Annexos

Migració serveis de tercers d’una organització a servidors propis

Table of Contents

Índex de Figures

Preparació servidor

Com he explicat a la introducció, per aquest servidor farem servir una Nintendo Switch. Nintendo no ho posa fàcil la instal·lació d'altres sistemes operatius, però gràcies a la comunitat de Switchroot, tenim varies versions de sistemes operatius (Android, Ubuntu, Fedora...) a punt per instal·lar.

He decidit instal·lar Fedora 37. No entraré en detalls sobre tot el procés d'instal·lació, he seguit els següents tutorials:

- https://wiki.switchroot.org/wiki/linux/linux-install-update-guide
- https://nh-server.github.io/switch-guide/user_guide/emummc/sending_payload/#mac-linux
- https://nh-server.github.io/switch-guide/user_guide/emummc/entering_rcm/
- Descarrega de firmware i altres components necessaris: https://www.sdsetup.com/console?switch
- https://www.youtube.com/watch?v=2VLIGQujFGk

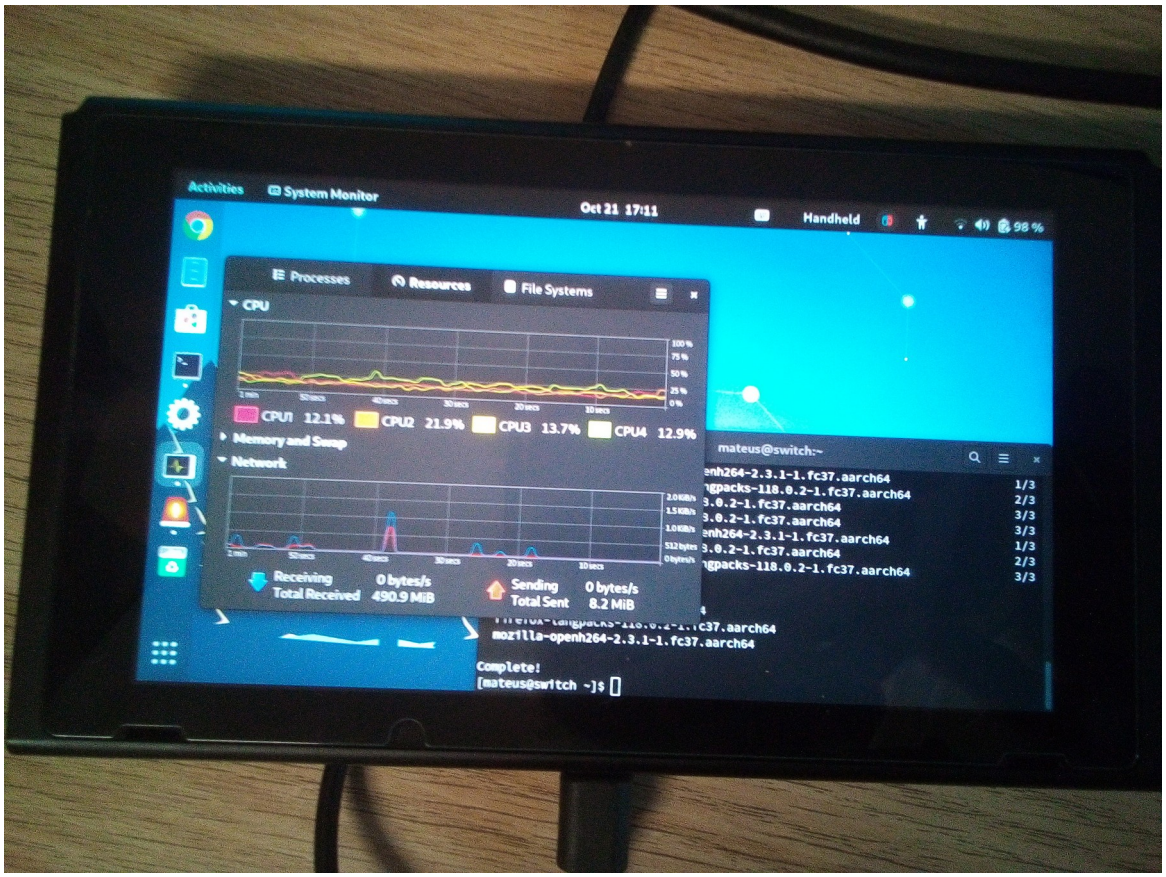


Figura 1: Monitor del sistema i terminal Switch

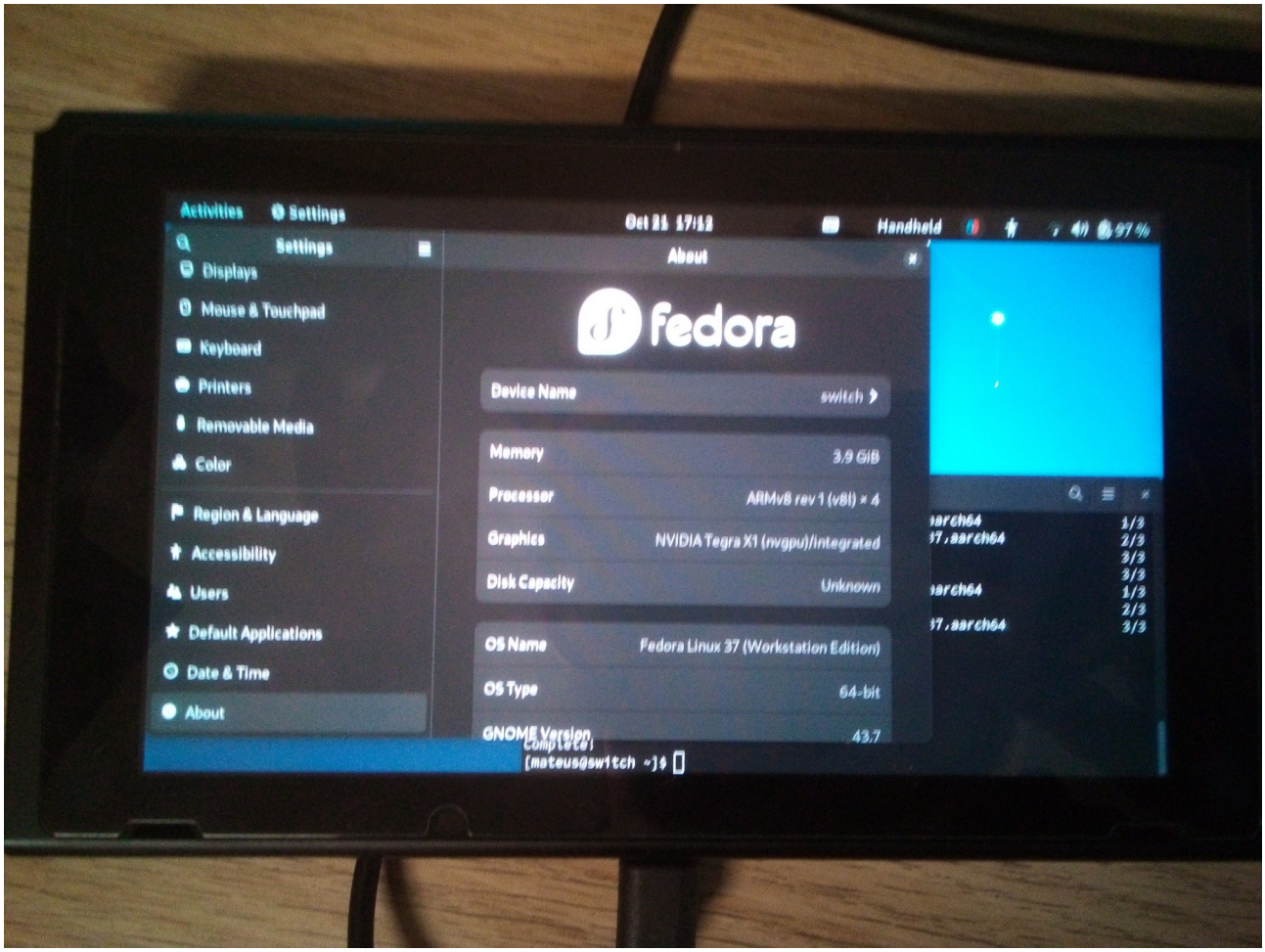


Figura 2: Informació sistema Switch

Com podem veure a la figura 2, la Nintendo Switch no té unes especificacions gaire elevades: només té 4 GB de RAM i una CPU de 4 nuclis, amb un percentatge d'us d'un 12% aproximadament només executant el que es veu en pantalla. Tot i així, és més que suficient per una petita organització.

SSH

Per connectar-nos al servidor remotament, podem fer servir ssh:

```
mateus@pop-os ~> ssh mateus@switchroot
The authenticity of host 'switchroot (192.168.10.168)' can't be established.
ED25519 key fingerprint is
SHA256:xc2xFRvNO2Y1wDKASZo4qN/DJ9iywp/tona8q0KUN6Q.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'switchroot' (ED25519) to the list of known hosts.
mateus@switchroot's password:
Last login: Sun Oct 22 12:50:47 2023
[mateus@switchroot ~]$
```

L'usuari del servidor és 'mateus', en aquest cas, el mateix que l'usuari del client. El nom del servidor és switchroot, el nom que s'ha ficat per defecte. Li diem que volem afegir el servidor a la llista d'ordinadors coneguts i ja podem accedir-hi.

SSH fent servir autentificació amb claus

Per defecte, ssh funciona amb contrasenyes, cada cop que ens connectem hem d'escriure la contrasenya de l'usuari remot. Una altre manera de fer-ho és a través de claus, un cop configurat és més comode, ja que no hem d'escriure la contrasenya i també serà util per fer script de backup amb scp, ja que voldrem que l'script s'executi sense cap interrupció per demanar input.

Primer creem la carpeta on ssh guardara les claus i canviem els permisos perquè només hi pugui accedir l'usuari en qüestió, root. En el meu cas, sembla que s'ha generat automàticament:

```
[root@switchroot ~]# ls -al | grep .ssh
drwx------  2 root   root       4096 Dec 20 07:50 .ssh
```

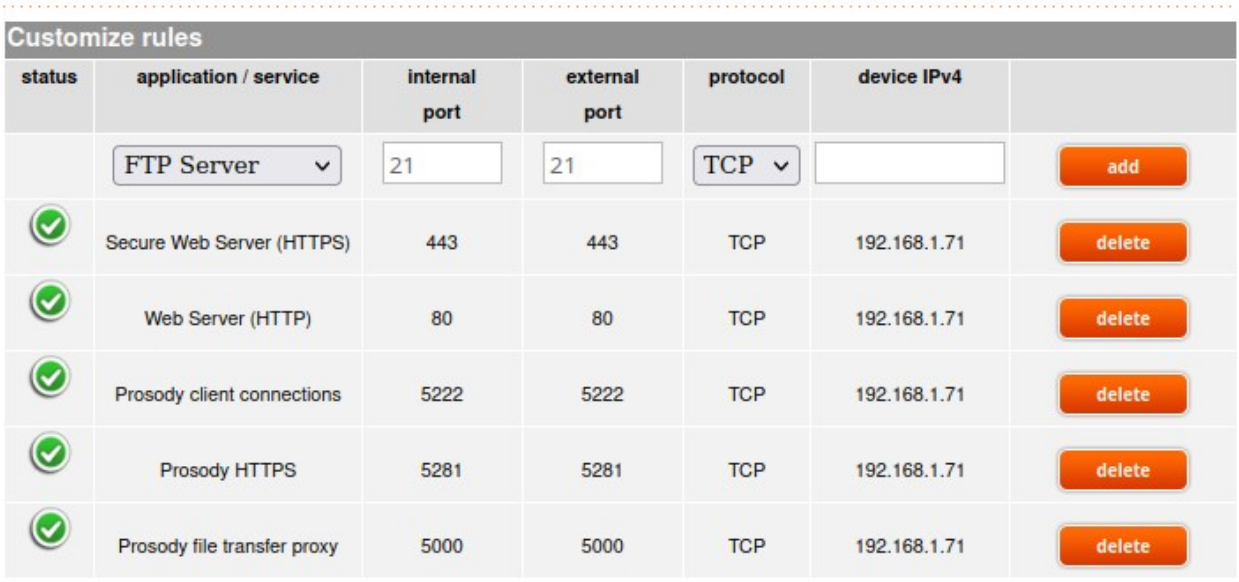
El següent pas és generar un parell de claus, en el meu cas ja les tinc generades. Ara hem de passar la clau pública a l'altre usuari. A continuació passo la clau pública a l'usuari mateus del meu ordinador personal:

```
[root@switchroot .ssh]# scp ~/.ssh/id_rsa.pub mateus@192.168.1.121:~/.ssh/authorized_keys
mateus@192.168.1.121's password:
id_rsa.pub                                    100%  569     3.4KB/s   00:00
```

I això és tot, ara podem accedir a l'usuari mateus des del servidor sense haver d'introduir la contrasenya.

Redirigir paquets enviats al router

El servidor està en una LAN i el router descarta tots els paquets de connexions que no haguem iniciat des de l'interior. Per poder connectar-nos des de l'exterior, podem fer servir el que s'anomena NAT (Network Address Translation): podem configurar el router perquè sempre redirigeixi els paquets de l'exterior amb un cert port a un ordinador dintre de la LAN:



Customize rules

status	application / service	internal port	external port	protocol	device IPv4	
	FTP Server	21	21	TCP		add
✓	Secure Web Server (HTTPS)	443	443	TCP	192.168.1.71	delete
✓	Web Server (HTTP)	80	80	TCP	192.168.1.71	delete
✓	Prosody client connections	5222	5222	TCP	192.168.1.71	delete
✓	Prosody HTTPS	5281	5281	TCP	192.168.1.71	delete
✓	Prosody file transfer proxy	5000	5000	TCP	192.168.1.71	delete

Figura 3: Configuració NAT del router

192.168.1.71 és l'adreça local del servidor.

També tenim l'opció de marcar el servidor com a DMZ (Demilitarized Zone), però redirigiria tots els paquets iniciat per l'exterior al nostre servidor. Fent servir NAT ens assegurem que només són redirigits els paquets que ens interessen, és a dir, fem servir el router de Firewall.

Com menys ports estiguin exposats a internet millor, ja que reduïm la possible superfície d'atac.

Desactivar interfície gràfics

Per defecte, el servidor inicia la interfície gràfica, i això fa que el servidor consumeixi cpu, ram, gpu innecessàriament, ja que no utilitzarem la interfície gràfica.

Fedora, com la majoria de distribucions avui en dia, utilitza systemd per iniciar el sistema. Systemd disposa de 4 opcions ('targets') que ens permeten definir fins a quin punt volem iniciar el sistema:

emergency.target	Només inicia els serveis necessaris per intentar arreglar el sistema.
rescue.target	Inicia tots els programes necessaris perquè el sistema sigui operatiu.
multi-user.target	Inicia tots els programes menys els necessaris per la interfície gràfica.
graphical.target	Inicia tots els 'targets' anteriors a més de tot el necessari perquè la interfície gràfica funcioni.

Cada 'target' inicia totes les seves 'targets' anteriors. Per defecte, el 'target' que tenim és graphical.target, ja que inicia Gnome i per això la interfície gràfica ha de funcionar. Per estar-ne segurs, podem comprovar-ho amb la següent comanda:

```
[mateus@switchroot ~]$ systemctl list-units --type target | egrep "eme|res|gra|mul" | head -1
  graphical.target       loaded active active Graphical Interface
```

Per canviar de ‘target’, podem fer servir la comanda `systemctl set-default`. Canviarem el ‘target’ a multi-user.target, de manera que iniciaran tots els serveis que hem configurat, però sense la interfície gràfica que consumeix recursos extra.

```
[mateus@switchroot ~]$ sudo systemctl set-default multi-user.target
[sudo] password for mateus:
Removed "/etc/systemd/system/default.target".
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/multi-user.target.
```

Script Systemd Trilium

Creem un script anomenat trilium.service a /etc/systemd/system. Especifiquem una descripció, li diem que necessita el servei de docker per executar-se i que s’ha d’executar després del servei de docker.

Especifiquem ‘Restart’ com a ‘always’ perquè si el servei deixa d’executar-se per alguna raó, es torni a executar automàticament. A ExecStart fiquem la comanda que iniciara el servei. Com que abans hem executat docker run, ja tenim el contenidor creat i el podem posar en marxa amb la comanda docker start i el nom que li hem donat al contenidor. ExecStop s’executara quan volquem aturar el servei i executara docker stop. L’opció -t 2 serveix per forçar l’aturada si després de 2 segons de rebre l’ordre d’aturada encara s'està executant.

Finalment, especifiquem WantedBy com a multi-user.target perquè s’executi quan el sistema arribi a aquest target:

```
[root@switchroot ~]# vi /etc/systemd/system/trilium.service
[Unit]
Description=Trilium container
Requires=docker.service
After=docker.service

[Service]
Restart=always
ExecStart=/usr/bin/docker start -a trilium
ExecStop=/usr/bin/docker stop -t 2 trilium

[Install]
WantedBy=multi-user.target
```

Iniciem i activem el servei:

```
[root@switchroot ~]# systemctl enable trilium
Created symlink /etc/systemd/system/multi-user.target.wants/trilium.service →
/etc/systemd/system/trilium.service.
[root@switchroot ~]# systemctl start trilium
```

Script Systemd Homepage

Primer de tot, movem el directori homepage a /usr/local/bin, juntament amb l'executable de Forgejo:

```
[root@switchroot ~]# mv homepage/ /usr/local/bin/homepage
```

A continuació, cream l'script al directori /etc/systemd/system. Aquest és el directori on systemd té tots els scripts de tots els serveis que pot iniciar.

```
[root@switchroot bin]# vi /etc/systemd/system/homepage.service
[Unit]
Description=Homepage
After=network.target

[Service]
Type=simple
User=mateus
ExecStart=/usr/bin/npm start --prefix /usr/local/bin/homepage
Restart=always

[Install]
WantedBy=multi-user.target
```

A la descripció simplement posem 'Homepage'. La propietat 'After' té el valor 'network.target', això vol dir que homepage no s'executara fins que no s'hagin iniciat totes les coses necesaries perquè ens puguem comunicar amb altres ordinadors. Això és necessari perquè homepage és un servidor, no té sentit iniciar-lo si encara no ens podem comunicar amb altres ordinadors.

A la categoria 'Service', especifiquem el tipus com a 'simple' i l'usuari com a 'mateus'. He fet servir el meu usuari perquè homepage no necessita privilegis root i sempre que sigui possible hem d'evitar executar programes com a root, ja que si algun atacant hi accedís tindria tots els privilegis. Amb 'ExecStart' indiquem la comanda que executara: `npm start` i especifico la propietat prefix perquè npm sàpiga on trobar el directori de Homepage. També especifiquem 'Restart' com a 'always', de manera que si Homepage falla per alguna raó, systemd tornara a iniciar el programa.

Finalment li indiquem que el servei s'inicii quan systemd inicii multi-user.target. multi-user.target és un conjunt de serveis que systemd executara quan s'inicii.

Activem i executem Homepage:

```
[root@switchroot bin]# systemctl enable homepage
Created symlink /etc/systemd/system/multi-user.target.wants/homepage.service →
/etc/systemd/system/homepage.service.
[root@switchroot bin]# systemctl start homepage
```

Copies de seguretat

Guardaré copies de seguretat al meu ordinador personal. En comptes de guardar-lo al meu usuari, he creat un usuari nou amb l'única finalitat de guardar les copies de seguretat del servidor:

```
[150ms][~]$ sudo useradd -m switchroot-backup
[89ms][~]$ sudo passwd switchroot-backup
New password:
Retype new password:
passwd: password updated successfully
```

És important poder establir una connexió ssh amb aquest usuari sense introduir la seva contrasenya, ja que les copies de seguretat s'haurien d'efectuar sense la nostra intervenció. Per això, configurarem l'autentificació amb claus com hem vist a l'apartat SSH fent servir autentificació amb claus.

Pàgina web

No hem de fer cap copia de seguretat de la pàgina web perquè ja hi ha una copia guardada a Forgejo.

Nextcloud

L'script de nextcloud és el més complex.

```
#!/bin/sh

sudo -u apache php /var/www/html/nextcloud/occ maintenance:mode --on

scp -Cr /var/www/html/nextcloud/{data,config,themes} switchroot-backup@pop-os:~/nextcloud/

mysqldump --single-transaction --default-character-set=utf8mb4 -u nextcloud_user -ppassword
nextcloud > nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
scp -C nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
switchroot-backup@pop-os:~/nextcloud/nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
rm nextcloud-sqlbkp_`date +"%Y%m%d"`.bak

sudo -u apache php /var/www/html/nextcloud/occ maintenance:mode --off
```

1. Primer de tot activem el mode de manteniment que evita que iniciïn sessió nous usuaris i bloqueja els usuaris que tenen una sessió oberta. Això és per evitar inconsistències en les dades.

2. A continuació, amb l'ajuda de scp, copiem a l'altre equip els directoris data, config, i themes situats al directori /var/www/html/nextcloud/. L'opció 'C' fa que comprimeixi els fitxers abans d'enviar-los, util quan hem d'enviar fitxers d'una mida considerable. L'opció 'r' serveix perquè copii tots els subdirectoris a més a més dels fitxers.
3. Fem un 'dump' (exportem) de la taula nextcloud que tenim a la base de dades MariaSQL. Enviem el fitxer resultant a l'altre equip i l'eliminem del servidor perquè no ocupi espai innecesariament.
4. Desactivem el mode de manteniment perquè tot torni a la normalitat.

Forgejo

Forgejo disposa de una utilitat 'dump' que ens crea una copia de seguretat en un fitxer zip. La comanda no permet l'execució fent servir root, així que faig servir l'usuari git, el mateix usuari que executa Forgejo. Guardo la copia de seguretat a /tmp i un cop enviada a l'altre sistema amb scp, elimino la copia del servidor per no ocupar emmagatzematge innecessàriament.

```
#!/bin/sh

systemctl stop forgejo

sudo -u git forgejo dump --config /etc/forgejo/app.ini -f /tmp/forgejo-dump.zip
scp /tmp/forgejo-dump.zip switchroot-backup@pop-os:~/forgejo/forgejo-dump.zip
rm /tmp/forgejo-dump.zip

systemctl start forgejo
```

Abans d'efectuar la copia aturo Forgejo i, un cop feta, l'activo de nou.

Prosody

Per fer una copia de seguretat de Prosody, hem de copiar el directori /etc/prosody el qual conté les configuracions, i el directori /var/lib/prosody el qual conté les dades:

```
#!/bin/sh

scp -r /etc/prosody/* switchroot-backup@pop-os:~/prosody/conf/
scp -r /var/lib/prosody/* switchroot-backup@pop-os:~/prosody/data/
```

L'opció -r serveix perquè a més de copiar els fitxers, copii els directoris recursivament també, sense aquesta opció només copiaria els fitxers i per tant la copia de seguretat no estaria completa.

Trilium

Per defecte, Trilium fa una copia de seguretat una vegada cada dia, setmana i mes i les guarda a la carpeta trilium-data/backups. L'únic que hem de fer és copiar aquests fitxers a l'altre equip fent servir scp. L'script és el següent:

```
#!/bin/sh

scp -C /root/trilium-data/backup/* switchroot-backup@pop-os:~/trilium/
```

Homepage

En aquest cas l'script és molt simple, només hem de copiar els fitxers de configuració que estan a /usr/local/bin/homepage/config/ a l'altre equip:

```
#!/bin/sh

scp /usr/local/bin/homepage/config/* switchroot-backup@pop-os:~/homepage/
```

Script mestre

Tenim 5 scripts en total. Per no haver d'executar-los un per un, podem crear un altre script amb l'única funció d'executar els altres scripts:

```
#!/bin/sh

./nextcloud_backup_to_other_host.sh
./forgejo_backup_to_other_host.sh
./prosody_backup_to_other_host.sh
./trilium_backup_to_other_host.sh
./homepage_backup_to_other_host.sh
```

Execució automàtica

Podem utilitzar crontab per executar l'script de forma automàtica. Per exemple, perquè l'script s'executi cada dilluns a les 23:00, podem utilitzar:

```
0 23 * * mon /home/root/backup_scripts/backup_to_another_host.sh
```

Com que la copia triga una gran quantitat de temps en fer-se degut a la gran quantitat de dades que hem de transmetre, lo millor és fer-la en una hora en que s'espera que hi hagi el mínim possible d'usuaris connectats.

Script per actualitzar Forgejo

Especificant la versió i l'arquitectura, aquest script que he fet baixa el nou l'executable i substitueix l'executable actual:

```
#!/bin/sh
# Pass the version as the first argument
# Pass the architecture as the second argument, arm64 by default

wget "https://codeberg.org/forgejo/forgejo/releases/download/v$1/forgejo-$1-linux-${2:-arm64}"

wget https://codeberg.org/forgejo/forgejo/releases/download/v$1/forgejo-$1-linux-${2:-arm64}.asc
gpg --verify forgejo-$1-linux-${2:-arm64}.asc forgejo-$1-linux-${2:-arm64}
while true; do
read -p "Proced with installation? (y/n)" yn
case $yn in
        [Yy] ) break;;
        [Nn] ) exit;;
        * ) echo Invalid response;;
esac
done

systemctl stop forgejo

rm forgejo-$1-linux-${2:-arm64}.asc
mv forgejo-$1-linux-${2:-arm64} /usr/local/bin/forgejo
chmod 755 /usr/local/bin/forgejo

systemctl start forgejo
```

Bibliografia

- Systemd targets
- Mirar quina target tenim actualment
- Canviar systemd target
- ssh autentificació amb claus
- Exemples scp
- Com crear usuaris a Linux
- Copia de seguretat de Nextcloud
- Copia de seguretat Forgejo
- Copia de seguretat Prosody
- Exemples cron