Engineer

Podman

Moteur de conteneurs Linux sans démon central

Ce que c’est

Podman est un moteur de conteneurs pour Linux qui construit, télécharge et exécute les mêmes images OCI que Docker, sans serveur central intermédiaire. Chaque commande `podman` accomplit son travail dans son propre processus, puis se termine. Un conteneur lancé continue de fonctionner sous un petit processus de surveillance appelé conmon, lui-même enfant de ce qui l’a lancé : un shell, un script ou systemd. Le projet se décrit comme sans démon central, open source et natif de Linux, et indique que la plupart des utilisateurs peuvent faire de `docker` un alias de `podman` et continuer à travailler.

Le nom abrège pod manager, gestionnaire de pods. Un pod, notion empruntée à Kubernetes, regroupe des conteneurs qui partagent un espace de noms réseau et peuvent donc se joindre sur localhost. Podman permet de créer les pods manuellement. `podman kube play` lance des conteneurs et des pods à partir d’un fichier YAML Kubernetes : c’est donc un outil raisonnable pour essayer un manifeste avant de le soumettre à un cluster.

Deux fonctionnalités le distinguent au quotidien. La première est le fonctionnement sans privilèges root, dit rootless. Un utilisateur ordinaire peut exécuter des conteneurs : les espaces de noms utilisateur du noyau associent cet utilisateur à l’UID 0 dans le conteneur, et sa plage d’identifiants subordonnés, définie dans `/etc/subuid` et `/etc/subgid`, aux autres UID. Le réseau des conteneurs rootless passe par pasta, une pile réseau en espace utilisateur, puisqu’un utilisateur non privilégié ne peut pas réécrire le filtre de paquets de l’hôte.

La seconde est Quadlet. Un fichier Quadlet décrit brièvement, de manière déclarative, un conteneur, un volume, un réseau, un pod ou la construction d’une image. Il reprend la syntaxe ini d’une unité systemd et se place dans

`/etc/containers/systemd/` pour le système ou `~/.config/containers/systemd/` pour un utilisateur. Au démarrage et à chaque `systemctl daemon-reload`, un générateur systemd transforme `web.container` en une unité `web.service` ordinaire. L'ancienne commande `podman generate systemd` fonctionne toujours, mais le projet la déclare obsolète au profit de Quadlet.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Podman ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Un logiciel conditionné pour se comporter à l'identique sur un portable, un serveur de test et en production, et monter en charge lors des semaines chargées.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 20 applications Docker sur Google Cloud et AWS — 20 applications conteneurisées déployées et surveillées

20 applications Docker sur Google Cloud et AWS Contactez-nous →

Pourquoi c'est important

Podman pose la question de la responsabilité de l'arbre des processus. Docker gère les conteneurs par l'intermédiaire de `dockerd`, avec ses propres politiques de redémarrage, sa journalisation et ses contrôles de l'état des conteneurs. Sur un serveur déjà géré par systemd, cela ajoute un second superviseur à côté du premier, sans vocabulaire commun entre les deux.

Avec Podman et Quadlet, systemd est le seul superviseur. Un conteneur devient un service comme un autre : `systemctl status` l'affiche, journald conserve sa sortie, `After=` et `Requires=` fixent son ordre par rapport à la base dont il dépend. `Restart=` et les limites de cgroup déjà comprises par systemd s'appliquent sans changement. Une personne qui sait lire un fichier d'unité sait donc déjà lire la configuration de la couche de conteneurs.

La sécurité entre aussi en jeu. Le démon Docker fonctionne en root, sauf si son mode rootless a été choisi. Le guide de sécurité de Docker lui-même recommande de n'en autoriser le contrôle qu'aux utilisateurs de confiance, car on peut donner à un conteneur n'importe quel répertoire de l'hôte. Podman ne maintient aucun démon privilégié en fonctionnement entre deux commandes. Une configuration rootless limite encore les privilèges : un processus qui s'échappe du conteneur se retrouve avec les droits d'un utilisateur non privilégié.

Le choix entre Docker et Podman dépend donc moins d'une compétition de fonctionnalités que de l'adéquation à l'environnement. Docker est plus fluide sur un portable de développement équipé de Docker Desktop, pour les équipes organisées autour de Compose et pour les outils qui attendent un socket Docker, même si Podman peut fournir une API compatible à la demande. Podman convient généralement mieux aux serveurs où systemd fait déjà autorité et où un démon qui réécrit le pare-feu constitue une contrainte plutôt qu'une commodité.

À quoi cela ressemble en pratique

Podman assure la couche de conteneurs dans deux projets récents. Auparavant, il servait d'outil de diagnostic pour un parc construit sur Docker.

Sur un hôte, les conteneurs tournent comme des unités Quadlet sous systemd, avec Podman à la place de Docker. La raison est le pare-feu : par défaut, Docker publie les ports des conteneurs avant les règles installées par le rôle de durcissement, alors que ces conteneurs utilisent le réseau de l'hôte sans aucun port publié. Les unités fonctionnent au niveau système plutôt qu'en rootless. La page présente ce compromis : une automatisation plus simple, au prix de l'isolation plus stricte qu'offrirait le mode rootless. Elle indique aussi vers quoi évoluer si le risque de sortie de conteneur prenait le pas sur la simplicité de l'automatisation. Le déploiement est séparé du provisionnement. Le compte root prépare une fois le chemin de déploiement. Chaque nouvelle version passe ensuite par un script fixe qu'un utilisateur non privilégié est autorisé à lancer grâce à une règle de portée limitée. Ce script ne fait que deux choses : construire l'image et redémarrer le service.

La même organisation tient sur très peu de matériel. Un hôte à un seul cœur doté de 512 Mo de mémoire exécute les points d'accès TLS publics de l'entreprise, sa forge Git et son service onion, avec une automatisation des sauvegardes implémentée séparément. Sa couche de conteneurs est composée d'unités Quadlet sous le même superviseur que le reste, plutôt que d'un démon avec son propre superviseur. Dans les échantillons enregistrés sur une journée, la mémoire disponible n'est jamais descendue sous environ 310 Mo, sur les 464 Mo utilisables.

Auparavant, pour vingt applications empaquetées avec Docker et réparties entre Google Cloud et AWS, le diagnostic passait par Podman, et Kubernetes servait au volet orchestration. Les images étaient identiques ; le choix portait sur l'outil qui les entourait.

Où les problèmes surviennent

La plupart des difficultés avec Podman viennent de l'attente d'un comportement identique à Docker ou à une simple unité systemd, alors qu'il tient un peu des deux.

Modifier un fichier Quadlet ne change rien. Le générateur s'exécute au démarrage et lors de `systemctl daemon-reload` : une modification sur disque reste donc invisible jusque-là. Ensuite, `systemctl enable` refuse le service généré, car systemd le considère comme transitoire. Le démarrage automatique se déclare dans la section `[Install]` du fichier lui-même :

```
# /etc/containers/systemd/web.container
[Unit]
Description=Web front end

[Container]
Image=docker.io/library/nginx:1.27
Network=host

[Service]
Restart=always

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

Après l'enregistrement, lancez `systemctl daemon-reload` , puis `systemctl start web.service` .

Un service rootless s'arrête à la déconnexion de son propriétaire. Les unités utilisateur vivent dans le gestionnaire systemd de cet utilisateur, qui s'arrête avec la dernière session. `loginctl enable-linger` pour cet utilisateur maintient le gestionnaire actif dès le démarrage, ce dont tout service rootless censé survivre à un redémarrage a besoin.

Un serveur web rootless demande le port 80 et se le voit refuser. Les ports inférieurs à 1024 exigent root ou la capacité `CAP_NET_BIND_SERVICE` , et cette règle s'applique à l'utilisateur sous lequel Podman s'exécute. Publiez un port élevé avec un proxy devant, ou abaissez volontairement le réglage `ip_unprivileged_port_start` du noyau en sachant qu'il s'applique à tous les utilisateurs de l'hôte.

Les fichiers d'un montage bind appartiennent à un inconnu. En mode rootless, l'UID de l'utilisateur correspond à root dans le conteneur, et chaque autre UID de conteneur est associé à la plage subordonnée. Un processus exécuté avec l'UID 999 à l'intérieur écrit des fichiers que l'hôte voit comme appartenant à un identifiant tel que 100998. `podman unshare` ouvre un shell dans la même correspondance d'identifiants, où `chown` se comporte comme le conteneur l'attend. L'option `U` d'un montage adapte automatiquement le propriétaire aux identifiants attendus par le conteneur.

Pour aller plus loin

- Qu'est-ce que Podman ? — Documentation du projet, à commencer par sa définition de l'outil.
- podman-systemd.unit — Référence Quadlet : types de fichiers, chemins de recherche et toutes les clés.
- Configuration et utilisation de base de Podman en environnement rootless — Identifiants subordonnés, espaces de noms utilisateur et réseau pasta.
- podman-unshare — Entrée dans l'espace de noms utilisateur rootless pour examiner ou corriger les propriétaires.
- podman-generate-systemd — Ancienne méthode de génération des unités, désormais obsolète au profit de Quadlet.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que Podman ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/podman/