

Búsqueda (Ctrl+Alt+E)

Cerrar

Actividad

Chat

Equipos

...

BeTek

LABORATORIO PRÁCTICO DE LINUX

CONTENIDO

OBJETIVOS	2
1. INSTALACIÓN VIRTUALBOX	3
2. DESCARGAR LA ISO DEL SISTEMA OPERATIVO.....	5
3. CREACIÓN MÁQUINA VIRTUAL	6
4. INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO CENTOS 7.....	9
5. CONEXIÓN POR SSH A LA MÁQUINA VIRTUAL.....	15
6. EJECUCIÓN DE COMANDOS BÁSICOS	19
7. CREACIÓN BASH SCRIPT	24
8. TAREAS DE PROGRAMACIÓN CRON EN LINUX	27
CONCLUSIONES.....	32

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.

OBJETIVOS

El objetivo general de realizar un laboratorio práctico de Linux es proporcionar una experiencia práctica y enriquecedora para los estudiantes, permitiéndoles aplicar los conceptos teóricos previamente vistos y adquirir habilidades prácticas en la instalación y el uso de Linux.

A continuación, los objetivos específicos:

- **Familiarización con Linux:** El objetivo principal es que los estudiantes se familiaricen con el entorno de trabajo proporcionado por el sistema operativo Linux. Esto incluye aprender a navegar por la línea de comandos, comprender la estructura de archivos, ejecutar comandos básicos y explorar las funcionalidades del sistema.
- **Aprender a instalar Linux en una máquina virtual:** Proporcionar a los estudiantes una experiencia práctica y concreta en la instalación y configuración de un sistema operativo Linux.
- **Aprender habilidades prácticas:** Los laboratorios prácticos permiten a los estudiantes aplicar conocimientos teóricos en situaciones reales. Al trabajar con Linux, los estudiantes pueden adquirir habilidades prácticas en administración de sistemas, configuración de redes, seguridad informática y más.

VirtualBox es un software de virtualización que permite a los usuarios ejecutar múltiples sistemas operativos en una sola máquina física. Es útil para probar diferentes sistemas operativos o programas en entornos seguros sin afectar el sistema principal.

Descargar VirtualBox: <https://www.virtualbox.org/wiki/Downloads>

NOTA: Si tiene otro software para virtualización de su preferencia, puede usarlo, no es obligatorio usar VirtualBox, el objetivo es poder virtualizar un sistema operativo Linux dentro de nuestro ordenador, creando lo que se conoce como máquina virtual.



Una vez descargado, le damos doble clic al archivo ejecutable y se nos abre el Wizard de instalación.

Un wizard o asistente es una herramienta que guía al usuario durante el proceso de configuración o instalación de un programa o aplicación, está diseñado para simplificar el proceso y evitar posibles errores o confusiones.



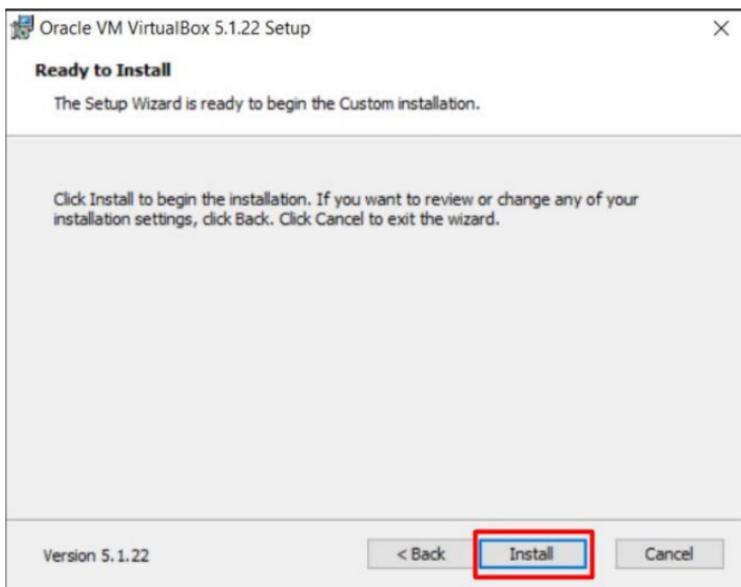
www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacen. Oficina
701.



Damos clic en instalar y esperamos que termine:

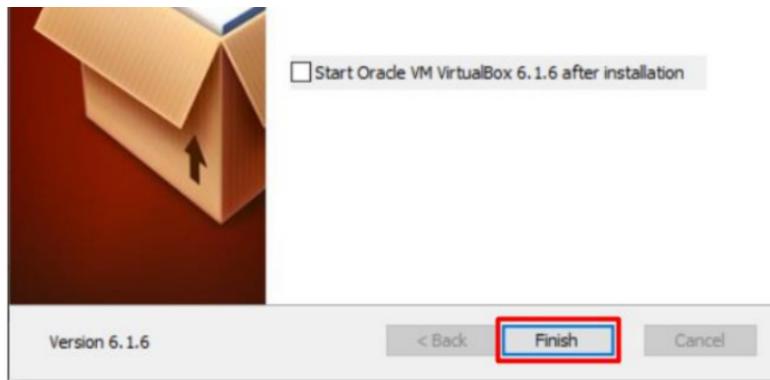


www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, AlmacenCentro. Oficina
701.

*BeTek





2. DESCARGAR LA ISO DEL SISTEMA OPERATIVO

Un archivo ISO, también conocido como imagen ISO, es un tipo de archivo que se utiliza para almacenar una copia exacta de un sistema de ficheros de una unidad óptica. Esto quiere decir que, si se copia un CD o DVD utilizando este formato, la copia resultante será un clon exacto de esa unidad óptica, pero en formato digital y cuando se monte en el ordenador será como si se utilizara el disco original. Suele ser el formato más utilizado para distribuir copias de sistemas operativos como Windows o los basados en GNU/Linux, pero también se puede utilizar para otras cosas, como para realizar copias digitales de seguridad de un CD, DVD o Bluray original.

http://edgeuno-bog2.mm.fcix.net/centos/7.9.2009/isos/x86_64/CentOS-7-x86_64-Minimal-2009.iso



NOTA: Esta es una versión básica o mínima.

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina 701.

*BeTek

3. CREACIÓN MÁQUINA VIRTUAL

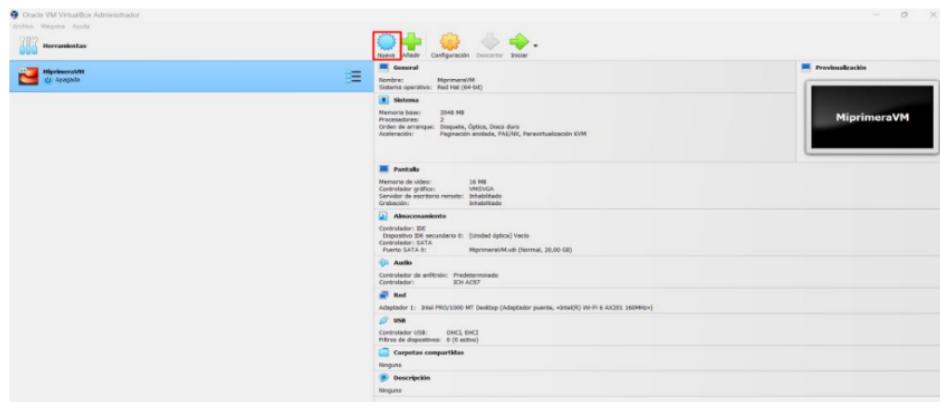
En VirtualBox las opciones “Nueva” y “Añadir” se refieren a diferentes acciones relacionadas con la creación y gestión de máquinas virtuales.

Al seleccionar “**Nueva**”, accedemos al asistente de creación de una máquina virtual, éste nos guía paso a paso para crear una máquina virtual desde cero. Podemos especificar detalles como el sistema operativo, la cantidad de memoria RAM, el tamaño del disco duro virtual y otros parámetros. Es útil cuando deseamos crear una máquina virtual completamente nueva.

Si ya tenemos una máquina virtual creada en el PC (por ejemplo, alguien nos proporcionó un archivo de máquina virtual), podemos seleccionar “**Añadir**”, luego

buscamos y agregamos esa máquina virtual existente a VirtualBox. Esta opción es útil cuando deseamos importar una máquina virtual previamente configurada y no necesitamos crearla desde cero.

En este laboratorio, vamos a crear una nueva máquina virtual desde cero:



Asignar como nombre a la máquina virtual MVNombreApellido, por ejemplo, MVLorenaJimenez.

Seleccionar la imagen ISO descargada en el punto 2.

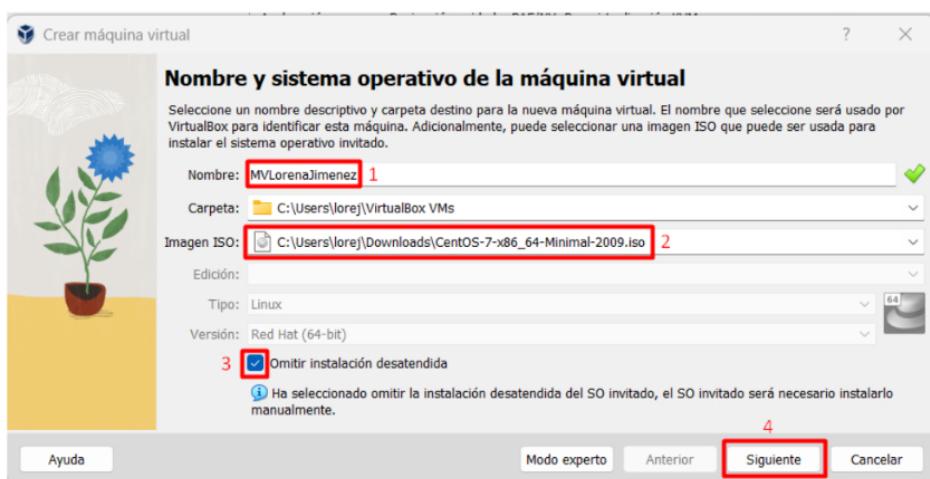
Omitir la instalación desatendida para poder configurar y personalizar el sistema operativo de forma manual y poder estar presente durante todo el proceso.

Clic en siguiente.

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.

*BeTek



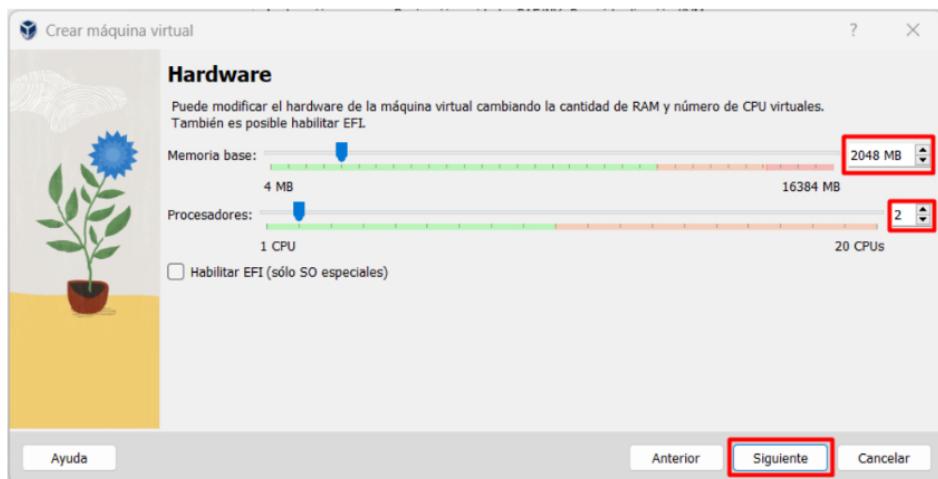
Dependiendo de cuántas máquinas virtuales vaya a tener encendidas al tiempo y de la capacidad de mi equipo físico, asigno los recursos a la máquina virtual.

la capacidad de mi equipo físico, asigne los recursos a la máquina virtual.

Para este laboratorio:

Asignar 2 GB de memoria, es decir, 2048 MB

Asignar 2 procesadores



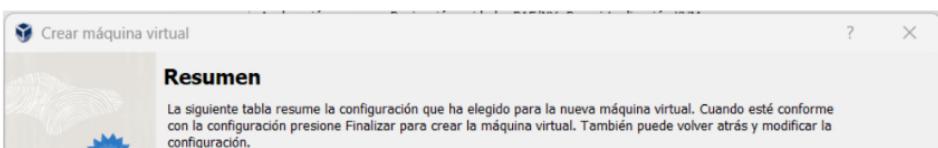
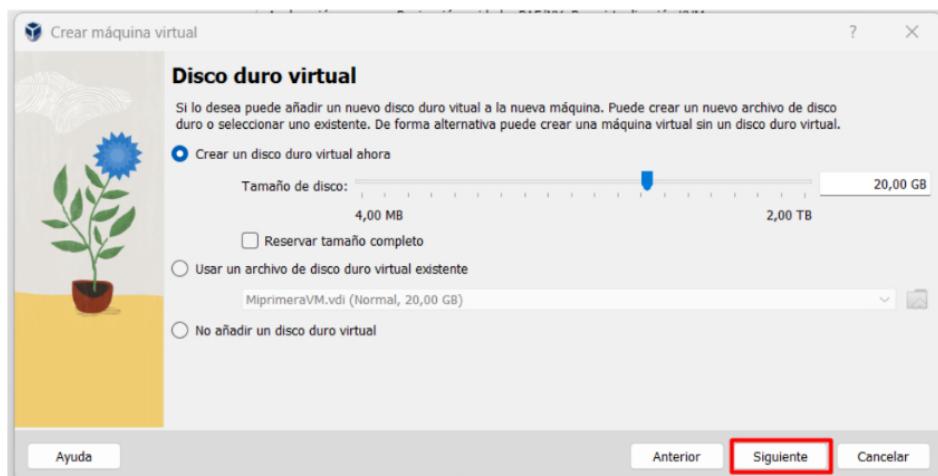
Creamos un disco virtual de 20 GB, no significa que la máquina virtual va a tomar toda esta capacidad, sino que va a crecer dinámicamente (de acuerdo a la necesidad) y puede llegar máximo hasta 20 GB:

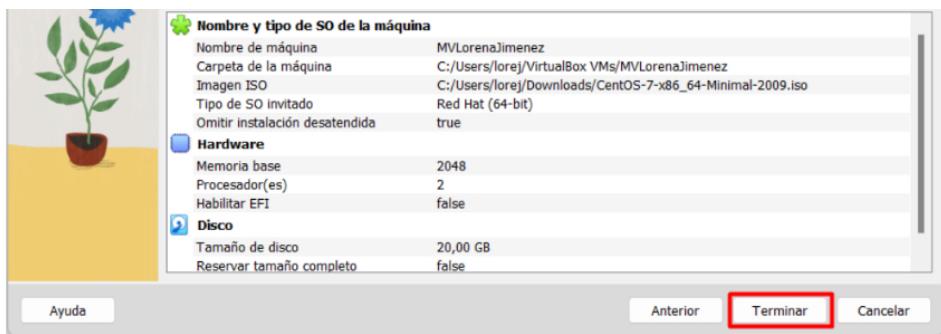
www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.

*BeTek





Seleccionamos la máquina virtual que acabamos de crear y le damos Iniciar:



www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.

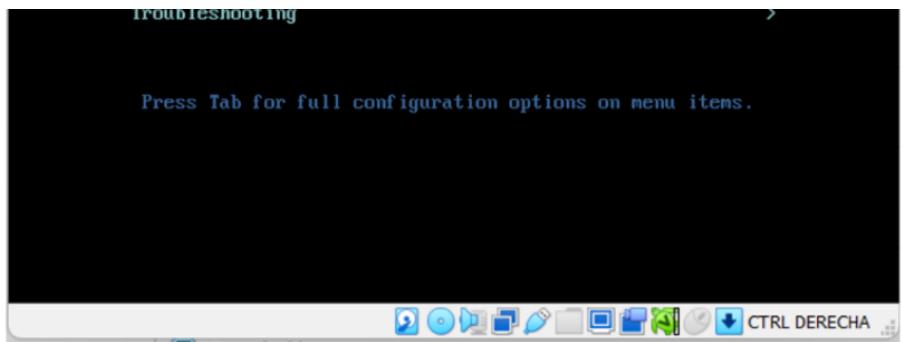
*BeTek

4. INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA OPERATIVO CENTOS 7

CentOS es una distribución de Linux de código abierto basada en Red Hat Enterprise Linux (RHEL). Es conocida como una de las distribuciones de Linux más populares y ampliamente utilizadas debido a su estabilidad, seguridad y compatibilidad con software empresarial.

Desplazarse con flecha direccional hacia arriba y al estar ubicados en Install CentOS 7 (se pone la letra en color blanco) dar enter:





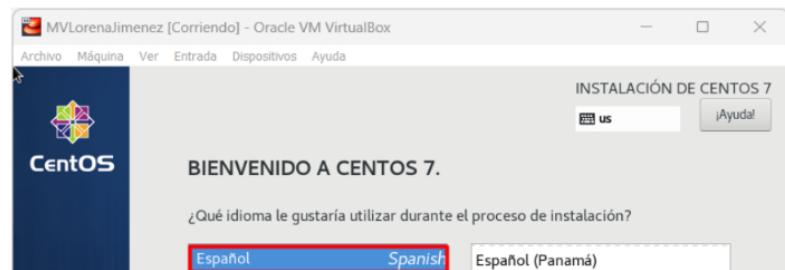
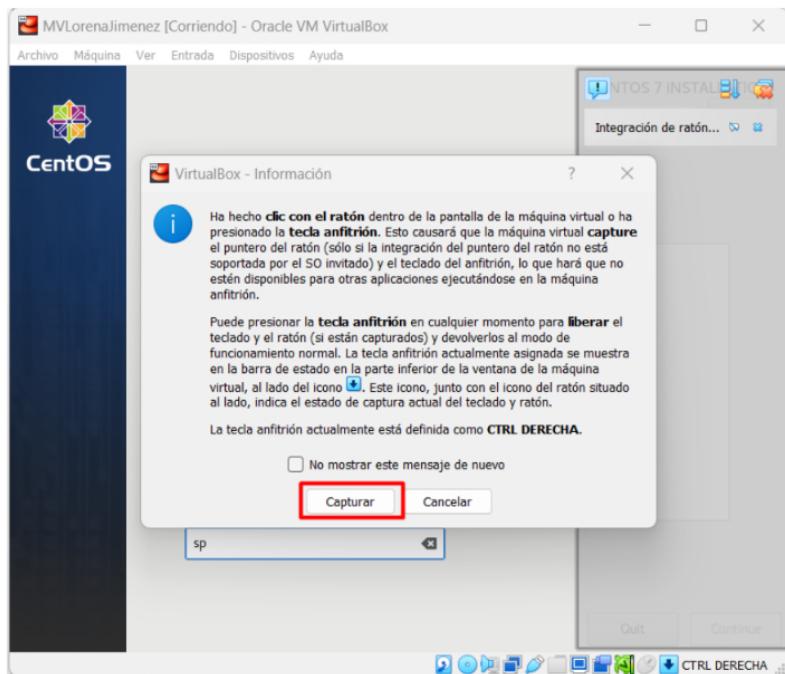
Seleccionar el idioma para la instalación, si prefieren inglés pueden dejarlo, sino lo cambiamos a español.

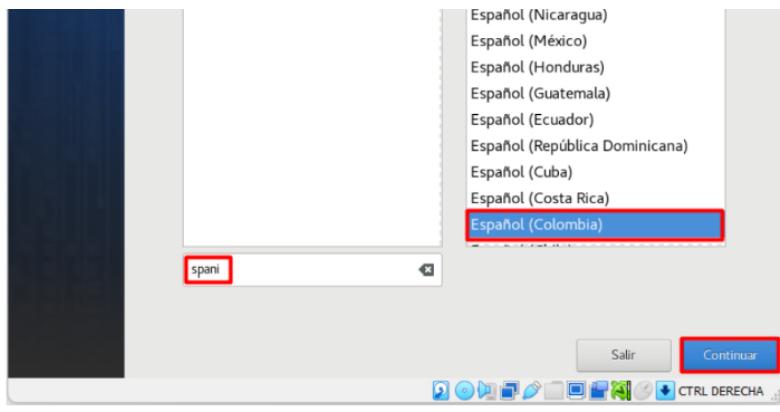
Como el mouse es de la máquina física y no de la máquina virtual, este mensaje pregunta si deseamos usar el mouse en la máquina virtual, damos Capturar para poder usarlo:

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.

*BeTek



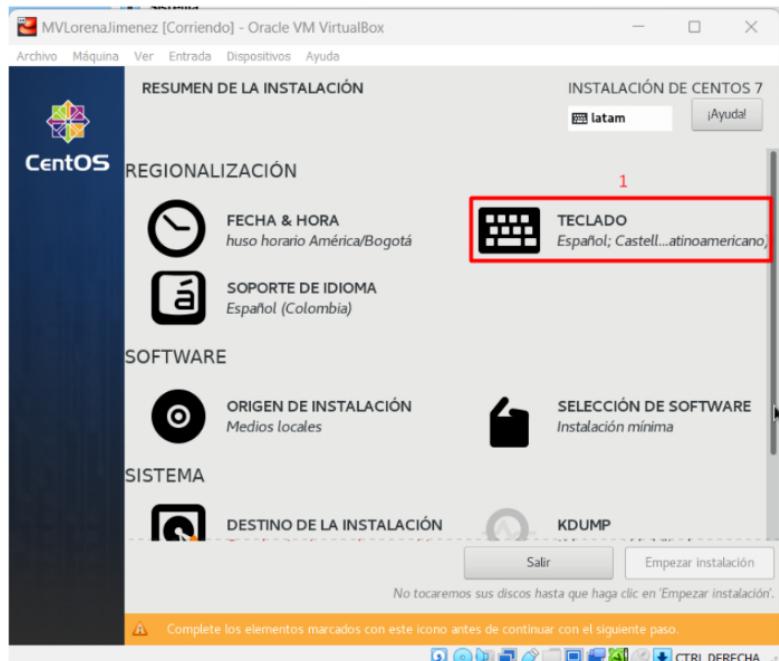


www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenento. Oficina
701.



Configurar el teclado, el destino de la instalación (lo dejamos como está por defecto) y la red:





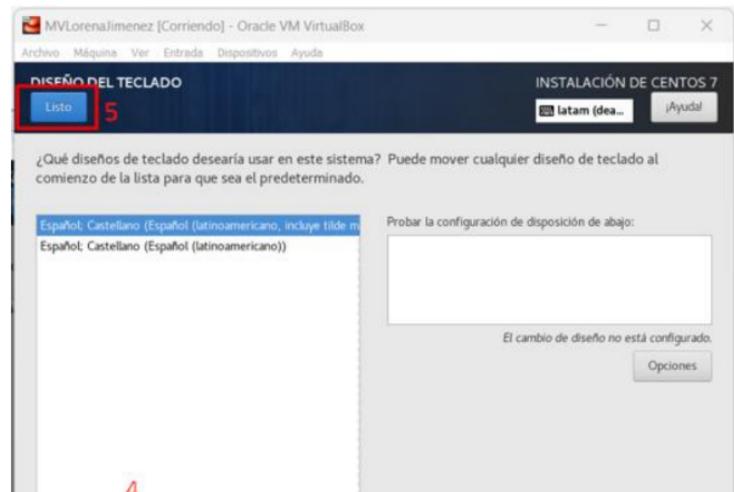
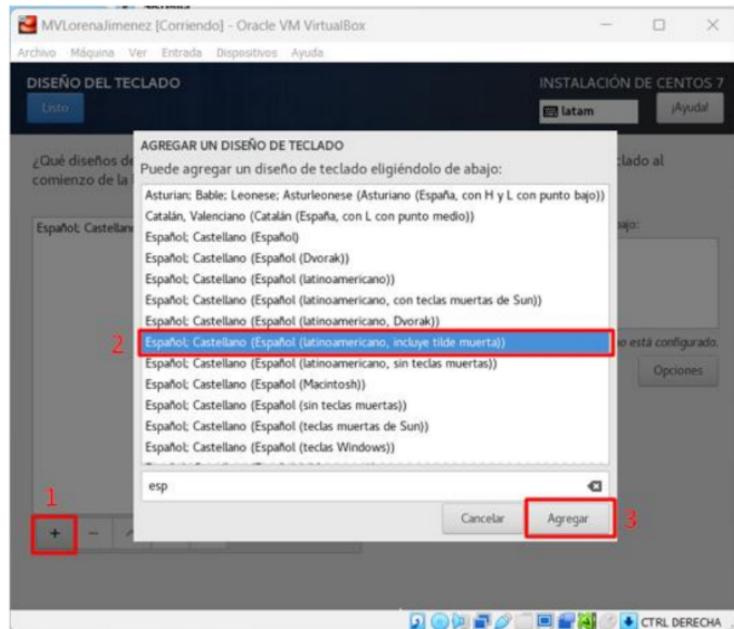
www.betek.la
@betek.la



Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.



Para el idioma del teclado agregamos español latinoamericano con tilde muerta y lo ponemos como primera opción:

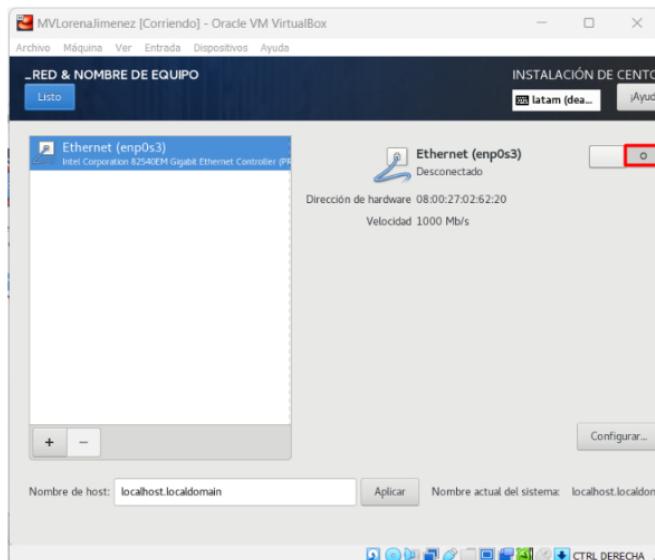


www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.

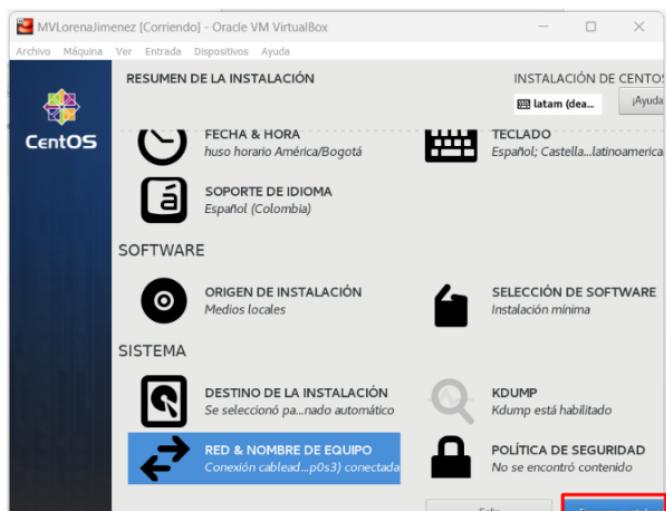


Encender la ethernet porque por defecto viene desactivada:



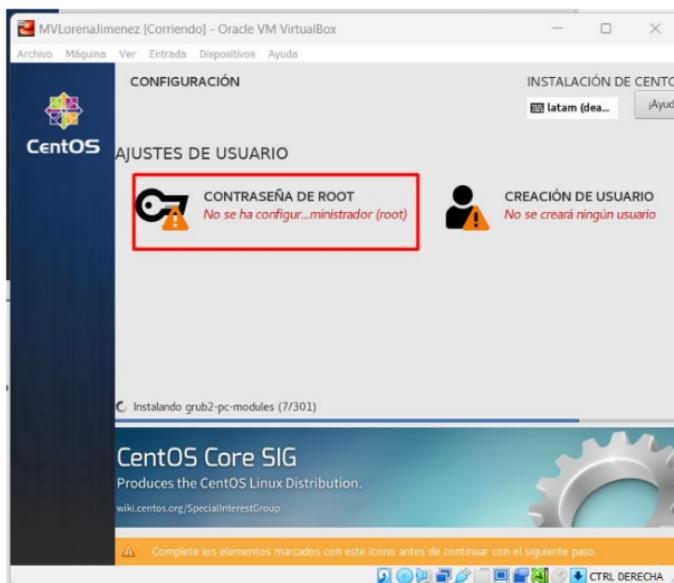
Ethernet es fundamental para conectar nuestro servidor con otros dispositivos en la red local sin requerir hardware físico adicional; es una solución eficiente para crear redes virtuales y segmentar el tráfico entre máquinas virtuales en un entorno seguro.

Empezamos la instalación:

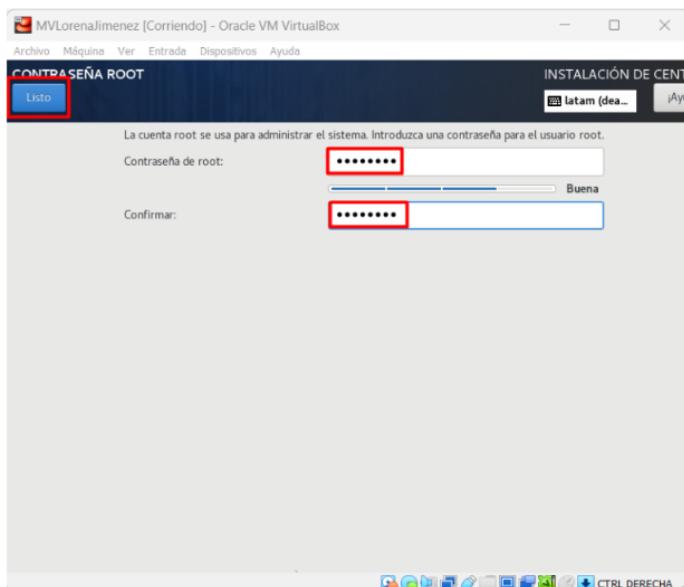




Configurar la contraseña del usuario root:

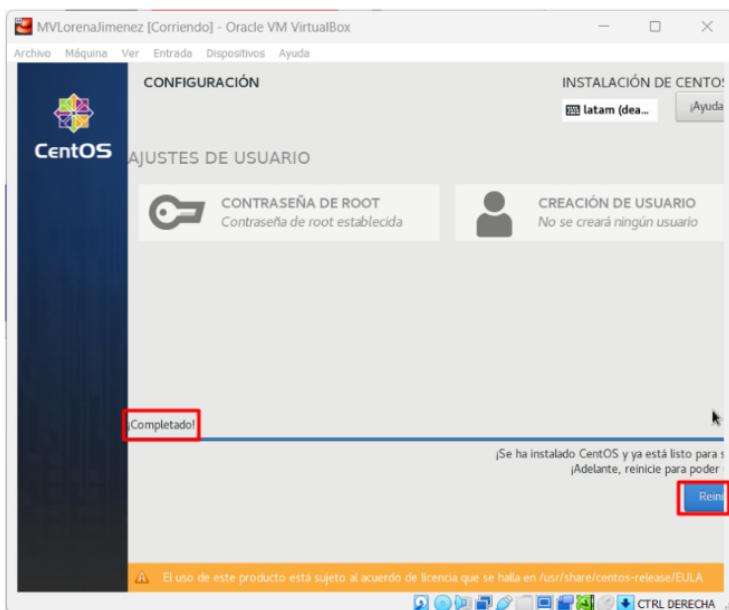


El usuario root en es aquel que tiene todos los permisos en el sistema operativo, también se le conoce como cuenta raíz, usuario raíz y super usuario. El usuario root puede acceder a cualquier archivo y ejecutar cualquier comando, es decir, es el súper administrador del sistema operativo.





Cuando termine la instalación, reiniciar:



5. CONEXIÓN POR SSH A LA MÁQUINA VIRTUAL

Nos logueamos con el usuario root y la contraseña que creamos durante la instalación.

Además, con el comando `ip a` vamos a consultar la información sobre las direcciones IP asignadas a mi máquina virtual, a verificar la configuración actual de las interfaces de red y a realizar ajustes si es necesario.

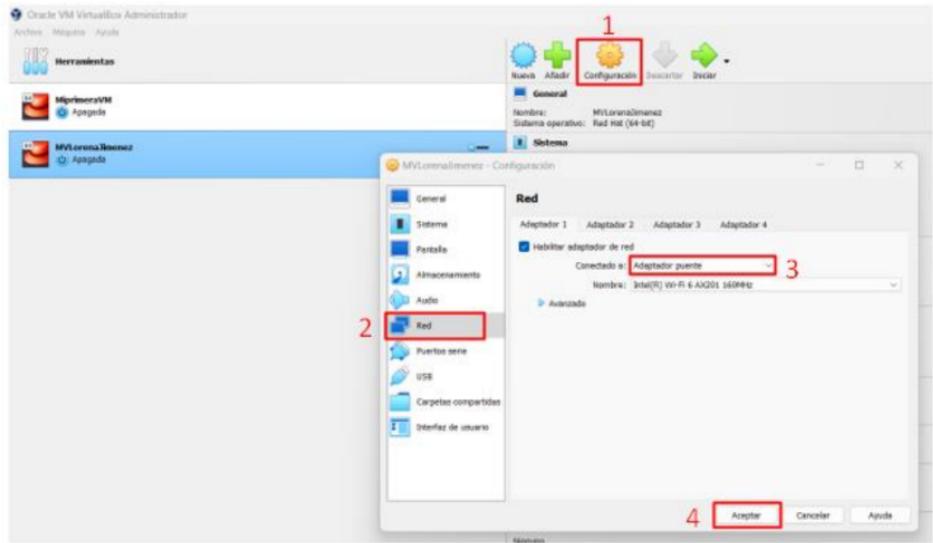
NOTA: Una dirección IP es una dirección única que identifica a un dispositivo en una red. Se expresa como un conjunto de cuatro números separados por puntos, que pueden variar desde el 0 al 255. La dirección IP permite la comunicación en internet entre el remitente y el destinatario de los datos.

```
MVLorenaJimenez [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
CentOS Linux 7 (Core)
Kernel 3.18.8-1168.el7.x86_64 on an x86_64
localhost login: root
Password:
Last login: Sun Mar 24 15:47:39 on ttys1
[root@localhost ~]# ip a
1: lo <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:00:27:f2:5e:67 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute dynamic enp0s3
        valid_lft 66306sec preferred_lft 86306sec
        inet6 fe80::3391:512:1115:37a3/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
[root@localhost ~]# shutdown -h now
```

Ahora vamos a hacer el primer diagnóstico de un problema común: La máquina virtual no tiene salida a internet.

Al ejecutar el comando **ip a** encontramos que el servidor tiene una dirección IP 10.0.2.15, éste es un direccionamiento para el entorno virtual y solo es visible desde el virtualizador, es decir, no tiene contacto con la red exterior, lo que incluye cualquier dispositivo incluyendo el router por donde sale a internet.

Apagamos la máquina virtual con el comando **shutdown -h now** y vamos a revisar la configuración de red:



El adaptador de red se encuentra en modo NAT, debemos pasarlo a modo BRIDGE o puente para que la máquina virtual tome un direccionamiento en la red que tenga salida a internet, ya que en NAT toma un direccionamiento 10.0.X.X para el entorno virtual sin salida a internet.

Luego de hacer el cambio, iniciamos nuevamente la máquina, nos conectamos a ella con el usuario root y ejecutamos el comando ***ip a*** para verificar que ya tiene una dirección IP 192.168.1.8, a la cual nos podemos conectar por ssh desde PowerShell.

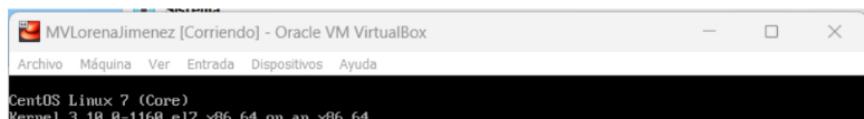


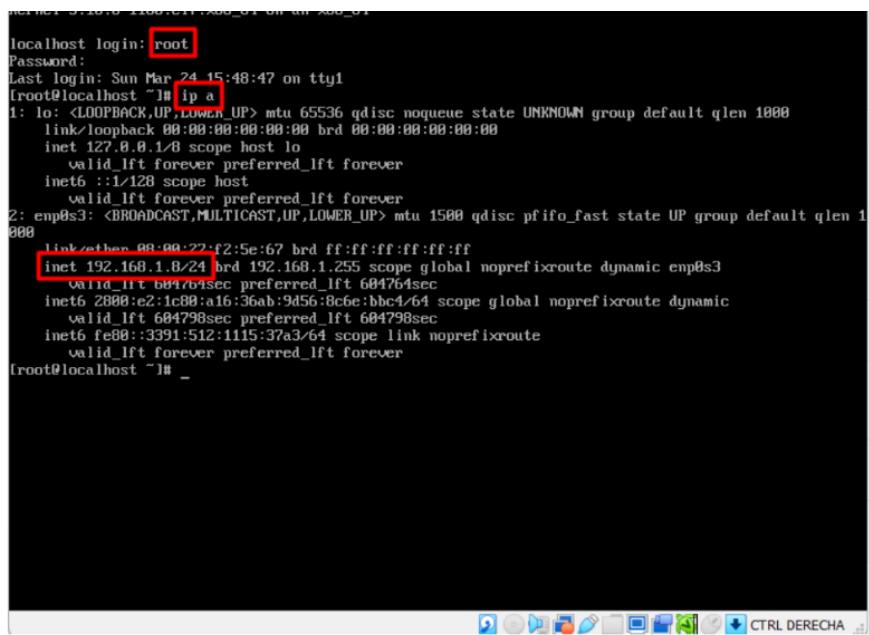
www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.

*BeTek





```
root@localhost ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback brd 00:00:00:00:00:00 state UP group default qlen 1000
        inet 127.0.0.1/8 brd 127.0.0.1 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 brd :: scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:22:f2:5e:67 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.8/24 brd 192.168.1.255 scope global noprefixroute dynamic enp0s3
            valid_lft 684764sec preferred_lft 684764sec
        inet6 2800:ec1c:0:a16:36ab:9d56:8c6e:bcb4/64 scope global noprefixroute dynamic
            valid_lft 684798sec preferred_lft 684798sec
        inet6 fe80::391:512:1115:37a3/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
[root@localhost ~]# _
```

SSH son las siglas de Secure Shell, es un protocolo de red seguro que se utiliza para acceder de forma remota a computadoras y servidores.

Proporciona una conexión cifrada entre el cliente y el servidor, lo que garantiza que la información transmitida, como contraseñas, comandos y datos, esté protegida contra posibles ataques de interceptación o manipulación, evitando que pueda filtrarse y que un tercero pueda ver esos datos.

PowerShell, originalmente conocido como Windows PowerShell, es una interfaz de línea de comandos (CLI) que permite ejecutar scripts y facilita la configuración, administración y automatización de tareas en múltiples plataformas.

PowerShell permite una comunicación remota a través de SSH para administrar sistemas tanto Linux como Windows.

Ejecutar el comando `ssh root@192.168.X.X` <reemplazamos por la IP de nuestra máquina virtual>

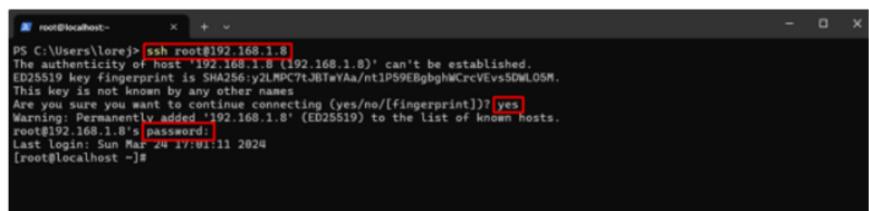
Ingresar con el usuario root y su respectiva contraseña:

www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacenamiento. Oficina
701.

*BeTek

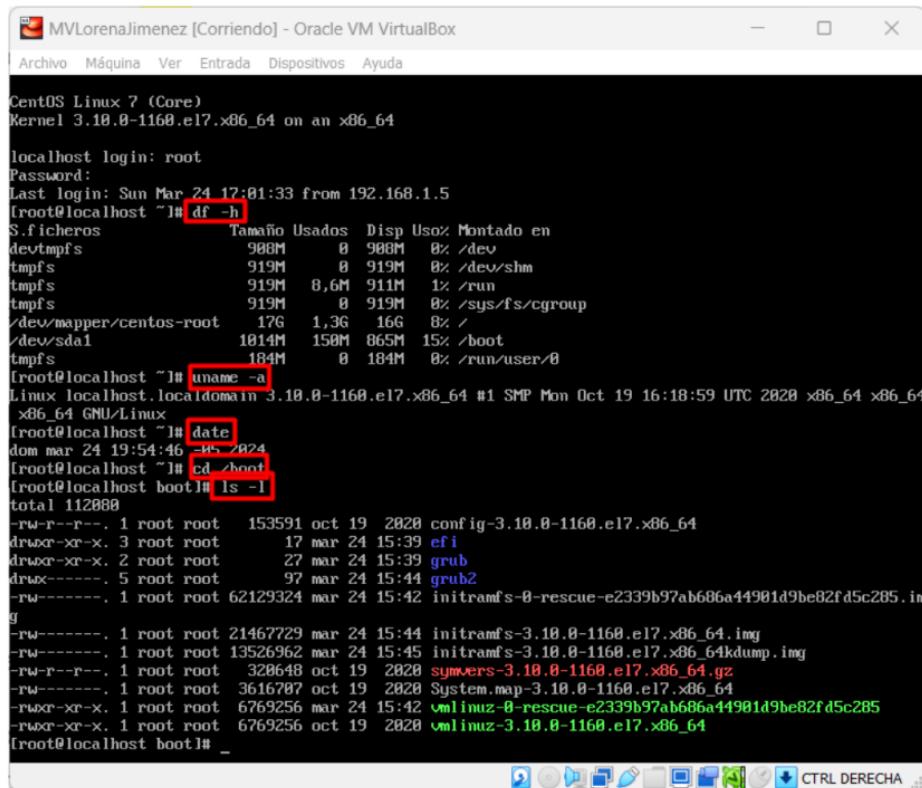


```
PS C:\Users\lorej> ssh root@192.168.1.8
The authenticity of host '192.168.1.8 (192.168.1.8)' can't be established.
ED25519 key fingerprint is SHA256:y2LMPC7tJB1wYAA/n1P59EBgbghMCrcVEvs5DWL0SM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added "192.168.1.8" (ED25519) to the list of known hosts.
root@192.168.1.8's password:
Last login: Sun Mar 24 17:01:11 2024
[root@localhost ~]# _
```

Ya estamos conectados a nuestra máquina virtual por SSH mediante PowerShell y podemos empezar a ejecutar comandos.

6. EJECUCIÓN DE COMANDOS BÁSICOS

Ejecutar algunos de los comandos de Linux más utilizados, como: Revisar el estado actual del sistema, comprobar el uso de espacio en disco, imprimir toda la información del sistema, revisar la hora y la fecha del sistema, desplazarse a un directorio, listar los archivos dentro de ese directorio, entre otros:



```
CentOS Linux 7 (Core)
Kernel 3.10.0-1160.el7.x86_64 on an x86_64

localhost login: root
Password:
Last login: Sun Mar 24 17:01:33 from 192.168.1.5
[root@localhost ~]# df -h
S. Ficheros Tamaño Usados Disp Uso% Montado en
/devtmpfs 980M 0 980M 0% /dev
tmpfs 919M 0 919M 0% /dev/shm
tmpfs 919M 8,6M 911M 1% /run
tmpfs 919M 0 919M 0% /sys/fs/cgroup
/dev/mapper/centos-root 17G 1,3G 16G 8% /
/dev/sda1 1014M 150M 865M 15% /boot
tmpfs 184M 0 184M 0% /run/user/0
[root@localhost ~]# uname -a
Linux localhost.localdomain 3.10.0-1160.el7.x86_64 #1 SMP Mon Oct 19 16:18:59 UTC 2020 x86_64 x86_64
x86_64 GNU/Linux
[root@localhost ~]# date
dom mar 24 19:54:46 -05 2024
[root@localhost ~]# cd /boot
[root@localhost boot]# ls -l
total 112088
-rw-r--r--. 1 root root 153591 oct 19 2020 config-3.10.0-1160.el7.x86_64
drwxr-xr-x. 3 root root 17 mar 24 15:39 efi
drwxr-xr-x. 2 root root 27 mar 24 15:39 grub
drwx----- 5 root root 97 mar 24 15:44 grub2
-rw-----. 1 root root 62129324 mar 24 15:42 initramfs-0-rescue-e2339b97ab686a44981d9be82fd5c285.ima
g
-rw-----. 1 root root 21467729 mar 24 15:44 initramfs-3.10.0-1160.el7.x86_64.img
-rw-----. 1 root root 13526962 mar 24 15:45 initramfs-3.10.0-1160.el7.x86_64kdump.img
-rw-r--r--. 1 root root 328648 oct 19 2020 symsvers-3.10.0-1160.el7.x86_64.gz
-rw-----. 1 root root 3616787 oct 19 2020 System.map-3.10.0-1160.el7.x86_64
-rw-r--r--. 1 root root 6769256 mar 24 15:42 vmlinuz-0-rescue-e2339b97ab686a44981d9be82fd5c285
-rw-r--r--. 1 root root 6769256 oct 19 2020 vmlinuz-3.10.0-1160.el7.x86_64
[root@localhost boot]# _
```

www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.

*BeTek

Crear un nuevo directorio y un archivo dentro de éste, luego con vi, vim o nano, editar dicho archivo:

```
[root@localhost ~]# cd /dev
[root@localhost dev]# mkdir nuevo-directorio
[root@localhost dev]# ls -l
total 0
```

NOTA: En Linux a las carpetas se les llaman directorios

```
[root@localhost dev]# cd nuevo-directorio/
```

```
[root@localhost nuevo-directorio]# ls -l  
total 0  
[root@localhost nuevo-directorio]# vi nuevo_archivo
```

Para insertar el texto digitamos **i**:

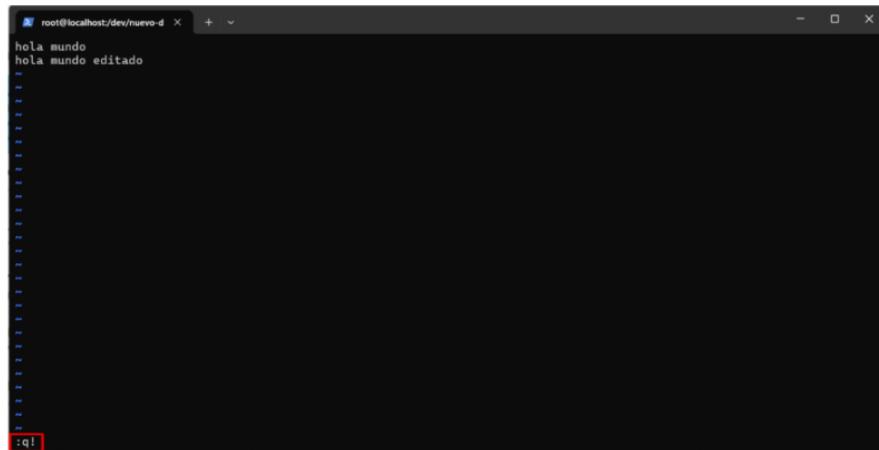
```
root@localhost:/dev/nuevo-d ~ + v
holamundo
holamundo editado
-- INSERT --
```

Para salir del editor digitamos **Ctrl + c**, si queremos guardar los cambios digitamos :**q!**, si queremos salir y guardar digitamos :**wq!**

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacen Centro. Oficina
701.

*BeTek



Modificar los permisos del archivo creado previamente.

NOTA: En Linux, cada archivo está asociado a tres clases de usuario: Propietario, miembro de grupo y otros. También tiene tres permisos: lectura (r), escritura (w) y ejecución (x).

Primero verificar los permisos actuales del archivo:

```
[root@localhost dev]# cd nuevo-directorio/  
[root@localhost nuevo-directorio]# ls -l  
total 4  
-rw-r--r--. 1 root root 30 mar 24 20:18 nuevo_archivo  
[root@localhost nuevo-directorio]#
```

En el archivo nuevo_archivo el propietario tiene permiso de lectura y escritura, los miembros de grupo tienen permiso de lectura y los demás usuarios tienen permiso de lectura.

Cambiamos el permiso del grupo a escritura, luego a lectura y escritura y finalmente, lo dejamos como estaba inicialmente solo de lectura:

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.

*BeTek

```
[root@localhost nuevo-directorio]# vi nuevo_archivo  
[root@localhost nuevo-directorio]# ls -l  
total 4  
-rw-r--r--. 1 root root 11 mar 24 20:55 nuevo_archivo  
[root@localhost nuevo-directorio]# chmod g=w nuevo_archivo  
[root@localhost nuevo-directorio]# ls -l  
total 4  
-rw--w-r--. 1 root root 11 mar 24 20:55 nuevo_archivo  
[root@localhost nuevo-directorio]# chmod g=rw nuevo_archivo  
[root@localhost nuevo-directorio]# ls -l  
total 4  
-rw-rw-r--. 1 root root 11 mar 24 20:55 nuevo_archivo  
[root@localhost nuevo-directorio]# chmod g=r nuevo_archivo  
[root@localhost nuevo-directorio]# ls -l  
total 4  
-rw-r--r--. 1 root root 11 mar 24 20:55 nuevo_archivo  
[root@localhost nuevo-directorio]#
```

Otra forma de asignar permisos es con números, en los entornos Unix básicamente cada permiso tiene el siguiente valor:

Lectura: 4

Escritura: 2

Ejecución: 1

Debemos aprender a jugar con estos números de la siguiente manera:

Si deseamos establecer un permiso de escritura usaremos el 6 (4 + 2= Lectura + Escritura)

Si deseamos que un usuario pueda ejecutar usaremos el 7 (4 + 2 + 1= Lectura + Escritura + Ejecución)

Tabla de valores permisos Linux

- 0: Sin permisos
- 1: Ejecución
- 2: Escritura
- 3: Lectura y escritura
- 4: Lectura
- 5: Lectura y ejecución
- 6: Lectura y escritura
- 7: Lectura, escritura y ejecución

NOTA: No usar el comando **chmod 777** (RWX / RWX / RWX), éste permite que todos los usuarios puedan leer, escribir y ejecutar en el archivo o carpeta.

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.



Lo recomendado es **chmod 755** (RWX / RW / RW), con este permiso el propietario del archivo puede leer, escribir y ejecutar el archivo mientras que los demás pueden leer y escribir el archivo mas no ejecutar.

```
[root@localhost nuevo-dIRECTORIO]# chmod 755 nuevo_archivo
[root@localhost nuevo-dIRECTORIO]# ls -l
total 4
-rwxr-xr-x. 1 root root 11 mar 24 20:55 nuevo_archivo
[root@localhost nuevo-dIRECTORIO]#
```

Cambiar el nombre de un archivo usando el comando **mv**, así:

```
mv nombre_archivo_antiguo.txt nombre_archivo_nuevo.txt
```

```
[root@localhost nuevo_dIRECTORIO]# ls -l
total 16
-rw-r--r--. 1 root root 11 mar 25 11:02 archivo_original
-rw-r--r--. 1 root root 110 mar 25 11:25 dias.sh
-rw-r--r--. 1 root root 119 mar 25 11:06 function.sh
-rw-r--r--. 1 root root 60 mar 25 11:51 incremento.sh
[root@localhost nuevo_dIRECTORIO]# vi function.sh
[root@localhost nuevo_dIRECTORIO]# vi function.sh
[root@localhost nuevo_dIRECTORIO]# mv function.sh condicional.sh
[root@localhost nuevo_dIRECTORIO]# ls -l
total 16
-rw-r--r--. 1 root root 11 mar 25 11:02 archivo_original
-rw-r--r--. 1 root root 119 mar 25 11:06 condicional.sh
-rw-r--r--. 1 root root 110 mar 25 11:25 dias.sh
```

```
-rw-r--r--. 1 root root 60 mar 25 11:51 incremento.sh  
[root@localhost nuevo_directorio]#
```

Eliminar el archivo creado, usando **rm -i** para que pida confirmación antes de borrar, si estamos seguros que queremos borrar lo hacemos con **rm -f** para que no pida confirmación:

```
-rwxr-xr-x. 1 root root 11 mar 24 20:55 nuevo_archivo  
[root@localhost nuevo-directorio]# rm -i nuevo_archivo  
rm: ¿borrar el fichero regular «nuevo_archivo»? (s/n) s  
[root@localhost nuevo-directorio]# ls -l  
total 0  
[root@localhost nuevo-directorio]#
```

Ejecutar **shutdown -h now** para apagar la máquina, si no le ponemos el now se demora un poco en apagar, si se lo ponemos la apaga de inmediato.

www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.



7. CREACIÓN BASH SCRIPT

Bash (Bourne-Again Shell) es un intérprete de comandos de Unix que lee y ejecuta varios comandos del shell e interactúa con el sistema operativo para ejecutarlos.

Cuando necesitamos ejecutar varios comandos bash, no tenemos que ejecutarlos manualmente de uno en uno, en su lugar es posible crear un archivo de script que contenga funciones bash para ejecutar esos comandos.

Un script de bash es un archivo de texto que contiene una serie de comandos que serán ejecutados por el intérprete de bash, es un lenguaje de programación que se utiliza principalmente en sistemas Unix y Linux

Cuando escribimos bash en un editor de texto, estamos compilando comandos o funciones bash, los cuales son un conjunto de comandos que pueden ser llamados numerosas veces tan solo usando el nombre de la función. El texto se guarda entonces como un archivo de script bash ejecutable con la extensión .sh.

Usamos el editor de texto vi (también se puede usar vim, nano o el editor de preferencia) para crear el archivo, introduciendo el comando: **vi function.sh**, esto abrirá un nuevo archivo .sh para editar.

Comenzamos escribiendo **#!/bin/bash** para que el shell sepa que debe ejecutar comandos utilizando el intérprete bash.

A continuación, vamos a ver algunos ejemplos sencillos para crear con éxito nuestros primeros bash script:

- El primer ejemplo utiliza un bucle for para imprimir todos los días de la semana:

```
#!/bin/bash
for days in Lunes Martes Miércoles Jueves Viernes Sábado Domingo
do
echo "Day: $days"
done
```

Los comandos bash de bucle son útiles para ejecutar comandos varias veces, hay varios tipos, el bucle **for** ejecuta el comando para una lista de elementos.

El comando **echo** es muy conocido y utilizado en muchos lenguajes de programación para imprimir la salida de texto estándar en el terminal.



En la línea 2, «days» se convierte automáticamente en una variable, cuyos valores son los nombres de los siguientes días. Luego, en el comando echo, utilizamos el símbolo \$ para llamar a los valores de la variable.

A screenshot of a terminal window titled 'root@localhost: /dev/nuevo_d'. The window shows a portion of a bash script in the vi editor. The first few lines of the script are: '#!/bin/bash' and 'for days in Lunes Martes Miércoles Jueves Viernes Sábado Domingo'. Below the script, there are several tilde (~) characters indicating the cursor's position in the editor.

Al terminar de utilizar el editor de texto vi, pulsamos **Ctrl+C** para cerrarlo y, a continuación, digitamos :wq! para guardar los cambios.

Para ejecutar un script bash en Linux, utilizamos el comando bash y especificamos el nombre del script a ejecutar. Para el bash que acabamos de crear sería **bash dias.sh**:

A screenshot of a terminal window showing the execution of the 'dias.sh' script. The command 'bash dias.sh' is highlighted with a red box. The output of the script is displayed below, showing the days of the week: 'Day: Lunes', 'Day: Martes', 'Day: Miércoles', 'Day: Jueves', 'Day: Viernes', 'Day: Sábado', and 'Day: Domingo'.

- El siguiente ejemplo utiliza un bucle while en el que el script evaluará una condición, si la condición es verdadera, seguirá ejecutando los comandos hasta que la salida ya no cumpla la condición definida.

www.betek.la

@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.



```
#!/bin/bash
i=0
while [ $i -le 5 ]
do
echo $i
((i++))
done
```

La variable (i) comienza con un valor 0 y el operador de incremento la aumentará en uno (i++). La condición establecida es menor o igual a cinco (le 5), por lo que el comando seguirá iterando hasta que la salida llegue a cinco. La salida de ese script será la siguiente:

```
[root@localhost nuevo_directorio]# vi incremento.sh
[root@localhost nuevo_directorio]# bash incremento.sh
0
1
2
3
4
5
[root@localhost nuevo_directorio]#
```

- El tercer ejemplo utiliza las sentencias **while** e **if**, ejecuta el bucle while cinco veces después de comprobar la sentencia condicional.

Muchos lenguajes de programación, incluido bash, utilizan sentencias condicionales como if, then y else para la toma de decisiones. Ejecutan comandos e imprimen salidas dependiendo de las condiciones.

La sentencia **if** va seguida de una expresión condicional, después le sigue **then** y el comando para definir la salida de la condición. El script ejecutará el comando si la condición expresada en la sentencia if es verdadera.

Sin embargo, si deseamos ejecutar un comando diferente si la condición es falsa, añadimos una sentencia else al script.



```
#!/bin/bash
isValid=true
count=1
while [ $isValid ]
do
echo $count
if [ $count -eq 5 ];
then
break
fi
((count++))
Done
```

En este script utilizamos las sentencias while e if para ejecutar el bucle while cinco veces después de comprobar la sentencia condicional.

La salida de este script será:

```
[root@localhost nuevo_directorio]# bash condicional.sh
1
2
3
4
5
[root@localhost nuevo_directorio]#
```

8. TAREAS DE PROGRAMACIÓN CRON EN LINUX

Cron y Crontab son herramientas esenciales en sistemas Linux para programar tareas de forma periódica y automatizada, liberándonos de la necesidad de ejecutar manualmente comandos o scripts, son excelentes herramientas para simplificar la administración de servidores.

Cron es un demonio (un proceso en segundo plano) que se ejecuta desde el inicio del sistema operativo, su función es comprobar si existen tareas programadas para su ejecución en momentos específicos.

Cron es un programador de trabajos basado en tiempo en Linux que permite a los usuarios automatizar tareas en intervalos específicos. Puede ejecutar scripts o comandos en un momento específico o en un conjunto de momentos. Cron es

esencial para los administradores de sistemas porque automatiza tareas como copias de seguridad, actualizaciones del sistema y rotación de registros.

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.



NOTA: Es fundamental que la hora y la zona horaria estén configuradas correctamente para que las ejecuciones coincidan con lo programado y satisfaga las necesidades para las que fue creado.

Para configurar el reloj del hardware con la hora actual del sistema, ejecutamos el comando: **hwclock -w**.

Crontab es la interfaz que nos permite agregar, modificar y eliminar tareas programadas en Cron, con Crontab podemos especificar qué comandos o scripts deseamos ejecutar y cuándo hacerlo.

Algunos pasos para trabajar con Crontab:

Agregar tareas: Utilizar **crontab -e** para abrir el editor de Crontab y añadir las tareas programadas.

Caracteres especiales: Usar caracteres especiales como *, -, /, etc., para definir intervalos de tiempo.

Gestión de los jobs: Listar, editar o eliminar tareas existentes con comandos como crontab -l, crontab -r, etc.

El archivo crontab contiene una lista de trabajos que ejecutará Cron. Cada trabajo consta de una línea con seis campos separados por espacios. Los campos representan el minuto, la hora, el día del mes, el mes, el día de la semana y el comando a ejecutar:

```
* * * * *  
- - - - -  
| | | | |  
+--- Día de la semana (0 - 7) [Ambos 0 y 7  
representan domingo]  
| | | +---- Mes (1 - 12)  
| | +----- Día del mes (1 - 31)  
| +----- Hora (0 - 23)  
+----- Minuto (0 - 59)
```

Los siguientes son los seis campos utilizados en el archivo crontab:

- Minuto: Este campo representa el minuto de la hora (0-59)
- Hora: Este campo representa la hora del día (0-23)
- Día del mes: Este campo representa el día del mes (1-31)



- Mes: Este campo representa el mes del año (1-12)
- Día de la semana: Este campo representa el día de la semana (0-6). El domingo está representado por 0 o 7
- Comando: Este campo representa el comando que se ejecutará

Algunos puntos a considerar:

Asterisco (*): Representa 'cada'. Si se pone en el campo de hora, significa «cada hora».

Guiones (-): Representa un rango. Por ejemplo, 1-5 en el campo de día de la semana significa «de lunes a viernes».

Comas (,): Se utilizan para separar valores. Por ejemplo, 1,15 en el campo de día del mes significa «el primero y el día quince del mes».

Ejemplos para tener claro el formato:

Ejecutar a las 6:30 PM todos los días: 30 18 * * *

Ejecutar a las 5:00 AM todos los lunes: 0 5 * * 1

Ejecutar cada hora, en punto, de lunes a viernes: 0 * * * 1-5

Ejecutar a las 2:30 PM y 4:30 PM todos los días: 30 14,16 * * *

Ejecutar a las 11:00 PM los primeros y últimos días del mes: 0 23 1,31 * *

Ejecutar cada 5 minutos: */5 * * * *

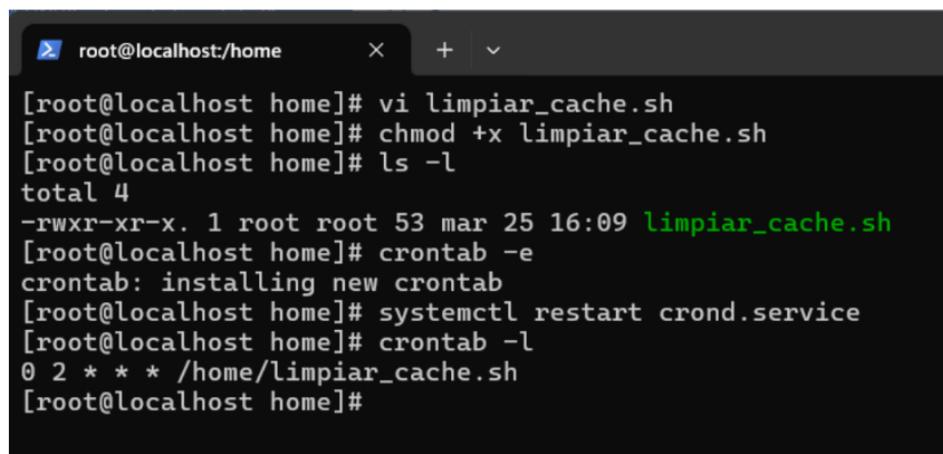
Ejecutar a las 8:45 AM todos los días excepto en diciembre: 45 8 * 1-11 *

Con los conceptos básicos claros, vamos a crear un script simple para automatizar la limpieza de la caché y vamos a programarlo para que se ejecute automáticamente con cron y crontab:

1. Crear el script: **vi limpiar_cache.sh**
2. Dentro del archivo, poner los siguientes comandos para liberar la memoria caché:
#!/bin/bash
sync; echo 3 > /proc/sys/vm/drop_caches

3. Guardar el archivo: En vi presionamos **Ctrl+C** para cerrarlo y luego digitamos **:wq!**
4. Darle permisos de ejecución: **chmod +x limpiar_cache.sh**
5. Editar el crontab para programar la tarea: **crontab -e**
6. Para ejecutar el script limpiar_cache.sh todos los días a las 2:00 AM, agregamos la siguiente línea al archivo:
0 2 * * * /ruta/al/limpiar_cache.sh
Reemplazamos /ruta/al/limpiar_cache.sh con la ubicación real del script, en este caso quedó así:
0 2 * * * /home/limpiar_cache.sh
7. Guardar los cambios y cerrar el editor: Crontab se encargará de ejecutar el script automáticamente según la programación establecida.

NOTA: Luego de modificar un archivo crontab, ya sea porque se agregó una nueva tarea o se modificó la configuración de una tarea existente, se recomienda reiniciar el servicio de cron para que las tareas programadas se ejecuten correctamente, para Centos 7 lo hacemos con el comando: **systemctl restart crond.service**:



```
root@localhost:~# vi limpiar_cache.sh
root@localhost:~# chmod +x limpiar_cache.sh
root@localhost:~# ls -l
total 4
-rwxr-xr-x. 1 root root 53 mar 25 16:09 limpiar_cache.sh
root@localhost:~# crontab -e
crontab: installing new crontab
root@localhost:~# systemctl restart crond.service
root@localhost:~# crontab -l
0 2 * * * /home/limpiar_cache.sh
root@localhost:~#
```

Para listar las tareas del Cron actual, usar el comando: **crontab -l**

Estos son ejemplos muy sencillos y básicos, apenas para familiarizarnos con el uso de bash script, sin embargo, hay mucho más que aprender, si quieras ser capaz de utilizar todo el potencial de bash debes practicar con los ejemplos que te hemos proporcionado y continuar explorando bash para lograr escribir scripts mejores y más eficientes.

Los ejemplos vistos en esta guía son muy didácticos, pero los scripts bash sirven en el día a día para prácticamente todo lo que tengamos que hacer y la principal razón para utilizarlos es la automatización.

Supongamos que necesitamos respaldar archivos todos los días a una hora específica, hacerlo manualmente no solo es una tarea que demanda tiempo y esfuerzo, sino que también puede provocar errores humanos. Aquí es donde los scripts bash ayudan, permitiendo programar esta tarea y olvidarnos de ella.

Además, podemos utilizarlos para automatizar prácticamente cualquier tarea en nuestros sistemas, desde tareas de administración hasta procesos de desarrollo de software, los scripts bash pueden hacerlo todo más eficiente.

Son muy útiles para la automatización de tareas rutinarias como realizar copias de seguridad de archivos, actualizar sistemas o limpiar directorios; también son de gran utilidad en la gestión de sistemas para monitorear el estado de diferentes servicios y recursos en los servidores y realizar tareas de mantenimiento.

En desarrollo los scripts bash pueden usarse para el despliegue de aplicaciones, para automatizar el proceso de despliegue de software, desde la descarga y la instalación hasta la configuración de aplicaciones y servicios, además en la automatización de tareas de desarrollo como la ejecución de pruebas, compilación de código y gestión de versiones.

Ya sea que estemos administrando un servidor, desarrollando software o simplemente buscando una manera de automatizar tareas habituales en nuestro PC, aprender a escribir scripts bash puede ser muy útil.

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.



- Este laboratorio práctico de Linux proporciona una introducción práctica al sistema operativo y sienta las bases para futuros estudios y proyectos relacionados con Linux, como el aprendizaje de la computación en la nube.
- Con la elaboración del laboratorio, los participantes tienen la oportunidad de familiarizarse con la línea de comandos de Linux, aprenden a navegar entre directorios, a crear archivos y directorios, y ejecutar comandos básicos, lo cual será de mucha utilidad para gestionar una nube por la línea de comandos (CLI) cuando no se puede o no se debe hacer por consola.
- A medida que avanzamos con el laboratorio encontramos desafíos y errores, por lo que la resolución de problemas fue una parte esencial del proceso, ayudando a desarrollar habilidades para solucionar problemas en un entorno Linux.

www.betek.la
@betek.la

Carrera 43 A # 34 – 155. Torre Norte, Almacentro. Oficina
701.