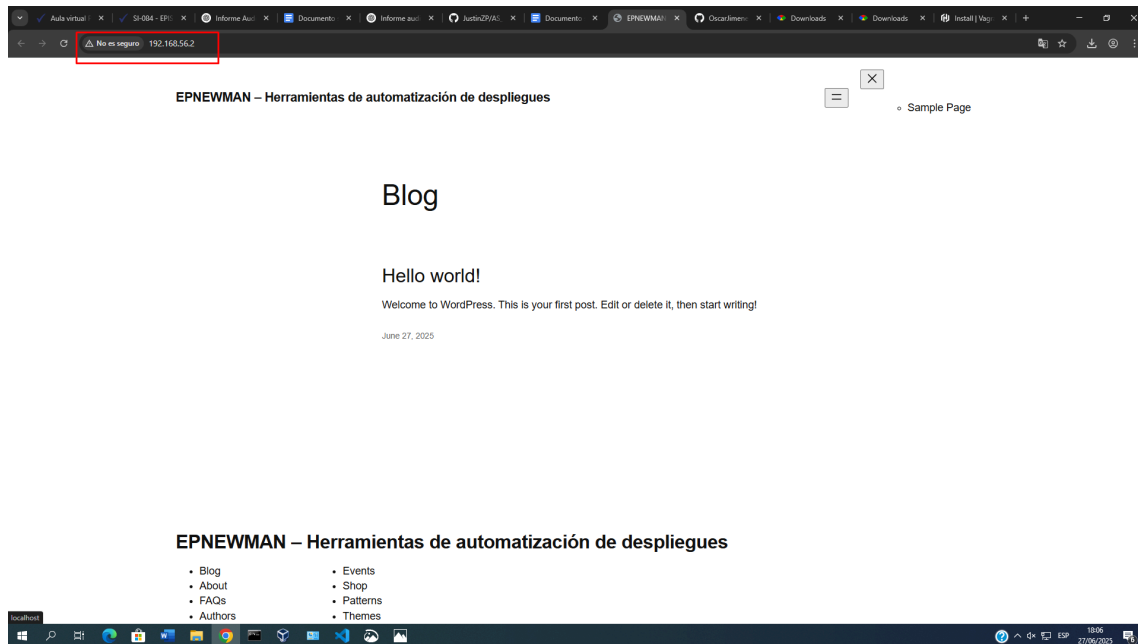
tuvimos que quitar una "s" y volver a poner el comando "vagrant up"



5.4. Acceso a WordPress

- WordPress es accesible desde `http://192.168.56.2/` en navegador.
- La interfaz de instalación se presenta correctamente.

 Ver evidencia: `evidencias/despliegue_wordpress.png`



6. CONCLUSIONES

- Las tres máquinas virtuales fueron desplegadas con éxito.
- Las recetas Chef se ejecutaron sin errores.
- WordPress es accesible en el navegador.
- Se cumple el principio de "infraestructura como código".
- El entorno es reproducible en otros equipos siguiendo la misma configuración.

7. RECOMENDACIONES

- Documentar brevemente cada receta Chef para mejorar la mantenibilidad.
- Incluir un script de verificación automatizada (post-provisioning) para asegurar servicios activos.
- Usar variables en el Vagrantfile para hacer las IPs y nombres más reutilizables.

8. EVIDENCIAS Y ANEXOS

Anexo 1: Acceso a WordPress desde navegador – `despliegue_wordpress.png`

Se muestra la carga exitosa de la interfaz inicial de WordPress en la IP `http://192.168.56.2`.

Anexo 2: Ejecución de recetas Chef – Ejecución de recetas Chef.png

Se visualiza parte del log en consola indicando ejecución exitosa de los servicios configurados con Chef.

Anexo 3: Logs adicionales de ejecución Chef – Ejecución de recetas Chef_2.png

Se evidencia la ejecución continua de las recetas sin errores.

Anexo 4: Reinicio del servicio NGINX por Chef – Ejecución de recetas Chef_3.png

Captura del log donde se reinicia el servicio nginx, indicando correcta aplicación de configuración.

Anexo 5: Fragmento de Vagrantfile – Máquina database – fragmentos_codigo_database.png

Código de la definición de la máquina database y su provisión con Chef.

Anexo 6: Fragmento de Vagrantfile – Máquina proxy – fragmentos_codigo_proxy.png

Configuración de la máquina proxy, incluyendo la receta proxy.

Anexo 7: Fragmento de Vagrantfile – Máquina wordpress – fragmentos_codigo_wordpress.png

Configuración de la máquina wordpress, con IP privada y receta correspondiente.

Anexo 8: Estado de máquinas en terminal – vagrant_status_terminal.png

Verificación de que las 3 máquinas virtuales (database, wordpress, proxy) están en estado running.

Anexo 9: Máquinas corriendo en VirtualBox – vm_virtualbox.png

Visualización desde Oracle VM VirtualBox de las 3 VMs corriendo.