# Dominando SSH: Una Guía Práctica para la Administración Remota Segura

## Introducción: El Cimiento de la Administración Remota

En el panorama tecnológico actual, la capacidad de gestionar sistemas de forma remota no es un lujo, sino una necesidad fundamental. Los servidores que impulsan nuestras aplicaciones, sitios web y servicios rara vez se encuentran debajo de nuestro escritorio; residen en centros de datos distribuidos por todo el mundo o en plataformas de computación en la nube.1 Para administrar, configurar y mantener estos sistemas, necesitamos una forma de comunicarnos con ellos de manera segura a través de redes que, por naturaleza, no son seguras, como Internet. La herramienta estándar de la industria para esta tarea es el protocolo SSH, o Secure Shell.

SSH es un protocolo de red criptográfico que permite a los usuarios operar servicios de red de forma segura sobre una red no segura.3 Proporciona un canal cifrado para la comunicación, garantizando la confidencialidad, integridad y autenticación de los datos transmitidos. Para comprender verdaderamente el valor de SSH, es instructivo mirar a su predecesor: Telnet.

### El "Porqué": Una Historia de Dos Protocolos

Antes de que SSH se convirtiera en el estándar, Telnet era el método principal para el acceso remoto a la línea de comandos. Desarrollado en 1969, en una era de redes académicas de alta confianza, Telnet fue revolucionario para su tiempo.5 Sin embargo, tiene un defecto fatal en el contexto de la Internet moderna: transmite toda la información, incluyendo nombres de usuario y contraseñas, en texto plano, sin ningún tipo de cifrado.7

Esto significa que cualquier persona con una herramienta de "sniffing" de red, como Wireshark, posicionada en cualquier punto entre el cliente y el servidor, puede interceptar y leer cada pulsación de tecla.2 Un atacante podría capturar fácilmente las credenciales de inicio de sesión y obtener acceso no autorizado al sistema, con consecuencias potencialmente catastróficas. Por esta razón, Telnet se considera obsoleto e inseguro para la administración remota a través de cualquier red no confiable y nunca debe usarse para este propósito en la actualidad.5

La propia existencia de SSH es una respuesta directa a esta vulnerabilidad. Fue creado en 1995 por Tatu Ylönen después de que presenciara un ataque de "sniffing" de contraseñas en la red de su universidad.4 Este origen subraya una lección fundamental: las herramientas de seguridad no son arbitrarias; son soluciones nacidas de amenazas del mundo real y de la evolución de los entornos de red. El paso de Telnet a SSH no fue simplemente una actualización tecnológica, sino un cambio de paradigma fundamental en la seguridad de la administración de redes, reconociendo que la Internet global es, por defecto, un entorno de baja confianza.

Para cristalizar las diferencias, la siguiente tabla ofrece una comparación directa.

**Tabla 1: SSH vs. Telnet - Una Instantánea de Seguridad**

| Característica | Telnet | SSH (Secure Shell) |
| --- | --- | --- |
| **Seguridad** | Inseguro. No proporciona cifrado. | Altamente seguro. Utiliza algoritmos de cifrado robustos. |
| **Transmisión de Datos** | Texto plano, vulnerable a la interceptación. | Cifrado, protegiendo la confidencialidad e integridad. |
| **Autenticación** | Envía credenciales en texto plano. | Utiliza criptografía de clave pública para una autenticación fuerte. |
| **Puerto por Defecto** | TCP 23 | TCP 22 |
| **Uso Moderno** | Obsoleto para acceso remoto; limitado a redes privadas y seguras o depuración de puertos. | Estándar de la industria para toda la administración remota segura. |

Datos compilados de.5

Este documento servirá como una guía exhaustiva y práctica para dominar SSH. Comenzaremos con los principios criptográficos que lo hacen seguro, pasaremos a la generación y gestión de identidades digitales, construiremos un entorno de laboratorio práctico con Vagrant para aplicar estos conocimientos y, finalmente, conectaremos esta habilidad fundamental con el mundo de la automatización de infraestructuras.

## Sección 1: Desmitificando la Magia - Los Principios Fundamentales de SSH

Para utilizar SSH de manera efectiva, es esencial comprender los dos pilares sobre los que se construye: su arquitectura cliente-servidor y la criptografía asimétrica que garantiza su seguridad. Estos conceptos pueden parecer abstractos al principio, pero son sorprendentemente intuitivos una vez que se desglosan.

### 1.1. La Danza Cliente-Servidor

Toda conexión SSH involucra dos actores principales que se comunican a través de la red 10:

* **El Cliente SSH:** Es el programa que se ejecuta en su máquina local, la que está utilizando para iniciar la conexión. Es la herramienta que envía sus comandos al sistema remoto. Ejemplos comunes incluyen el comando ssh de OpenSSH, que viene preinstalado en prácticamente todos los sistemas Linux y macOS, y clientes como PuTTY para Windows.10
* **El Servidor SSH:** Es un programa, a menudo un servicio o demonio llamado sshd, que se ejecuta constantemente en la máquina remota. Su trabajo es escuchar las solicitudes de conexión entrantes, típicamente en el puerto TCP 22, y gestionar el proceso de autenticación y la sesión segura.3

El flujo es simple: el cliente inicia una solicitud de conexión al servidor. El servidor responde, y juntos negocian los parámetros de la conexión segura, autentican al usuario y establecen un canal cifrado a través del cual se pueden enviar comandos y recibir respuestas.

### 1.2. El Lenguaje del Secreto: Una Introducción a la Criptografía de Clave Pública

Aquí es donde reside la "magia" de SSH. En lugar de depender de una contraseña compartida que debe transmitirse a través de la red, SSH utiliza un sistema mucho más elegante y seguro conocido como **criptografía de clave pública** o **criptografía asimétrica**. Se llama asimétrica porque utiliza un par de claves diferentes pero matemáticamente relacionadas para realizar sus funciones.14

#### El Par de Claves

Cuando generas una identidad SSH, en realidad estás creando dos archivos de claves:

* **La Clave Privada (ej. id\_ed25519):** Esta es tu posesión más secreta. Es el equivalente digital de la única llave física que abre tu casa. **Nunca, bajo ninguna circunstancia, debe ser compartida**.12 Su propósito es demostrar tu identidad sin revelarse.
* **La Clave Pública (ej. id\_ed25519.pub):** Esta es la contraparte que puedes compartir libremente. Piensa en ella como una cerradura de alta seguridad que puedes instalar en las puertas de todos los servidores a los que deseas acceder.12 Cualquiera puede tener una copia de tu cerradura (la clave pública), pero solo tú tienes la llave (la clave privada) que puede abrirla.

#### Analogía: El Buzón Seguro

Para visualizar cómo funciona este sistema, imaginemos un buzón con dos propiedades especiales:

1. **Confidencialidad:** La ranura del buzón (tu **clave pública**) está abierta. Cualquiera que la tenga puede depositar un mensaje cifrado dentro. Sin embargo, una vez dentro, el buzón se cierra herméticamente. La única forma de abrirlo y leer el mensaje es usando tu **clave privada** única. Esto garantiza que solo el destinatario previsto pueda descifrar la comunicación.15
2. **Autenticación y Firma Digital:** El sistema también funciona a la inversa, que es el mecanismo principal utilizado para el inicio de sesión SSH. Puedes usar tu **clave privada** para "firmar" digitalmente un mensaje. Cualquiera que tenga tu **clave pública** puede verificar esa firma. Si la verificación es exitosa, prueba dos cosas de manera irrefutable: que el mensaje provino de ti (autenticidad) y que no ha sido alterado en tránsito (integridad). Esto se debe a que solo el poseedor de la clave privada podría haber creado esa firma específica.14

Durante un inicio de sesión SSH, el servidor, que ya tiene tu clave pública, te envía un "desafío" (un mensaje aleatorio). Tu cliente SSH usa tu clave privada para "firmar" este desafío y devuelve la respuesta. El servidor usa tu clave pública para verificar la firma. Si coincide, el servidor sabe que te estás conectando con un cliente que posee la clave privada correspondiente y te concede el acceso. Tu clave privada nunca abandona tu máquina.20

Este sistema resuelve elegantemente el problema fundamental de la autenticación en redes no seguras: cómo probar quién eres sin enviar tu secreto (la contraseña o la clave privada) a través de un canal que podría ser interceptado.16

La genialidad de este enfoque radica en la separación de dos actos distintos: el de *conceder acceso* y el de *probar la identidad*. En un sistema tradicional de contraseñas, estos dos actos están fusionados; conocer la contraseña lo es todo. En SSH, colocar la clave pública en un servidor es un acto administrativo único para conceder acceso a un titular de identidad. El posterior proceso de desafío-respuesta criptográfico durante cada inicio de sesión es el acto de probar esa identidad. Este desacoplamiento es una mejora de seguridad profunda. Si un atacante comprometiera el servidor y robara el archivo que contiene las claves públicas autorizadas, no ganaría la capacidad de suplantar a los usuarios, porque no posee sus claves privadas. Esta resiliencia es un principio de seguridad central que a menudo se pasa por alto en las explicaciones básicas.

## Sección 2: El Kit de Herramientas SSH - Generando y Gestionando tu Identidad Digital

Con los conceptos teóricos claros, es hora de pasar a la práctica. Crear y gestionar tus claves SSH es el primer paso práctico para establecer conexiones seguras. Esta sección te guiará a través de las herramientas y las mejores prácticas para manejar tu identidad digital de manera profesional.

### 2.1. Forjando tus Claves con ssh-keygen

La herramienta estándar en sistemas basados en Unix (Linux, macOS) para crear pares de claves SSH es ssh-keygen. Es un programa de línea de comandos que te guiará a través del proceso.

Para generar un par de claves moderno y seguro, se recomienda utilizar el algoritmo **Ed25519**. Es más rápido y ofrece una seguridad superior en comparación con el antiguo estándar RSA.22

Abre una terminal en tu máquina local y ejecuta el siguiente comando:

Bash

ssh-keygen -t ed25519 -C "tu\_email@ejemplo.com"

* -t ed25519: Especifica el tipo (type) de clave a crear.
* -C "...": Añade un comentario (Comment) a la clave. Es una buena práctica usar tu correo electrónico para identificar fácilmente a quién pertenece la clave.

El comando iniciará un diálogo interactivo:

Bash

Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/usuario/.ssh/id\_ed25519):

1. **Ubicación del archivo:** El programa te sugiere una ubicación y nombre de archivo por defecto: el directorio oculto .ssh dentro de tu carpeta de inicio. Esta es la convención estándar y se recomienda aceptarla presionando Enter, a menos que tengas una razón específica para cambiarla.25

Bash

Enter passphrase (empty for no passphrase):  
Enter same passphrase again:

1. **Frase de contraseña (Passphrase):** Este es un paso de seguridad crucial. Una frase de contraseña **cifra tu clave privada en tu disco local**. Esto añade una segunda capa de protección. Si alguien robara tu ordenador y obtuviera el archivo de tu clave privada, no podría usarla sin conocer también esta frase de contraseña.24 Es muy recomendable establecer una frase de contraseña fuerte y única.

Una vez completado, ssh-keygen mostrará una confirmación y un "randomart" visual de tu clave:

Bash

Your identification has been saved in /home/usuario/.ssh/id\_ed25519  
Your public key has been saved in /home/usuario/.ssh/id\_ed25519.pub  
The key fingerprint is:  
SHA256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx tu\_email@ejemplo.com  
The key's randomart image is:  
+----+  
  
|.o+o. |  
|. =.o. |  
|. \* +.. |  
| o B... |  
| S = . |  
|. \*.. |  
| o =. |  
| + E. |  
|.. |  
+---------+

Ahora tienes dos nuevos archivos en tu directorio ~/.ssh/:

* id\_ed25519: Tu clave privada. ¡Mantenla segura!
* id\_ed25519.pub: Tu clave pública. Esta es la que compartirás. 28

### 2.2. El Anillo del Maestro de Llaves: Usando ssh-agent

Si estableciste una frase de contraseña, te darás cuenta de que tienes que escribirla cada vez que usas tu clave SSH. Esto es seguro, pero puede volverse tedioso. Para solucionar esto, existe ssh-agent.

ssh-agent es un programa auxiliar que se ejecuta en segundo plano y mantiene tu clave privada descifrada en la memoria durante tu sesión de inicio de sesión. De esta manera, solo necesitas introducir tu frase de contraseña una vez, cuando añades la clave al agente.29

Para usarlo, sigue estos pasos:

1. **Iniciar el agente:**  
   Bash  
   eval "$(ssh-agent -s)"  
     
   Este comando inicia el agente y establece las variables de entorno necesarias para que otros programas puedan encontrarlo.
2. **Añadir tu clave al agente:**  
   Bash  
   ssh-add ~/.ssh/id\_ed25519  
     
   Se te pedirá tu frase de contraseña. Una vez introducida, la clave se carga en el agente.

A partir de este momento, cualquier conexión SSH que realices utilizará la clave del agente y no te volverá a pedir la frase de contraseña hasta que reinicies tu sistema o termines la sesión del agente.

### 2.3. Higiene de Seguridad Profesional

Un par de claves SSH no es solo un archivo; es una credencial de identidad digital de larga duración. Debe ser gestionada con la misma seriedad que un pasaporte o un documento de identidad oficial. A diferencia de una contraseña, que se puede cambiar fácilmente, una clave SSH se despliega en potencialmente cientos de servidores. Revocarla es un proceso mucho más complejo. Por lo tanto, proteger el origen (la clave privada) es de suma importancia. Adoptar una mentalidad de "identidad" en lugar de "archivo de inicio de sesión" desde el principio inculca una disciplina de seguridad profesional.

* **Regla de Oro:** Nunca compartas tu clave privada. No la envíes por correo electrónico, no la subas a un repositorio de código, no la copies en una unidad USB no cifrada.17
* **Compartimentación:** Utiliza diferentes pares de claves para diferentes entornos (por ejemplo, uno para proyectos personales, otro para el trabajo, otro para un cliente específico). Esto limita el "radio de explosión" si una clave se ve comprometida. Un atacante que obtenga tu clave personal no podrá acceder a tus sistemas de trabajo.27
* **Rotación de Claves:** Es una buena práctica de seguridad generar periódicamente nuevos pares de claves y retirar los antiguos. La frecuencia depende de la sensibilidad del entorno. Para uso personal, rotar las claves cada uno o dos años es razonable. En entornos empresariales de alta seguridad, la rotación puede ser trimestral o semestral. Esto mitiga el riesgo de que una clave comprometida sin ser detectada permanezca válida indefinidamente.27

## Sección 3: El Laboratorio Práctico - Construyendo tu Patio de Juegos Seguro con Vagrant

La teoría es esencial, pero la verdadera comprensión proviene de la práctica. Para evitar el problema común de "funciona en mi máquina, pero no en la tuya", utilizaremos Vagrant para crear un entorno de laboratorio estandarizado, simple y reproducible. Vagrant es una herramienta que automatiza la creación y configuración de máquinas virtuales, asegurando que todos en la clase trabajen con una configuración idéntica.33

El valor principal de Vagrant en este contexto no es solo la automatización, sino la *abstracción y la consistencia*. Resuelve el problema pedagógico eliminando variables irrelevantes como configuraciones de red locales, reglas de firewall o versiones de software conflictivas. Vagrant crea un "laboratorio en una caja" autocontenido. La red privada que definiremos asegura que la conectividad de red sea idéntica para cada estudiante, permitiendo que la lección se centre exclusivamente en los conceptos de SSH que se están enseñando, en lugar de en la frustrante resolución de problemas ambientales.

### 3.1. El Plano: Tu Vagrantfile Multi-Máquina

En un directorio de tu elección en tu máquina local, crea un archivo llamado Vagrantfile (sin extensión) y pega el siguiente contenido. Este archivo es el "plano" que Vagrant usará para construir nuestro laboratorio.

Ruby

# -\*- mode: ruby -\*-  
# vi: set ft=ruby :  
  
# La versión de la API de Vagrantfile, siempre debe ser "2"  
Vagrant.configure("2") do |config|  
  
 # Definimos nuestra primera máquina virtual, la llamaremos 'controller'.  
 # Esta será la máquina desde la que iniciaremos nuestras conexiones SSH.  
 config.vm.define "controller" do |controller|  
 # Especificamos la imagen base del sistema operativo que usaremos.  
 # 'ubuntu/focal64' corresponde a Ubuntu 20.04 LTS.  
 controller.vm.box = "ubuntu/focal64"  
  
 # Asignamos un nombre de host a la máquina para identificarla fácilmente.  
 controller.vm.hostname = "controller"  
  
 # Creamos una red privada. Esto permite la comunicación entre las VMs  
 # y entre la máquina anfitriona y las VMs, pero no es accesible desde el exterior.  
 controller.vm.network "private\_network", ip: "192.168.56.10"  
 end  
  
 # Definimos nuestra segunda máquina virtual, el 'nodo' al que nos conectaremos.  
 config.vm.define "node1" do |node1|  
 node1.vm.box = "ubuntu/focal64"  
 node1.vm.hostname = "node1"  
 # Le asignamos una IP diferente dentro de la misma subred privada.  
 node1.vm.network "private\_network", ip: "192.168.56.11"  
 end  
end

Desglosemos las directivas clave:

* config.vm.define: Así es como Vagrant gestiona entornos con múltiples máquinas. Cada bloque define configura una VM separada.34
* vm.box: La imagen base del sistema operativo que Vagrant descargará y utilizará.33
* vm.hostname: Establece un nombre de host legible dentro de la VM.34
* vm.network "private\_network": Esta es la directiva más importante para nuestro laboratorio. Crea una red virtual aislada. Las dos VMs podrán comunicarse entre sí utilizando las direcciones IP estáticas que hemos definido (192.168.56.10 y 192.168.56.11), y tu máquina anfitriona también podrá comunicarse con ellas. Esta red no es accesible desde Internet, proporcionando un entorno seguro y consistente.39

### 3.2. Dando Vida al Laboratorio

Con el Vagrantfile guardado, abre una terminal en el mismo directorio y ejecuta los siguientes comandos:

1. **Crear e iniciar las VMs:**  
   Bash  
   vagrant up  
     
   La primera vez que ejecutes este comando, Vagrant descargará la imagen de Ubuntu (si no la tienes ya) y luego creará, configurará e iniciará ambas máquinas virtuales. Esto puede tardar unos minutos.
2. **Verificar el estado:**  
   Bash  
   vagrant status  
     
   Deberías ver una salida que indique que tanto controller como node1 están en estado running.
3. **Acceder a la máquina controller:**  
   Bash  
   vagrant ssh controller  
     
   Este comando te conectará por SSH a la máquina virtual controller. Notarás que no te pide contraseña. Esto se debe a que Vagrant gestiona automáticamente un par de claves SSH inseguras para esta conveniencia inicial. Tu terminal ahora mostrará el prompt de la VM, algo como vagrant@controller:~$.

### 3.3. El Primer Apretón de Manos (Basado en Contraseña)

Antes de introducir la complejidad de la autenticación por clave, es fundamental verificar que nuestras máquinas pueden comunicarse entre sí a través de la red privada que hemos definido. Este paso aísla posibles problemas de red de los problemas de autenticación.

Desde dentro de la VM controller (a la que te conectaste en el paso anterior), intenta conectarte a node1 usando su IP privada:

Bash

ssh vagrant@192.168.56.11

La primera vez que te conectes, verás una advertencia sobre la autenticidad del host. Hablaremos de esto en detalle más adelante. Por ahora, escribe yes y presiona Enter.

A continuación, se te pedirá una contraseña. Vagrant preconfigura todas sus VMs con un usuario vagrant cuya contraseña es vagrant. Escribe vagrant y presiona Enter.

Si todo ha ido bien, ¡felicidades! Ahora estarás en el prompt de node1 (vagrant@node1:~$). Has verificado con éxito la conectividad de red. Escribe exit para volver a la controller. Este primer apretón de manos, aunque inseguro por usar una contraseña conocida, es un paso de diagnóstico vital.

## Sección 4: El Camino hacia la Automatización - Implementando SSH sin Contraseña

Ahora que hemos confirmado la conectividad de red, el siguiente paso es eliminar la necesidad de contraseñas. La autenticación basada en claves (a menudo llamada "SSH sin contraseña") es la piedra angular de la administración de sistemas automatizada. Mostraremos tanto el método recomendado como el manual para asegurar una comprensión completa del proceso.

Primero, asegúrate de estar en la VM controller. Si seguiste la sección anterior, ya deberías estar allí. Si no, usa vagrant ssh controller desde tu máquina local. Ahora, genera un par de claves SSH dentro de la VM controller como aprendiste en la Sección 2:

Bash

ssh-keygen -t ed25519 -C "vagrant@controller"

Presiona Enter en todas las preguntas para aceptar los valores por defecto y no establecer una frase de contraseña por ahora (para simplificar el laboratorio).

### 4.1. Concediendo Acceso: El Rol de ~/.ssh/authorized\_keys

En el servidor (en nuestro caso, node1), hay un archivo especial que actúa como el guardián del acceso basado en claves: ~/.ssh/authorized\_keys. Este archivo es simplemente una lista de claves públicas. Cada clave pública en este archivo representa una identidad que está *autorizada* para iniciar sesión en esa cuenta de usuario sin una contraseña.17

Cuando un cliente intenta conectarse, el servidor comprueba si la clave pública del cliente está en este archivo. Si es así, procede con el desafío criptográfico para verificar que el cliente posee la clave privada correspondiente.17

### 4.2. El Camino Fácil: ssh-copy-id

La forma más sencilla y recomendada de instalar tu clave pública en un servidor remoto es usando la utilidad ssh-copy-id. Esta herramienta está diseñada específicamente para esta tarea y maneja todos los detalles por ti.28

Desde la VM controller, ejecuta el siguiente comando:

Bash

ssh-copy-id vagrant@192.168.56.11

Se te pedirá la contraseña del usuario vagrant en node1 por última vez. Escribe vagrant.

El comando ssh-copy-id realiza varias acciones importantes entre bastidores:

1. Se conecta al servidor remoto usando la contraseña proporcionada.
2. Comprueba si el directorio ~/.ssh existe en el servidor. Si no, lo crea con los permisos correctos (700).
3. Comprueba si el archivo ~/.ssh/authorized\_keys existe. Si no, lo crea con los permisos correctos (600).
4. Añade (anexa) el contenido de tu clave pública local (~/.ssh/id\_ed25519.pub) en una nueva línea dentro del archivo authorized\_keys en el servidor.25

La existencia de una herramienta como ssh-copy-id es un ejemplo del principio de "convención sobre configuración" y encapsula un flujo de trabajo de mejores prácticas de seguridad. Aunque el proceso manual es factible, tiene varios puntos de fallo, como olvidar crear el directorio, usar permisos incorrectos o sobrescribir accidentalmente el archivo. ssh-copy-id automatiza la secuencia correcta, haciendo el proceso no solo más rápido, sino también más seguro y fiable. Al aprender ambos métodos, no solo se entiende *cómo* funciona (manualmente), sino también la *forma profesional* de hacerlo (con la herramienta adecuada).

### 4.3. El Método Manual: Entendiendo la Mecánica

Para reforzar la comprensión de lo que acaba de suceder, realicemos el proceso manualmente. Esto es invaluable para la resolución de problemas y para entender los fundamentos.

1. **Paso 1 (Cliente controller):** Muestra el contenido de tu clave pública y cópialo al portapapeles.  
   Bash  
   cat ~/.ssh/id\_ed25519.pub  
     
   La salida será una larga cadena de caracteres que empieza con ssh-ed25519 AAAA.... Selecciona y copia toda la línea.
2. **Paso 2 (Servidor node1):** Conéctate a node1 usando la contraseña.  
   Bash  
   ssh vagrant@192.168.56.11  
     
   Introduce vagrant como contraseña.
3. **Paso 3 (Servidor node1):** Ahora, en node1, crea el directorio y el archivo necesarios y añade la clave pública.  
   Bash  
   # Crea el directorio.ssh si no existe y establece permisos estrictos  
   mkdir -p ~/.ssh  
   chmod 700 ~/.ssh  
     
   # Pega la clave pública copiada en el archivo authorized\_keys  
   # ¡Asegúrate de usar '>>' (anexar), no '>' (sobrescribir)!  
   echo "pega-aqui-la-cadena-de-la-clave-publica" >> ~/.ssh/authorized\_keys  
     
   # Establece permisos estrictos en el archivo authorized\_keys  
   chmod 600 ~/.ssh/authorized\_keys  
     
   La configuración de permisos con chmod es vital. El demonio SSH (sshd) es muy estricto por seguridad y se negará a usar el archivo authorized\_keys si sus permisos o los del directorio ~/.ssh son demasiado abiertos (es decir, si otros usuarios pueden escribir en ellos).22
4. **Paso 4 (Servidor node1):** Sal de la sesión de node1 para volver a controller.  
   Bash  
   exit

### 4.4. El Momento de la Verdad

Ya sea que hayas usado ssh-copy-id o el método manual, el resultado debería ser el mismo. Es hora de probarlo.

Desde tu VM controller, intenta conectarte a node1 de nuevo:

Bash

ssh vagrant@192.168.56.11

Si todo se configuró correctamente, serás conectado instantáneamente al prompt de node1 **sin que se te pida una contraseña**. Este es el resultado deseado y el requisito fundamental para la automatización.

## Sección 5: Resolución de Problemas y Flujos de Trabajo Profesionales

Incluso con una configuración correcta, te encontrarás con ciertos comportamientos de SSH que pueden ser confusos al principio. El más común es la advertencia sobre la "identificación del host remoto". Entender por qué ocurre y cómo manejarlo correctamente es una marca de competencia profesional.

### 5.1. "Confianza en el Primer Uso": El Archivo known\_hosts Explicado

Este archivo, ubicado en el **cliente** en ~/.ssh/known\_hosts, es un componente de seguridad crítico diseñado para protegerte contra **ataques de intermediario (Man-in-the-Middle, MITM)**.45

SSH opera bajo un modelo llamado "Confianza en el Primer Uso" (Trust on First Use, TOFU):

1. **La Primera Conexión:** La primera vez que te conectas a un servidor (por ejemplo, 192.168.56.11), tu cliente SSH no tiene forma de saber si realmente está hablando con el servidor legítimo. Podría ser un atacante interceptando la conexión. Por lo tanto, te muestra la "huella digital" (fingerprint) de la clave pública del servidor y te pregunta si confías en él.25  
   The authenticity of host '192.168.56.11 (192.168.56.11)' can't be established.  
   ECDSA key fingerprint is SHA256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx.  
   Are you sure you want to continue connecting (yes/no)?
2. **Estableciendo la Confianza:** Si escribes yes, tu cliente SSH guarda la dirección del servidor (192.168.56.11) junto con su clave pública en tu archivo local ~/.ssh/known\_hosts.48
3. **Conexiones Posteriores:** En cada conexión futura a ese mismo servidor, tu cliente SSH compara la clave pública que el servidor presenta con la que tiene guardada en known\_hosts. Si coinciden, la conexión continúa silenciosamente. Si **no coinciden**, la conexión se detiene y se muestra una advertencia de seguridad grave.

### 5.2. La Temida Advertencia: "REMOTE HOST IDENTIFICATION HAS CHANGED!"

Esta advertencia es probablemente el "error" de SSH más común y alarmante que encontrarás.

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @  
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!  
Someone could be eavesdropping on you right now (man-in-the-middle attack)!  
It is also possible that a host key has just been changed.  
...

Este mensaje significa que la clave pública presentada por el servidor al que intentas conectarte es diferente de la que tienes guardada en tu archivo known\_hosts para esa misma dirección IP o nombre de host.49 Es una comprobación de seguridad crítica.

Hay razones legítimas e ilegítimas para que esto ocurra:

* **Razones Legítimas (muy comunes en entornos de desarrollo):**
  + El sistema operativo del servidor fue reinstalado, lo que generó un nuevo conjunto de claves de host.49 Esto es exactamente lo que sucede cuando ejecutas  
    vagrant destroy y luego vagrant up en el mismo proyecto. La VM node1 es destruida y una nueva, con una nueva identidad (clave de host), es creada en la misma dirección IP.
  + La dirección IP del servidor fue reasignada a una máquina completamente diferente.49
* **Razón Ilegítima (el peligro real):**
  + Un atacante ha interceptado tu conexión y se está haciendo pasar por el servidor legítimo para robar tus datos (un ataque MITM).50

**Cómo Solucionarlo (de la Manera Correcta):**

1. **Verificar:** Primero, determina por qué cambió la clave. En nuestro laboratorio con Vagrant, la causa es conocida y segura: hemos reconstruido la VM. En un entorno de producción, deberías contactar al administrador del sistema para confirmar que el cambio fue intencionado.
2. **Eliminar la Clave Antigua:** Una vez que has verificado que el cambio es legítimo, debes eliminar la entrada *antigua y obsoleta* de tu archivo known\_hosts. La forma más segura y sencilla de hacerlo es con el comando ssh-keygen:  
   Bash  
   ssh-keygen -R 192.168.56.11  
     
   Este comando busca y elimina (Remove) todas las entradas para el host 192.168.56.11 de tu archivo known\_hosts.49 Es preferible a editar el archivo manualmente, especialmente porque los nombres de host a menudo se "hashean" (se ofuscan) en el archivo por seguridad, haciéndolos ilegibles.47
3. **Reconectar:** Después de eliminar la clave antigua, el siguiente intento de conexión se comportará como la "primera vez" de nuevo. Se te mostrará la nueva huella digital y se te pedirá que la aceptes.

Este "error" es, de hecho, la herramienta pedagógica más importante en todo el proceso de aprendizaje de SSH. Inicialmente, se percibe como un fallo, pero en realidad es una demostración del éxito del protocolo: el sistema de seguridad está funcionando como se diseñó. Es el momento en que el concepto abstracto de un ataque MITM se convierte en una realidad tangible en la terminal. Al aprender a distinguir entre las causas legítimas e ilegítimas y al aplicar la solución correcta (ssh-keygen -R) en lugar de una peligrosa (como desactivar la comprobación de claves de host), se aprende la disciplina operativa y el pensamiento crítico en seguridad.

### 5.3. Agilizando Conexiones con el Archivo config de SSH

A medida que trabajas con más sistemas, escribir ssh usuario@direccion-ip-larga cada vez se vuelve repetitivo. SSH proporciona un archivo de configuración del lado del cliente, ~/.ssh/config, para crear alias y preconfigurar opciones de conexión.31

#### 5.3.1. Ejemplo Básico de Alias

Crea o edita el archivo ~/.ssh/config en tu máquina controller y añade el siguiente contenido:

# Alias para nuestra máquina virtual node1  
Host node1  
 HostName 192.168.56.11  
 User vagrant

Guarda el archivo. Ahora, en lugar de escribir el comando largo, simplemente puedes ejecutar:

Bash

ssh node1

SSH leerá tu archivo de configuración, encontrará la entrada Host node1 y usará automáticamente el HostName y el User especificados.55 Esta es una mejora de productividad inmensa.

#### 5.3.2. Ejemplo Avanzado: Gestión de Múltiples Identidades

Un flujo de trabajo profesional a menudo implica interactuar con diferentes sistemas que requieren diferentes claves SSH (por ejemplo, una para el trabajo, otra para proyectos personales en GitHub). El archivo config es la herramienta perfecta para gestionar esto de forma transparente.

Supongamos que tienes dos pares de claves:

* ~/.ssh/id\_ed25519\_work para conectarte a los servidores de tu empresa.
* ~/.ssh/id\_ed25519\_github para interactuar con GitHub.

Puedes configurar tu archivo ~/.ssh/config de la siguiente manera:

# Conexión al servidor de trabajo  
Host work-server  
 HostName 172.16.10.50  
 User admin  
 IdentityFile ~/.ssh/id\_ed25519\_work  
  
# Configuración para GitHub  
Host github.com  
 HostName github.com  
 User git  
 IdentityFile ~/.ssh/id\_ed25519\_github  
 IdentitiesOnly yes  
  
# Alias para nuestro laboratorio Vagrant (usa la clave por defecto)  
Host node1  
 HostName 192.168.56.11  
 User vagrant

Con esta configuración:

* ssh work-server se conectará automáticamente a 172.16.10.50 como usuario admin usando la clave id\_ed25519\_work.
* Cualquier comando git que interactúe con github.com (como git pull) usará automáticamente la clave id\_ed25519\_github. La directiva IdentitiesOnly yes es importante aquí para evitar que SSH intente ofrecer otras claves, lo que a veces puede causar errores de autenticación si se ofrecen demasiadas.
* ssh node1 seguirá funcionando como antes, usando la clave por defecto.

Esta técnica de compartimentación de identidades es una práctica de seguridad y productividad fundamental para cualquier profesional de DevOps.

Para consolidar los conceptos, la siguiente tabla resume los archivos clave que hemos discutido.

**Tabla 2: Archivos Clave de SSH y su Propósito**

| Nombre del Archivo | Ubicación | Propósito |
| --- | --- | --- |
| id\_ed25519 | Cliente (~/.ssh/) | Tu identidad secreta (clave privada). **Nunca la compartas.** |
| id\_ed25519.pub | Cliente (~/.ssh/) | Tu cerradura compartible (clave pública). Se instala en los servidores. |
| authorized\_keys | Servidor (~/.ssh/) | Lista de claves públicas (identidades) autorizadas para iniciar sesión en esa cuenta. |
| known\_hosts | Cliente (~/.ssh/) | Lista de servidores en los que confías y sus claves públicas para prevenir ataques MITM. |
| config | Cliente (~/.ssh/) | Tus atajos y configuraciones personalizadas para simplificar las conexiones. |

## Sección 6: Un Puente hacia el Futuro - Por Qué Esto es Crucial para Ansible

Haber dominado la configuración de SSH sin contraseña no es solo un ejercicio académico o una simple conveniencia. Es la adquisición de una habilidad fundamental que desbloquea el siguiente nivel de la administración de sistemas moderna: la automatización de la infraestructura con herramientas como Ansible.

### 6.1. El Modus Operandi de Ansible

Ansible es una potente herramienta de automatización de TI que te permite configurar, desplegar software y orquestar tareas avanzadas como despliegues continuos o actualizaciones sin tiempo de inactividad. Su filosofía se basa en un diseño **sin agentes**, lo que significa que no requiere la instalación de ningún software o cliente especial en las máquinas que gestiona.56

Su arquitectura es simple:

* Un **Nodo de Control:** La máquina donde Ansible está instalado (como nuestra VM controller). Desde aquí se ejecutan los "Playbooks" (archivos YAML que describen el estado deseado de un sistema).58
* **Nodos Gestionados:** Los servidores o dispositivos que Ansible configura (como nuestra VM node1).56

La pregunta clave es: si no hay agentes, ¿cómo se comunica el nodo de control con los nodos gestionados? La respuesta es: **a través de SSH**.59

### 6.2. El Imperativo del Acceso sin Contraseña

Aquí es donde todo lo que hemos aprendido converge. Imagina que necesitas aplicar una actualización de seguridad a 100 servidores. Si Ansible tuviera que detenerse y pedirte una contraseña para cada uno de esos 100 servidores, el propósito de la automatización se perdería por completo. El proceso debe ser completamente **no interactivo** para ser escalable y eficiente.62

La autenticación basada en claves SSH que acabas de configurar es la **tecnología habilitadora** para esta automatización. Ansible, ejecutándose en el nodo de control, utilizará las claves SSH (idealmente cargadas en un ssh-agent) para autenticarse en todos los nodos gestionados de forma transparente, segura y sin intervención humana.59

Por lo tanto, dominar SSH sin contraseña no es una opción, sino un **prerrequisito indispensable** para practicar la gestión de configuración moderna y las metodologías DevOps.65 La relación entre SSH y Ansible ilustra perfectamente el principio de "bloques de construcción" en la tecnología. SSH, por sí solo, es una herramienta para el acceso interactivo. Ansible, por sí solo, es un motor abstracto para describir el estado del sistema. Sin embargo, cuando se combinan, crean un paradigma completamente nuevo: la Infraestructura como Código (IaC). Ansible

*aprovecha* el transporte seguro y no interactivo que proporciona una configuración SSH robusta para convertir archivos de configuración declarativos en sistemas concretos y funcionales. Dominar un protocolo fundamental como SSH no solo te enseña esa habilidad específica, sino que te da la llave para utilizar una clase completamente nueva de herramientas más potentes y abstractas que definen la industria actual.

## Conclusión: Tu Fundación para la Automatización de Infraestructuras

A lo largo de esta guía, hemos emprendido un viaje completo a través del mundo de Secure Shell. Partimos de comprender el "porqué" de su existencia al contrastarlo con la inseguridad inherente de Telnet. Profundizamos en el "cómo", desmitificando la criptografía de clave pública que forma su núcleo seguro. Luego, pasamos al "qué", con una guía práctica y paso a paso para generar claves, construir un laboratorio consistente con Vagrant e implementar la crucial autenticación sin contraseña. Finalmente, hemos conectado los puntos hacia el "qué sigue", estableciendo cómo esta habilidad es la base indispensable para la automatización con Ansible.

Has aprendido a crear y gestionar tu identidad digital, a instalarla de forma segura en sistemas remotos y a solucionar los problemas más comunes que surgen en el proceso. Más importante aún, has comprendido los principios de seguridad subyacentes que hacen que todo el sistema sea robusto.

La capacidad de acceder a sistemas remotos de forma segura y programática no es simplemente una habilidad más en el repertorio de un técnico; es, sin duda, la habilidad más fundamental para cualquier carrera en DevOps, Ingeniería en la Nube, Ingeniería de Fiabilidad de Sitios (SRE) o Administración de Sistemas. Es el cimiento sobre el cual se construyen todas las prácticas de automatización, orquestación y gestión de infraestructuras a escala. El conocimiento que has adquirido aquí es tu primer y más importante paso en el camino hacia el dominio de la infraestructura moderna.

## Apéndice A: Una Nota sobre Terminología - Claves SSH vs. Certificados SSH

A lo largo de esta guía, hemos utilizado el término "claves SSH" para referirnos a los pares de claves pública/privada que has generado. En el lenguaje coloquial y en muchos contextos técnicos, esto es perfectamente aceptable. Sin embargo, para un nivel de conocimiento experto, es importante hacer una distinción precisa con otro concepto: los **Certificados SSH**. La confusión inicial entre "certificados digitales" y "claves" es común, y esta aclaración añade una capa de profundidad a tu comprensión.

* **Claves SSH (lo que hemos usado):** El modelo se basa en la confianza explícita. Para que el servidor node1 confíe en el cliente controller, debes copiar manualmente la clave pública de controller en el archivo authorized\_keys de node1. Si tuvieras 500 servidores, tendrías que repetir este proceso 500 veces. Si un nuevo ingeniero se une al equipo, su clave pública debe añadirse a los 500 servidores. Si un ingeniero se va, su clave debe ser eliminada de los 500 servidores. Como puedes ver, este modelo no escala bien.66
* **Certificados SSH (la solución a escala empresarial):** Este modelo introduce un intermediario de confianza: una **Autoridad de Certificación (CA)**.20
  1. **Establecer la Confianza:** En lugar de configurar cada servidor para que confíe en docenas de claves de usuario individuales, configuras cada servidor una sola vez para que confíe en la clave pública de tu CA interna.20
  2. **Emitir Certificados:** Cuando un usuario necesita acceso, no se distribuye su clave pública. En su lugar, el usuario envía su clave pública a la CA. La CA usa su propia clave privada para "firmar" la clave pública del usuario, creando un **Certificado SSH**. Este certificado es un objeto criptográfico que vincula la clave pública del usuario a una identidad (por ejemplo, "usuario: juanperez", "grupos: dev, db-admin") y, crucialmente, tiene una **fecha de caducidad** corta (por ejemplo, 8 horas).68
  3. **Autenticación:** El usuario se conecta al servidor presentando este certificado de corta duración. El servidor, que confía en la CA, verifica la firma del certificado y, si es válida, concede el acceso basado en los permisos definidos dentro del propio certificado.

**Ventajas de los Certificados SSH:**

* **Gestión Centralizada:** El acceso se gestiona en un solo lugar (la CA). Para añadir o revocar el acceso de un usuario, solo interactúas con la CA; no es necesario tocar ninguno de los servidores de destino.67
* **Credenciales de Corta Duración:** Los certificados caducan automáticamente, lo que elimina el riesgo de que claves antiguas y olvidadas permanezcan activas y puedan ser explotadas.66
* **Control de Acceso Granular:** Los certificados pueden contener metadatos que restringen lo que un usuario puede hacer, aplicando el principio de mínimo privilegio.68

En resumen, mientras que las **claves SSH** son perfectas para individuos, equipos pequeños y los escenarios que hemos practicado, los **Certificados SSH** son la solución robusta y escalable para gestionar el acceso en entornos empresariales grandes y complejos.68 Comprender esta diferencia es una marca distintiva de un conocimiento profundo y profesional de la seguridad SSH.

## Apéndice B: Laboratorio Manual con VirtualBox (Sin Vagrant)

Aunque Vagrant es una herramienta excelente para la reproducibilidad, es fundamental entender cómo configurar un entorno similar manualmente. Este apéndice te guiará a través de la creación de un laboratorio de dos máquinas virtuales Ubuntu en VirtualBox que pueden comunicarse entre sí, sin usar Vagrant.

### Paso 1: Configurar una Red "Solo-Anfitrión" en VirtualBox

Para que las VMs se comuniquen entre sí y con tu máquina anfitriona en una red privada, usaremos una red "Solo-Anfitrión" (Host-Only). 72

1. Abre VirtualBox y ve a Archivo -> Administrador de Redes de Anfitrión. 77
2. Haz clic en Crear. VirtualBox creará una nueva red, probablemente llamada vboxnet0.
3. Selecciona la nueva red y haz clic en Propiedades.
4. En la pestaña Adaptador, asegúrate de que la dirección IP del anfitrión sea algo como 192.168.56.1. Esta será la IP de tu propia máquina dentro de esta red privada. 77
5. En la pestaña Servidor DHCP, puedes habilitar el servidor si quieres que las IPs se asignen automáticamente, pero para este laboratorio, configuraremos IPs estáticas para tener un control total. Puedes dejarlo deshabilitado.
6. Haz clic en Aplicar.

### Paso 2: Crear y Configurar las Máquinas Virtuales

Ahora, crea dos máquinas virtuales Ubuntu Server. Llamémoslas ubuntu-control y ubuntu-node1. Durante la creación, o después, ve a la configuración de red de cada VM.

Para cada una de las dos VMs, configura dos adaptadores de red: 79

* **Adaptador 1 (para acceso a Internet):**
  + **Conectado a:** NAT
  + Esto permitirá a la VM acceder a Internet para descargar paquetes (como el servidor SSH). 75
* **Adaptador 2 (para la comunicación interna):**
  + **Conectado a:** Adaptador Solo-Anfitrión
  + **Nombre:** Selecciona la red que creaste en el paso 1 (ej. vboxnet0). 77

### Paso 3: Configurar IPs Estáticas en Ubuntu

Inicia ambas VMs. Por defecto, el segundo adaptador de red no estará configurado. Debemos asignarle una IP estática a cada máquina. Las versiones modernas de Ubuntu usan netplan para la configuración de red.

1. Identifica los nombres de las interfaces:  
   En cada VM, ejecuta ip a para ver las interfaces de red. Verás una llamada lo (loopback), y otras dos, probablemente enp0s3 (para NAT) y enp0s8 (para Solo-Anfitrión). Anota el nombre de la segunda interfaz. 84
2. Configura netplan en ubuntu-control:  
   Edita el archivo de configuración de netplan (el nombre puede variar, busca en /etc/netplan/).  
   Bash  
   sudo nano /etc/netplan/00-installer-config.yaml  
     
   Modifica el archivo para que se parezca a esto, reemplazando enp0s8 si tu nombre de interfaz es diferente:  
   YAML  
   network:  
    ethernets:  
    enp0s3:  
    dhcp4: true  
    enp0s8:  
    dhcp4: no  
    addresses:  
    - 192.168.56.10/24  
    version: 2
3. Configura netplan en ubuntu-node1:  
   Haz lo mismo en la segunda VM, pero con una IP diferente:  
   Bash  
   sudo nano /etc/netplan/00-installer-config.yaml  
     
   YAML  
   network:  
    ethernets:  
    enp0s3:  
    dhcp4: true  
    enp0s8:  
    dhcp4: no  
    addresses:  
    - 192.168.56.11/24  
    version: 2
4. Aplica la configuración:  
   En ambas VMs, ejecuta el siguiente comando para aplicar los cambios: 82  
   Bash  
   sudo netplan apply  
     
   Verifica que las IPs se han asignado correctamente con ip a.

### Paso 4: Instalar OpenSSH y Probar la Conexión

1. Instala el servidor SSH:  
   En ambas máquinas virtuales, asegúrate de que el servidor OpenSSH esté instalado.  
   Bash  
   sudo apt update  
   sudo apt install openssh-server  
     
   86
2. Verifica la conectividad de red:  
   Desde ubuntu-control, haz ping a ubuntu-node1:  
   Bash  
   ping 192.168.56.11  
     
   Si recibes respuesta, la red está funcionando. 92
3. Prueba la conexión SSH con contraseña:  
   Desde ubuntu-control, intenta conectarte a ubuntu-node1:  
   Bash  
   ssh tu\_usuario@192.168.56.11  
     
   La primera vez te preguntará si confías en el host. Escribe yes. Luego, introduce la contraseña de tu usuario en ubuntu-node1. Si te conectas, ¡todo está listo para el siguiente paso!

### Paso 5: Configurar el Acceso sin Contraseña

Este proceso es idéntico al descrito en la Sección 4, pero ahora lo aplicas a tu entorno manual.

1. **En ubuntu-control**, genera tu par de claves:  
   Bash  
   ssh-keygen -t ed25519  
     
   Acepta los valores por defecto.
2. **En ubuntu-control**, copia tu clave pública a ubuntu-node1:  
   Bash  
   ssh-copy-id tu\_usuario@192.168.56.11  
     
   Te pedirá la contraseña de tu\_usuario en ubuntu-node1 por última vez.
3. Prueba final:  
   Desde ubuntu-control, conéctate de nuevo:  
   Bash  
   ssh tu\_usuario@192.168.56.11  
     
   Ahora deberías conectarte instantáneamente, sin que se te pida contraseña. Has replicado con éxito el entorno del laboratorio de forma manual.

#### Obras citadas

1. ¿Qué es SSH (Secure Shell)? - IONOS, fecha de acceso: septiembre 30, 2025, <https://www.ionos.com/es-us/digitalguide/servidores/herramientas/protocolo-ssh/>
2. SSH, Telnet y las diferencias para conectar en un servidor | Alura Cursos Online, fecha de acceso: septiembre 30, 2025, <https://www.aluracursos.com/blog/ssh-telnet-y-las-diferencias-para-conectar-en-un-servidor>
3. www.cloudflare.com, fecha de acceso: septiembre 30, 2025, <https://www.cloudflare.com/es-es/learning/access-management/what-is-ssh/#:~:text=SSH%20(Secure%20Shell)%20es%20un,trav%C3%A9s%20de%20redes%20no%20seguras.>
4. Protocolo SSH: uso, versiones e implementaciones - Stackscale, fecha de acceso: septiembre 30, 2025, <https://www.stackscale.com/es/blog/protocolo-ssh/>
5. Telnet vs. SSH: How Is SSH Different From Telnet? - phoenixNAP, fecha de acceso: septiembre 30, 2025, <https://phoenixnap.com/kb/telnet-vs-ssh>
6. Telnet vs SSH Explained - YouTube, fecha de acceso: septiembre 30, 2025, <https://www.youtube.com/watch?v=tZop-zjYkrU&pp=0gcJCf8Ao7VqN5tD>
7. Comparación Entre Telnet y SSH | PDF | Cubierta segura | Autenticación - Scribd, fecha de acceso: septiembre 30, 2025, <https://es.scribd.com/document/660953889/Comparacion-entre-Telnet-y-SSH>
8. Difference between SSH and Telnet - GeeksforGeeks, fecha de acceso: septiembre 30, 2025, <https://www.geeksforgeeks.org/computer-networks/difference-ssh-telnet/>
9. SSH vs. Telnet: La opción más segura para el acceso remoto [2025] - Geekflare, fecha de acceso: septiembre 30, 2025, <https://geekflare.com/es/ssh-vs-telnet-for-remote-access/>
10. SSH: qué es y cómo funciona este protocolo | Blog de Arsys, fecha de acceso: septiembre 30, 2025, <https://www.arsys.es/blog/ssh>
11. SSH (Secure Shell) - Paessler, fecha de acceso: septiembre 30, 2025, <https://www.paessler.com/es/it-explained/ssh>
12. ¿Qué es el protocolo SSH? ¿Como funciona? - Wallarm, fecha de acceso: septiembre 30, 2025, <https://lab.wallarm.com/what/que-es-el-protocolo-ssh/?lang=es>
13. Habilitación de la autenticación basada en claves RSA en sistemas operativos UNIX y Linux, fecha de acceso: septiembre 30, 2025, <https://www.ibm.com/docs/es/sia?topic=kbaula-enabling-rsa-key-based-authentication-unix-linux-operating-systems-2>
14. Diferencias entre clave pública y clave privada al cifrar datos - RedesZone, fecha de acceso: septiembre 30, 2025, <https://www.redeszone.net/tutoriales/seguridad/diferencias-cifrado-clave-publica-privada/>
15. Criptografía asimétrica - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 30, 2025, <https://es.wikipedia.org/wiki/Criptograf%C3%ADa_asim%C3%A9trica>
16. Criptografía de clave pública - IBM, fecha de acceso: septiembre 30, 2025, <https://www.ibm.com/docs/es/integration-bus/10.0.0?topic=overview-public-key-cryptography>
17. Cómo utilizar la autenticación de clave pública SSH | Linode Docs, fecha de acceso: septiembre 30, 2025, <https://www.linode.com/docs/guides/use-public-key-authentication-with-ssh/?lang=es>
18. Lección 3: Sistemas de cifra con clave pública (intypedia) - YouTube, fecha de acceso: septiembre 30, 2025, <https://www.youtube.com/watch?v=On1clzor4x4>
19. Clave pública, clave privada y la frases secretas - MtPelerin, fecha de acceso: septiembre 30, 2025, <https://www.mtpelerin.com/es/blog/clave-publica-clave-privada-frases-secretas>
20. Cómo Funcionan los Certificados SSH | Keytos, fecha de acceso: septiembre 30, 2025, <https://www.keytos.io/blog/resistente-a-phishing/como-funcionan-los-certificados-ssh.html>
21. Criptografía 101: Clave Pública, Clave Privada y Cómo Funcionan Juntas - Kiteworks, fecha de acceso: septiembre 30, 2025, <https://www.kiteworks.com/es/cumplimiento-de-cmmc/criptografia-clave-publica-clave-privada/>
22. How to Set Up SSH Keys on Ubuntu: A Comprehensive Guide - DigitalOcean, fecha de acceso: septiembre 30, 2025, <https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-ubuntu-22-04>
23. Generating a new SSH key and adding it to the ssh-agent - GitHub Docs, fecha de acceso: septiembre 30, 2025, <https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>
24. Cómo configurar claves SSH: gestión eficaz de claves para conexiones remotas seguras, fecha de acceso: septiembre 30, 2025, <https://www.hostinger.com/es/tutoriales/como-configurar-claves-ssh>
25. Cómo configurar la autenticación basada en claves de SSH en un servidor Linux, fecha de acceso: septiembre 30, 2025, <https://www.digitalocean.com/community/tutorials/how-to-configure-ssh-key-based-authentication-on-a-linux-server-es>
26. How to Set Up SSH Keys on Ubuntu 20.04 | Guide - CloudPanel, fecha de acceso: septiembre 30, 2025, <https://www.cloudpanel.io/tutorial/set-up-ssh-keys-on-ubuntu-20-04/>
27. Explicación de las claves SSH: Guía para un acceso remoto rápido y seguro | DataCamp, fecha de acceso: septiembre 30, 2025, <https://www.datacamp.com/es/tutorial/ssh-keys>
28. SSH Copy ID for Copying SSH Keys to Servers, fecha de acceso: septiembre 30, 2025, <https://www.ssh.com/academy/ssh/copy-id>
29. How to understand ssh-keygen and ssh-copy-id? - Unix & Linux Stack Exchange, fecha de acceso: septiembre 30, 2025, <https://unix.stackexchange.com/questions/279923/how-to-understand-ssh-keygen-and-ssh-copy-id>
30. Cómo generar claves SSH para GitHub - Kinsta, fecha de acceso: septiembre 30, 2025, <https://kinsta.com/es/blog/generar-claves-ssh/>
31. La guía definitiva para SSH: Configurando claves SSH - freeCodeCamp, fecha de acceso: septiembre 30, 2025, <https://www.freecodecamp.org/espanol/news/la-guia-definitiva-para-ssh-configuracion-de-claves-ssh/>
32. Cómo Generar Claves SSH Seguras Para GitHub - K2WebHost, fecha de acceso: septiembre 30, 2025, <https://www.k2webhost.com/blog/como-generar-claves-ssh-seguras-para-github/>
33. Vagrant Tutorial For Beginners: Getting Started Guide - DevOpsCube, fecha de acceso: septiembre 30, 2025, <https://devopscube.com/vagrant-tutorial-beginners/>
34. Create Multi-vm setup using Vagrant file (Iac), fecha de acceso: septiembre 30, 2025, <https://aveshdevops.hashnode.dev/multi-vm-setup-in-vagrant-iac-day-1790>
35. Manage multi-machine environments | Vagrant - HashiCorp Developer, fecha de acceso: septiembre 30, 2025, <https://developer.hashicorp.com/vagrant/tutorials/get-started/multi-machine>
36. Multi-Machine | Vagrant - HashiCorp Developer, fecha de acceso: septiembre 30, 2025, <https://developer.hashicorp.com/vagrant/docs/multi-machine>
37. Multi Server setup with vagrant - Mouhsen Ibrahim Site, fecha de acceso: septiembre 30, 2025, <https://mohsensy.github.io/sysadmin/2019/08/16/multi-server-setup-with-vagrant.html>
38. Basic Usage - Networking | Vagrant - HashiCorp Developer, fecha de acceso: septiembre 30, 2025, <https://developer.hashicorp.com/vagrant/docs/networking/basic_usage>
39. Private Networks - Vagrant - HashiCorp Developer, fecha de acceso: septiembre 30, 2025, <https://developer.hashicorp.com/vagrant/docs/networking/private_network>
40. How To Use Vagrant Networking - Progress Software, fecha de acceso: septiembre 30, 2025, <https://www.progress.com/blogs/how-to-use-vagrant-networking>
41. Allow two or more vagrant VMs to communicate on their own network - Stack Overflow, fecha de acceso: septiembre 30, 2025, <https://stackoverflow.com/questions/24867252/allow-two-or-more-vagrant-vms-to-communicate-on-their-own-network>
42. How to Use ssh-keygen to Generate a New SSH Key? - SSH Communications Security, fecha de acceso: septiembre 30, 2025, <https://www.ssh.com/academy/ssh/keygen>
43. Cómo configurar llaves SSH adicionales en una instancia - OVHcloud, fecha de acceso: septiembre 30, 2025, <https://help.ovhcloud.com/csm/es-public-cloud-compute-additional-ssh-keys?id=kb_article_view&sysparm_article=KB0050572>
44. Configuración de la autenticación de clave pública a través de SSH - IBM, fecha de acceso: septiembre 30, 2025, <https://www.ibm.com/docs/es/db2/11.1.0?topic=profile-setting-up-public-key-authentication-over-ssh>
45. www.lenovo.com, fecha de acceso: septiembre 30, 2025, <https://www.lenovo.com/es/es/glossary/ssh/#:~:text=%C2%BFQu%C3%A9%20significa%20known_hosts%20en%20SSH,almacenada%20en%20su%20archivo%20known_hosts.>
46. Cómo verificar la autenticidad del host SSH en Linux - LabEx, fecha de acceso: septiembre 30, 2025, <https://labex.io/es/tutorials/linux-how-to-verify-ssh-host-authenticity-in-linux-415844>
47. What is known\_hosts File in Linux - Sysadmin, fecha de acceso: septiembre 30, 2025, <https://sysadmin.info.pl/en/blog/known-hosts-file/>
48. ¿Qué se supone que debe haber en mi carpeta .ssh/known\_hosts? : r/linuxquestions - Reddit, fecha de acceso: septiembre 30, 2025, <https://www.reddit.com/r/linuxquestions/comments/1awnpdr/what_is_supposed_to_be_in_my_sshknown_hosts_folder/?tl=es-419>
49. How to Solve SSH Host Key Mismatch Error in Linux, macOS, Windows? - MonoVM, fecha de acceso: septiembre 30, 2025, <https://monovm.com/blog/remote-host-identification-has-changed-error/>
50. Fix "Warning: Remote Host Identification Has Changed" Error - Kinsta, fecha de acceso: septiembre 30, 2025, <https://kinsta.com/knowledgebase/warning-remote-host-identification-has-changed/>
51. SSH: Remote Host identification has changed - linux - Super User, fecha de acceso: septiembre 30, 2025, <https://superuser.com/questions/1536782/ssh-remote-host-identification-has-changed>
52. "Remote host identification has changed" warning when connecting over SSH - Ask Ubuntu, fecha de acceso: septiembre 30, 2025, <https://askubuntu.com/questions/9800/remote-host-identification-has-changed-warning-when-connecting-over-ssh>
53. Why are connections to GitHub over SSH throwing an error "Warning: Remote Host Identification Has Changed"? - Stack Overflow, fecha de acceso: septiembre 30, 2025, <https://stackoverflow.com/questions/75830783/why-are-connections-to-github-over-ssh-throwing-an-error-warning-remote-host-i>
54. SSH config: un ejemplo de archivo de configuración de SSH, explicado - GitHub Gist, fecha de acceso: septiembre 30, 2025, <https://gist.github.com/aguadopd/818dc386e9f7c26550dd5e72d6e7eeff>
55. Cómo crear alias para conexiones ssh - YouTube, fecha de acceso: septiembre 30, 2025, <https://www.youtube.com/watch?v=iKDT0CqB0Mw>
56. Ansible: ¿qué es y cómo funciona? - Red Hat, fecha de acceso: septiembre 30, 2025, <https://www.redhat.com/es/topics/automation/learning-ansible-tutorial>
57. Qué es Ansible y cómo funciona - Grupo Castilla, fecha de acceso: septiembre 30, 2025, <https://www.grupocastilla.es/ansible/>
58. Introducción a Ansible con ejemplos para administradores de sistemas - Red Hat, fecha de acceso: septiembre 30, 2025, <https://www.redhat.com/es/blog/system-administrators-guide-getting-started-ansible-fast>
59. www.redhat.com, fecha de acceso: septiembre 30, 2025, <https://www.redhat.com/es/topics/automation/learning-ansible-tutorial#:~:text=De%20forma%20predeterminada%2C%20Ansible%20utiliza,sudo%20para%20adquirir%20nuevos%20privilegios.>
60. ansible.builtin.ssh connection – connect via SSH client binary, fecha de acceso: septiembre 30, 2025, <https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ssh_connection.html>
61. Curso de iniciación a Ansible para la automatización de operaciones, fecha de acceso: septiembre 30, 2025, <https://ualmtorres.github.io/CursoAnsible/tutorial/>
62. Autenticar a los usuarios que ejecutan módulos de Ansible en dispositivos Junos, fecha de acceso: septiembre 30, 2025, <https://www.juniper.net/documentation/mx/es/software/junos-ansible/ansible/topics/topic-map/junos-ansible-authenticating-users.html>
63. Enabling password-less SSH for Ansible - IBM, fecha de acceso: septiembre 30, 2025, <https://www.ibm.com/docs/en/storage-ceph/7.1?topic=installation-enabling-password-less-ssh-ansible>
64. Cómo configurar SSH sin contraseña en Linux - Hostinger, fecha de acceso: septiembre 30, 2025, <https://www.hostinger.com/es/tutoriales/configurar-ssh-sin-contrasena-linux>
65. Iniciarse en Ansible - Ediciones ENI, fecha de acceso: septiembre 30, 2025, <https://www.ediciones-eni.com/libro/ansible-administre-la-configuracion-de-sus-servidores-y-el-despliegue-de-sus-aplicaciones-9782409029783/iniciarse-en-ansible>
66. Por qué es importante la gestión de claves SSH en la seguridad moderna - Keyfactor, fecha de acceso: septiembre 30, 2025, <https://www.keyfactor.com/es/blog/why-ssh-key-management-matters-in-modern-security/>
67. SSH Certificates: A Complete Guide - Infisical, fecha de acceso: septiembre 30, 2025, <https://infisical.com/blog/ssh-certificates-guide>
68. SSH vs. X.509 Certificates - Smallstep, fecha de acceso: septiembre 30, 2025, <https://smallstep.com/blog/ssh-vs-x509-certificates/>
69. Why SSH Certificates Can Be A Better Option For Remote Access Than SSH Keys, fecha de acceso: septiembre 30, 2025, <https://www.appviewx.com/blogs/why-ssh-certificates-can-be-a-better-option-for-remote-access-than-ssh-keys/>
70. ¿Cuál es la Diferencia Entre los Certificados SSH y X509? - Keytos, fecha de acceso: septiembre 30, 2025, <https://www.keytos.io/blog/resistente-a-phishing/cual-es-la-diferencia-entre-certificados-ssh-y-x509.html>
71. How to Create an SSH Certificate Authority in Azure - Keytos Security, fecha de acceso: septiembre 30, 2025, <https://www.keytos.io/blog/passwordless/how-to-create-an-ssh-certificate-authority.html>
72. 6 Virtual Networking - Oracle Help Center, fecha de acceso: septiembre 30, 2025, <https://docs.oracle.com/en/virtualization/virtualbox/7.1/user/networkingdetails.html>
73. Host only/Internal network? - virtualbox.org, fecha de acceso: septiembre 30, 2025, <https://forums.virtualbox.org/viewtopic.php?t=92253>
74. 6.7. Host-Only Networking, fecha de acceso: septiembre 30, 2025, <https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/network_hostonly.html>
75. Choosing the Right VirtualBox Network Adapter: NAT, Host-Only, or Bridged? - Dev Genius, fecha de acceso: septiembre 30, 2025, <https://blog.devgenius.io/choosing-the-right-virtualbox-network-adapter-nat-host-only-or-bridged-188003a260de>
76. What is VirtualBox's Host-Only networking mode? - Backup Education, fecha de acceso: septiembre 30, 2025, <https://backup.education/showthread.php?tid=3193>
77. VirtualBox host-only network - ssh to remote machine - Code Maven, fecha de acceso: septiembre 30, 2025, <https://code-maven.com/virtualbox-host-only-network-ssh-to-remote-machine>
78. Creating a New Host-Only Adapter in VirtualBox - School of ..., fecha de acceso: septiembre 30, 2025, <https://carleton.ca/scs/2024/creating-a-new-host-only-adapter-in-virtualbox/>
79. VirtualBox: How to set up networking so both host and guest can ..., fecha de acceso: septiembre 30, 2025, <https://serverfault.com/questions/225155/virtualbox-how-to-set-up-networking-so-both-host-and-guest-can-access-internet>
80. SSH to VirtualBox Host-Only Adapter Interface from WSL2 Machine - codemonday, fecha de acceso: septiembre 30, 2025, <https://www.codemonday.com/blogs/ssh-to-virtualbox-host-only-adapter-interface-from-wsl2-machine>
81. VirtualBox Host SSH to Guest - Super User, fecha de acceso: septiembre 30, 2025, <https://superuser.com/questions/424083/virtualbox-host-ssh-to-guest>
82. Ubuntu Server 22.10 on Virtualbox - host-only network and NAT can't coexist together?, fecha de acceso: septiembre 30, 2025, <https://askubuntu.com/questions/1450443/ubuntu-server-22-10-on-virtualbox-host-only-network-and-nat-cant-coexist-toge>
83. How to SSH to a VirtualBox guest externally through a host? [closed] - Stack Overflow, fecha de acceso: septiembre 30, 2025, <https://stackoverflow.com/questions/5906441/how-to-ssh-to-a-virtualbox-guest-externally-through-a-host>
84. how to get the ipaddress of a virtual box running on local machine [closed] - Stack Overflow, fecha de acceso: septiembre 30, 2025, <https://stackoverflow.com/questions/42953785/how-to-get-the-ipaddress-of-a-virtual-box-running-on-local-machine>
85. How to find IP Address of newly started headless VM - virtualbox.org, fecha de acceso: septiembre 30, 2025, <https://forums.virtualbox.org/viewtopic.php?t=90295>
86. OpenSSH server - Ubuntu documentation, fecha de acceso: septiembre 30, 2025, <https://documentation.ubuntu.com/server/how-to/security/openssh-server/>
87. Setting up an Ubuntu Desktop installation for SSH, quickly - Jeff Geerling, fecha de acceso: septiembre 30, 2025, <https://www.jeffgeerling.com/blog/2025/setting-ubuntu-desktop-installation-ssh-quickly>
88. How to Install and Configure SSH on Ubuntu 22.04 - Hostman, fecha de acceso: septiembre 30, 2025, <https://hostman.com/tutorials/how-to-install-and-configure-ssh-on-ubuntu-22-04/>
89. How to enable SSH with OpenSSH on Ubuntu 24.04 - Server - IONOS, fecha de acceso: septiembre 30, 2025, <https://www.ionos.com/digitalguide/server/configuration/ubuntu-2404-ssh/>
90. How to Configure SSH on Ubuntu Server [Beginner's Guide], fecha de acceso: septiembre 30, 2025, <https://itsfoss.com/set-up-ssh-ubuntu/>
91. How to install Open SSH on Ubuntu Desktop - YouTube, fecha de acceso: septiembre 30, 2025, <https://www.youtube.com/watch?v=-mojs4Eo52s>
92. Setup 2 Ubuntu boxes in VirtualBox to communicate with each other, fecha de acceso: septiembre 30, 2025, <https://code-maven.com/setup-2-ubuntu-boxes-in-virtualbox-to-communicate-with-each-other>