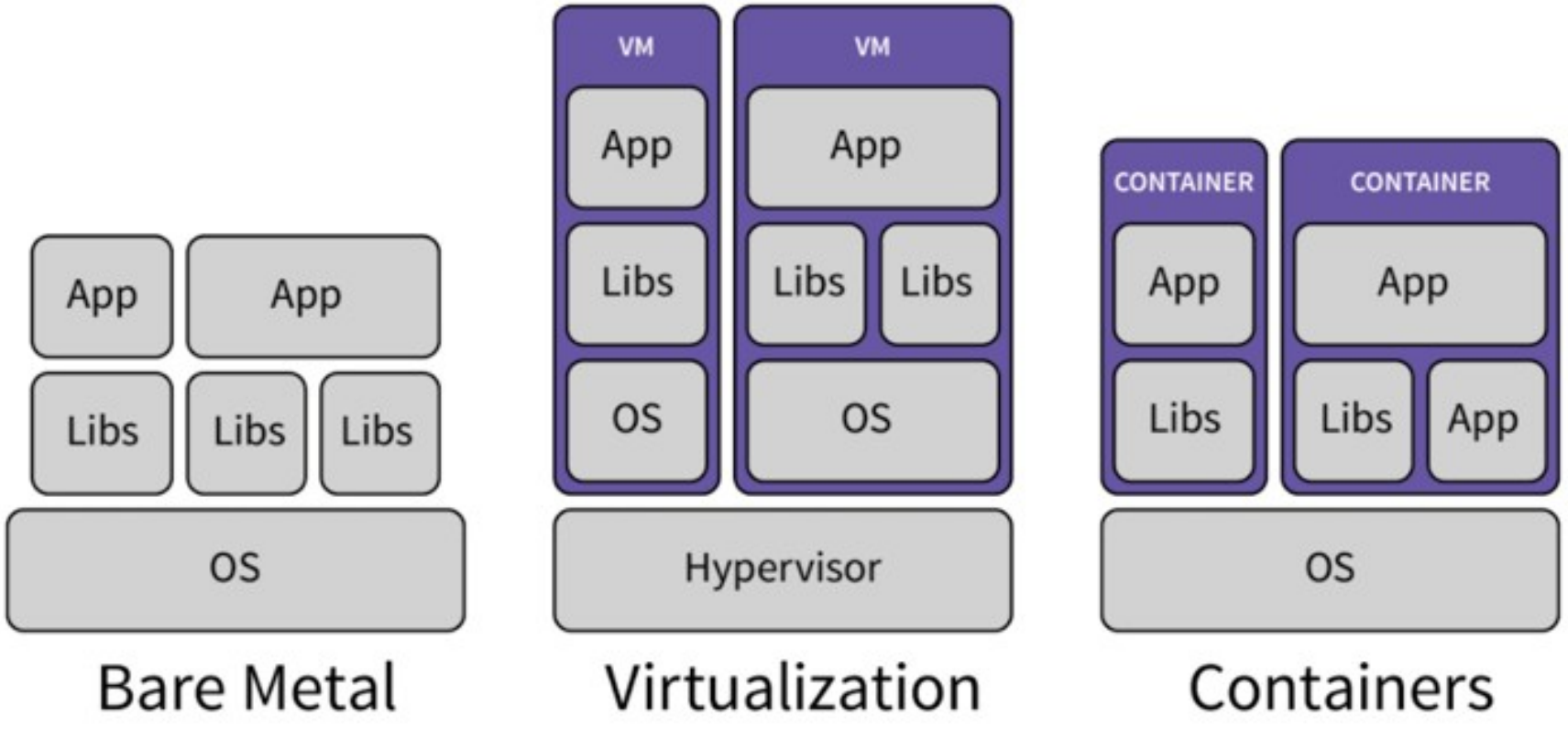
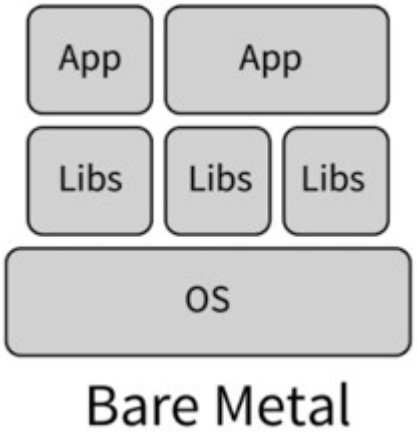
podman

Déployer une application



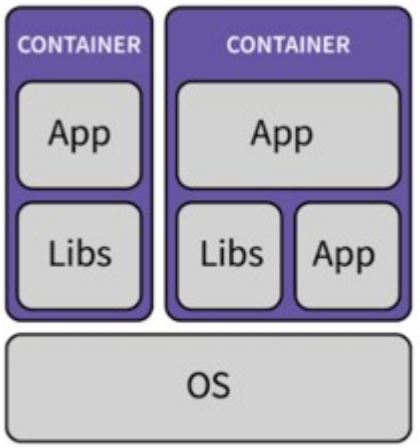


- Manque de flexibilité
- Performance, accès direct au matériel

Virtualization

- Surconsommation de ressource

- Isolation (quasi) complète
- Plus flexible qu’en bare metal
- Possibilité d’exécuter un autre système



Containers

- Surconsommation de ressource
- Même kernel que l’hôte

- Bonne isolation
- Grande flexibilité

Comparaison podman docker

Podman

- Rootless
- Microservices
  (Intégration avec systemd)
- Intégration avec Kubernetes
  (podman generate kube)

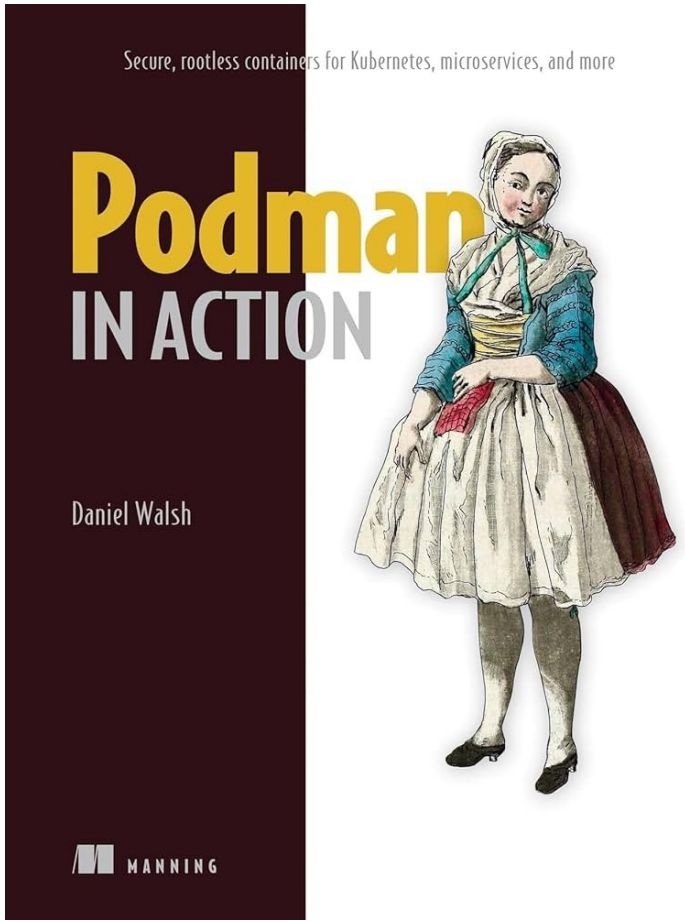
Docker

- Stack en root
  (possibilité de faire du rootless depuis la version 19)
- Docker compose

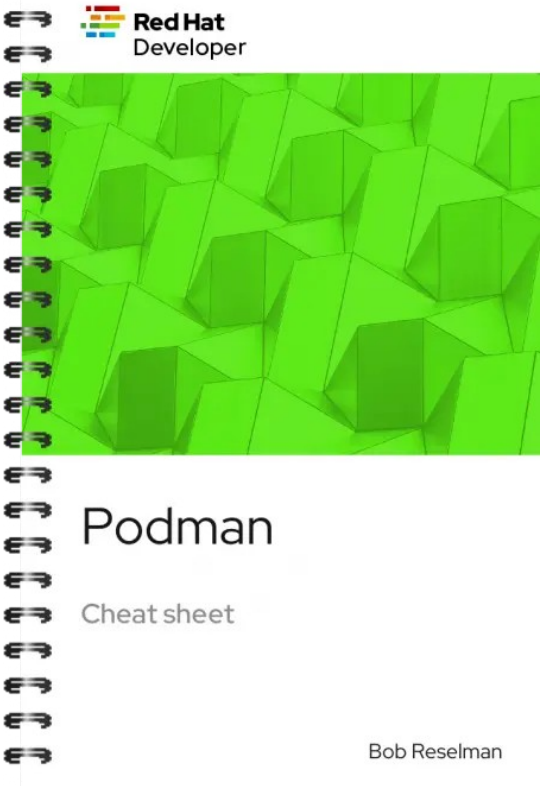
Utilisation de podman

Documentation

docs.podman.io



Red Hat Developer



Rootless

- **Réseau**
  → slirp4netns
  → port >= 1024
- **UIDs /etc/subuid et /etc/subgid** ***(shadow-utils)***

  format USERNAME:UID:RANGE

  exemple /etc/subuid:

  bob:100000:65536

  alice:165536:65536

Création d’un 1er conteneur

- Exemple avec apache

  podman create --name apache -p 8080:80 docker.io/library/httpd

  podman start apache

- Lister les conteneurs

  podman ps -a

Création d’un 1er conteneur

- En une commande

  podman run -dt --name apache -p 8080:80 docker.io/library/httpd

Création d’un 1er conteneur

- Voir les logs

  podman logs apache

  podman logs -f apache
  podman logs --follow apache

Création d’un 1er conteneur

- Stopper/détruire le conteneur

  podman stop apache

  podman rm apache

  podman kill apache

  podman rm -f apache *(tuer et supprimer)*

Création d’un 1er conteneur

- Ouvrir un terminal dans un conteneur

  podman exec -it nom-conteneur /bin/sh

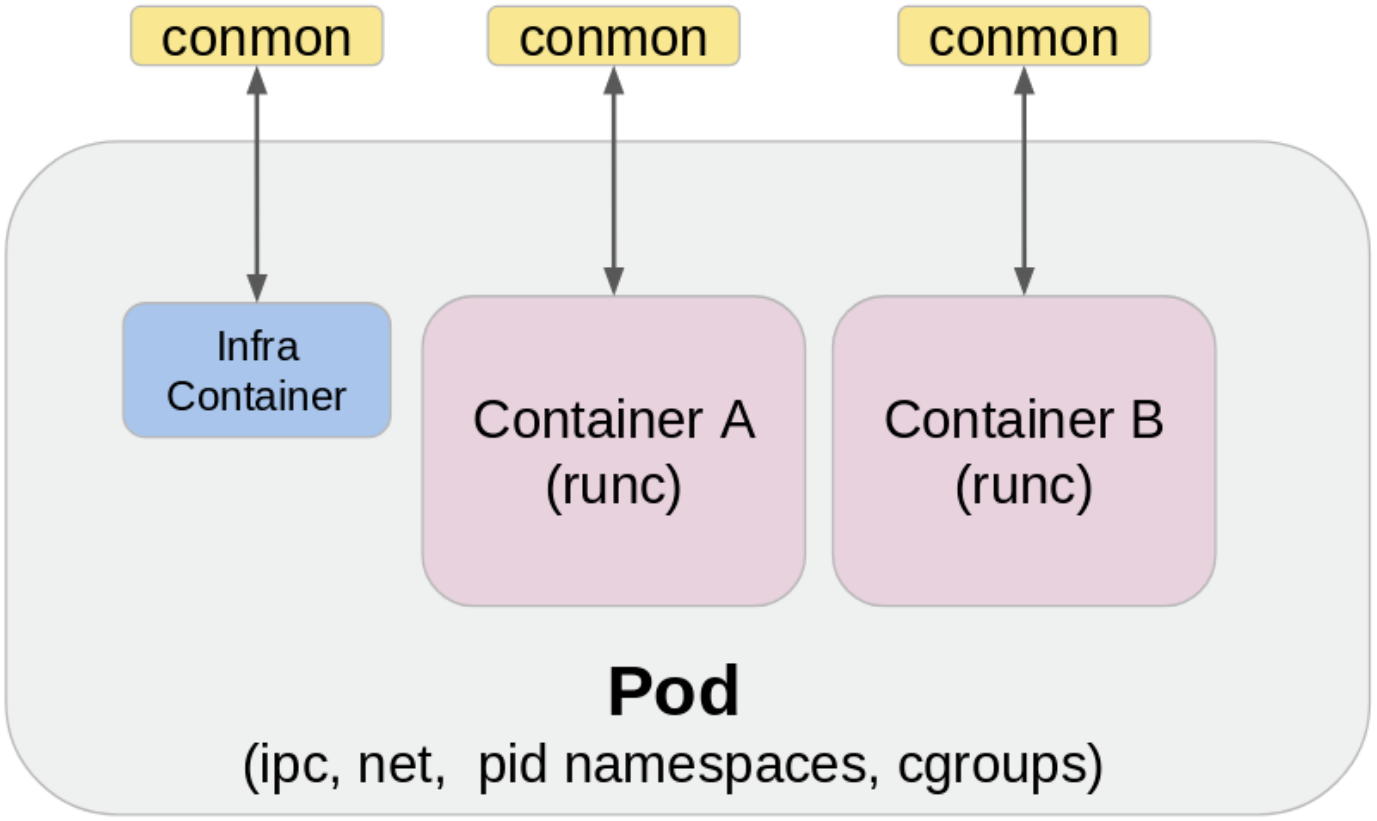
Création d’un 1er conteneur

- Variable d’environnement

  podman run … -e NOM_VARIABLE=”toto”

- Volumes
- podman run … -v /home/fire/Downloads:/test
  podman run … -v /home/fire/Downloads:/test:ro

Création d’un 1er pod



Création d’un 1er pod

- Exemple

  podman pod create --name mon-premier-pod -p 3000:3000 -p 2223:2223

- Ajouter des conteneurs

  podman run -d --pod=gitea …

Création d’un 1er pod

- Avec des limites

  podman pod create --name mon-premier-pod -p 3000:3000 -p 2223:2223 --cpu-shares=512 --memory=1024m

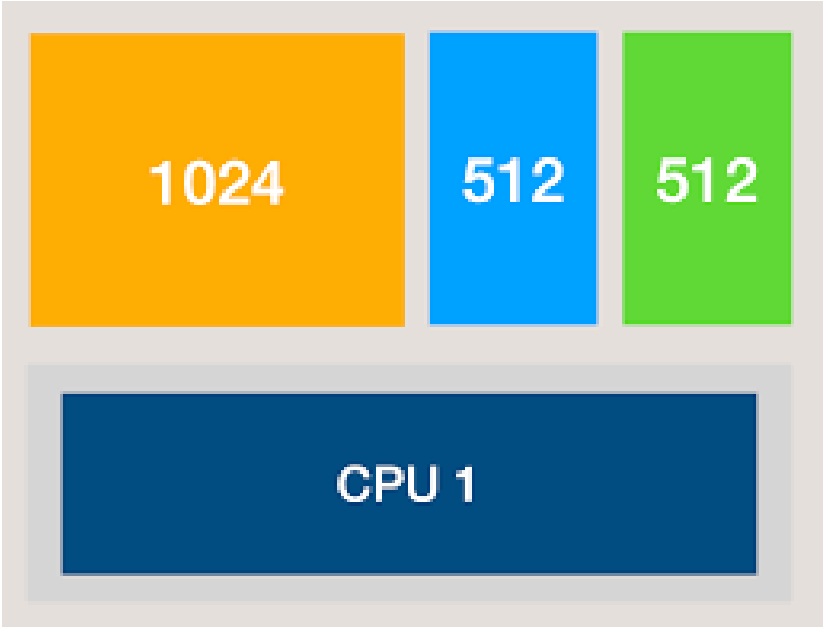
- Voir les ressources utilisées

  podman stats [nom-conteneur]

Création d’un 1er pod

- cpu-shares

  podman pod create ... --cpu-shares=512 ...



Sécurité

SElinux

- Sécuriser les accès aux fichiers (volumes dans les conteneurs)

  → est-ce que 2 conteneurs ont le droit d'accéder à des fichiers en commun?

  → est-ce que l'hôte doit avoir accès aux fichiers d'un conteneur?

SElinux

→ **options pour les volumes**

- Accès partagé

  /path/on/host:/path/podman:z

- Accès privé

  /path/on/host:/path/podman:Z

SElinux

- Règles particulières

  → udica

  podman ps *(récupérer l'id du conteneur)*

  podman inspect <id> | udica my_container

Gestion des secrets

- Podman secrets

  → permet d’éviter d’exposer ses secrets dans les fichiers de configuration

  echo -n "<postgres-pass>" > /tmp/secret

  podman secret create postgres_pass /tmp/secret

  podman run -d --pod=nextcloud --secret postgres_pass,type=env,target=POSTGRES_PASSWORD ...

Gestion avec des services

- Systemd

  cd ~/.config/systemd/user/

  podman generate systemd --restart-policy=on-failure --files --new --name mon-pod

  systemctl --user daemon-reload

  systemctl --user enable --now pod-mon-pod.service

Mises à jours

- Automatiser les mises à jour des conteneurs

  → avec un label et un tag d’image

  Exemple:

  podman run -dt --name apache -p 8080:80 --label io.containers.autoupdate=registry docker.io/library/httpd:latest

  podman auto-update --dry-run

  systemctl --user enable podman-auto-update.{service,timer}

Mises à jours

- rollbacks

  → sdnotify

  l’image doit être crée avec “systemd-notify --ready;” dedans (indiquer le bon fonctionnement du conteneur)

  option --rollback lors de la création du conteneur (true par défaut)

Création d’images

Image OCI

- OCI = Open Container Initiative

  → par habitude fichiers dockerfile pour construire les images

  → on peut aussi utiliser buildah en ligne de commande

Image OCI

- Exemple de fichier Dockerfile

Dockerfile

```
1  FROM python:bookworm
2  COPY requirements.txt /
3  RUN pip3 install --upgrade pip
4  RUN pip3 install -r /requirements.txt
5
6  COPY ./src /app
7  COPY ./web /web
8
9  WORKDIR /app
10
11 EXPOSE 8080
12
13 ENV POSTGRES_HOST=127.0.0.1
14 ENV POSTGRES_PORT=5432
15
16 CMD ["gunicorn","--config", "gunicorn_config.py", "app:app"]
```

Image OCI

- Construction depuis un fichier dockerfile

  podman build --tag mon-image:1.0.0 -f ./Dockerfile

- Lister les images locales

  podman images

- Ajout de tag

  podman tag <image:tag>/<image-id> image:tag2

Image OCI

- Envoi de l’image sur un registre

  podman push registry/username/image:tag

- Gitea fonctionne très bien comme registre pour ses propres images

  podman login gitea.example.com

  podman push gitea.example.com/{owner}/{image}:{tag}

Image OCI

- Exemple de construction avec buildah

  buildah from centos

  buildah run centos-working-container yum install httpd -y
  echo "Hello from Red Hat" > index.html

  buildah copy centos-working-container index.html /var/www/html/index.html

  buildah config --entrypoint "/usr/sbin/httpd -DFOREGROUND" centos-working-container

  buildah commit centos-working-container redhat-website

Questions ?