Administration générale

Tout ce qui peut s'appliquer à n'importe quel Fedora, Firewall, ssl, VM et container, ...

- Liste des services à proposer
- Backup avec borg
- Firewall
    - Firewalld
    - Nftables
- Postgres
- SSL
- Zones DNS
- Raid 5
- SSL de LAN
- Proftpd
- Libvirt
- Knot DNS (notes personnelles)
- Uwsgi
- SELinux
- PostgreSQL
- Podman
- Remote Wireshark

Liste des services à proposer

Listes des services Fede.re / ppsfleet.navy

Services ouverts

- pad.fede.re
- poll.fede.re
- maps.ppsfleet.navy (mettre cartes.app ? ou garder simplestreetmap ?)

Services avec compte

- Peertube
- RSS
- matrix

Services pour les proches

- Nextcloud
  - fichier
  - calendrier
  - todo list
  - liste de contacts
- Adresse email
- Jellyfin
- Wiki (recette de cuisine + creer son livre ?)

Idées

- systeme de cagnotte (pas sur que ca existe)
- partager ses photos (nextcloud permet ça)
- IA (generer des photos, transcrire du texte, chatbot pour le wiki)

Backup avec borg

Les backups sont gérées avec Borg et Borgmatic et sont exécutées tous les jours à 5:00, et gardés pendant une période de 10 jours.

Les éléments suivants sont inclus dans la sauvegarde :

- /etc
- /srv
- /var
- /data/peertube-data
- /data/mails
- les bases de données MySQL et Postgres

Sont exclus les dossiers de cache.

Commande utiles

Il est préférables d'utiliser le wrapper `borgmatic` pour effectuer les opérations étant donné qu'il est déjà configuré, ainsi :

- on peut utiliser `borgmatic list` pour lister les archives ;
- les archives peuvent être montées en tant que système de fichier via FUSE avec `borgmatic mount --archive ARCHIVE --mount-point PATH`, `ARCHIVE` est le nom d'une archive ou latest pour monter la dernière, le système de fichier peut être unmount avec `borgmatic umount` ;

Bases de données

Le dump des bases est dans `run/root/borgmatic`

- les bases de données peuvent être restaurées avec `borgmatic restore`. **Attention cette action supprime TOUTES les bases de données du serveur pour les remplacer et doit être effectuer par le même utilisateur qui effectue les sauvegardes (root).**
- Pour restaurer uniquement une (ou plusieures) base, on peux utiliser la commande `borgmatic restore --archive latest --database nextcloud [--database autre...]`

A savoir

- Une métric est envoyée à prometheus à chaque étape de la sauvegarde, une alerte est envoyé si il n'y a pas eu de sauvegarde dans les dernières 24h ou si la sauvegarde à échouée.
- Pour exclure un dossier des sauvegardes on peut le rajouter à la liste dans le fichier de configuration (`/etc/borgmatic/config.yaml`) ou créer un fichier .nobackup dans ce dossier.
- Créer un fichier `CACHEDIR.TAG` dans un dossier marque ce dossier comme dossier de cache et l'exclue des sauvegardes.

Firewall

Firewalld

Cheatsheet

- Ouvrir un port :

```
firewall-cmd --permanent --add-port=22/tcp
systemctl reload firewalld
```

- Lister les ports ouverts : `sudo firewall-cmd --list-all`
- Ajouter des rich rules : `firewall-cmd --permanent --add-rich-rule='rule family="ipv6" source address="2001:db8:cafe:bc68::1" port port="9100" protocol="tcp" accept'`
- Forwarder un port :
  - `firewall-cmd --permanent --add-forward-port=port=3724:proto=tcp:toport=:toaddr=10.114.0.12`

Liens utiles

- https://www.rootusers.com/how-to-use-firewalld-rich-rules-and-zones-for-filtering-and-nat/

Firewall

Nftables

Cheatsheet

Example de commandes:

- Lister toutes les règles : `nft list ruleset`
- Lister la table filter de famille inet : `nft list table inet filter`
- Lister les règles en incluant les positions : `nft list table inet filter -a`
- Inserer une règle (ouvire le port 8080 tcp) a une position dans la table filter de famille inet après la règle à la position 5: `nft add rule inet filter input position 5 tcp dport 8080 accept`
- Supprimer la règle à la position 22: `nft delete rule inet filter input handle 22`
- Pour seulement supprimer les tables gérées dans le fichier de config, remplacer `flush ruleset` par `table inet filter; delete table inet filter` au début du ficher. La première instruction s'assure que la table existe avec de la supprimer pour de pas avoir d'erreur lors de la première exécution.

Le fichier de configuration est /etc/sysconfig/nftables.conf, pour recharger les règles après modification il peut être directement exécuté, ce qui est équivalent à nft -f /etc/sysconfig/nftables.conf. Pour ne pas affecter les règles gérer par podman il ne faut pas recharger le service nftables, cela écraserait toute autre règle rendant les containers injoignables.

Liens utiles

- https://wiki.archlinux.org/index.php/nftables
- https://wiki.nftables.org/wiki-nftables/index.php/Quick_reference-nftables_in_10_minutes
- https://wiki.nftables.org/wiki-nftables/index.php/Scripting
- https://wiki.nftables.org/wiki-nftables/index.php/Performing_Network_Address_Translation_(NAT)

Postgres

Changement de version majeure

Sauvegarder la base de données avant la mise à jour pour éviter des incompatibilité avec des modules

1. Dump la base de donnée : `postgres@alshain $ pg_dumpall > backup`
2. Stoper le service `root@alshain # systemctl stop postgresql`
3. Sauvegarde l'ancien répertoire de postgres `root@alshain # mv /var/lib/pgsql{,.old}`
4. Recréer la base de données

```
root@alshain # mkdir /var/lib/pgsql
root@alshain # chown postgres:postgres /var/lib/pgsql
root@alshain # sudo -i postgres
postgres@alshain $ postgres -D /var/lib/pgsql/data
```

5. Importer les données `postgres@alshain $ psql -d postgres -f ../pgsql.old/backup`
6. Relancer le service `roooot@alshain # systemctl start postgresql`

Sinon sans backup préalable de la DB:

1. Mounter une sauvegarde du serveur :

```
root@alshain # borgmatic mount --archive latest --mount-point /mnt
```

2. Récupéré les lib manquantes des modules (et leurs dépendances):

```
root@alshain # cp /mnt/usr/lib64/pgsql/*.so /usr/lib64/pgsql/postgresql-14/lib/
root@alshain # cp /mnt/usr/lib64/libproj.so.* /usr/lib64/
```

3. Lancer l'upgrade de postgres

```
root@alshain # sudo -iu postgres postgresql-setup --upgrade
```

SSL

Créer un certificat pour un site web 101

1. Ajouter à votre fichier conf (de préférence vers l'entete de celui-ci):
   ```
   include /etc/nginx/snippets/letsencrypt-acme-challenge.conf;
   ```
2. Utilisez la commande :
   ```
   sudo certbot --authenticator webroot --installer nginx
   ```
3. Lorsque demandé, spécifiez le webroot, il est le suivant :
   ```
   /srv/www/letsencrypt
   ```
   Si vous choisissez d'être redirigé automatiquement, vérifiez que les adresses ip automatiquement remplies par certbot soient bien indiquées et non sous la forme [::]:443

Révoquer un certificat

```
certbot revoke --cert-path /PATH/TO/fullchain.pem --key-path /PATH/TO/privkey.pem
```

Renouveler un certificat

```
#!/bin/bash

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

certbot renew >> /etc/letsencrypt/monthly.log 2>&1
service nginx restart
```

Avec crontab

Mettre `PATH=$PATH:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin` au début du crontab (crontab -e)

Http-101

/etc/nginx/snippets/letsencrypt-acme-challenge.conf :

```
################################################################################
#
# This config enables to access /.well-known/acme-challenge/xxxxxxxxxx
# on all our sites (HTTP), including all subdomains.
# This is required by ACME Challenge (webroot authentication).
# You can check that this location is working by placing ping.txt here:
# /var/www/letsencrypt/.well-known/acme-challenge/ping.txt
# And pointing your browser to:
# http://xxx.domain.tld/.well-known/acme-challenge/ping.txt
#
# Sources:
# https://community.letsencrypt.org/t/howto-easy-cert-generation-and-renewal-with-nginx/3491
#
################################################################################

# Rule for legitimate ACME Challenge requests (like /.well-known/acme-challenge/xxxxxxxxx)
# We use ^~ here, so that we don't check other regexes (for speed-up). We actually MUST cancel
# other regex checks, because in our other config files have regex rule that denies access to
files with dotted names.
location ^~ /.well-known/acme-challenge/ {

    # Set correct content type. According to this:
    # https://community.letsencrypt.org/t/using-the-webroot-domain-verification-method/1445/29
    # Current specification requires "text/plain" or no content header at all.
    # It seems that "text/plain" is a safe option.
    default_type "text/plain";

    # This directory must be the same as in /etc/letsencrypt/cli.ini
    # as "webroot-path" parameter. Also don't forget to set "authenticator" parameter
    # there to "webroot".
    # Do NOT use alias, use root! Target directory is located here:
    # /var/www/common/letsencrypt/.well-known/acme-challenge/
    root         /var/www/letsencrypt;
}
```

```
# Hide /acme-challenge subdirectory and return 404 on all requests.
# It is somewhat more secure than letting Nginx return 403.
# Ending slash is important!
location = /.well-known/acme-challenge/ {
    return 404;
}
```

Zones DNS

Après altair

Les fichiers sont dans le dossier: `/var/lib/knot/zones` sur Altair.

Le serial n'a pas besoin d'être mis à jours après édition de la zone.

Pour reload `systemctl reload knot` ou `knotc zone-reload <nom de la zone>`

Avant Altair (obsolète):

ppsfleet.navy

La zone externe ppsfleet.navy est signée avec DNSSEC, pour rajouter une entrée dans la zone il faut dabord modifier le fichier `/var/named/ppsfleet.navy.d/ppsfleet.navy.zone` puis signer la zone avec le script `/var/named/sign-zone.sh ppsfleet.navy`.

signe.zone.sh

Le script s'utilise de la manière suivante :
`/var/named/sign-zone.sh [nom de la zone]`

Pour qu'il fonctionne la zone dois être placée dans `/var/named/[nom de la zone].d/[nom de la zone].zone`. Ce script s'occupe de vérifier la validité de la zone, mettre à jour le serial, signer la zone et de recharger le service.

```
#!/bin/bash

set -e

OLD_DIR=$(pwd)

today=$(date +%Y%m%d)
```

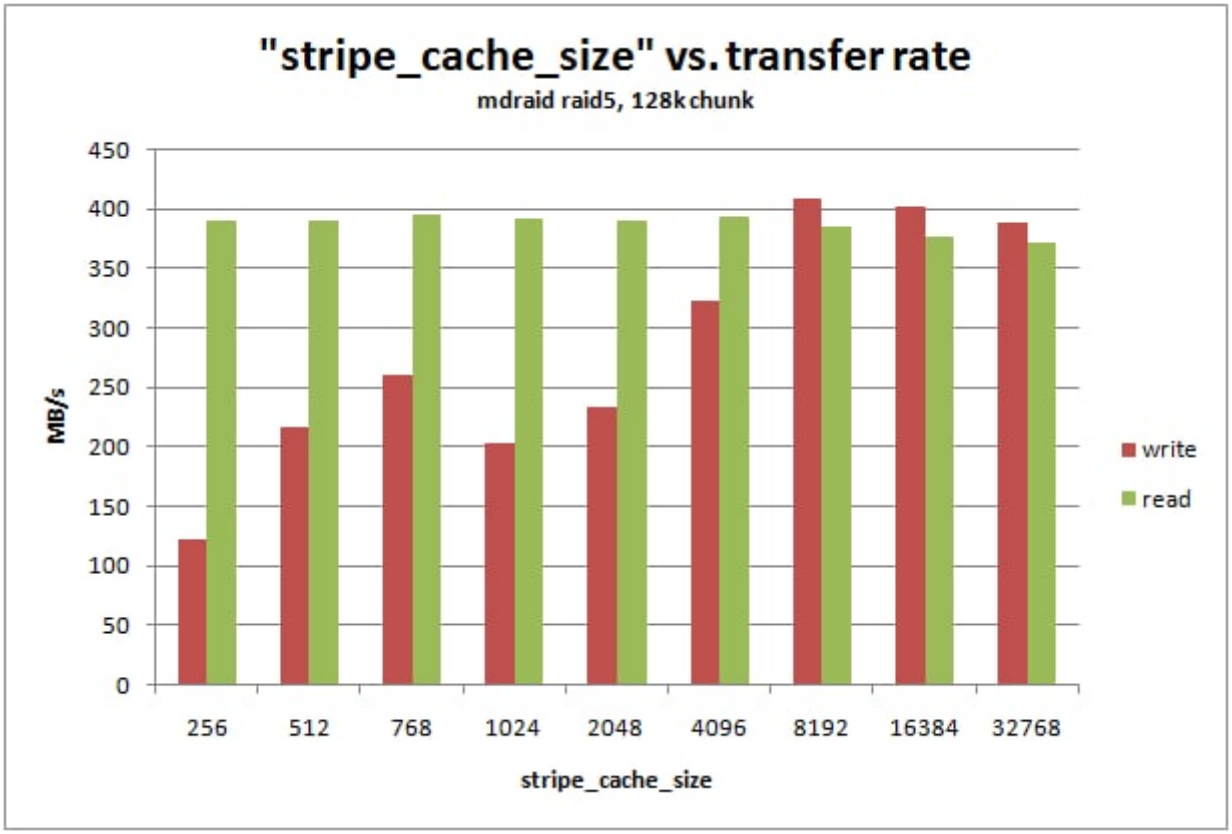
```
cd /var/named/$1.d/

named-checkzone $1 $1.zone
serial=$(named-checkzone $1 $1.zone | grep --color=never serial | cut -d' ' -f5)
serialDate=${serial:0:8}
serialIncr=${serial:8:2}
echo $serial
echo $today
if [ $today == $serialDate ]; then
  newSerial=$today$(printf %02d $(($serialIncr+1)))
else
  newSerial=${today}01
fi

sed -i "0,/$serial/s//$newSerial/" $1.zone

sudo -u named dnssec-signzone -A -3 $(head -c 1000 /dev/urandom | sha1sum | cut -b 1-16) -N
keep -o $1 -t $1.zone
systemctl reload named
cd $OLD_DIR
```

Raid 5

Performance du raid5

On peut améliorer la vitesse d'écriture sur le raid 5 en lui allouant plus de ram:

```
echo 32768 > /sys/block/md126/md/stripe_cache_size
```



source:

- https://lucatnt.com/2013/06/improve-software-raid-speed-on-linux/
- https://mannygonzalez.blogspot.com/2015/02/speeding-up-raid-operations-on-synology.html

SSL de LAN

Si vous voulez seulement mettre à jour les certificats car ils ont expirés, allez à la section UPDATE DES CERTIFS.

/!\ Des fois, le renew ne fonctionne pas, il suffit de le refaire /!

Les certifs sont générés sur ozone.

Pour générer les certificats pour un site dont les DNS sont sur le serveur et gérés avec bind9, et dont on veut avoir le SSL en LAN, il faut effectuer la commande suivante :

```
certbot certonly --manual --preferred-challenges=dns --manual-public-ip-logging-ok --manual-auth-hook /etc/letsencrypt/authenticator.sh --manual-cleanup-hook /etc/letsencrypt/cleanup.sh -d admin.lan.air-eisti.fr
```

- --manual : permet de génerer le certificat grâce à des hook (les scripts shells)
- --preferred-challenges=dns : demande de générer le certificat pour DNS-01 et permet de l'utiliser sur des DNS "locaux". Il faut déployer un record DNS TXT pour le temps de la validation
- --manual-public-ip-logging-ok : accepte automatiquement que l'IP soit loggé, sinon il faut le confirmer à la main
- --manual-auth-hook /etc/letsencrypt/authenticator.sh : execute authenticator.sh avant de générer les certificats, permet de déployer le record DNS TXT
- --manual-cleanup-hook /etc/letsencrypt/cleanup.sh : execute cleanup.sh après la génération des certificats, permet d'enlever le record DNS TXT
- -d admin.lan.air-eisti.fr spécifie le nom de domaine

avec authenticator.sh :

```
#!/bin/bash

DOMAIN=$CERTBOT_DOMAIN
VALIDATION=$CERTBOT_VALIDATION

DNS_PATH="/etc/bind/db.ext.air-eisti.fr"

# On cherche à modifier le sérial, qui est au format AAAAMMJJXX avec XX le numéro de version du jour, par défaut 00
# On incrémente de 1 à chaque modification effectuée le même jour que la précédente
```

```
DATE=$(date +%Y%m%d)
# Variable settant la recherche du commentaire présent sur la ligne du serial
NEEDLE="serial"
curr=$(/bin/grep -e "${NEEDLE}$" $DNS_PATH | /bin/sed -n "s/^\s*\([0-
9]*\)\s*;\s*${NEEDLE}\s*/\1/p")
if [ ${#curr} -lt ${#DATE} ]; then
   serial="${DATE}00"
else
   prefix=${curr::-2}
   if [ "$DATE" -eq "$prefix" ]; then # same day
     num=${curr: -2} # last two digits from serial number
     num=$((10#$num + 1)) # force decimal representation, increment
     serial="${DATE}$(printf '%02d' $num )" # format for 2 digits
   else
     serial="${DATE}00" # just update date
   fi
fi
/bin/sed -i -e "s/^\(\s*\)[0-9]\{0,\}\(\s*;\s*${NEEDLE}\)$/\1${serial}\2/" ${DNS_PATH}

# Déploie le DNS TXT record pour valider la demande de certificat
SOUSDOMAINE=${DOMAIN%*.*.*} # enleve .air-eisti.fr
echo "_acme-challenge.$SOUSDOMAINE IN TXT \"$VALIDATION\"" >> ${DNS_PATH}

# Application des changements
service bind9 restart
```

et cleanup.sh :

```
#!/bin/bash

DOMAIN=$CERTBOT_DOMAIN
VALIDATION=$CERTBOT_VALIDATION

DNS_PATH="/etc/bind/db.ext.air-eisti.fr"

# On cherche à modifier le sérial, qui est au format AAAAMMJJXX avec XX le numéro de version
du jour, par défaut 00
# On incrémente de 1 à chaque modification effectuée le même jour que la précédente
```

```
DATE=$(date +%Y%m%d)
# Variable settant la recherche du commentaire présent sur la ligne du serial
NEEDLE="serial"
curr=$(/bin/grep -e "${NEEDLE}$" $DNS_PATH | /bin/sed -n "s/^\s*\([0-
9]*\)\s*;\s*${NEEDLE}\s*/\1/p")
if [ ${#curr} -lt ${#DATE} ]; then
   serial="${DATE}00"
else
   prefix=${curr::-2}
   if [ "$DATE" -eq "$prefix" ]; then # same day
     num=${curr: -2} # last two digits from serial number
     num=$((10#$num + 1)) # force decimal representation, increment
     serial="${DATE}$(printf '%02d' $num )" # format for 2 digits
   else
     serial="${DATE}00" # just update date
   fi
fi
/bin/sed -i -e "s/^\(\s*\)[0-9]\{0,\}\(\s*;\s*${NEEDLE}\)$/\1${serial}\2/" ${DNS_PATH}

# Déploie le DNS TXT record pour valider la demande de certificat
SOUSDOMAINE=${DOMAIN%*.*.*} # enleve .air-eisti.fr
/bin/sed -i -e "/_acme-challenge.$SOUSDOMAINE IN TXT \"$VALIDATION\"/d" ${DNS_PATH}

# Application des changements
service bind9 restart
```

Quand on ne sera plus sous OPNSense, il faudra probablement ajouter la copie du cert.pem et du privkey.pem dans l'endroit où il faut à cleanup.sh, ils sont localisés dans /etc/letsencrypt/live/ Actuellement il faut copier coller dans le clicodrome d'OPNSense.

/!\ Apparemment, certbot 0.19 renew automatiquement, mais c'est pas sur, certbot renew --dry-run le fait en tout cas /!

De plus, on ne peux pas renew automatiquement les certificats (ils ont une durée de vie de 90 jours), donc il faut réeffectuer la commande pour recréer tous les certificats et re copier-coller dans OPNSense.

On peux faire une tache cron pour refaire la commande en y ajoutant --force-renewal (sinon certbot demande si on veut renew ou ne rien faire) à interval réguliers.

exemple pour un renew tout les mois

```
0 0 1 * * certbot certonly --manual --force-renewal --preferred-challenges=dns --manual-public-ip-logging-ok --manual-auth-hook /etc/letsencrypt/authenticator.sh --manual-cleanup-hook /etc/letsencrypt/cleanup.sh -d admin.lan.air-eisti.fr
```

Update des certifs

Si tout ce passe bien, le renew est automatique, du coup il suffit de faire les étapes suivantes.

Pour les VM du core et du portail captif :

depuis ozone, mettez les clefs dans votre home, accessible par votre user. Les clefs sont dans /etc/letsencrypt/archive/urlDuSite, n'oubliez pas d'enlever les anciennes et les garder celle avec le plus récent numéro, et enlever ce numéro (ex: privkey3.pem -> privkey.pem)

Il faudra le faire pour les 3 domaines suivants (avec ip de la VM pour se co en ssh):

```
portal.lan.air-eisti.fr (machine : portal.lan.air-eisti.fr / 10.82.0.66)
core.lan.air-eisti.fr (machine : main-web.net.air-eisti.fr / 10.82.0.65)
lpmng.lan.air-eisti.fr (machine : main-web.net.air-eisti.fr / 10.82.0.65)
```

en réseau de lan, il faut être dans le vlan admin puis :

```
ssh root@10.82.0.40
(demander mdp à admin)
ssh -i first_key_openstack centos@IPDeLaVM
(ça va calculer pendant un petit peu de temps, vous pouvez à la place récupérer la clef depuis barium puis
ssh -i first_key_openstack centos@portal.lan.air-eisti.fr depuis votre session (en fermant le ssh depuis barium))
```

Puis

```
scp -r votreUser@air-eisti.fr:~/urlDuSite ./urlDuSite
cp -R urlDuSite /etc/key/
```

pour tester si la maj a bien été faite :

```
curl urldusite
```

Proftpd

Installation

Paquets necessaires : `proftpd`, `proftpd-mysql`, `mysql`.

Configuration

- `/etc/proftpd.conf` : fichier de configuration principal, il faudrait éviter de l'éditer directement.
- `/etc/proftpd` : dossier contenant la suite de la configuration séparée en différement fichiers en fonction des fonctionnalités :
  - `/etc/proftpd/modules.conf` : contient la liste des modules proftpd chargés ;
  - `/etc/proftpd/sql.conf` : configuration de la liaison avec mysql pour gérer les utilisateurs virtuels (actuellement l'authentification mysql est la méthode principale et authoritaire);
  - `/etc/proftpd/tls.conf`: configuration du chiffrage des connexion avec TLS.

Gestion des utilisateurs avec MySQL

La base de donnée utilisée pour l'authentification est `timonier`, accessible avec l'utilisateur homonyme, le mot de passe peut être trouvé dans le fichier de configuration de sql pour proftpd (voir ci-dessus).

La structure de la base de donnée est la suivante :

- Table `ftpusers`, le mot de passe du champs `password` doit être hashé avec la méthode native de mysql `PASSWORD()`:

id	username	password	uid	gid	homedir	shell

- Table `ftpgroups`, doit contenir une entrée pour chaque utilisateur de la table `ftpuser` :

groupname	gid	members

Pour plus de renseignements sur les type des champs, le script suivant à été utilisé pour créer la base de donnée :

```
drop table if exists ftpusers;
drop table if exists ftpgroups;

create table `ftpusers` (
        `id` int(10) unsigned not null auto_increment,
        `username` varchar(32) not null,
        `password` varchar(42) not null,
        `uid` smallint(6) not null,
        `gid` smallint(6) not null,
        `homedir` varchar(255) not null,
        `shell` varchar(32) not null,
        primary key(`id`),
        unique key (`username`, `uid`)
) engine=InnoDB default charset=utf8;

create table `ftpgroups` (
        `groupname` varchar(32) not null,
        `gid` smallint(10) not null,
        `members` varchar(255) default ''
) engine=InnoDB default charset=utf8
```

Libvirt

Installation des paquets

`dnf install virt-install libvirt-python libvrt-client libvrt-daemon libvirt-daemon-ddriver-qemu qemu-kvm qemu-img`

Démarrer les services `libvrtd` et `virtlogd`

Création de la machine virtuelle

Exemple pour l'installation de debian 8 depuis les dépots d'images

```
virt-install \
        --name=mail \
        --file=/var/lib/libvirt/images/mail.dsk \
        --file-size=8 \
        --vcpus=2 --ram=2048 \
        --location http://cdn-fastly.deb.debian.org/debian/dists/jessie/main/installer-amd64/
\
        --extra-args "console=ttyS0,115200n8 serial" \
        --os-type linux \
        --os-variant generic \
        --network bridge=virbr0 \
        --graphics none \
        --console pty,target_type=serial
```

Repertoires partagés

Configuration de l'host

1. Dans `/etc/libvirt/qemu.conf` trouver et décommenter les lignes

```
#user = "root"
#group = "root"
```

```
#dynamic_ownership = 1
```

Modier ensuite les valeurs pour

```
user = "qemu" #Ou l'utilisateur avec lequel qemu est censé être lancé
group = "qemu" #Idem
dynamic_ownership = 0 #Requis pour les configurations qui vont suivre
```

2. Redémarrer libvirt `systemctl restart libvirtd`.
3. Modifier ensuite les paramètre de la VM (`virsh edit [domain]`) et rajouter dans le noeud `<devices>`

```
<filesystem type='mount' accessmode='mapped'>
      <source dir='/data/mails'/> <!-- Répertoire a partager (sur l'host) -->
      <target dir='sharedDirMails'/> <!-- label avec lequel le périphérique virtuel va être
accessible -->
</filesystem>
```

4. Créer le répertoire `/data/mails` avec qemu:qemu comme propriétaire.
5. Redémarrer la machine virtuelle `virsh reboot [domain]`.

Configuration du guest

1. Editer le fichier `/etc/modules` pour être sur que les bons modules soient chargés :

```
# cat >>/etc/modules <<EOF
loop
virtio
9p
9pnet
9pnet_virtio
EOF
```

2. Charger les modules `systemctl restart kmod`
3. Créer le répertoire `/opt/share/mails`
4. Le répertoire peut maintenant être monté `mount sharedDirMails /opt/share/mail -t 9p -o trans=virtio,version=9p2000.L`
5. Pour monter le répertoire au démarrage de la VM, editer `/etc/fstab` en ajoutant

```
sharedDirMails  /opt/share/mails        9p      trans=virtio,version=9p2000.L 0 2
```

Commandes utiles

- `virsh list --all` : liste toutes les vm peut importe l'état
- `virsh shutdown [domain]` `virsh reboot [domain]` `virsh start [domain]` : arrete / redémarre ou démarre la vm `[domain]`
- `virsh destroy [domain]` : force l'arret de la machine
- `virsh undefine [domain]` : oublie la vm `[domain]`
- `virsh net-dhcp-leases --network default` : récupère la liste des leases dhcp pour le réseau par défaut

Liens utils

- https://raymii.org/s/articles/virt-install_introduction_and_copy_paste_distro_install_commands.html
- http://rabexc.org/posts/p9-setup-in-libvirt
- http://www.linux-kvm.org/page/9p_virtio

Knot DNS (notes personnelles)

Backup et restore

Utilisation de restauration en ligne pour éviter la suppression des zones listées via une zone catalogue (à comme effet indésirable de supprimer les clés)

```
# Backup
mkdir /var/lib/knot/backups/
chown knot:knot /var/lib/knot/backups/
knotc zone-backup +backupdir /var/lib/knot/backups/
```

```
# Réstauration
systemctl stop knot
mv /var/lib/knot{,.bak}
mkdir /var/lib/knot
cp -r /my/backups /var/lib/knot/backups
cp -r /var/lib/knot/backups/{keys,catalog,timers} /var/lib/knot

# L'arborescence doit être recréée à la main pour une restauration hors ligne
mkdir /var/lib/knot/{catalog-,}zones
cp /var/lib/knot/backups/*-catalog.zone /var/lib/knot/catalog-zones
cp /var/lib/knot/backups/*.zone /var/lib/knot/zones
systemctl start knot

# Vérifier la restauration des clés dnssec

kdig familier.net.eu.org. @localhost dnskey
grep DNSKEY /var/lib/knot/backups/familier.net.eu.org.zone
```

Références :

- https://www.knot-dns.cz/docs/3.0/html/operation.html#data-and-metadata-backup

Uwsgi

Plugin sous fedora

```
- uwsgi-logger-file
- uwsgi-plugin-python3
- uwsgi-plugin-common
```

SELinux

Pour afficher toutes les règles fcontext du système :

```
semanage fcontext -l
```

Pour donner à un dossier et tout ses sous-dossiers/fichiers les même contexte qu'un autre, utilisez "l'équivalence"

```
semanage fcontext -a -e /home /mnt/nextcloud-lamal
```

Application automatique des régles

Editer /etc/selinux/restorecond.conf.

Attention: ce n'est pas récursif, donc * ne s'applique qu'au 1er niveau

Comprendre un message d'AVC

Dans l'exemle suivant d'AVC de qbittorrent

type=AVC msg=audit(1685394119.816:61526): avc: denied { **name_bind** } for pid=157511 comm="qbittorrent-nox" src=8080 scontext=system_u:system_r:**qbittorrent.process** :s0:c557,c831 tcontext=system_u:object_r:**http_cache_port_t**:s0 tclass=**tcp_socket** permissive=0

`name_bind` est l'action

Web

Autoriser un dossier à être lu par le serveur Web.

```
chcon -Rt httpd_sys_content_t /path/to/www # temporairement
semanage fcontext -a -t httpd_sys_content_t '/path/to/www(/.*)?' # definitivement
```

Pour executer du php il faut l'attribut `httpd_sys_script_exec_t`
Pour ecrire des fichiers `httpd_sys_rw_content_t`

Conteneurs

Autoriser un conteneur à accéder à un dossier mappé sur l'hôte (flag `Z`).

```
podman run --rm -v /path/to/volume:/data:Z debian
```

Creer une policy custom (podman):

Extraire les settings du conteneur et run udica dessus :

```
podman inspect [id/nom du pod] > conteneur.json


udica -j conteneur.json mon_conteneur
```

Editer le fichier .cil créé, et ajouter/modifier selon ce que vous voulez autoriser

Exemple pour qbittorrent :

```
(block qbittorrent
    (blockinherit container)
    (blockinherit restricted_net_container)
    (allow process process ( capability ( chown dac_override fowner fsetid kill
net_bind_service setfcap setgid setpcap setuid sys_chroot )))

    # Autorise a se bind et a contacter tous les ports
    (allow process port_type ( tcp_socket (  name_bind name_connect )))
    (allow process port_type ( udp_socket (  name_bind )))

    # Autorise a acceder aux fichier avec le contexte httpd_user_ra_content_t
    (allow process httpd_user_ra_content_t ( dir ( add_name create getattr ioctl lock open
read remove_name rmdir search setattr write )))
    (allow process httpd_user_ra_content_t ( file ( append create getattr ioctl lock map open
read rename setattr link unlink write )))
    (allow process httpd_user_ra_content_t ( fifo_file ( getattr read write append ioctl lock
open )))
    (allow process httpd_user_ra_content_t ( sock_file ( append getattr open read write link
create setattr unlink execute rename )))
```

```
    (allow process default_t ( dir ( add_name create getattr ioctl lock open read remove_name
rmdir search setattr write )))
    (allow process default_t ( file ( append create getattr ioctl lock map open read rename
setattr unlink write )))
    (allow process default_t ( fifo_file ( getattr read write append ioctl lock open )))
    (allow process default_t ( sock_file ( append getattr open read write )))
    (allow process var_t ( dir ( add_name create getattr ioctl lock open read remove_name
rmdir search setattr write )))
    (allow process var_t ( file ( append create getattr ioctl lock map open read rename
setattr unlink write )))
    (allow process var_t ( fifo_file ( getattr read write append ioctl lock open )))
    (allow process var_t ( sock_file ( append getattr open read write )))
)
```

Définir le contexte (httpd_user_ra_content_t ici) des dossiers qui sont accedés par le conteneur (et d'autres process éventuels)

```
semanage fcontext -a -t httpd_user_ra_content_t "/var/opt/qbittorrent/config(/.*)?"
semanage fcontext -a -t httpd_user_ra_content_t "/data/downloads(/.*)?"
restorecon -Rv /var/opt/qbittorrent/config/  # Ajouter l'option -F si ça marche pas
restorecon -Rv /data/downloads/ # Pareil
```

importer la policy :

```
semodule -i qbittorrent.cil /usr/share/udica/templates/{base_container.cil,net_container.cil}
```

relancer le conteneur, verifier les alertes selinux, autoriser ce qui génère les alertes, supprimer et reimporter la policy, relancer le conteneur, repeat until it works...

```
semodule -r qbittorent  # Supprime la policy
tail -f /var/log/audit/audit.log  # Lookout for les AVC here
```

PostgreSQL

Mise à jour

```
# Stocker l'ancienne version de PostgreSQL, utile pour plus tard
set PG_VERSION $(cat /var/lib/pgsql/data/PG_VERSION)

# Monter la dernière sauvegarde pour récupérer les libs manquantes
mkdir /mnt/backup
borgmatic mount --archive latest --mount-point /mnt/backup

# Copier les anciennes libs dans le dossier des anciennes libs
cp /mnt/backup/usr/lib64/pgsql/*.so /usr/lib64/pgsql/postgresql-$PG_VERSION/lib/

# Déplacer les données de l'ancienne version
mv /var/lib/pgsql/data /var/lib/pgsql/data-old

# Initialiser le répertoire de données pour la nouvelle version
mkdir /var/lib/pgsql/data
chown postgres: /var/lib/pgsql/data
sudo -iu postgres /usr/bin/initdb --pgdata=/var/lib/pgsql/data --auth=ident

# Faire la migration de données
sudo -iu postgres /usr/bin/pg_upgrade \
  --old-bindir=/usr/lib64/pgsql/postgresql-$PG_VERSION/bin \
  --new-bindir=/usr/bin \
  --old-datadir=/var/lib/pgsql/data-old \
  --new-datadir=/var/lib/pgsql/data \
  --link --old-port=5432 --new-port=5432 --username=postgres

# Copier les fichiers de configuration
cp /var/lib/pgsql/data-old/postgresql.conf /var/lib/pgsql/data
cp /var/lib/pgsql/data-old/pg_hba.conf /var/lib/pgsql/data

# Démarrer PostgreSQL
systemctl restart postgresql
```

```
# Optimiser la nouvelle version
sudo -iu postgres /usr/bin/vacuumdb -U postgres --all --analyze-in-stages

# Nettoyage
sudo -iu postgres ./delete_old_cluster.sh
umount /mnt/backup
```

Troubleshooting

The database was created using collation version X.XX, but the operating system provides version Y.YY

```
sudo -iu postgres

for t in `psql -tAc "SELECT datname FROM pg_database WHERE datistemplate = false"`; do psql -d $t -c "REINDEX DATABASE $t"; psql -d $t -c "ALTER DATABASE $t REFRESH COLLATION VERSION" ; done

for t in `psql -tAc "SELECT datname FROM pg_database WHERE datistemplate = true"`; do  psql -c "ALTER DATABASE $t REFRESH COLLATION VERSION" ; done
```

Podman

Startup / service

Creation et installation d'un service pour lancer un pod rootless :

```
root$ loginctl enable-linger keycloak
root$ sudo -iu keycloak
keycloak$ export XDG_RUNTIME_DIR=/run/user/$(id -u $USER)
keycloak$ mkdir ~/.config/containers/systemd/ -p
keycloak$ podlet generate container keycloak-26.2.4 > ~/.config/containers/systemd/keycloak-26.2.4.container
keycloak$ systemctl --user daemon-reload
keycloak$ systemctl --user start keycloak-26.2.4
```

Creer le fichier de service systemd niveau root, pour le lancement au boot:

Dans `/etc/systemd/system/`

```
[Unit]
Description=Start keycloak
After=postgresql.service

[Service]
Type=oneshot
ExecStart=systemctl --machine=keycloak@ --user start keycloak-26.4.5

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

Puis enable le service.

Gestion du service user :

Option 1 :

```
root$ sudo -iu <user>
<user># export XDG_RUNTIME_DIR=/run/user/$(id -u $USER)
<user># systemctl --user <command>
```

Option 2 :

```
root$ systemctl --machine=keycloak@ --user <start/stop/status/...> <service>
```

Remote Wireshark

Use wireshark remotely

create a fifo file, owned by your user:

```
mkfifo ~/ftcpdump
```

In another terminal, send the data into wireshark

on your local machine do:

```
ssh server "cat ~/ftcpdump" | wireshark -k -i
```

(do not forget to filter out ssh)

Output tcpdump data into the fifo

from the remote machine :

```
tcpdump <filter> -U -s 0 -w ~user/ftcpdump
```