:3 Fiches de Révision — Virtualisation, Réseau et Conteneurisation

---

UwU Fiche 1 : Introduction à Proxmox

🔹 1. Présentation Générale

Proxmox VE (Virtual Environment) est un **hyperviseur open-source de type 1** (*bare-metal*) basé sur **Debian Linux**.

Virtualisation fournie :

- **Machines Virtuelles (VM)** avec **KVM** (*Kernel-based Virtual Machine*).
- **Conteneurs (CT)** avec **LXC** (*Linux Containers*).

Fonctionnalités principales :

- Administration via une **interface Web** complète.
- **Clusterisation** : regroupe plusieurs serveurs Proxmox.
- **Gestion du stockage** : local, réseau (Ceph, SAN…).
- **Sauvegarde et restauration** intégrées.
- **Licence** : GNU AGPL (support commercial disponible).
- **Version Community** : gratuite, sans support, affiche une alerte de souscription.

🔹 2. Accès et Interface

- **Accès Web UI** : `https://<adresse_ip_hôte>:8006`
- **Interface principale :**
  - Vue d'ensemble du *Datacenter* ou du *Nœud*.
  - Arborescence : Nœuds, VMs, CTs, Stockage.
  - Panneau central : résumé, console, options.
  - Journal des tâches en bas.

🔹 3. Gestion des Images

- **Types d'images :**
  - **ISO** : installation d'OS pour VMs.
  - **Templates CT** : images préconfigurées pour LXC.
- **Stockage** : dans les espaces configurés (ex : `local`).
- **Ajout d'images :**
  - ISO → upload local ou téléchargement direct.
  - Templates CT → téléchargement via l'UI depuis la liste officielle.

---

4. Création de Machines

VM (Machine Virtuelle)

1. Cliquer sur **Create VM**.
2. Choisir une **image ISO**.
3. Configurer : nom, OS, stockage, CPU, RAM, réseau.
4. Attacher l'ISO comme CD virtuel.
5. Démarrer la VM → installation via la console.

CT (Conteneur)

1. Cliquer sur **Create CT**.
2. Sélectionner un **template CT** téléchargé.
3. Configurer : nom, stockage, CPU, RAM, réseau (IP statique ou DHCP).
4. Démarrer le CT → système déjà prêt.

---

5. Sauvegardes et Restauration

- **Par VM/CT** : onglet *Backup*.
  → Créer, lister, restaurer, supprimer.
- **Niveau Datacenter** :
  - Vue globale des sauvegardes.
  - Planification via *Backup Jobs* (rétention, fréquence…).

UwU Fiche 2 : Réseau dans VirtualBox

1. Rappels Réseau

- **Modèle OSI** : 7 couches (Physique → Application).
- **Protocoles clés :**
  - IP (adressage réseau global)
  - TCP/UDP (transport, ports)
- **Adresse IPv4** : 32 bits (4 octets).
  Masque de sous-réseau (`/24` = 255.255.255.0).
- **Communication :**
  - Même réseau → directe via switch.
  - Réseaux différents → via routeur.

---

2. Configuration Réseau Linux avec `nmcli`

- **Visualiser :**
  - `nmcli device status`
  - `nmcli connection show`
- **Créer une connexion :**

```
nmcli connection add type ethernet con-name "profil" ifname eth0
```

- **Configurer IP statique :**

```
nmcli connection modify "profil" ipv4.addresses 192.168.1.10/24
ipv4.gateway 192.168.1.1 ipv4.dns "8.8.8.8" ipv4.method manual
```

- **Activer / désactiver :**

```
nmcli connection up "profil"
nmcli connection down "profil"
```

- **Fichiers :** `/etc/NetworkManager/system-connections/*.nmconnection`

---

3. Modes Réseau VirtualBox

Mode	Description
Not attached	Pas de connexion.
NAT	Accès Internet via IP hôte, non joignable.
NAT Network	Réseau virtuel entre VMs, accès externe partagé.
Bridged Adapter	VM sur le même réseau physique que l'hôte.
Internal Network	Réseau isolé entre VMs uniquement.
Host-only Adapter	Réseau entre l'hôte et les VMs uniquement.
Cloud / Generic	Modes avancés (expérimentaux).

UwU Fiche 3 : Introduction à la Virtualisation

1. Concepts

- **Virtualisation** : exécution d'un système invité sur un hôte.
- **Hôte** : machine physique.
- **Invité** : VM ou conteneur.

---

2. Bénéfices

- **Économiques :** mutualisation, consolidation.
- **Opérationnels :** déploiement rapide, tests multi-OS.
- **Sécurité :** isolation, sauvegarde, migration, PRA.

- **Flexibilité :** scalabilité, mobilité (VDI).

---

3. Types de Virtualisation

Type	Description
Émulation complète	Simule tout le matériel (lent, très flexible).
Hyperviseur Type 1	Directement sur le matériel (Proxmox, ESXi).
Hyperviseur Type 2	Sur un OS hôte (VirtualBox, VMware Workstation).
Conteneurisation	Isolation de processus (Docker, LXC).
Paravirtualisation	OS invité conscient de la virtualisation.
Traduction d'appels système	Compatibilité (ex : Wine, WSL).

---

4. VirtualBox

- **Type :** Hyperviseur de Type 2.
- **Plateformes :** Windows, Linux, macOS.
- **Usage :** création de VMs sur poste local.

UwU Fiche 4 : Docker — Images et Conteneurs

1. Principe

Docker isole des **processus** partageant le même noyau.
Performant et léger, idéal pour le déploiement d'applications.

---

2. Image vs Conteneur

Élément	Description
Image	Modèle en lecture seule. (Analogie : classe POO)
Conteneur	Instance en exécution. (Analogie : objet POO)

---

3. Création d'Images

- **Via Dockerfile :**

```
docker build -t monimage:1.0 .
```

- **Via Commit :**

```
docker commit <id_conteneur> monimage:debug
```

🔹 4. Dockerfile — Instructions Clés

```
FROM debian:bookworm
RUN apt-get update && apt-get install -y nginx
COPY . /app
WORKDIR /app
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

🔹 5. Système de Couches

Chaque instruction crée une **couche**.
Les couches sont partagées → gain d'espace et de temps.
⚠ Supprimer un fichier dans une couche supérieure ne réduit pas la taille totale.

🔹 6. Partage d'Images

- **Sauvegarde locale :**
  `docker image save monimage -o image.tar`
- **Chargement :**
  `docker image load -i image.tar`
- **Docker Hub :**
  `docker push monuser/monrepo:latest`

UwU Fiche 5 : Docker — Stockage et Compose

🔹 1. Persistance des Données

Type	Description
Volume	Géré par Docker. Recommandé.
Bind Mount	Dossier de l'hôte monté dans le conteneur.
tmpfs	En mémoire, temporaire.

🔹 2. Utilisation

```
docker run -v monvolume:/data nginx
docker run --mount type=bind,source="$(pwd)",target=/app node
```

---

3. Docker Compose

- Fichier : `compose.yaml`
- Commandes :

```
docker-compose up -d
docker-compose down -v
```

- Exemple :

```
services:
  web:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - ./site:/usr/share/nginx/html
```

---

4. Introduction à Kubernetes

- **Objectif :** orchestration des conteneurs.
- **Composants :**
  - Master Node : API server, scheduler, etcd.
  - Worker Nodes : exécutent les *Pods*.
- **Outils :** `kubectl`, dashboard, cloud (EKS, GKE, AKS).

UwU Fiche 6 : Serveurs Web

1. Rôle

- Sert du contenu HTTP/HTTPS (statique ou dynamique).
- Exemple d'URL : `https://www.example.com:443/index.html`

---

2. Apache HTTP Server

- **Installation :** `apt install apache2`
- **Structure :**
  - `/etc/apache2/sites-available/`
  - `/etc/apache2/sites-enabled/`
- **Commandes :** `a2ensite`, `a2enmod`, `systemctl reload apache2`

Exemple VirtualHost :

```
<VirtualHost *:80>
  ServerName www.example.com
  DocumentRoot /var/www/example
  Require all granted
</VirtualHost>
```

- 3. Nginx

  - **Installation :** `apt install nginx`
  - **Structure :** `/etc/nginx/sites-available/`
  - **Exemple :**

```
server {
  listen 80;
  server_name example.com;
  root /var/www/example;
}
```

- 4. Reverse Proxy (Nginx)

```
location /api/ {
  proxy_pass http://127.0.0.1:5000/;
}
```

Usages : répartition de charge, sécurité, SSL termination, cache.

UwU Fiche 7 : Réseau Docker

- 1. Concepts

  - Un conteneur est une instance d'image.
  - **Modes :**
    - *Détaché (-d)* : arrière-plan.
    - *Interactif (-it)* : terminal ouvert.

- 2. Exposition des Ports

```
docker run -p 8080:80 nginx
```

- `-p` : mappe un port hôte → conteneur.
- `-P` : mappe automatiquement tous les ports exposés.

🔹 3. Réseaux Docker

Pilote	Description
bridge	Par défaut, privé à l'hôte.
host	Partage l'interface de l'hôte.
overlay	Réseau multi-hôte (Swarm).
macvlan	Adresse MAC propre, visible sur LAN.
ipvlan	Même MAC, IPs séparées.
none	Pas de réseau du tout.

Commandes utiles :

```
docker network ls
docker network create --driver bridge monreseau
docker network inspect monreseau
```

Fin des fiches de révision — Virtualisation & Conteneurisation