



# UNIVERSIDAD DE GRANADA

## **Centro de Procesamiento de Datos**

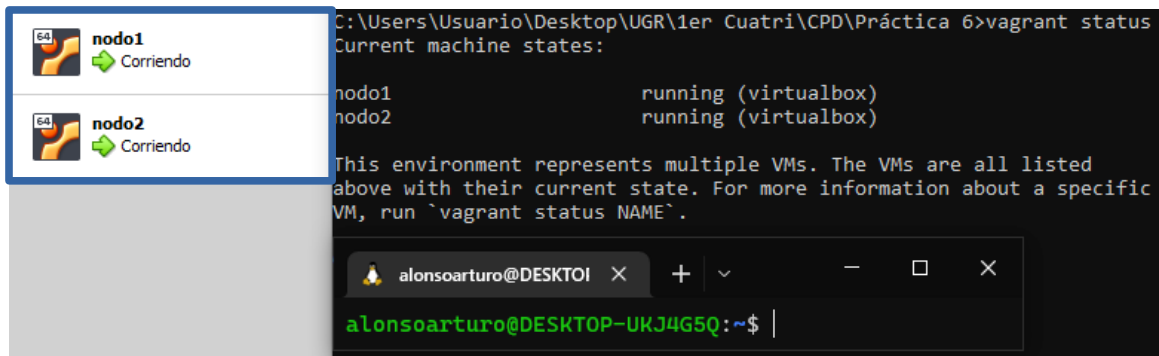
### **Práctica 6 – Copias de Seguridad**

---

Arturo Alonso Carbonero

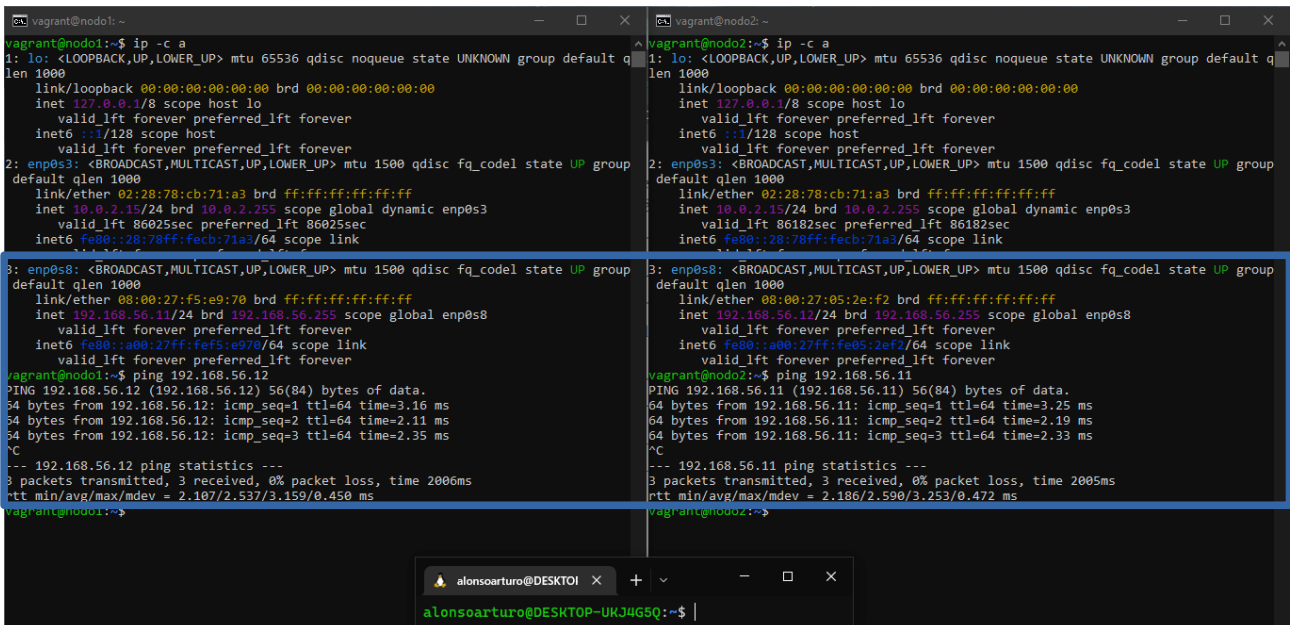
## Desarrollo

En primer lugar, ejecutamos el Vagrantfile de la práctica mediante el comando **vagrant up** y comprobamos que las máquinas se han creado correctamente con **vagrant status**.



*Máquinas creadas correctamente*

Comprobamos, mediante la orden **ping**, por ejemplo, si las máquinas son visibles entre ellas.



*Visibilidad entre las máquinas*

En la máquina cliente, en este caso la máquina *nodo2*, ejecutamos el comando **ssh-keygen -t rsa** para crear el par de claves público-privada para poder realizar conexiones a través de SSH entre las máquinas sin hacer uso de contraseña. La opción **-t** permite seleccionar el algoritmo para la autenticación de las claves, el cual será **rsa**.

```
vagrant@nodo2: ~  
vagrant@nodo2:~$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/vagrant/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/vagrant/.ssh/id_rsa  
Your public key has been saved in /home/vagrant/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:NcpVzJfcJCcFS4vI3EsNfZTa7vjiYtYyNIX4krIdIao vagrant@nodo2  
The key's randomart image is:  
+---[RSA 3072]-----+  
|  
| ooo=B*|  
| o +o==Bo|  
| O =.*|  
| o * + + |  
| . S + O .|  
| . . + + |  
| . + + o o|  
| E . . * + |  
| o =.O.|  
+---[SHA256]-----+
```

Creación de claves

Copiamos la clave pública en la máquina servidor, *nodo1*, haciendo uso del comando **ssh-copy-id**.

```
vagrant@nodo2:~$ ssh-copy-id nodo1  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/vagrant/.ssh/id_rsa.pub"  
The authenticity of host 'nodo1 (192.168.56.11)' can't be established.  
ECDSA key fingerprint is SHA256:V+N7iH1M1CwH2XEF3EPHt9OjYBQ7zXgNHCqVTDVSdA.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
vagrant@nodo1's password:  
Number of key(s) added: 1  
Now try logging into the machine, with: "ssh 'nodo1'"  
and check to make sure that only the key(s) you wanted were added.
```

Copia de la clave pública en el servidor

Para probar que la conexión es adecuada, accedemos a *nodo1*, desde *nodo2*, sin hacer uso de ningún tipo de autenticación. Para ello, ejecutamos **ssh nodo1**.

```
vagrant@nodo2:~$ ssh nodo1
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-164-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu Nov  2 15:51:08 UTC 2023

System load:  0.01               Processes:            108
Usage of /:   3.7% of 38.70GB    Users logged in:     1
Memory usage: 19%               IPv4 address for enp0s3: 10.0.2.15
Swap usage:   0%                IPv4 address for enp0s8: 192.168.56.11

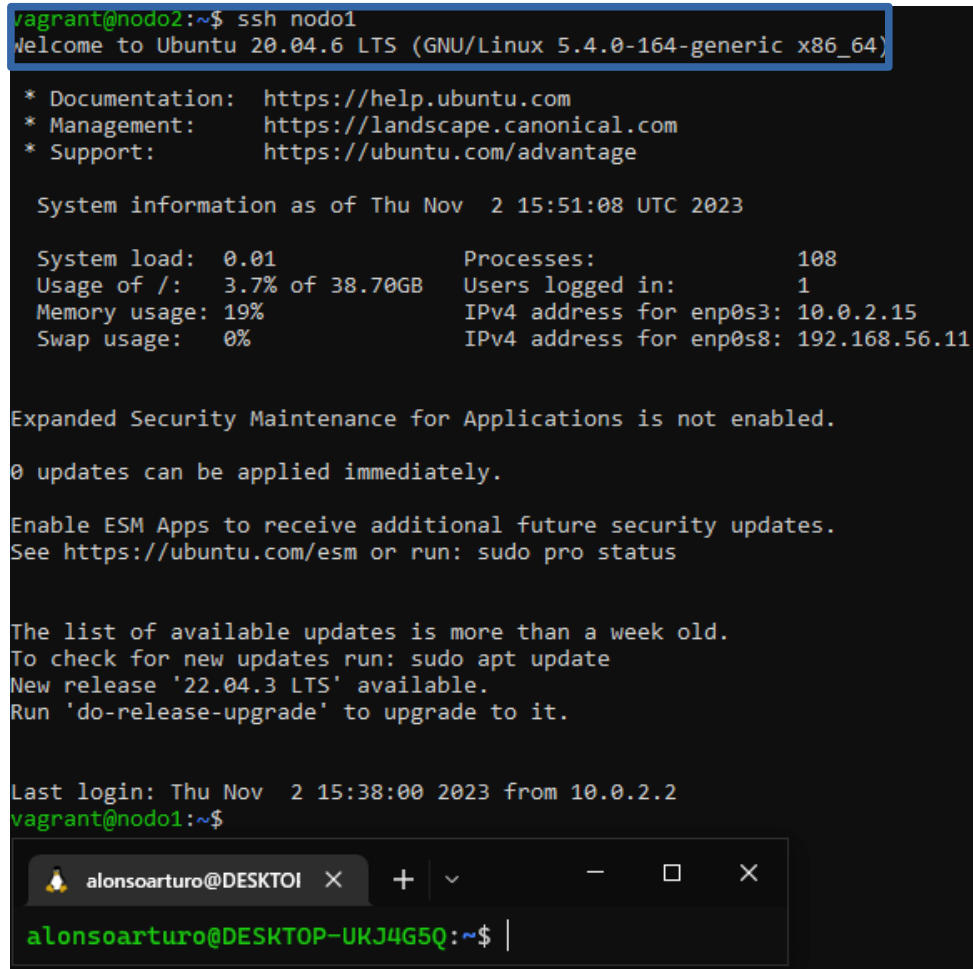
Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Nov  2 15:38:00 2023 from 10.0.2.2
vagrant@nodo1:~$
```



*Conexión SSH realizada*

## Rsync

Para sincronizar un directorio entre ambas máquinas, creamos uno en *nodo2* haciendo uso de **mkdir**. Sincronizamos el directorio mediante el comando **rsync**, indicando el directorio a sincronizar y la ubicación en la máquina destino. Las opciones utilizadas son:

- **-a** → Indica ‘modo archivo’, es decir, copia los datos de forma recursiva, preservando enlaces simbólicos, propietarios, permisos, etc.
- **-v** → Indica ‘verbose’ y muestra información durante el proceso.
- **-z** → Emplea compresión durante la operación.

Para comprobar el proceso, tal y como se muestra en la imagen, realizamos un cambio en el directorio y re-sincronizamos, comprobando en *nodo1*, a la izquierda, que los cambios se han aplicado.

The image shows two terminal windows side-by-side, illustrating the rsync process. The left window is titled 'vagrant@nodo1: ~' and the right window is titled 'vagrant@nodo2: ~'. Both windows show the output of 'ls -lai' commands, displaying file permissions, sizes, owners, and timestamps. The right window also shows the execution of 'mkdir test1', 'cd test1', 'touch prueba.txt', and 'tree' commands. The bottom of the image shows a window titled 'alonsoarturo@DESKTOP-UKJ4G5Q: ~\$' with a command prompt.

```
vagrant@nodo1:~$ ls -lai
total 32
80001 drwxr-xr-x 4 vagrant vagrant 4096 Nov  2 16:05 .
1561 drwxr-xr-x 4 root root 4096 Nov  2 15:31 ..
72148 -rw-r--r-- 1 vagrant vagrant  5 Nov  2 15:51 .bash_history
146 -rw-r--r-- 1 vagrant vagrant 220 Oct 11 21:56 .bash_logout
72648 -rw-r--r-- 1 vagrant vagrant 3771 Oct 11 21:56 .bashrc
256055 drwx----- 2 vagrant vagrant 4096 Nov  2 15:31 .cache
72344 -rw-r--r-- 1 vagrant vagrant 807 Oct 11 21:56 .profile
80004 drwx----- 2 vagrant vagrant 4096 Nov  2 15:32 .ssh
vagrant@nodo1:~$ ls -lai
total 36
80001 drwxr-xr-x 5 vagrant vagrant 4096 Nov  2 16:07 .
1561 drwxr-xr-x 4 root root 4096 Nov  2 15:31 ..
72148 -rw-r--r-- 1 vagrant vagrant  5 Nov  2 15:51 .bash_history
146 -rw-r--r-- 1 vagrant vagrant 220 Oct 11 21:56 .bash_logout
72648 -rw-r--r-- 1 vagrant vagrant 3771 Oct 11 21:56 .bashrc
256055 drwx----- 2 vagrant vagrant 4096 Nov  2 15:31 .cache
72344 -rw-r--r-- 1 vagrant vagrant 807 Oct 11 21:56 .profile
80004 drwx----- 2 vagrant vagrant 4096 Nov  2 15:32 .ssh
256208 drwxrwxr-x 2 vagrant vagrant 4096 Nov  2 16:07 test1
vagrant@nodo1:~$ tree
.
└── test1
    1 directory, 0 files
vagrant@nodo1:~$ tree
.
└── test1
    ├── prueba.txt
    1 directory, 1 file
vagrant@nodo1:~$

vagrant@nodo2:~$ mkdir test1
vagrant@nodo2:~$ ls -lai
total 32
80001 drwxr-xr-x 5 vagrant vagrant 4096 Nov  2 16:07 .
1561 drwxr-xr-x 4 root root 4096 Nov  2 15:34 ..
146 -rw-r--r-- 1 vagrant vagrant 220 Oct 11 21:56 .bash_logout
72648 -rw-r--r-- 1 vagrant vagrant 3771 Oct 11 21:56 .bashrc
256055 drwx----- 2 vagrant vagrant 4096 Nov  2 15:34 .cache
72344 -rw-r--r-- 1 vagrant vagrant 807 Oct 11 21:56 .profile
80004 drwx----- 2 vagrant vagrant 4096 Nov  2 15:50 .ssh
256052 drwxrwxr-x 2 vagrant vagrant 4096 Nov  2 16:07 test1
vagrant@nodo2:~$ rsync -avz test1 vagrant@nodo1:
sending incremental file list
test1/
sent 76 bytes received 20 bytes 38.40 bytes/sec
total size is 0 speedup is 0.00
vagrant@nodo2:~$ cd test1/
vagrant@nodo2:~/test1$ touch prueba.txt
vagrant@nodo2:~/test1$ cd ..
vagrant@nodo2:~$ tree
.
└── test1
    └── prueba.txt
    1 directory, 1 file
vagrant@nodo2:~$ rsync -avz test1 vagrant@nodo1:
sending incremental file list
test1/
test1/prueba.txt
sent 140 bytes received 39 bytes 71.60 bytes/sec
total size is 0 speedup is 0.00
vagrant@nodo2:~$
```

Sincronización del directorio test1

## Servidor Git

En primer lugar, creamos el repositorio local en la máquina *nodo2*, en el directorio **/home/vagrant/devel/test2**, mediante la orden **git init**.

```
vagrant@nodo2:~/devel/test2$ git init
Initialized empty Git repository in /home/vagrant/devel/test2/.git/

alonsoarturo@DESKTOI x + v - □ x
alonsoarturo@DESKTOP-UKJ4G5Q:~$ |
```

*Creación del repositorio local*

De igual forma, creamos el repositorio remoto en el máquina *nodo1*.

```
vagrant@nodo1:~/test2.git$ git init --bare
Initialized empty Git repository in /home/vagrant/test2.git/

alonsoarturo@DESKTOI x + v - □ x
alonsoarturo@DESKTOP-UKJ4G5Q:~$ |
```

*Creación del repositorio remoto*

Sincronizamos ambos repositorios mediante el comando **git remote add origin** y subimos los cambios mediante el comando **git push**. La opción **-u** enlaza las ramas local y remota automáticamente, por lo que solo es necesario la primera vez que se hace un *push*.

```
vagrant@nodo2:~/devel/test2$ git remote add origin vagrant@nodo1:/home/vagrant/test2.git
vagrant@nodo2:~/devel/test2$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 209 bytes | 69.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To nodo1:/home/vagrant/test2.git
* [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

alonsoarturo@DESKTOI x + v - □ x
alonsoarturo@DESKTOP-UKJ4G5Q:~$ |
```

*Sincronización de repositorios*

Realizamos un cambio en el directorio local y subimos los cambios mediante **git add \***. Una vez hemos subido los archivos, hacemos efectivo el cambio en el repositorio remoto mediante la orden **git commit**. La opción **-m** es para añadir un mensaje al *commit*. Mediante la orden **git push**, subimos los archivos definitivamente.

```
vagrant@nodo2:~/devel/test2$ git add *
vagrant@nodo2:~/devel/test2$ git commit -m "prueba"
[master 6326f4c] prueba
Committer: ArturoAcf <vagrant@nodo2.vm>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

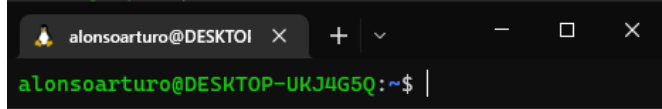
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 prueba.txt
vagrant@nodo2:~/devel/test2$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 236 bytes | 59.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To nodo1:/home/vagrant/test2.git
 423e134..6326f4c master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
alonsoarturo@DESKTOI × + ▾ − □ ×
alonsoarturo@DESKTOP-UKJ4G5Q:~$ |
```

*Resultado final*

## Kopia

Instalamos la clave de GPG para verificar la autenticidad de las versiones. Es necesario crear el directorio `/etc/apt/keyrings` si no existe.

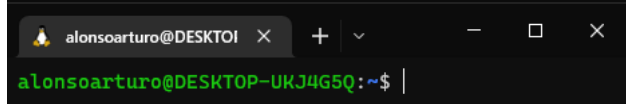
```
vagrant@nodo2:~$ sudo mkdir /etc/apt/keyrings
vagrant@nodo2:~$ curl -s https://kopia.io/signing-key | sudo gpg --dearmor -o /etc/apt/keyrings/kopia-keyring.gpg
vagrant@nodo2:~$ ls /etc/apt/keyrings/
kopia-keyring.gpg
```



*Instalación de la llave de GPG*

Registramos la fuente APT y actualizamos los paquetes.

```
vagrant@nodo2:~$ echo "deb [signed-by=/etc/apt/keyrings/kopia-keyring.gpg] http://packages.kopia.io/apt/ stable main" |
sudo tee /etc/apt/sources.list.d/kopia.list
deb [signed-by=/etc/apt/keyrings/kopia-keyring.gpg] http://packages.kopia.io/apt/ stable main
```

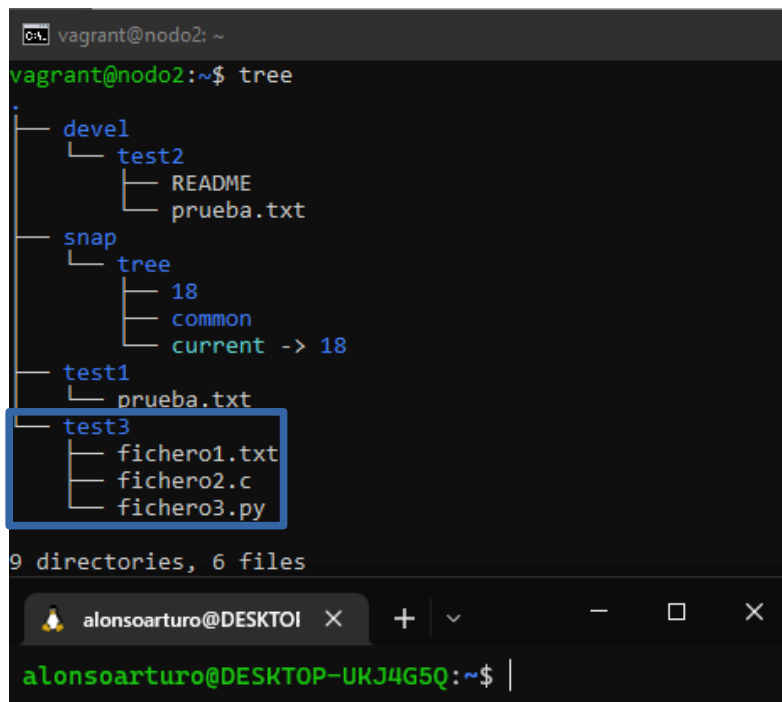


*Registro de fuente*

Para instalar *Kopia* definitivamente, ejecutamos el comando **sudo apt install kopia**. Para la práctica, creamos un directorio, `/test3`, que contenga varios archivos.

```
vagrant@nodo2: ~
vagrant@nodo2:~$ tree
.
├── devel
│   └── test2
│       ├── README
│       └── prueba.txt
├── snap
│   └── tree
│       ├── 18
│       ├── common
│       └── current -> 18
├── test1
│   └── prueba.txt
└── test3
    ├── fichero1.txt
    ├── fichero2.c
    └── fichero3.py

9 directories, 6 files
```



*Directorio de la práctica*



Mediante la orden **kopia repository create**, creamos un repositorio para poder recuperar las copias de seguridad que hagamos.

```
vagrant@nodo2:~$ kopia repository create filesystem --path /tmp/repoCPD_P6
Enter password to create new repository:
Re-enter password for verification:
Initializing repository with:
  block hash:      BLAKE2B-256-128
  encryption:      AES256-GCM-HMAC-SHA256
  splitter:        DYNAMIC-4M-BUZZHASH
Connected to repository.

NOTICE: Kopia will check for updates on GitHub every 7 days, starting 24 hours after first use.
To disable this behavior, set environment variable KOPIA_CHECK_FOR_UPDATES=false
Alternatively you can remove the file "/home/vagrant/.config/kopia/repository.config.update-info.json".

Retention:
  Annual snapshots:      3      (defined for this target)
  Monthly snapshots:     24     (defined for this target)
  Weekly snapshots:      4      (defined for this target)
  Daily snapshots:       7      (defined for this target)
  Hourly snapshots:     48      (defined for this target)
  Latest snapshots:     10      (defined for this target)
  Ignore identical snapshots: false (defined for this target)
Compression disabled.

To find more information about default policy run 'kopia policy get'.
To change the policy use 'kopia policy set' command.

NOTE: Kopia will perform quick maintenance of the repository automatically every 1h0m0s
and full maintenance every 24h0m0s when running as vagrant@nodo2.

See https://kopia.io/docs/advanced/maintenance/ for more information.

NOTE: To validate that your provider is compatible with Kopia, please run:

$ kopia repository validate-provider
```

### Creación del repositorio

A continuación, nos conectamos al repositorio mediante la orden **kopia repository connect**, usando la contraseña que habremos introducido en el paso anterior.

```
vagrant@nodo2: ~
vagrant@nodo2:~$ kopia repository connect filesystem --path /tmp/repoCPD_P6
Enter password to open repository:
Connected to repository.

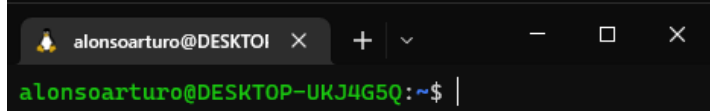
NOTICE: Kopia will check for updates on GitHub every 7 days, starting 24 hours after first use.
To disable this behavior, set environment variable KOPIA_CHECK_FOR_UPDATES=false
Alternatively you can remove the file "/home/vagrant/.config/kopia/repository.config.update-info.json".
```

### Conexión al repositorio

Para poder usar el *nodo1* como servidor de copia, este repositorio debe crearse en dicha máquina. En este caso, esto se ha realizado en una sola máquina, ya que no ha sido posible realizar la conexión al repositorio remoto de *Kopia* en el *nodo1*, pero el procedimiento es el mismo.

Posteriormente, creamos la copia de seguridad del directorio mediante el comando **kopia snapshot create**.

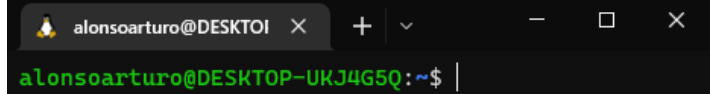
```
vagrant@nodo2:~$ kopia snapshot create test3
Snapshotting vagrant@nodo2:/home/vagrant/test3 ...
* 0 hashing, 3 hashed (0 B), 0 cached (0 B), uploaded 192 B, estimating...
Created snapshot with root ke4bb852b4303be0b623a41eff604f374 and ID f21e5085cf1a0255922713b2dd2b7abb in 0s
Running full maintenance...
Looking for active contents...
Looking for unreferenced contents...
GC found 0 unused contents (0 B)
GC found 0 unused contents that are too recent to delete (0 B)
GC found 2 in-use contents (315 B)
GC found 2 in-use system-contents (1.1 KB)
Rewriting contents from short packs...
Total bytes rewritten 0 B
Not enough time has passed since previous successful Snapshot GC. Will try again next time.
Skipping blob deletion because not enough time has passed yet (59m59s left).
Cleaned up 0 logs.
Cleaning up old index blobs which have already been compacted...
Finished full maintenance.
```



*Creación de la copia de seguridad*

Una vez hemos creado la copia, podemos volver a realizar una copia con el mismo comando del paso previo. Las copias en *kopia* son siempre incrementales. Si no se ha realizado ningún cambio, el proceso es instantáneo y la copia es la misma, por lo que se mantiene el identificador.

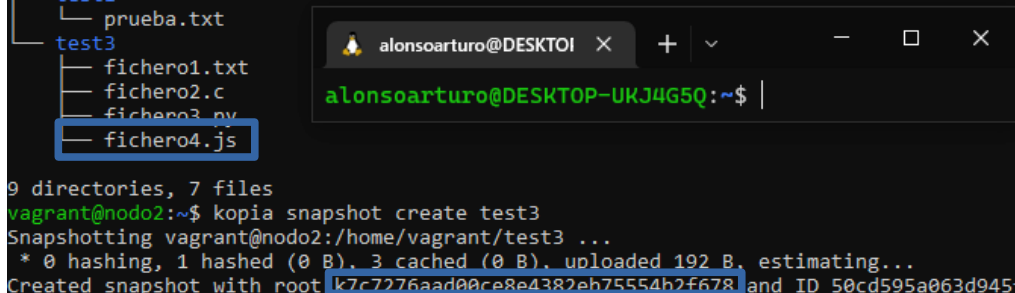
```
vagrant@nodo2:~$ kopia snapshot create test3
Snapshotting vagrant@nodo2:/home/vagrant/test3 ...
* 0 hashing, 0 hashed (0 B), 3 cached (0 B), uploaded 0 B, estimating...
Created snapshot with root ke4bb852b4303be0b623a41eff604f374 and ID 94155f44b2391f71fc68d35729029f98 in 0s
```



*Misma copia*

*Kopia* lee el directorio y crea una nueva *snapshot* si se ha realizado algún cambio.

```
vagrant@nodo2:~$ tree
.
├── devel
│   └── test2
│       ├── README
│       └── prueba.txt
├── snap
│   └── tree
│       ├── 18
│       ├── common
│       └── current -> 18
├── test1
│   └── prueba.txt
└── test3
    ├── fichero1.txt
    ├── fichero2.c
    ├── fichero3.py
    └── fichero4.js
```



*Nueva copia*

Podemos listar las copias de seguridad creadas mediante el comando **kopia snapshot list**.

```
vagrant@nodo2:~$ kopia snapshot list
vagrant@nodo2:/home/vagrant/test3
 2023-11-06 15:09:09 UTC ke4bb852b4303be0b623a41eff604f374 0 B drwxrwxr-x files:3 dirs:1 (latest-2..3)
+ 1 identical snapshots until 2023-11-06 15:12:12 UTC
 2023-11-06 15:13:30 UTC k7c7276aad00ce8e4382eb75554b2f678 0 B drwxrwxr-x files:4 dirs:1 (latest-1, hourly-1, daily-1, weekly-1, monthly-1, annual-1)
```

*Copias creadas*

Mediante el comando **kopia diff**, al cual le pasamos los identificadores de dos copias, podemos ver la diferencia entre ambas.

```
vagrant@nodo2:~$ kopia diff ke4bb852b4303be0b623a41eff604f374 k7c7276aad00ce8e4382eb75554b2f678
modification times differ: 2023-11-06 09:28:49.22704583 +0000 UTC 2023-11-06 15:13:15.57901158 +0000 UTC
added file ./fichero4.js (0 bytes)
```

*Diferencia entre copias*

Con **kopia ls**, y el ID de una copia, podemos ver su contenido.

```
vagrant@nodo2:~$ kopia ls ke4bb852b4303be0b623a41eff604f374
fichero1.txt
fichero2.c
fichero3.py
vagrant@nodo2:~$ kopia ls k7c7276aad00ce8e4382eb75554b2f678
fichero1.txt
fichero2.c
fichero3.py
fichero4.js
```

*Contenido de las copias*

Los directorios se almacenan en objetos JSON. Podemos ver la información en dicho formato mediante el comando **kopia content show -j** y el ID de la copia.

```
vagrant@nodo2: ~  
vagrant@nodo2:~$ kopia content show -j k7c7276aad00ce8e4382eb75554b2f678  
{  
  "stream": "kopia:directory",  
  "entries": [  
    {  
      "name": "fichero1.txt",  
      "type": "f",  
      "mode": "0664",  
      "mtime": "2023-11-06T09:28:40.379045746Z",  
      "uid": 1000,  
      "gid": 1000,  
      "obj": "c05683f4aa016746feff833dff0e2040"  
    },  
    {  
      "name": "fichero2.c",  
      "type": "f",  
      "mode": "0664",  
      "mtime": "2023-11-06T09:28:45.655045796Z",  
      "uid": 1000,  
      "gid": 1000,  
      "obj": "c05683f4aa016746feff833dff0e2040"  
    },  
    {  
      "name": "fichero3.py",  
      "type": "f",  
      "mode": "0664",  
      "mtime": "2023-11-06T09:28:49.22704583Z",  
      "uid": 1000,  
      "gid": 1000,  
      "obj": "c05683f4aa016746feff833dff0e2040"  
    },  
    {  
      "name": "fichero4.js",  
      "type": "f",  
      "mode": "0664",  
      "mtime": "2023-11-06T15:13:15.57901158Z",  
      "uid": 1000,  
      "gid": 1000,  
      "obj": "c05683f4aa016746feff833dff0e2040"  
    }  
  ]  
}
```

*Información en JSON*

Para restaurar una copia, ejecutamos **kopia snapshot restore** indicando el ID de la copia y el directorio de destino.

```
vagrant@nodo2:~$ kopia snapshot restore k7c7276aad00ce8e4382eb75554b2f678 neoCopia  
Restoring to local filesystem (/home/vagrant/neoCopia) with parallelism=8...  
Processed 5 (0 B) of 4 (0 B).  
Restored 4 files, 1 directories and 0 symbolic links (0 B).  
  
vagrant@nodo2:~$ ls  
devel neoCopia snap test1 test3  
vagrant@nodo2:~$ ls neoCopia/  
fichero1.txt fichero2.c fichero3.py fichero4.js
```

*Restauración de la copia*

## Minio

Tras ejecutar todos los comandos indicados en el enunciado de la práctica, me he topado con el error de la siguiente imagen a la hora de crear el contenedor de *Minio*.

```
vagrant@nodo1:~$ docker run -p 9000:9000 --name minio1 -v /home/vagrant/minio_data:/data -e "MINIO_ACCESS_KEY=CpDrand  
om20" -e "MINIO_SECRET_KEY=scretkcPd20CY" minio/minio server /data &  
[1] 4823  
vagrant@nodo1:~$ Fatal glibc error: CPU does not support x86-64-v2
```

*Error encontrado*

Desgraciadamente, no ha sido posible resolver dicho error, por lo que no se especificará nada más en este apartado de la memoria.

## Referencias

### Desarrollo

- <https://www.ssh.com/academy/ssh/keygen>

### Rsync

- <https://linux.die.net/man/1/rsync>
- <https://github.com/ArturoAcf/Servidores-Web-de-Altas-Prestaciones/blob/main/AlonsoCarboneroArturoP2.pdf>

### Git

- <https://practicalgit.com/blog/do-you-always-need-u-in-push.html>

### Kopia

- <https://kopia.io/docs/installation/>
- <https://kopia.io/docs/getting-started/>
- <https://kopia.io/docs/reference/command-line/common/>

### Minio

- <https://min.io/>
- <https://community.veeam.com/kasten-k10-support-92/fatal-glibc-error-cpu-does-not-support-x86-64-v2-4936>
- <https://github.com/keycloak/keycloak/issues/17290>
- <https://access.redhat.com/solutions/6833751>
- <https://forums.centos.org/viewtopic.php?t=78733>