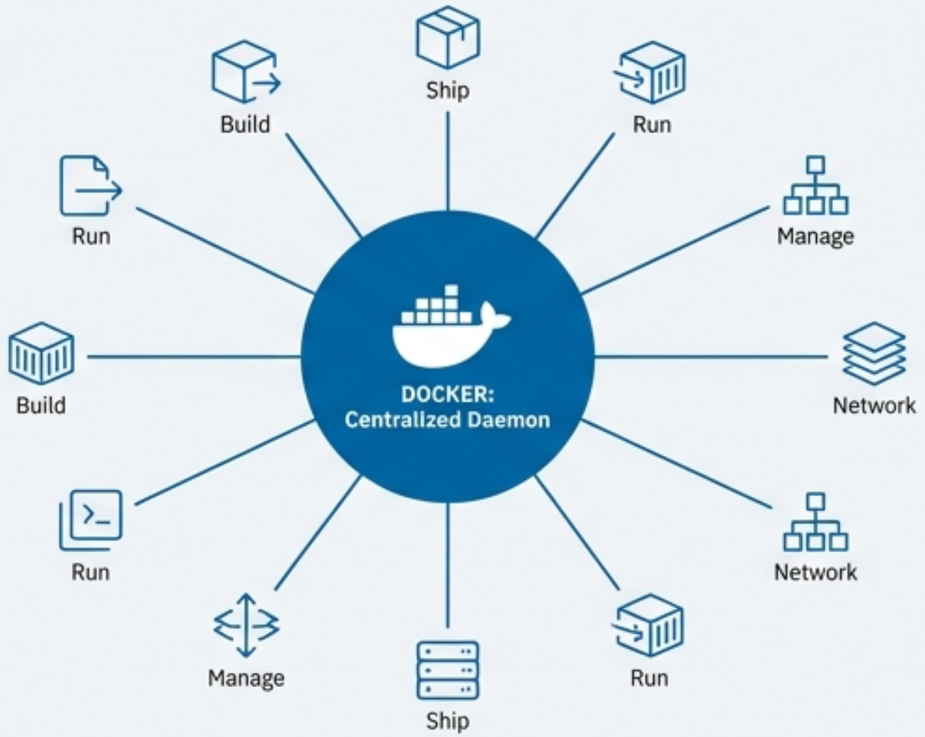
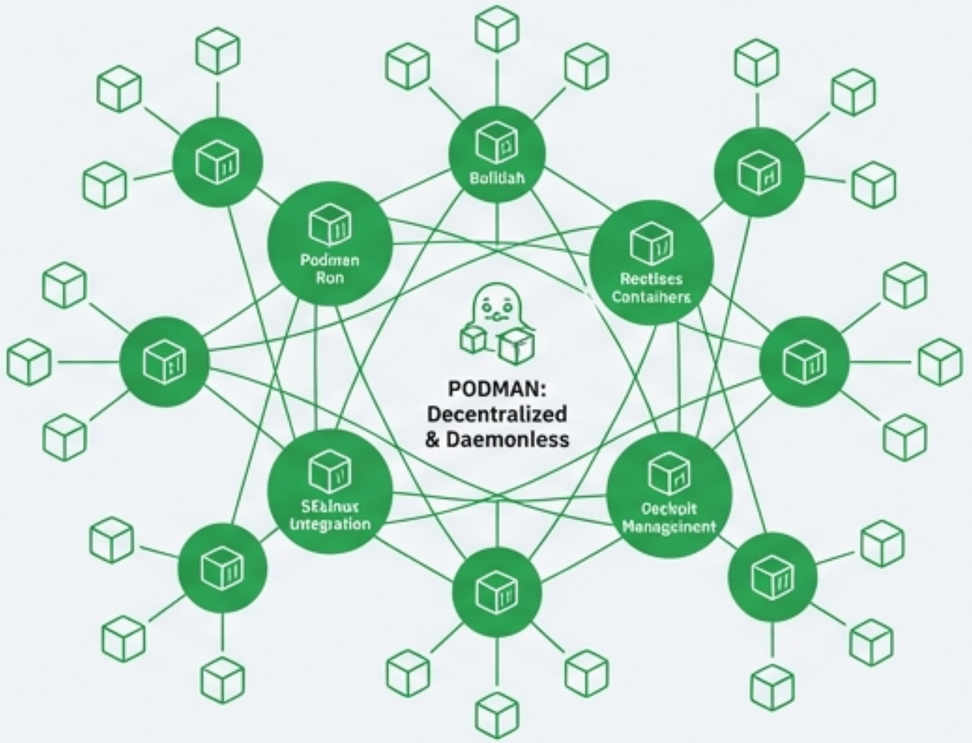
L'Écosystème des Conteneurs en 2025 : Maîtriser Podman au-delà de Docker

Un guide stratégique pour l'administrateur moderne, de l'architecture à la gestion avec Cockpit et SELinux.

La conteneurisation a mûri. Au-delà de la simple exécution d'applications, le choix d'un runtime est devenu une décision d'architecture, de sécurité et d'écosystème. Cette présentation explore les deux philosophies qui dominent aujourd'hui et vous donne les clés pour naviguer ce choix de manière éclairée.



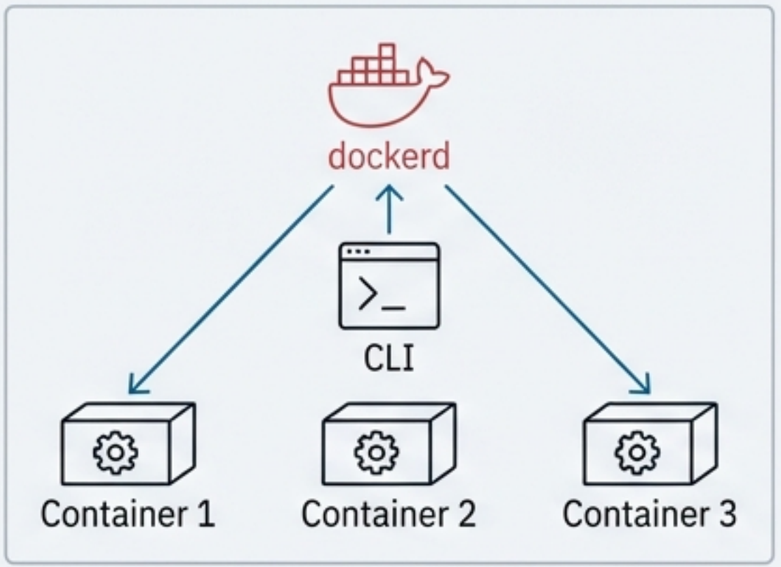
Daemon-based Architecture



Daemonless & Rootless Architecture

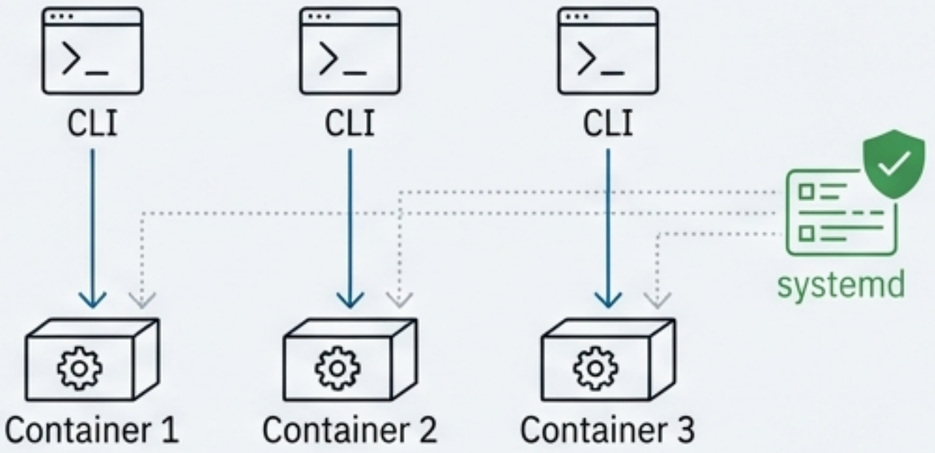
Deux Philosophies, Une Norme (OCI)

Le Modèle Centralisé : La Tour de Contrôle



- **Architecture:** Un service d'arrière-plan persistant (`dockerd`) gère le cycle de vie de tous les conteneurs.
- **Avantage:** Pratique et centralisé. Démarrage de conteneur individuel marginalement plus rapide.
- **Risque:** Point de défaillance unique (SPOF). Si le daemon tombe, tous les conteneurs s'arrêtent.

L'Approche Distribuée : La Flotte Autonome



- **Architecture:** "Daemonless". Chaque conteneur est un processus enfant de la commande qui l'a lancé.
- **Avantage:** Pas de SPOF, pas de service root persistant. Supervision possible via `systemd`.
- **Bilan:** Zéro surcharge mémoire au repos (`idle overhead`).

L'Impact sur la Sécurité : Le Principe du Moindre Privilège

Le 'Rootless' comme standard

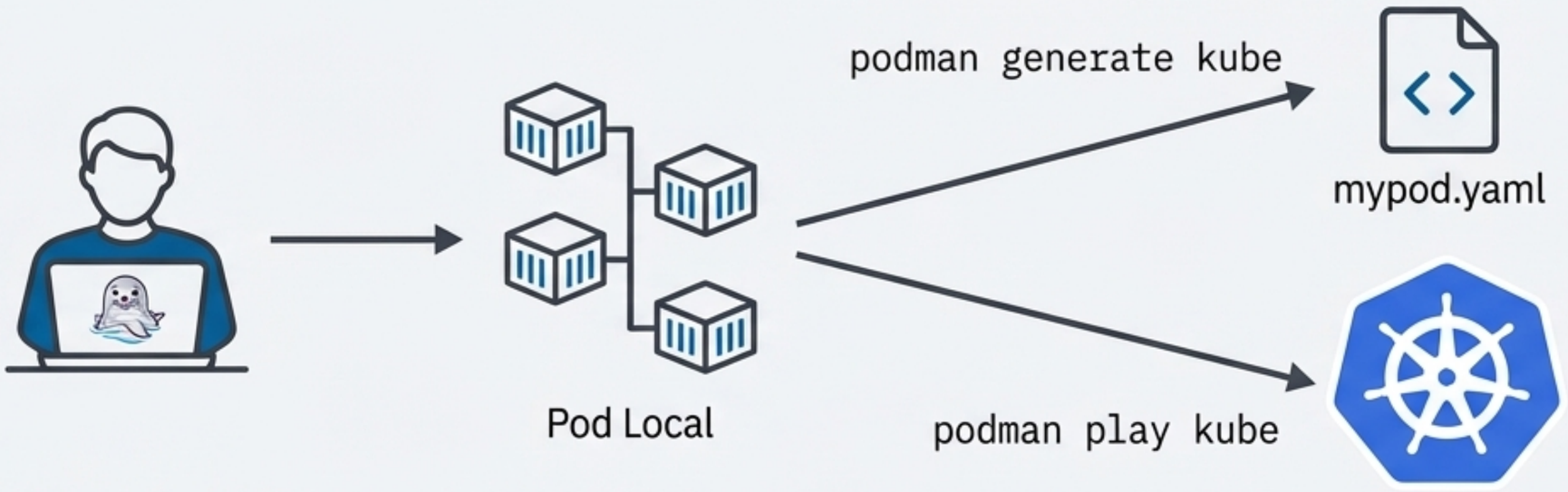
- **Podman** : Conçu nativement pour le mode 'rootless'. Le root *dans* le conteneur est mappé à un utilisateur non-privilégié sur l'hôte. C'est la configuration par défaut.
- **Docker** : Le support 'rootless' a été ajouté plus tardivement. Le mode 'rootful' reste le défaut dans de nombreux déploiements.

> Le défaut majeur de Docker est qu'il exécute les conteneurs en tant que root, donnant à l'application un accès complet au système. Podman élimine ce risque en exécutant les conteneurs avec les permissions d'un utilisateur normal."

Surface d'attaque et intégration

- **Podman** : Minimise les risques en évitant un daemon privilégié et en utilisant des permissions par défaut plus strictes. Son intégration avec SELinux est plus profonde.
- **Docker** : Le daemon `dockerd` représente une surface d'attaque potentielle. Supporte SELinux, mais l'intégration est moins native dans certains environnements.

Conçu pour Kubernetes : Du Local à l'Orchestration



La Transition Transparente vers K8s

Podman adopte une conception "Kubernetes-first". Il permet de créer et gérer des *pods* (groupes de conteneurs) localement, reflétant la brique de base de Kubernetes.

`podman generate kube mypod > mypod.yaml` Exporte la configuration d'un pod Podman en cours d'exécution vers un manifeste YAML compatible Kubernetes.

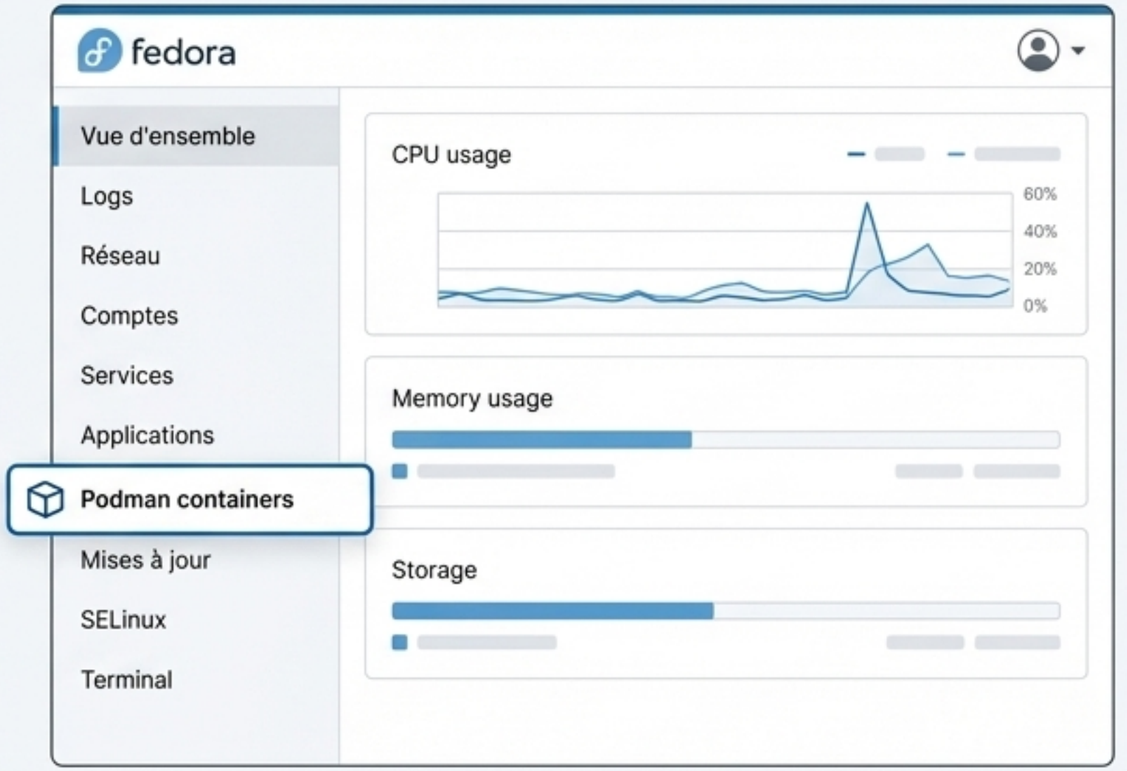
`podman play kube mypod.yaml` Déploie un conteneur ou un pod directement à partir d'un manifeste Kubernetes.

Bilan Stratégique Podman est un excellent choix pour les équipes qui migrent leurs charges de travail vers Kubernetes, en réduisant la friction entre l'environnement de développement et celui de production.

Le Poste de Commande : Gérer Podman avec **Cockpit** sur **Fedora Server**

Qu'est-ce que Cockpit ?

Cockpit est une interface d'administration web pour serveurs Linux. Sur Fedora Server, il est installé par défaut et intègre une gestion complète des conteneurs Podman via le composant cockpit-podman.



Mise en Place (Check-list)

- Installer les paquets (si nécessaire) :

```
sudo dnf install cockpit cockpit-podman
```

- Activer le service :

```
sudo systemctl enable --now cockpit.socket
```

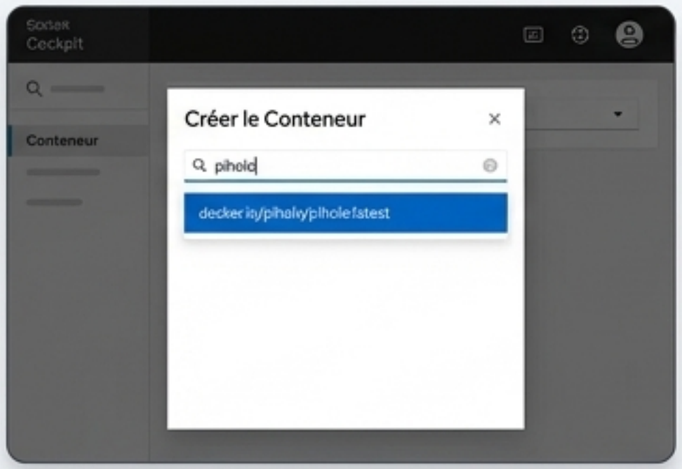
- Démarrer l'API Podman pour l'utilisateur :

```
systemctl --user start podman.socket
```

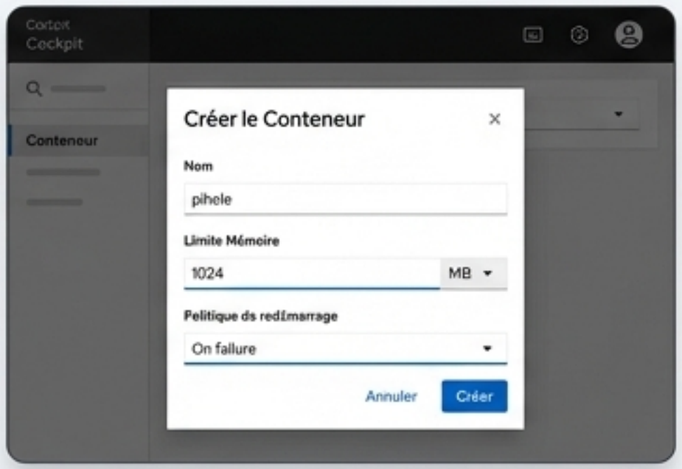
ℹ Essentiel pour que Cockpit puisse interagir avec vos conteneurs.

Connectez-vous simplement via votre navigateur à https://<ip_serveur>:9090 avec vos identifiants système.

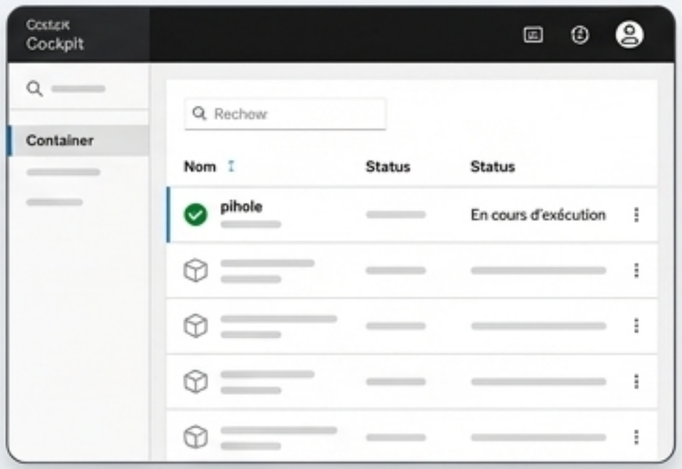
Cas Pratique : Déployer Pi-hole en quelques clics



1. Recherche d'image



2. Configuration Initiale



3. Déploiement Réussi

Préparation sur l'hôte

```
mkdir -p ~/podman/pihole/etc-pihole
mkdir -p ~/podman/pihole/etc-dnsmasq.d
```

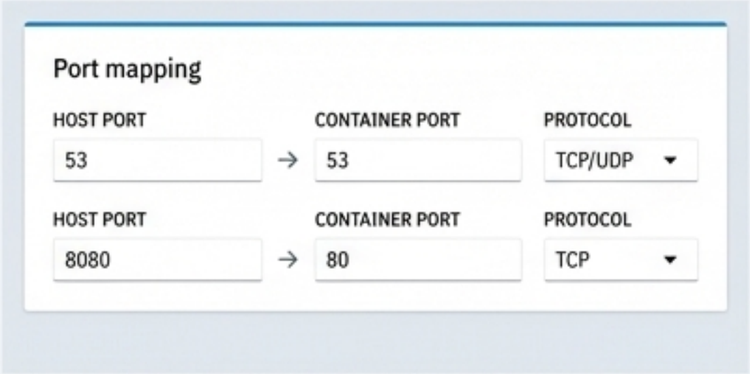
Création du Conteneur (dans Cockpit)

- **Nom:** `pihole`
- **Image:** `docker.io/pihole/pihole:latest` (Montre la capacité de rechercher et tirer des images de registres publics).
- **Limite Mémoire:** 1024 MB
- **Politique de redémarrage:** `On failure`

De la recherche de l'image à la configuration initiale, Cockpit rationalise le déploiement sans masquer la puissance de Podman.

Maîtriser la Configuration : Ports, Volumes et Variables

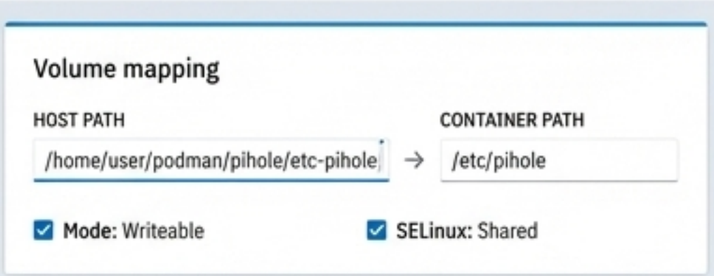
Exposition des Ports



Mappage des ports du conteneur vers l'hôte pour rendre le service accessible sur le réseau.

Exemple (Pi-hole) :
HOST PORT: 53 → CONTAINER PORT: 53 (TCP/UDP)
HOST PORT: 8080 → CONTAINER PORT: 80 (TCP)

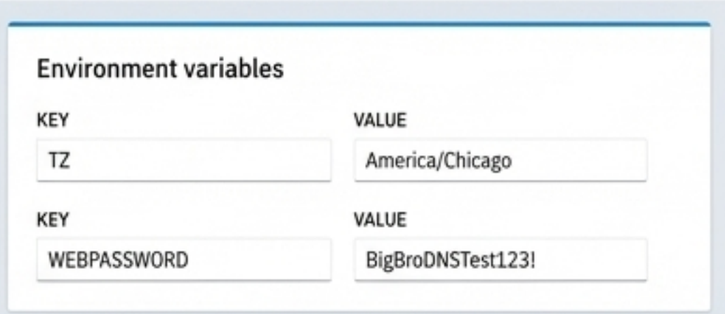
Persistance des Données avec les Volumes



Liaison des répertoires de l'hôte aux répertoires du conteneur pour assurer la persistance des données.

Exemple (Pi-hole) :
HOST PATH: /home/user/podman/pihole/etc-pihole → CONTAINER PATH: /etc/pihole
Options Clés : Mode (Writeable), SELinux (Shared)

Injection de Configuration via Variables d'Environnement

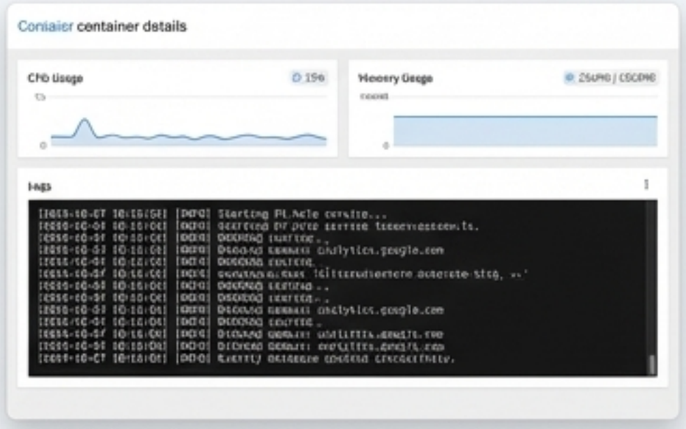


Passage de paramètres essentiels au conteneur au démarrage.

Exemple (Pi-hole) :
KEY: TZ → VALUE: America/Chicago
KEY: WEBPASSWORD → VALUE: BigBroDNSTest123!

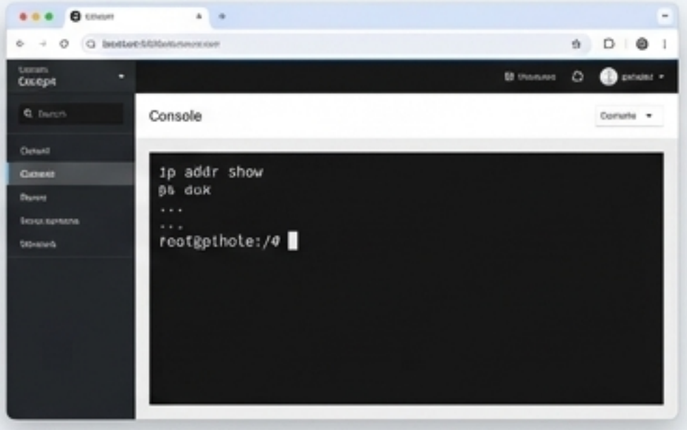
Au-delà de la Création : Inspection, Console et Réseau

Monitoring et Logs en Temps Réel



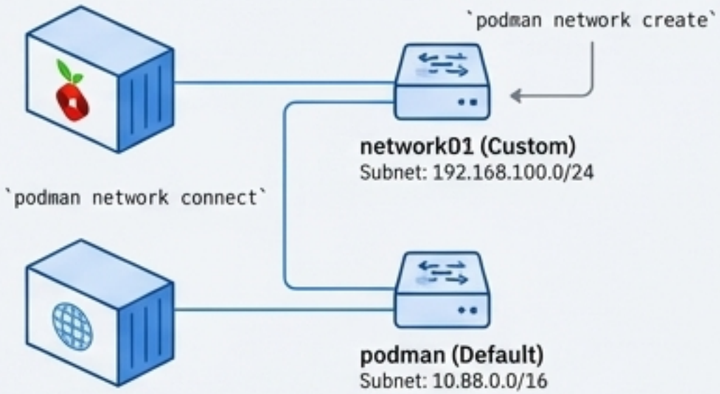
Visualisez la consommation CPU et mémoire de chaque conteneur et accédez aux logs en direct pour un débogage rapide.

Accès Console Direct



Plus besoin de `podman exec`. Obtenez un shell à l'intérieur de votre conteneur directement depuis votre navigateur.

Gestion du Réseau Podman



Créez et gérez des réseaux dédiés pour isoler vos applications. Cockpit vous permet de configurer des subnets personnalisés.

L’Épreuve du Terrain : Quand SELinux s’en Mêle

> **Nous avons installé un conteneur Syncthing, et en lisant les logs dans Cockpit, nous avons découvert que nous avions des problèmes avec SELinux qui refusait les permissions... Nous pourrions le désactiver, mais nous voulons apprendre à gérer un serveur de la manière la plus sécurisée possible.”**
>
> — Steve Daley (mowest), Fedora Discussion Forum

Analyse du Problème

- C’est l’un des obstacles les plus courants pour les nouveaux utilisateurs de Podman sur des systèmes comme Fedora ou RHEL.
- Le symptôme : un conteneur qui échoue au démarrage sans erreur évidente, souvent lié à l’accès aux volumes mappés.
- La tentation : `setenforce 0`. Une solution de facilité qui compromet la sécurité.

La Bonne Approche : Diagnostiquer, Pas Désactiver

La Tentation

~~setenforce 0~~

La Solution

Étape 1 : Le Diagnostic via Cockpit

Cockpit intègre une interface pour SELinux. Lorsqu'un conteneur est bloqué, une alerte est générée, expliquant la cause et suggérant la commande exacte pour créer une exception.

SELinux est en train de bloquer.

Commandes

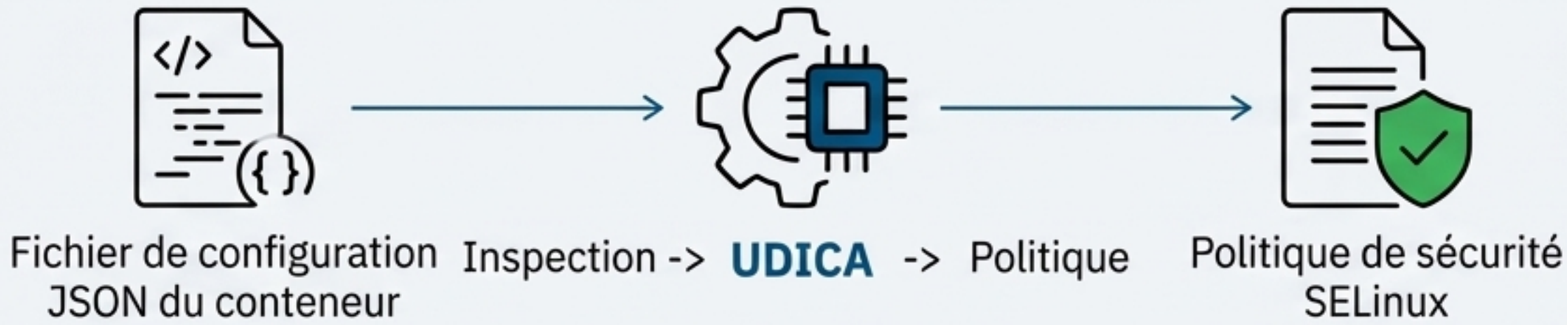
```
ausearch -m avc -ts recent ... | audit2allow -M my_container
semanage fcontext -a .../containers/ my_container.pp
```

Ressources Clés

- Le blog de **Dan Walsh** (expert SELinux & conteneurs).
- Le site web : **stopdisablingselinux.com**
- Le "**SELinux Coloring Book**" pour une compréhension conceptuelle.

SELinux n'est pas un ennemi, c'est un mécanisme de sécurité puissant.
L'écosystème Fedora fournit les outils pour le maîtriser.

La Maîtrise Proactive : Générer des Politiques SELinux avec UDICA



Qu'est-ce que UDICA ?

UDICA (User Defined Image Content Analysis) est un outil qui analyse la configuration d'un conteneur (basée sur son JSON d'inspection) et génère un profil de sécurité SELinux sur mesure.

Mentionné dans la documentation Red Hat Enterprise Linux 8 comme l'outil de choix pour créer des politiques SELinux pour les conteneurs.

Avantages

Précision
Crée une politique du "moindre privilège" en n'autorisant que ce qui est strictement nécessaire pour le fonctionnement du conteneur.

Automatisation
Idéal pour les pipelines CI/CD et le déploiement sécurisé à grande échelle.

Intégration
S'intègre parfaitement dans l'écosystème Podman et RHEL.

Avec des outils comme Cockpit pour le diagnostic et UDICA pour la politique, la gestion de la sécurité SELinux devient une partie intégrante et maîtrisée du cycle de vie des conteneurs.

Le Bon Outil pour le Bon Usage : Un Cadre de Décision

Scénario	Outil Privilégié	Justification
Serveurs Linux multi-utilisateurs	**Podman**	Architecture 'rootless' par défaut, isolation supérieure entre utilisateurs.
Pipelines CI/CD avec builds sécurisés	**Podman** (+ Buildah)	Permet les builds d'images en mode 'rootless', un gain majeur pour la sécurité.
Développement Desktop pour un cluster Kubernetes	**Podman Desktop**	Alignement natif avec les concepts de K8s (`pod`, `generate kube`). Pas de restrictions de licence.
Pipelines existants utilisant l'API Docker	**Docker**	Compatibilité maximale de l'API. (Note : Podman offre une API compatible, mais Docker reste la référence).
Charges de travail de conteneurs Windows	**Docker**	Écosystème historiquement plus mature pour les conteneurs natifs Windows.

Source des scénarios : Adapté de ‘Linux Journal, August 2025’.

Une Vision d’Ensemble : L’Écosystème des Outils Libres

Podman : Gérer et Exécuter.
Le runtime sans daemon pour les conteneurs et les pods.

Udica : Sécuriser.
Le générateur de politiques SELinux pour renforcer l'isolation.

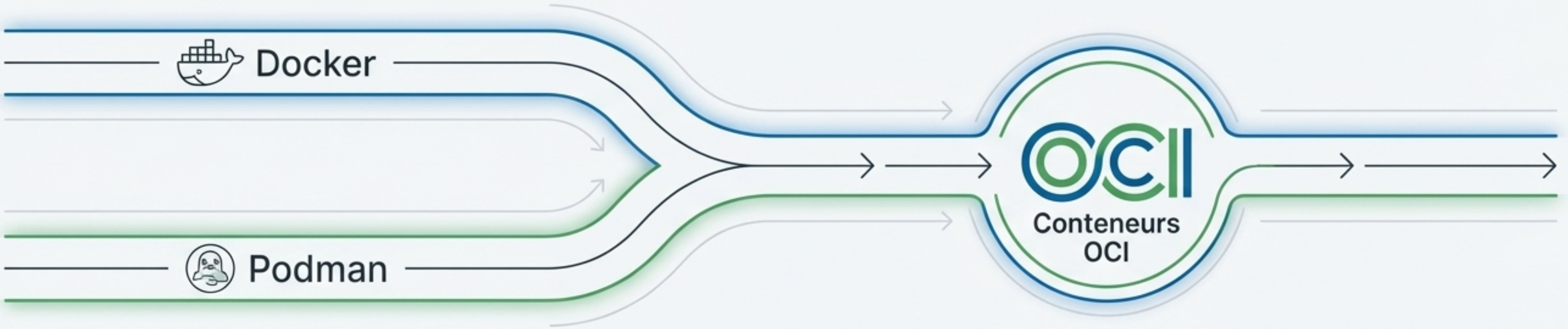
Buildah : Construire.
Un outil spécialisé dans la création d'images OCI, en mode rootless.

Skopeo : Inspecter et Transférer.
Permet de travailler avec des images sur des registres distants sans avoir à les tirer localement.

Cockpit : Administrer.
L'interface web unifiée pour la gestion du système et des conteneurs.

La force de Podman réside dans son intégration au sein dune boîte à outils modulaire, ouverte et spécialisée, offrant flexibilité et sécurité à chaque étape du cycle de vie d'un conteneur.

Convergence, Pas Remplacement



Synthèse

- Docker et Podman représentent deux approches matures et puissantes.
- **Docker** excelle en compatibilité et en facilité d'embarquement pour les débutants.
- **Podman** offre une sécurité avancée par défaut et une approche nativement alignée sur Kubernetes.

La Perspective pour 2025 et au-delà

- La rivalité s'estompe au profit du **"bon outil pour le bon travail"**.
- Grâce aux standards OCI, les images sont portables et les workflows peuvent rester cohérents.
- L'avenir est à la flexibilité, où les développeurs et administrateurs peuvent mélanger et assortir les outils en fonction des besoins précis de chaque projet.

Le véritable gagnant de cette évolution, c'est l'administrateur moderne, qui dispose désormais d'un choix éclairé pour construire des systèmes plus résilients et sécurisés.