Guía Exhaustiva para la Creación y Personalización de Boxes de Vagrant con Ubuntu Server

Este informe técnico proporciona una guía detallada y profesional para la construcción, configuración y despliegue de un entorno de desarrollo basado en Vagrant, partiendo desde cero. El documento abarca el proceso completo, desde la creación de una máquina virtual base con Ubuntu Server 25.04 hasta la automatización del aprovisionamiento de usuarios y la configuración de autenticación SSH personalizada. El objetivo es capacitar a los ingenieros de sistemas y desarrolladores para que obtengan un control total sobre sus entornos virtualizados, trascendiendo el uso de boxes preconfiguradas.

Parte I: Construcción de la Máquina Virtual Base con Ubuntu 25.04

La base de cualquier entorno Vagrant robusto es una imagen de máquina virtual (VM) limpia, estable y correctamente configurada. Esta sección detalla el proceso de creación de esta VM fundamental utilizando Oracle VirtualBox y la última versión de Ubuntu Server. Las decisiones tomadas en esta fase inicial son críticas y tienen un impacto directo en la fiabilidad y el rendimiento del box final.

1.1 Adquisición de las Herramientas Necesarias

Antes de comenzar la construcción de la VM, es imperativo obtener el software correcto. Este proceso consta de dos componentes principales: el hipervisor (VirtualBox) y la imagen del sistema operativo (ISO de Ubuntu Server).

- **Oracle VirtualBox**: VirtualBox es el hipervisor que ejecutará nuestra máquina virtual. Es un requisito previo para utilizar el proveedor de VirtualBox en Vagrant, que es el más común y el que se utilizará en esta guía.[1] Se debe descargar la última versión compatible con el sistema operativo anfitrión (en este caso, Windows) desde el sitio web oficial de VirtualBox.
- **Imagen ISO de Ubuntu Server 25.04**: El siguiente paso es obtener el medio de instalación del sistema operativo. Las páginas oficiales de lanzamientos de Ubuntu son la fuente autorizada para estas imágenes.[3] Para este proyecto, se buscará y descargará

específicamente la imagen de instalación del servidor para la arquitectura de 64 bits (amd64), que se distribuye como un archivo con la extensión .iso.[4]

Asesoramiento Profesional: Versiones LTS frente a Versiones Intermedias

La solicitud especifica el uso de Ubuntu 25.04. Es fundamental entender la naturaleza de esta versión. Ubuntu 25.04 es una versión "intermedia" o no LTS (Long-Term Support).[6] Las versiones LTS, como la 24.04, reciben soporte de seguridad y mantenimiento durante cinco años, lo que las convierte en una base estable y predecible para entornos de producción y desarrollo a largo plazo. En contraste, las versiones intermedias como la 25.04 solo reciben soporte durante nueve meses.[6]
El propósito principal de Vagrant es crear entornos de desarrollo reproducibles, estables y duraderos. Construir un box base sobre un sistema operativo con un ciclo de vida corto introduce una deuda técnica significativa; el box se volverá obsoleto y vulnerable rápidamente, requiriendo una reconstrucción completa en menos de un año. Para cualquier proyecto profesional o de larga duración, la estabilidad y el soporte extendido son primordiales. Por lo tanto, aunque esta guía procederá con Ubuntu 25.04 según lo solicitado, la recomendación profesional es utilizar siempre la última versión LTS disponible para la creación de boxes base.

1.2 Configuración de la Máquina Virtual en VirtualBox

Con las herramientas necesarias descargadas, el siguiente paso es configurar la estructura de la máquina virtual dentro de VirtualBox. Se utilizará el asistente "Nueva Máquina Virtual", prestando especial atención a las configuraciones que son cruciales para la compatibilidad con Vagrant.

- **Nombre y Tipo**: La VM debe recibir un nombre descriptivo, como ubuntu-25.04-base. Si el nombre incluye la palabra "Ubuntu", VirtualBox detectará automáticamente el tipo y la versión del sistema operativo, simplificando la configuración.[8]
- **Asignación de Hardware**: Los recursos de hardware deben ser suficientes para un rendimiento fluido. Se recomienda un mínimo de 2 GB de RAM y 2 núcleos de CPU, siendo 4 GB o más lo ideal para tareas más intensivas.[8] El disco duro virtual debe tener un tamaño mínimo de 25 GB y configurarse como "almacenamiento de expansión dinámica" para que solo ocupe el espacio que realmente utiliza en el disco anfitrión.[8] Se seleccionará el formato de disco VMDK (Virtual Machine Disk), ya que es un estándar común y portátil.[2]
- **Asociación de la Imagen ISO**: La imagen ISO de Ubuntu Server 25.04 descargada previamente se debe adjuntar a la unidad óptica virtual de la VM. Esto permite que la VM arranque desde el instalador del sistema operativo.[8]

La Configuración Crítica del Adaptador de Red NAT

Una de las configuraciones más importantes y a menudo pasadas por alto es la del adaptador de red. Para que Vagrant funcione correctamente con su configuración predeterminada, el primer adaptador de red (Adaptador 1) de la VM **debe** estar configurado en modo NAT (Network Address Translation).[11]
El mecanismo de comunicación por defecto de Vagrant, que permite comandos como vagrant ssh sin necesidad de conocer la dirección IP de la VM, se basa en el reenvío de puertos del hipervisor. Vagrant configura automáticamente una regla que reenvía un puerto alto en la máquina anfitriona (por ejemplo, el puerto 2222) al puerto 22 (el puerto estándar de SSH) dentro de la VM invitada. Esta funcionalidad de reenvío de puertos es una característica intrínseca del modo de red NAT de VirtualBox. Si el primer adaptador no está configurado como NAT, Vagrant no puede establecer este canal de comunicación inicial, lo que provocará que el comando vagrant up falle con un error de tiempo de espera al intentar conectar por SSH. Por lo tanto, esta no es una simple recomendación, sino un requisito fundamental para la funcionalidad básica de Vagrant.

1.3 Instalación Guiada de Ubuntu Server

Una vez configurada la VM, se procede a arrancarla para iniciar el proceso de instalación de Ubuntu Server. Durante este proceso, se deben tomar decisiones clave que alinearán el sistema operativo con los requisitos de Vagrant.

- **Creación del Usuario Inicial**: Durante el asistente de instalación, se solicitará la creación de un usuario. Es de vital importancia que este usuario se nombre vagrant y que su contraseña se establezca como vagrant.[2] Vagrant está preconfigurado para utilizar estas credenciales por defecto al interactuar con boxes base. Usar un nombre de usuario o contraseña diferente requeriría una configuración adicional en el Vagrantfile para cada proyecto.
- **Selección de Software**: El instalador de Ubuntu ofrece la opción de instalar conjuntos de software adicionales. Es absolutamente necesario seleccionar la opción "Instalar servidor OpenSSH".[14] Sin un demonio SSH ejecutándose en la máquina invitada, Vagrant no tendrá ningún servicio al que conectarse, haciendo imposible cualquier tipo de comunicación o gestión.
- **Particionamiento del Disco**: Para simplificar, se utilizará el esquema de particionamiento guiado que utiliza todo el disco. Esto es suficiente para un entorno de desarrollo.
- **Finalización y Reinicio**: Una vez que el instalador complete la copia de archivos y la configuración, la VM se reiniciará. Después del reinicio, se debe iniciar sesión en la consola por primera vez con el usuario vagrant y la contraseña vagrant para proceder con la siguiente fase de configuración.

Parte II: Transformación de la VM en una Plantilla Preparada para Vagrant

Con Ubuntu Server instalado, la VM es funcional pero aún no está lista para ser utilizada por Vagrant. Esta sección detalla los pasos de configuración post-instalación que se deben realizar dentro de la VM para hacerla compatible con los protocolos de automatización y comunicación de Vagrant. Estos pasos son los que distinguen a una VM genérica de un box base de Vagrant verdaderamente funcional.

2.1 Establecimiento de Privilegios de Superusuario para el Usuario 'vagrant'

Vagrant necesita ejecutar comandos administrativos en la máquina invitada para realizar tareas de aprovisionamiento, como instalar software o modificar archivos de configuración. Para evitar la necesidad de interacción manual (introducir una contraseña), es un requisito indispensable configurar sudo sin contraseña para el usuario vagrant.
Este es un pilar fundamental de la automatización. Las herramientas de gestión de configuración como Vagrant operan de forma no interactiva. Si un comando de aprovisionamiento como sudo apt-get install solicitara una contraseña, no habría una terminal (TTY) para recibir esa entrada. El proceso se detendría indefinidamente, causando que vagrant up falle. La configuración de sudo sin contraseña es el nexo causal que permite el aprovisionamiento automatizado y no interactivo.
El método recomendado y más seguro para lograr esto es crear un nuevo archivo en el directorio /etc/sudoers.d/, en lugar de modificar directamente el archivo principal /etc/sudoers. Esto hace que la configuración sea modular y más fácil de gestionar.[2]
El comando para realizar esta configuración es:

Bash

```
echo "vagrant ALL=(ALL) NOPASSWD:ALL" | sudo tee /etc/sudoers.d/vagrant
```

Después de ejecutar este comando, el usuario vagrant podrá ejecutar cualquier comando con sudo sin que se le solicite su contraseña.

2.2 Configuración de SSH para el Arranque Inicial de Vagrant

Para que Vagrant pueda establecer su primera conexión SSH con un box recién creado, necesita un método de autenticación predecible. La solución estándar en el ecosistema de Vagrant es el uso de un par de claves SSH "inseguras" conocidas públicamente. La clave pública de este par se instala en todos los boxes base, y la clave privada correspondiente se distribuye con cada instalación de Vagrant.
Este mecanismo de "secreto compartido" resuelve el problema de cómo Vagrant puede autenticarse en cualquier box genérico sin un intercambio previo de claves. Sin embargo, utilizar una clave conocida globalmente representa un riesgo de seguridad significativo.[15] Para mitigar esto, Vagrant implementa un ciclo de vida de claves en dos fases:

1. **Fase de Arranque (Bootstrap)**: En el primer vagrant up de una nueva instancia, Vagrant se conecta utilizando la clave privada insegura para autenticarse con la clave pública insegura que está en el box.
2. **Fase de Reemplazo**: Inmediatamente después de conectarse, Vagrant genera un par de claves SSH *nuevo y único* para esa instancia específica. Inserta la nueva clave pública en el archivo ~/.ssh/authorized_keys de la VM y elimina la clave pública insegura. A partir de ese momento, todas las comunicaciones futuras con esa instancia de VM utilizarán el nuevo par de claves seguras.[16]

Para que nuestro box personalizado participe en este proceso, debemos instalar manualmente la clave pública insegura. Los pasos son los siguientes [2]:

Bash

```
# Crear el directorio.ssh si no existe
mkdir -p /home/vagrant/.ssh

# Establecer los permisos correctos para el directorio
chmod 0700 /home/vagrant/.ssh

# Descargar la clave pública insegura de Vagrant y guardarla como authorized_keys
wget --no-check-certificate \
    https://raw.githubusercontent.com/hashicorp/vagrant/main/keys/vagrant.pub \
    -O /home/vagrant/.ssh/authorized_keys

# Establecer los permisos correctos para el archivo de claves
chmod 0600 /home/vagrant/.ssh/authorized_keys

# Asegurarse de que el usuario y grupo 'vagrant' sean los propietarios del directorio y su contenido
chown -R vagrant:vagrant /home/vagrant/.ssh
```

2.3 Integración de las VirtualBox Guest Additions

Las VirtualBox Guest Additions son un conjunto de controladores y utilidades del sistema que mejoran drásticamente el rendimiento y la integración entre la máquina anfitriona y la invitada. Son un requisito indispensable para funcionalidades clave de Vagrant, como las carpetas sincronizadas (synced folders).[2]
La funcionalidad de carpetas sincronizadas depende de un módulo del kernel en el sistema invitado llamado vboxsf.[7] Este módulo es instalado por las Guest Additions. Para que el instalador de las Guest Additions pueda compilar y cargar este módulo, el sistema invitado debe tener instaladas las cabeceras del kernel correspondientes a la versión del kernel en ejecución, así como las herramientas de compilación básicas (como gcc). Existe una clara cadena de dependencias: la falta de las cabeceras del kernel o de las herramientas de compilación provocará un fallo en la instalación de las Guest Additions, lo que a su vez impedirá que el módulo vboxsf esté disponible, resultando finalmente en que las carpetas sincronizadas no funcionen.
El proceso de instalación correcto es el siguiente:

1. **Instalar Dependencias**: Actualizar la lista de paquetes e instalar las herramientas de compilación, DKMS (Dynamic Kernel Module Support) y las cabeceras del kernel.
   Bash
   sudo apt-get update
   sudo apt-get install -y build-essential dkms linux-headers-$(uname -r)

2. **Montar la Imagen de las Guest Additions**: Desde el menú de la ventana de la VM en VirtualBox, seleccionar "Dispositivos" y luego "Insertar imagen de CD de las 'Guest Additions'...". Esto montará una imagen ISO virtual en la unidad de CD-ROM de la VM.
3. **Ejecutar el Instalador**: Montar el CD-ROM virtual en el sistema de archivos y ejecutar el script de instalación.
   Bash
   sudo mount /dev/cdrom /mnt
   sudo /mnt/VBoxLinuxAdditions.run

Una vez completada la instalación, es recomendable reiniciar la máquina virtual para asegurarse de que todos los nuevos módulos del kernel se carguen correctamente.

2.4 Preparación de la VM para el Empaquetado

El último paso antes de crear el box es realizar una limpieza para reducir su tamaño final y optimizar la compresión. Un box más pequeño se descarga y se despliega más rápido.

- **Limpiar la Caché de APT**: Eliminar los paquetes .deb descargados que ya no son necesarios.

Bash
sudo apt-get clean

- **Poner a Cero el Espacio Libre en Disco**: Este es un paso de optimización crucial. Se crea un archivo grande lleno de ceros que ocupa todo el espacio libre del disco virtual. Luego, este archivo se elimina. Cuando VirtualBox o la herramienta de empaquetado compriman el disco virtual, las áreas llenas de ceros se comprimirán de manera extremadamente eficiente, reduciendo drásticamente el tamaño del archivo final del box.
  Bash
  sudo dd if=/dev/zero of=/EMPTY bs=1M
  sudo rm -f /EMPTY

  Es posible que el comando dd termine con un error de "no space left on device", lo cual es esperado y normal.
- **Apagar la VM**: Finalmente, apagar la máquina virtual de forma segura.
  Bash
  sudo shutdown -h now

La VM está ahora completamente preparada y lista para ser empaquetada como un box de Vagrant reutilizable.

Parte III: Empaquetado y Despliegue de su Primer Box Personalizado

Con la máquina virtual meticulosamente preparada, el siguiente paso es convertirla en un archivo .box portátil que Vagrant pueda gestionar. Esta sección cubre el flujo de trabajo desde la línea de comandos para empaquetar la VM, importarla al inventario local de Vagrant y lanzarla por primera vez.

3.1 Creación del Archivo del Box

Vagrant proporciona un comando nativo, vagrant package, para exportar una VM existente a un formato de box. Este comando se ejecuta desde la terminal del sistema anfitrión (Windows, en este caso).
Para empaquetar una VM que fue creada y gestionada directamente en VirtualBox (y no a través de un Vagrantfile previo), es fundamental utilizar el indicador --base. Este indicador le dice a Vagrant que se dirija a una VM específica por su nombre o UUID tal como aparece en la interfaz de VirtualBox.[18] También se utilizará el indicador --output para especificar un nombre

de archivo de salida descriptivo para nuestro box.[18]
Desde una terminal en el directorio de su proyecto en Windows, ejecute el siguiente comando, reemplazando "ubuntu-25.04-base" con el nombre exacto que le dio a su VM en VirtualBox:

PowerShell

```
vagrant package --base "ubuntu-25.04-base" --output "ubuntu-25.04.box"
```

Este proceso puede tardar varios minutos, ya que Vagrant exporta la configuración de la VM y comprime su disco duro virtual en un único archivo de archivo (.box). Al finalizar, tendrá un archivo llamado ubuntu-25.04.box en su directorio actual.

3.2 Importación y Verificación del Box Local

El archivo .box recién creado debe ser añadido al inventario de boxes de Vagrant para que pueda ser utilizado en proyectos. Esto se hace con el comando vagrant box add.
Un punto común de error, especialmente en Windows, es la sintaxis de la ruta al archivo local. Vagrant puede interpretar incorrectamente las rutas de estilo Windows (con barras invertidas) como URLs. La forma más robusta y recomendada de especificar una ruta de archivo local es utilizando el protocolo file:///, seguido de la ruta absoluta al archivo .box.[19]
El comando vagrant box add toma dos argumentos principales: un nombre lógico para el box (por ejemplo, ubuntu/25.04) y la ruta al archivo. El nombre lógico es cómo se referenciará al box en los Vagrantfile.

PowerShell

```
vagrant box add ubuntu/25.04 file:///C:/ruta/a/su/proyecto/ubuntu-25.04.box
```

Después de que el comando se complete, se puede verificar que el box ha sido añadido correctamente al inventario local con el comando vagrant box list.[19] La salida debería mostrar ubuntu/25.04 junto con su proveedor (virtualbox) y la versión (generalmente 0 si no se especifica).

Tópico Avanzado: Nomenclatura y Versionado de Boxes

Para uso personal, un nombre simple como ubuntu/25.04 es suficiente. Sin embargo, en un entorno de equipo o para una gestión más avanzada, el control de versiones del box (por ejemplo, 1.0.0, 1.1.0 después de aplicar actualizaciones) es esencial. El comando vagrant box

add para archivos locales no permite especificar una versión directamente.
La solución profesional para esto es crear un archivo metadata.json simple junto al archivo .box.[22] Este archivo de metadatos permite definir explícitamente el nombre, las versiones y los proveedores del box, permitiendo una gestión de versiones adecuada incluso para boxes locales.
Un ejemplo de metadata.json sería:

JSON

```
{
  "name": "mi-organizacion/ubuntu-25.04",
  "versions": [
    {
      "version": "1.0.0",
      "providers": [
        {
          "name": "virtualbox",
          "url": "file:///C:/ruta/a/su/proyecto/ubuntu-25.04.box"
        }
      ]
    }
  ]
}
```

Luego, el box se añadiría usando la ruta al archivo JSON en lugar del archivo .box:

PowerShell

```
vagrant box add metadata.json
```

Este enfoque, aunque más elaborado, es la práctica recomendada para la distribución y gestión de boxes en un entorno profesional.

3.3 El Lanzamiento Inaugural

Con el box personalizado añadido al inventario, es hora de lanzarlo.

1. **Crear un Directorio de Proyecto**: Cree una nueva carpeta para este proyecto de Vagrant.
2. **Inicializar el Vagrantfile**: Dentro de la nueva carpeta, cree un archivo llamado Vagrantfile con el siguiente contenido mínimo, asegurándose de que el valor de

config.vm.box coincida exactamente con el nombre lógico que le dio al box en el paso anterior.[23]

```
Ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/25.04"
end
```

3. **Arrancar la Máquina**: Desde la terminal en el directorio del proyecto, ejecute vagrant up. Vagrant leerá el Vagrantfile, encontrará el box ubuntu/25.04 en su inventario local, lo importará a VirtualBox y lo arrancará.
4. **Conectar y Verificar**: Una vez que la máquina esté en funcionamiento, conéctese a ella a través de SSH con vagrant ssh. Debería obtener un prompt de shell dentro de su VM Ubuntu 25.04.

Como validación final del ciclo de vida de la clave SSH discutido anteriormente, se puede inspeccionar el archivo de clave privada generado por Vagrant en el anfitrión. Este se encontrará en .vagrant/machines/default/virtualbox/private_key dentro del directorio del proyecto. El contenido de esta clave será diferente de la clave insegura estándar de Vagrant, proporcionando una prueba tangible de que el mecanismo de reemplazo de clave ha funcionado como se esperaba.

Parte IV: Automatización de la Configuración del Entorno con Aprovisionamiento

Habiendo creado y lanzado con éxito un box base personalizado, el siguiente paso es aprovechar el poder de la automatización de Vagrant para configurar el entorno de la VM dinámicamente en el momento de la creación. Esta sección se centra en cumplir el requisito de crear un nuevo usuario con una contraseña configurable directamente desde el Vagrantfile, utilizando el aprovisionador de shell.

4.1 El Poder del Aprovisionador de Shell

El aprovisionamiento es el proceso mediante el cual Vagrant instala software, modifica configuraciones y realiza otras tareas de configuración dentro de la máquina invitada después de que esta arranca por primera vez.[24] Vagrant soporta múltiples aprovisionadores, desde sistemas de gestión de configuración completos como Ansible o Puppet hasta el más simple y directo: el aprovisionador de shell.[25]

El aprovisionador de shell permite ejecutar scripts de shell (Bash, en el caso de Linux) en la máquina invitada. Estos scripts pueden ser "en línea" (definidos directamente en el Vagrantfile) o externos (un archivo .sh que reside en la máquina anfitriona). Para la tarea de crear un usuario, un script externo es más limpio, modular y reutilizable.

4.2 Desarrollo de un Script de Creación de Usuarios Reutilizable

Se creará un script de Bash simple pero robusto llamado create_user.sh en el mismo directorio que el Vagrantfile. Este script aceptará dos argumentos desde la línea de comandos: el nombre de usuario a crear y la contraseña a establecer.
El script utilizará comandos estándar de Linux: useradd para crear la cuenta de usuario con su directorio personal, y chpasswd para establecer la contraseña de forma no interactiva, lo cual es esencial para la automatización.[26]
Versión Inicial del Script (create_user.sh):

Bash

```
#!/bin/bash

# Asignar argumentos a variables para mayor claridad
USERNAME=$1
PASSWORD=$2

# Crear el usuario con un directorio home (-m) y shell bash (-s)
sudo useradd -m -s /bin/bash "$USERNAME"

# Establecer la contraseña para el nuevo usuario
echo "${USERNAME}:${PASSWORD}" | sudo chpasswd

# Opcional: Añadir el nuevo usuario al grupo 'sudo' para darle privilegios de administrador
sudo usermod -aG sudo "$USERNAME"

echo "Usuario '$USERNAME' creado con éxito."
```

Mejora del Script: Idempotencia

Un principio fundamental en la automatización de la infraestructura es la **idempotencia**. Un script o proceso idempotente es aquel que puede ejecutarse múltiples veces sobre el mismo sistema y siempre producirá el mismo estado final, sin generar errores en ejecuciones

posteriores. El script anterior no es idempotente; si se ejecuta una segunda vez (por ejemplo, con vagrant provision), el comando useradd fallará porque el usuario ya existe.
Para convertirlo en una herramienta de nivel profesional, se debe modificar el script para que verifique si el usuario ya existe antes de intentar crearlo.
Versión Idempotente y Robusta del Script (create_user.sh):

Bash

```
#!/bin/bash

USERNAME=$1
PASSWORD=$2

# Verificar si el usuario ya existe
if id -u "$USERNAME" >/dev/null 2>&1; then
  echo "El usuario '$USERNAME' ya existe. Omitiendo la creación."
else
  echo "Creando el usuario '$USERNAME'."
  sudo useradd -m -s /bin/bash "$USERNAME"
  echo "${USERNAME}:${PASSWORD}" | sudo chpasswd
  sudo usermod -aG sudo "$USERNAME"
  echo "Usuario '$USERNAME' creado con éxito."
fi
```

Esta versión mejorada es mucho más segura y fiable para su uso en flujos de trabajo de aprovisionamiento repetibles.

4.3 Un Vagrantfile Dinámico

Ahora, se modificará el Vagrantfile para ejecutar este script. Para que sea fácilmente configurable, el nombre de usuario y la contraseña se definirán como variables de Ruby en la parte superior del archivo.
Se utilizará la configuración del aprovisionador de shell, especificando la ruta al script externo con la opción path. Los valores de las variables de Ruby se pasarán al script como argumentos utilizando la opción args.[25]
Contenido del Vagrantfile Configurable:

Ruby

```
# -*- mode: ruby -*-
```

```
# vi: set ft=ruby :

# --- Ajustes de Usuario Configurables ---
# Modifique estos valores para cambiar el usuario a crear
NEW_USERNAME = "david"
NEW_PASSWORD = "david"
# ------------------------------------

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/25.04"

  # Aprovisionador para crear un usuario dinámicamente
  config.vm.provision "shell",
    path: "create_user.sh",
    args:
end
```

Con esta configuración, al ejecutar vagrant up en un proyecto nuevo, Vagrant arrancará la VM desde el box base y luego ejecutará automáticamente el script create_user.sh, pasándole "david" y "david" como argumentos. Esto creará el usuario david con su contraseña correspondiente. Si se necesita crear un usuario diferente, solo es necesario modificar las variables NEW_USERNAME y NEW_PASSWORD en el Vagrantfile y volver a aprovisionar con vagrant provision o vagrant reload --provision.

Parte V: Asegurando el Acceso con una Clave SSH Personalizada

Esta sección final aborda el requisito más avanzado: configurar Vagrant para que utilice un par de claves SSH preexistente de la máquina anfitriona de Windows para la autenticación, eludiendo por completo la gestión de claves predeterminada de Vagrant. Esto proporciona un control total sobre el acceso SSH y alinea el entorno de Vagrant con las prácticas de seguridad estándar que utilizan claves personales.

5.1 Deconstrucción de la Autenticación SSH de Vagrant

Para implementar con éxito la autenticación con clave personalizada, es crucial comprender los diferentes mecanismos de SSH que Vagrant emplea. La interacción entre la clave insegura, el par de claves generado automáticamente y las directivas del Vagrantfile como config.ssh.insert_key y config.ssh.private_key_path es compleja pero fundamental.[16]

Se pueden identificar tres modos principales de operación SSH en Vagrant:

1. **Modo de Arranque (Bootstrap)**: Utiliza la clave privada insegura global (~/.vagrant.d/insecure_private_key) para la conexión inicial a cualquier box base público, que contiene la clave pública insegura correspondiente.
2. **Modo Seguro Predeterminado**: Este es el comportamiento por defecto (config.ssh.insert_key es true). En el primer vagrant up, Vagrant se conecta usando la clave insegura, genera un nuevo par de claves único para la instancia de la VM, inserta la nueva clave pública en authorized_keys en el invitado y elimina la clave insegura. La nueva clave privada se almacena en el anfitrión en .vagrant/machines/default/virtualbox/private_key.[17]
3. **Modo de Clave Personalizada**: El usuario toma el control total. Se deshabilita la inserción automática de claves y se especifica explícitamente qué clave privada debe usar Vagrant. El usuario es entonces responsable de asegurar que la clave pública correspondiente esté presente en la máquina invitada.

La siguiente tabla compara estos tres modos para clarificar sus diferencias:

Característica	Modo de Arranque (Conexión Inicial)	Modo Seguro Predeterminado	Modo de Clave Personalizada
config.ssh.insert_key	true (implícito)	true (predeterminado)	false (explícito)
config.ssh.private_key_path	Predeterminado (apunta a la clave insegura)	Predeterminado (ignorado en favor de la clave generada)	Ruta a la clave privada del usuario (ej. ~/.ssh/id_rsa)
Clave Privada Usada para Conexión	Clave insegura de Vagrant	Clave generada por Vagrant para la instancia	Clave privada especificada por el usuario
Estado de authorized_keys en el Invitado	Contiene la clave pública insegura	La clave pública insegura es reemplazada por la clave pública generada	Gestionado manualmente por el usuario (vía aprovisionamiento)

5.2 Configuración del Vagrantfile para Autenticación Personalizada

Para habilitar el modo de clave personalizada, se deben realizar dos cambios cruciales en el Vagrantfile.

1. **Deshabilitar la Inserción de Claves**: La directiva config.ssh.insert_key = false es el interruptor maestro. Le indica a Vagrant que no debe realizar su proceso automático de generación e inserción de claves.[16] Vagrant confiará en que la clave especificada en private_key_path ya está autorizada en la máquina invitada.
2. **Especificar la Ruta de la Clave Privada**: La directiva config.ssh.private_key_path debe apuntar a la ubicación de la clave privada del usuario en la máquina anfitriona de

Windows.[32] Para maximizar la portabilidad y evitar problemas con las barras invertidas de Windows, es una buena práctica usar la expansión de ruta de Ruby.

La configuración en el Vagrantfile se vería así:

Ruby

```
config.ssh.insert_key = false
config.ssh.private_key_path = [File.expand_path("~/.ssh/id_rsa")]
```

5.3 Aprovisionamiento de la Clave Pública

Con la gestión automática de claves de Vagrant deshabilitada, ahora es nuestra responsabilidad colocar la clave pública del usuario en el archivo ~/.ssh/authorized_keys de la máquina invitada. Este paso es el que realmente concede el acceso.

Un método robusto y claro para lograr esto es un proceso de dos pasos que utiliza tanto el aprovisionador de archivos como el de shell. El aprovisionador de archivos puede tener problemas de permisos al intentar escribir directamente en directorios protegidos. Por lo tanto, una estrategia más fiable es:

1. Usar el aprovisionador file para copiar la clave pública del anfitrión a una ubicación temporal y de escritura pública en el invitado, como /tmp/.
2. Usar un aprovisionador shell (que se ejecuta con privilegios de sudo por defecto) para mover la clave desde /tmp/ a la ubicación final (~/.ssh/authorized_keys), y luego establecer los permisos y la propiedad correctos.

El Vagrantfile completo que implementa la creación de usuario y la autenticación con clave personalizada sería el siguiente:

Ruby

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# --- Ajustes de Usuario Configurables ---
NEW_USERNAME = "david"
NEW_PASSWORD = "david"
# ----------------------------------------

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/25.04"
```

```
  # --- Configuración de SSH Personalizada ---
  config.ssh.insert_key = false
  config.ssh.private_key_path = [File.expand_path("~/.ssh/id_rsa")]

  # --- Aprovisionamiento ---

  # 1. Crear el usuario 'david'
  config.vm.provision "create_user", type: "shell",
    path: "create_user.sh",
    args:

  # 2. Copiar la clave pública del anfitrión al invitado
  config.vm.provision "file",
    source: File.expand_path("~/.ssh/id_rsa.pub"),
    destination: "/tmp/id_rsa.pub"

  # 3. Mover la clave pública a su lugar y establecer permisos
  config.vm.provision "install_ssh_key", type: "shell", inline: <<-SHELL
    # Asegurar que el directorio.ssh exista para el usuario vagrant
    mkdir -p /home/vagrant/.ssh
    chmod 700 /home/vagrant/.ssh
    # Añadir la clave al usuario vagrant y establecer permisos
    cat /tmp/id_rsa.pub >> /home/vagrant/.ssh/authorized_keys
    chmod 600 /home/vagrant/.ssh/authorized_keys
    chown -R vagrant:vagrant /home/vagrant/.ssh

    # Hacer lo mismo para el nuevo usuario 'david'
    mkdir -p /home/#{NEW_USERNAME}/.ssh
    chmod 700 /home/#{NEW_USERNAME}/.ssh
    cat /tmp/id_rsa.pub >> /home/#{NEW_USERNAME}/.ssh/authorized_keys
    chmod 600 /home/#{NEW_USERNAME}/.ssh/authorized_keys
    chown -R #{NEW_USERNAME}:#{NEW_USERNAME} /home/#{NEW_USERNAME}/.ssh
  SHELL
end
```

5.4 Validación Final

La prueba definitiva de que la configuración ha sido exitosa es conectarse a la VM sin usar el comando vagrant ssh, utilizando en su lugar un cliente SSH estándar desde la máquina anfitriona de Windows (por ejemplo, el cliente OpenSSH incluido en PowerShell o Git Bash).

1. **Destruir y Recrear el Entorno**: Para asegurar una prueba limpia, ejecute vagrant

destroy -f seguido de vagrant up.

2. **Obtener la Información de Conexión**: Vagrant todavía gestiona el reenvío de puertos. Ejecute vagrant port para ver a qué puerto del anfitrión se ha reenviado el puerto 22 del invitado.
   PowerShell
   > vagrant port
   ==> default: The forwarded ports for the machine are listed below. Please note that
   ==> default: forwarded ports can only be accessed from the machine running Vagrant.
   ==> default:
   ==> default: 22 (guest) => 2222 (host)

3. **Conectar Directamente**: Abra una terminal en Windows y utilice el comando ssh estándar, especificando la clave privada, el usuario, el host (127.0.0.1) y el puerto reenviado.
   PowerShell
   ssh -i C:/Users/YourUser/.ssh/id_rsa david@127.0.0.1 -p 2222

   (Asegúrese de reemplazar YourUser con su nombre de usuario de Windows).

Si la conexión se establece con éxito sin solicitar una contraseña, se ha logrado el objetivo final: la autenticación a la máquina Vagrant está ahora completamente controlada por su par de claves SSH personal.

Conclusiones

Este informe ha detallado el proceso integral de creación de un entorno Vagrant personalizado desde sus componentes más básicos. Siguiendo los pasos descritos, se ha pasado de una imagen ISO de un sistema operativo a un box de Vagrant reutilizable, y de ahí a un entorno de desarrollo dinámicamente aprovisionado y asegurado con credenciales SSH personalizadas.
Los puntos clave del proceso son:

1. **La Preparación es Fundamental**: La creación de un box base robusto requiere una configuración meticulosa dentro de la VM, incluyendo la creación del usuario vagrant, la configuración de sudo sin contraseña, la instalación de la clave insegura de Vagrant y la integración de las VirtualBox Guest Additions. Cada uno de estos pasos es un requisito previo para la automatización posterior.
2. **La Automatización Requiere Idempotencia**: Los scripts de aprovisionamiento deben diseñarse para ser idempotentes, permitiendo que se ejecuten de forma segura y repetida sin causar errores. Esta es una práctica esencial en la ingeniería de sistemas moderna.
3. **El Control de SSH es Posible y Deseable**: Aunque la gestión de claves predeterminada de Vagrant es conveniente, es posible y a menudo preferible tomar el

control total de la autenticación SSH. Comprender la interacción entre config.ssh.insert_key y config.ssh.private_key_path es la clave para implementar una solución de acceso segura y personalizada que se alinee con las prácticas de seguridad estándar.

Al dominar estas técnicas, los usuarios de Vagrant pueden trascender las limitaciones de los boxes públicos y construir entornos de desarrollo que se adapten con precisión a sus necesidades, mejorando la reproducibilidad, la seguridad y la eficiencia en sus flujos de trabajo.

Obras citadas

1. How to Install, Use and Uninstall Vagrant on Ubuntu 24.04 - Greenwebpage Community, fecha de acceso: octubre 21, 2025, https://greenwebpage.com/community/how-to-install-use-and-uninstall-vagrant-on-ubuntu-24-04/
2. Building a Vagrant Box from Start to Finish - EngineYard, fecha de acceso: octubre 21, 2025, https://www.engineyard.com/blog/building-a-vagrant-box-from-start-to-finish/
3. Get Ubuntu | Download, fecha de acceso: octubre 21, 2025, https://ubuntu.com/download
4. Ubuntu 25.04 (Plucky Puffin), fecha de acceso: octubre 21, 2025, https://releases.ubuntu.com/plucky/
5. Ubuntu 25.04 (Plucky Puffin) - Index of / - USP, fecha de acceso: octubre 21, 2025, http://sft.if.usp.br/releases/releases/25.04/
6. Download | Ubuntu MATE, fecha de acceso: octubre 21, 2025, https://ubuntu-mate.org/download/
7. tylert/packer-build: Packer Automated VM Image and Vagrant Box Builds - GitHub, fecha de acceso: octubre 21, 2025, https://github.com/tylert/packer-build
8. How to run an Ubuntu Desktop virtual machine using VirtualBox 7 ..., fecha de acceso: octubre 21, 2025, https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox
9. How to Install Ubuntu 25.04 on VirtualBox: A Step-by-Step Guide - kifarunix.com, fecha de acceso: octubre 21, 2025, https://kifarunix.com/install-ubuntu-25-04-on-virtualbox-step-by-step-guide/
10. How to install Ubuntu on VirtualBox?, fecha de acceso: octubre 21, 2025, https://askubuntu.com/questions/142549/how-to-install-ubuntu-on-virtualbox
11. Creating a Base Box - VirtualBox Provider | Vagrant - HashiCorp Developer, fecha de acceso: octubre 21, 2025, https://developer.hashicorp.com/vagrant/docs/providers/virtualbox/boxes
12. Create Vagrant Base Box from Scratch, fecha de acceso: octubre 21, 2025, https://mudongliang.github.io/2018/08/10/create-vagrant-base-box-from-scratch.html
13. Writing a Vagrantfile and making a base box | by Brian Seel - Medium, fecha de acceso: octubre 21, 2025,

https://medium.com/@brian.seel/writing-a-vagrantfile-and-making-a-base-box-9a9e011675c4
14. How to install Ubuntu 25.04 Server edition on VirtualBox Step by step guide in 8 MInutes|2025 Update - YouTube, fecha de acceso: octubre 21, 2025, https://www.youtube.com/watch?v=m_MVreU38IY
15. How to use local image as a vagrant box - Reddit, fecha de acceso: octubre 21, 2025, https://www.reddit.com/r/vagrant/comments/46p6t8/how_to_use_local_image_as_a_vagrant_box/
16. Why does vagrant use 2 ssh private keys? - Stack Overflow, fecha de acceso: octubre 21, 2025, https://stackoverflow.com/questions/46256727/why-does-vagrant-use-2-ssh-private-keys
17. Vagrant cloud, box and insecure key pairs, fecha de acceso: octubre 21, 2025, https://mudongliang.github.io/2023/07/14/vagrant-cloud-box-and-insecure-key-pairs.html
18. vagrant package - Command-Line Interface - HashiCorp Developer, fecha de acceso: octubre 21, 2025, https://developer.hashicorp.com/vagrant/docs/cli/package
19. Vagrant: Create local box - GitHub Gist, fecha de acceso: octubre 21, 2025, https://gist.github.com/kekru/a76ba9d0592ce198f09f6ba0cefa5afb
20. windows - How to add a downloaded .box file to Vagrant? - Stack ..., fecha de acceso: octubre 21, 2025, https://stackoverflow.com/questions/22065698/how-to-add-a-downloaded-box-file-to-vagrant
21. Vagrant Cheat Sheet - gist/GitHub, fecha de acceso: octubre 21, 2025, https://gist.github.com/wpscholar/a49594e2e2b918f4d0c4
22. Is there a way for Vagrant to import a local box file with a specified version? - Super User, fecha de acceso: octubre 21, 2025, https://superuser.com/questions/810071/is-there-a-way-for-vagrant-to-import-a-local-box-file-with-a-specified-version
23. Run a Vagrant box - Ubuntu Public Images documentation, fecha de acceso: octubre 21, 2025, https://documentation.ubuntu.com/public-images/public-images-how-to/run-a-vagrant-box/
24. Basic Usage - Provisioning | Vagrant - HashiCorp Developer, fecha de acceso: octubre 21, 2025, https://developer.hashicorp.com/vagrant/docs/provisioning/basic_usage
25. Shell Scripts - Provisioning | Vagrant | HashiCorp Developer, fecha de acceso: octubre 21, 2025, https://developer.hashicorp.com/vagrant/docs/provisioning/shell
26. Automating User Account Creation with Password in Shell ..., fecha de acceso: octubre 21, 2025, https://www.baeldung.com/linux/automate-user-account-creation
27. How to automatically add user account AND password with a Bash script? - Stack Overflow, fecha de acceso: octubre 21, 2025,

https://stackoverflow.com/questions/2150882/how-to-automatically-add-user-account-and-password-with-a-bash-script
28. How to create user and password in one script for 100+ users, fecha de acceso: octubre 21, 2025, https://unix.stackexchange.com/questions/419063/how-to-create-user-and-password-in-one-script-for-100-users
29. Vagrant documentation about the default insecure key, is still not updated #8058 - GitHub, fecha de acceso: octubre 21, 2025, https://github.com/hashicorp/vagrant/issues/8058
30. new to Vagrant, getting ssh private key errors - Google Groups, fecha de acceso: octubre 21, 2025, https://groups.google.com/g/vagrant-up/c/FeXfcxpQlVA
31. How to disable ssh when running vagrant up - Stack Overflow, fecha de acceso: octubre 21, 2025, https://stackoverflow.com/questions/49217621/how-to-disable-ssh-when-running-vagrant-up
32. config.ssh.private_key_path - Vagrant, fecha de acceso: octubre 21, 2025, https://friendsofvagrant.github.io/v1/docs/config/ssh/private_key_path.html
33. config.ssh - Vagrantfile | Vagrant - HashiCorp Developer, fecha de acceso: octubre 21, 2025, https://developer.hashicorp.com/vagrant/docs/vagrantfile/ssh_settings