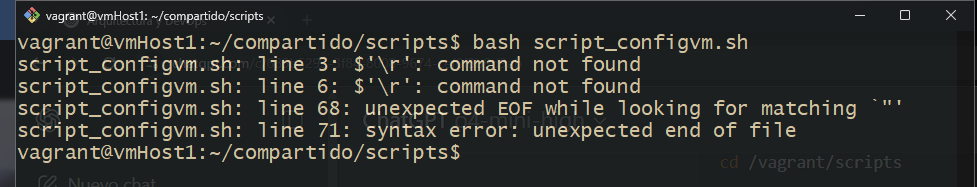
**Corrigiendo errores de scripts.**  
  
R2 script\_configvm.sh  
  
Al momento de ejecutar el script en la vm, figura el siguiente error:  


El error de las lineas 3 y 6 viene de esto:  
Cada línea termina con \r\n en lugar de solo \n, y el \r (carriage-return) aparece como parte de la línea cuando Bash lo interpreta.  
  
Error linea 68 y 71:   
En tu script tienes comillas tipográficas (” … ”) mezcladas con comillas ASCII ("), y además en varios sitios abres con " y cierras con ”. Eso confunde al parser y hace que nunca encuentre la comilla de cierre.  
  
Para normalizar los saltos de linea usamos el siguiente comando:  
sed -i 's/\r$//' MI\_SCRIPT.SH

Luego notamos que habia algunos errores con las comillas en algunos comandos de echo. Modificamos el script normalizando las comillas dobles.

En el script, corrio sin problemas de sintaxis, pero en el segundo paso, al conectar por ssh sin contraseña, me solicito la contraseña de vagrant y del equipo, y cuando lo coloque, se quedo trabado en el siguiente mensaje:  
  


Parece ser que hay un problema en la red privada entre los dos equipos, probamos hacer ping de testing a produccion y no llegan, probamos hacer ping inverso de produc a testing y si llegan.

Parece ser que producción estaba tomando una ip por dhcp a pesar de que lo forzamos en el vagrantfile a que tome la ip 192.168.56.6/24 Lo que hicimos hasta ahora es modificar desde la vm de produccion la ip y fijarla mediante los siguientes comandos:  
  
# Cambia el método a manual y fija la IP

sudo nmcli connection modify "Wired connection 1" \

ipv4.method manual \

ipv4.addresses 192.168.56.6/24 \

ipv4.gateway 192.168.56.1 \

ipv4.dns 192.168.56.1 \

connection.autoconnect yes

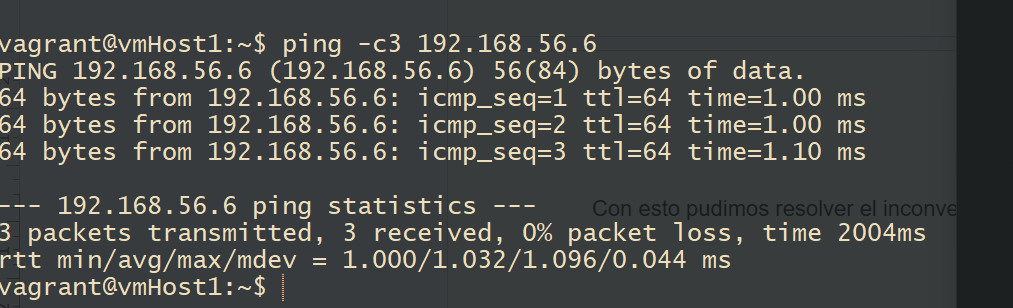
# Baja y sube la conexión para aplicar cambios

sudo nmcli connection down "Wired connection 1"

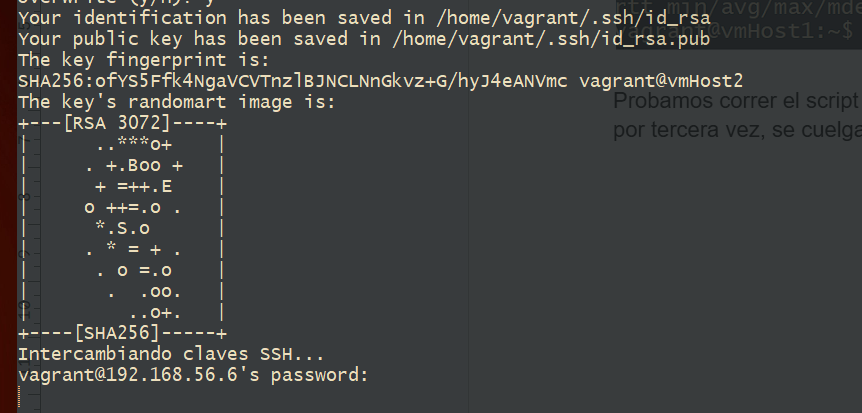
sudo nmcli connection up "Wired connection 1"

* ipv4.addresses 192.168.56.6/24 fija tu IP.
* ipv4.gateway apunta al host-only (normalmente 192.168.56.1).
* ipv4.dns puede ser el mismo gateway o cualquier DNS que prefieras.
* connection.autoconnect yes hace que arranque al reinicio.

Con esto pudimos resolver el inconveniente del ping:



Probamos correr el script pero se al solicitarnos la contraseña de vagrant y su confirmacion por tercera vez, se cuelga en la siguiente parte

LO ANTERIOR HAY QUE AUTOMATIZARLO CON ESOS SCRIPTS DESDE EL VAGRANTFILE O CORRIENDO EL SCRIPT DESDE UN SCRIPT Y PROVEERLO EN EL VAGRANTFILE PARA QUE AUTOMATICAMENTE REALICE ESA CONFIG.  
  


No es automatico, solicita las contraseñas manualmente, mas allá de eso, se traba en esa instancia y no avanza.  
  
Consultando con chatgpt y teniendo en cuenta que solicitaba de manera manual las contraseñas, use un programa que pasa automaticamente las claves ssh, para que no tengamos que interactuar y poner las contraseñas. El programa se llama sshpass y se encarga de hacer que el pasaje de claves ssh sea no interactivo.   
  
Modifique los 2 bloques donde se solicitaban las contraseñas y declare que cuando las solicite indique la contraseña que por defecto coloca vagrant, “vagrant”. Tambien designe que cuando pida confirmacion (poner yes o no) acepte sin solicitarlo.

Tambien modifique el bloque donde configuro el tema de que pida contraseña usando sudo en la parte 3 del script, deje la parte 4 tal cual, hay que validar que paquetes necesitamos realmente para que no consuma tanta memoria ni demore tanto la instalación.  
  
Segun chatgpt los cambios que hicimos son estos:

**1.Hosts**

**Antes**: usábamos echo … | sudo tee -a /etc/hosts en cada línea y hacíamos backup manual.

**Ahora**: usamos un heredoc único con sudo tee -a /etc/hosts <<EOF…EOF para añadir ambas entradas.

**2.Generación de claves SSH**

**Antes**: dentro de cada VM, con ssh host ssh-keygen…, te pedía contraseña y confirmación.

**Ahora**: generamos la clave una sola vez en la VM “testing” con ssh-keygen -q… y usamos sshpass + ssh-copy-id para copiar la pública a ambas VMs sin pedir password ni prompts.

**SSHPASS ES EL APT QUE INSTALAMOS PARA QUE RESUELVA LA INTERACCION CON ESTE SCRIPT.**

**2.StrictHostKeyChecking**

**Antes**: al primer SSH preguntaba “confías en esta llave?” y había que escribir “yes”.

**Ahora**: añadimos -o StrictHostKeyChecking=no a todos los ssh y ssh-copy-id para saltarnos ese prompt.

**3.Sudoers**

**Antes**: se hacía en un bloque aparte, copiabas tu propia sesión a cada VM.

**Ahora**: añadimos un paso 3 que, vía SSH no interactivo, escribe el archivo /etc/sudoers.d/vagrant en ambas máquinas y ajusta permisos (chmod 440), para que vagrant use sudo sin contraseña.

**4.Instalación de software**

**Antes**: teníamos un bloque largo de ssh host <<EOF … apt install… y otro para Fedora con dnf.

**Ahora**: lo movimos a un paso 4 limpio que, igualmente por SSH no interactivo, ejecuta apt-get update && install -y … en Ubuntu y dnf update && install -y … en Fedora, con mensajes claros.

**¿QUE RECOMIENDO?**

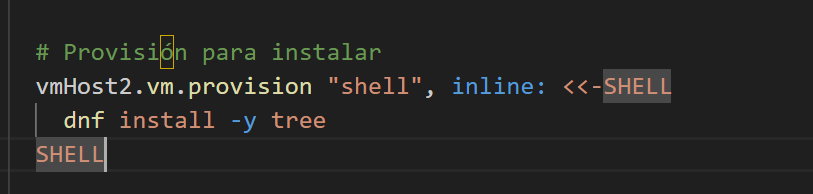
**APRENDER COMO FUNCIONA SSHPASS PARA EXPLICARLE AL PROFE PORQUE LO USAMOS, PORQUE AÑADIMOS EL STRICTHOSTKEYCHECKING.   
PASAR EL SCRIPT PÓR CHATGPT Y PEDIRLE QUE DE MANERA SIMPLIFICADA PERO ABARCANDO TODOS LOS PASOS, EXPLIQUE EL FUNCIONAMIENTO.**

**DEJO LAS DOS VERSIONES PUSHEADAS EN EL REPO, CONJUNTAMENTE DE LA VERSIÓN DEL VAGRANTFILE MODIFICADA Y LA VERSION ANTERIOR DEL VAGRANTFILE.**

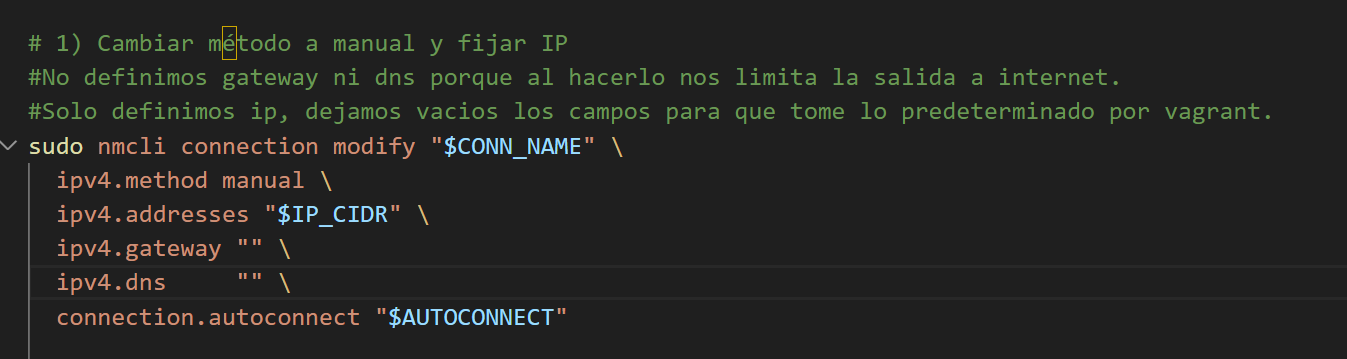
**REALICE UN SCRIPT PARA QUE SE CONFIGURE LA RED EN LA VM DE PRODU PARA PODER HACER EL CRUCE DE SSH DE AMBAS VMS. ESTE SCRIPT SE EJECUTA DESDE LA RUTA DE SCRIPTS AUTOMATICAMENTE AL CREAR LAS VM CON EL VAGRANTFILE. (SCRIPT\_CONFIG\_RED\_PRODUCCION.SH)**

Al provisionar al vagrantfile de este script, daba error.

Esto ocurría porque se colocaba una linea antes de la provision del script de instalacion de la ultima linea:



Al configurarse el adaptador con los nuevos parametros para la red (ip, gateway, dns, etc) limitaba la conexion hacia internet, y solo conectaba con la vm de testing, para que esto no ocurra saqué los parámetros de gateway y dns para que tome los predeterminados del sistema de vagrant/oracle o quien administre esa configuracion que en principio SI daba conexion a internet.



Asi, dejo funcionando la conexión entre equipos provisionando la ip definida, y la conexión a internet dejando la configuración por defecto de esos parámetros vacios.

**El script en cuestion se llama script\_config\_red\_produccion.sh**

**El script de configvm\_2 presentó otro problema, si bien copia de testing a produ la clave ssh, no lo hacia en viceversa. Con los siguientes bloques agregados y la creacion de variables que contienen la info de usuario y password, tambien facilito el script.   
Agrego bloque de copiado de produ a testing para que asi pueda ser bidireccional, y ademas, creo otro bloque donde si en produ no tenia las claves creadas, las crea. (paso 4 y 5 del script) Aparte, tambien separé del script, la instalación de paquetes para que sea mas facil testear el prompt y separar este (podemos unificarlo o dejar una version unificada). En el bloque donde copia de produ a testing, agregue un if donde si sshpass no está instalado en el equipo, lo instala y sigue con el script.**

**Tambien añadí un paso 0 donde instala sshpass en el equipo de testing.**

**El script se llama script\_configvm\_v3.sh**