BACKUPS REMOTOS

Índice

RESUMEN

En este proyecto tratare de crear backups en remoto de forma segura y automatizada, para ello empleare varias herramientas como ssh, crontab y rsync. Con lo que de esta forma y el script que usare podre crear backups automatizados y con logs.

PRIMEROS PASOS

Lo primero que tratare de hacer para empezar con este proyectos sera crear una clave ssh que usare para la conexión automatizada por ssh con el servidor. Para ello empleare los siguientes comandos.

```
$ ssh-keygen
```

Con este comando creare una clave ssh en mi usuario, en el momento en el que me pida contraseña no le pondré nada para que la automatización funcione correctamente. Después para usar la clave ssh usare el siguiente comando:

```
$ ssh-copy-id usuario@direccion_servidor
```

De esta forma lo que haré sera copiar esa clave en el servidor para que cada vez que quiera conectarme y lo haga con esa clave no

me pida contraseña ni nada.
Otra cosa que haría para mejorar la seguridad seria crear un usuario con una contraseña robusta y directorios propios para los backups en el lado del servidor, y también haría las copias a través de alguna VPN para que el trafico quede cifrado y directo al servidor, de esta forma evitaría que algún atacante pueda hacer algo. Con todo esto la parte de ssh ya queda configurada.

SCRIPT PARA BACKUP

Lo siguiente que haré sera crear un script básico en bash para poder ejecutarlo de forma automática con los parámetros que yo quiera y rutas. Dejare aun así el script en mi pagina para que quien quiera descargarlo y usarlo que pueda.

```
cat backup.sh
#!/bin/bash

ORIGEN="/home/yerai/backup_ejemplo"
DESTINO="yerai@192.168.0.18:/home/yerai/backup_ejemplo"
USUARIO="backup"
SERVIDOR="192.168.0.18"
LOG_ARCHIVO="/var/log/backup.log"
FECHA=$(date  +%Y%m%d_%H%M%S)
DIAS=7

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_ARCHIVO"
}

error_exit() {
    log "ERROR: $1"
    exit 1
}

command -v rsync >/dev/null 2>&1 || error_exit "Rsync no instalado."

log "Intentando conexion SSH..."
ssh -o ConnectTimeout=5 -o BatchMode=yes "$USUARIO@$SERVIDOR" "echo 'Conexion Completada'" || error_exit "Conexion no establecida"

log "Creando directorio para backup..."
ssh "$USUARIO@$SERVIDOR" "mkdir -p $DESTINO/$FECHA"

log "Iniciando backup..."
rsync -avz --delete -e ssh --exclude='.cache' --exclude='*.tmp' "$ORIGEN/" "$USUARIO@$SERVIDOR:$DESTINO/$FECHA/" && log "Backup OK: $FECHA" || error_exit "Fallo rsync"

if [ $? -eq 0 ]; then
        log "Backup completado con exito: $FECHA"
        log "Limpiando backups mayores a $DIAS dias..."
        ssh "$USUARIO@$SERVIDOR" "find $DESTINO -maxdepth 1 -type d -mtime +$DIAS -exec rm -rf {} +"
else
        error_exit "Fallo en el backup"
fi

log "Backup terminado"
exit 0
```

En la imagen esta el script completo, ahora voy a desglosarlo para poder entenderlo.

Lo primero que pongo es el interprete que usare para el script que es bash, siempre habrá que ponerlo para crear el script.

Después me dispongo a marcar todas las variables necesarias, que en este caso nos harán falta los siguientes.

- ORIGEN: Aquí pondré el directorio que quiero hacerle backup en la maquina host.

- DESTINO: Aquí ira el directorio donde se guardara el backup.

- USUARIO: En esta variable pondré el usuario que usare para acceder por ssh.

- SERVIDOR: Aquí ira la dirección del servidor ya sea una ip o un dominio.

- LOG_ARCHIVO: Aquí creare un archivo donde podre ver los logs del backup para verlos cuando necesite.

- FECHA: Esta variable cuando la llame me imprimirá la fecha en la que se use.

- DÍAS: Esta variable la usare para poner cuantos días duraran los backups en el sistema.

Ahora me dispondré a crear dos funciones una es “log” y otra es “error_exit”.

Como se ve en la función de “log” haré que cuando llame a la función, me imprima la fecha y con “tee” haré que me lo imprima en pantalla y en la variable anteriormente definida.

Y en la función “error_exit” he hecho que cuando le llame porque hay un error lo marque en el log como error y que salga del script.

Después de de que haya definido las variables y las funciones ya puedo empezar con el script. Para ello empezare mirando verificando que el sistema disponga de rsync y si no lo tengo que me lance un error y me saque.

Después debo determinar y asegurar que la conexión SSH puede completarse, para ello empleo el comando ssh con 2 opciones, una es “ConnectTimeout” que me servirá para determinar cuanto tiempo quiero que dure la conexión, dado que solo quiero verificar si hay

conexión o no, y si no la hay que me lance un error y le definiré “BatchMode” en “yes” para que no me pregunte nada, dado que queremos automatizar la tarea.

El siguiente paso por el pasa el script es en mirar y crear en caso de que no haya un directorio en el destino definido anteriormente, y dentro de ese directorio otro con la fecha, para tener en todo momento localizados los backups. De esta forma si necesito restaurar en especifico alguna fecha podre sin problema.

Lo siguiente que haré sera ejecutar el backup con rsync y diferentes parámetros que explico ahora.
Como se ven con rsync empleare una serie de parámetros, con estos parámetros lo que haré sera hacer que la copia sea lo mas fiel posible, que me vaya imprimiendo todo lo que va transfiriendo y que comprima durante la transferencia para que use menos ancho de banda. Después con “—delete” haré que la copia sea idéntica a la que hay en el directorio origen. Y después excluiré los archivos que contengan “.cache” o “.tmp” dado que suelen ser archivos de programas que no me interesan copiar, y le pondré en caso de error un log también.

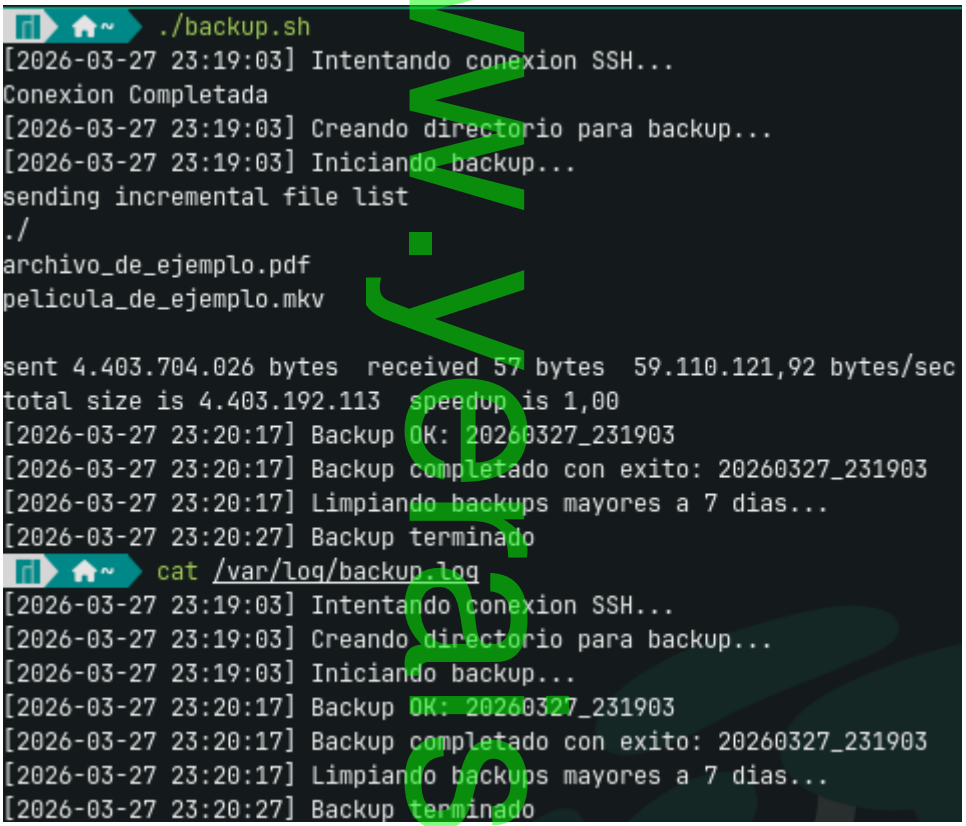
Y por ultimo en el script lo que voy a añadir va a ser una condicional que me verificara los backups que hay en el servidor y si tienen una longevidad mayor a 7 días los eliminara de forma automática.

Una vez con el script listo lo haré ejecutable y lo ejecutare para verificar que todo funciona correctamente. Para ello empleare primero un comando para hacer ejecutable el script y después el otro para ejecutarlo.

```
$ chmod +x backup.sh
```

```
$ ./backup.sh
```

De esta forma ejecuto el script y en la siguiente imagen dejare el log para ver como quedaría.



Como se ve en la imagen tengo el mismo script y después en la ruta definida de logs también guardados para poder acceder en cualquier momento y ver si todo esta bien o no.

Con esto ya habría terminado la parte del script y verificación del mismo, con lo que pasare al siguiente paso.

AUTOMATIZACIÓN DE BACKUP

Ahora me dispondré a automatizar el script, para ello voy a usar cron, que es un programa que estará en mi Linux 24/7 corriendo, y me permitirá definir cada cuanto quiero lanzar mi script. Voy a explicar como funciona la programación y como editar el crontab. Para ello lo primero que hay que saber que “cron” es el programa que esta en ejecución y que “crontab” es la tabla donde irán la programación por script. Los primero comando para empezar son los siguientes:

```
$ crontab -e
```
Con este comando se edita la tabla de programación.

```
$ crontab -l
```
Con este se listan las programaciones que hay disponibles.

```
$ crontab -r
```
Con este se puede eliminar la tabla de programación

Una vez con esto en mente me dispondré a añadir en mi crontab el script que he creado anteriormente. Para ello usare el siguiente comando:

```
$ crontab -e
```

Una vez dentro, voy a hacer que este script se lance una vez al día a las 2 de la madrugada que es cuando menos trafico tengo.

Para ello voy a enseñar primero como se añaden y como funciona lo de la programación. Una vez dentro del crontab dejara editar un archivo, con lo que voy a hacer el ejemplo de mi script:

```
0 2 * * * /home/yerai/backup.sh
```

Como se ve en el ejemplo las horas funcionan con esos asteriscos, que cada uno significa un valor determinado. Voy a hacer la lista de izquierda a derecha y son 5 valores:

```
MINUTOS | HORAS | DÍA DEL MES | MES | DÍA DE LA SEMANA
```

Ahora se puede ver mas claro, en los minutos podre 0 para que sean a en punto, en la hora pondré 2 para que sea a las 2 de la madrugada, y en los demás pondré * dado que así por defecto le asignara todos. Con lo que con este resultado conseguiré que todos los días a las 2 de la mañana haya podido automatizar mi script y cuando necesite revisar los logs para ver si todo esta en buen estado o ha habido algún fallo y donde.

CONCLUSIÓN

De esta manera he conseguido crear un backup totalmente automatizado y eficaz, a parte de tener unos logs para poder

leerlos y hacer verificación. Lo bueno de este script y usar cron es que es muy escalable, con lo que puedo añadir en cualquier momento todos los backups que necesite sin necesidad de modificar mas cosas que el crontab y las variables del script. Como he mencionado antes para añadir alguna capas de seguridad recomendaría añadir alguna cosa mas, como por ejemplo un usuario dedicado y con los permisos justos para los backups, hacer las copias a través de alguna VPN para que todo funcione por un túnel privado y que no haya ningún intermediario, sin duda alguna un buen Firewall a nivel de red, y en caso de que sean copias muy criticas cifrar las copias con contraseñas robustas para que nadie tenga acceso. Con esto dicho queda este proyecto terminado.