# Podman : et si vous changiez de moteur de conteneurs ?

BAPTISTE VÉRÉ and ANTOINE MARTINELLI

CCS Concepts: • **Security and privacy → Virtualization and security**.

Additional Key Words and Phrases: virtualization, containers, information security

Contents



Authors' address: Baptiste Véré; Antoine Martinelli.

# 1 LES CONTENEURS LXC

## 1.1 Principes

Podman est un moteur de conteneurs LXC [5], c'est-à-dire qu'il est simplement une construction permettant de lancer ainsi que de gérer des conteneurs LXC. Le conteneur LXC ou LinuX Containers est un système de virtualisation utilisant l'isolation comme méthode de cloisonnement pour le système d'exploitation. Cela permet donc de créer des environnements Linux isolés des uns des autres mais partageant le même noyau. Ces conteneurs sont issus des fonctionnalités présentes dans le noyau Linux. La plus importante est la présence des cgroups qui permettent de limiter, compter et isoler les ressources telles que la mémoire, le processeur, etc.

## 1.2 Les conteneurs, des machines virtuelles ?

Les conteneurs ne sont pas des machines virtuelles. En effet, contrairement à une machine virtuelle, chaque conteneur utilise le même noyau Linux, ainsi que les mêmes bibliothèques partagées. En somme, contrairement à la virtualisation, aucun élément physique n'est réellement émulé. Cela permet notamment de meilleurs performances, mais induit une moins bonne isolation par rapport au système hôte et entre les conteneurs [3].

# 2 *PODMAN*, L'OUTIL

Considérant que Podman n'est ni un ni un client ni un serveur, et qu'il agit directement sur les conteneurs, on peut dire que c'est un utilitaire. Cet utilitaire ne fait pas les appels systèmes nécessaires à la gestion des conteneurs par lui-même, il passe par Buildah pour la construction des images et Runc pour la gestion des conteneurs.

## 2.1 Une simple copie de Docker ?

Podman n'est-il donc qu'un Docker-bis ?

*2.1.1 Similarités.* La communication la plus classique (et l'idée répandue) autour de Podman est la suivante : Podman serait comme Docker. Tellement similaire à Docker qu'un simple ‘alias docker=podman‘ permettrait une utilisation transparente pour un utilisateur habitué à la syntaxe des commandes Docker [5]. Cela à l'avantage de faciliter grandement une migration.

*2.1.2 Différences.* L'objectif affiché par les développeurs de Podman n'est pas de faire de Podman un clone de Docker [5]. Aussi, les différences suivantes sont voulues et permettent de répondre à d'autres besoins (ou lacunes) de Docker. Podman est dit "root less" car contrairement à Docker il peut tourner avec des droits utilisateurs standards, et non uniquement root. Cela permet de résoudre plusieurs problèmes de sécurité que peut rencontrer l'utilisation de Docker comme nous le verrons dans la partie 3. Podman est aussi dit "Daemon less", contrairement à Docker il n'y a pas de Deamon qui gère tout : construire, télécharger, gérer les images ainsi que lancer et gérer les conteneurs. Docker a été pensé comme un programme qui s'exécute en arrière-plan sous la forme d'un client qui se connecte au Daemon ce qui va permettre un contrôle par l'utilisateur. Podman est lui construit différemment comme un simple programme s'exécutant ponctuellement qui va ensuite déléguer son travail à d'autres logiciels comme Buildah pour la construction d'image ou RunC pour le lancement d'un conteneur [2, 4].

## 2.2 Podman, l'écosystème

- Buildah : Dans "l'écosystème" Podman, Buildah est l'outil se chargeant de la construction des images ainsi que de leur gestion [2]. Cet outil ne sert que pour cette tâche et ne gère pas les conteneurs. Mais il possède d'autres fonctionnalités comme prendre en charge la création d'images à l'aide de Dockerfile mais permet également de simplement créer un répertoire OCland pour y mettre du contenu qui sera utilisé pour la création d'une image OCI et de le pousser vers n'importe quel registre de conteneurs. Ces conteneurs

ou images peuvent ensuite être utilisés par Docker, Podman ... ou n'importe quel conteneur basé sur les normes OCI (Open Container Initiative) [3].

- RunC : RunC est l'outil servant à la création et la gestion du cycle de vie des conteneurs. Il ne s'agit pas d'un programme spécifique à Podman et peut tout comme Buildah être utilisé de manière indépendante. [6].
- Stockage : Si Podman est utilisé par un utilisateur root, les conteneurs seront stockés dans le fichier /etc/containers/storage.conf. Si il est utilisé par un autre utilisateur, un espace de stockage spécifique sera situé dans /.local/share/containers. Cela permet d'isoler les espaces de stockages des différents utilisateurs. Ce fonctionnement comporte des avantages certains pour l'aspect sécurité, mais peut induire une duplication éventuelle dans le cas où plusieurs utilisateurs utilisent la même image.
- Sécurité des conteneurs : Le fait de pouvoir faire tourner le moteur de conteneurs sans privilèges root permet de s'assurer que même si un attaquant arrive à obtenir un accès root dans le conteneur, il ne lui sera pas possible de prendre le contrôle complet de la machine mais simplement d'effectuer ce que peut faire l'utilisateur ayant lancé le conteneur. De plus, comme Podman utilise un modèle fork/execution contrairement à Docker qui utilise un modèle client/serveur, il n'est pas possible de compromettre le serveur (qui n'existe pas) ce qui réduit la surface d'attaque.

## 3 LES PODS

### 3.1 Principes

Podman ne serait pas Podman sans le concept de pod. Un pod est un groupe de conteneurs travaillant dans un environemment similaire. Par exemple, deux conteneurs dans un pod partagent, par défaut, un même réseau. Le pod est un concept apporté en premier lieu par Kubernetes [4]. La définition du pod dans Podman est donc très similaire à celle de Kubernetes. En ce sens, Podman est plus facilement intégrable à une architecture Kubernetes. C'est un parti pris évident des développeurs qui peut donc être un avantage si on utilise cette technologie. Tout pod contient (par défaut) au minimum un conteneur : le conteneur infra. Ce conteneur occupant le PID 1 permet de coordonner le namespace des conteneurs du pod, ainsi que de s'assurer du nettoyage des processus zombies [4, 6]. Il est ensuite possible d'ajouter des conteneurs ayant une utilité dans le pod ainsi créé.

### 3.2 Orchestration

Un outil très utilisé avec Docker est docker-compose. Il permet de lancer un ensemble de conteneurs à partir d'une définition dans un unique fichier yaml [4]. Bien que de tels outils non officiels existent avec Podman, ce n'est pas le parti pris officiel. En revanche, Podman implémente nativement une manière de lancer un noeud Kubernetes. Il est possible de générer un fichier de configuration (aussi en yaml) avec une simple commande à partir d'un pod en cours d'exécution, ce qui permet une reproductibilité facile d'un environnement.

## 4 LE MONDE DES CONTENEURS

### 4.1 Docker, une position de leader de plus en plus érodée

Docker domine largement dans l'utilisation des conteneurs cependant les projets de plus en plus nombreux autour des conteneurs tendent à affaiblir sa domination. En 2017, Docker représentait 99% des conteneurs utilisés mais depuis ce nombre est en constante diminution avec une part de 83% en 2018 et 79% en 2019 [1, 7]. En effet, le marché ne fait que se diversifier et d'évoluer par l'apparition de nombreux autres environnements d'exécution de conteneurs, tels que CoreOS RKT, Mesos, LXC.

### 4.2 Et Podman dans tout ça ?

Podman fait partie des conteneurs LXC et représente moins de 1% des conteneurs employés. Il est en effet fortement concurrencé par CoreOS RKT qui représente 12% des conteneurs. Un autre concurrent Mesos représente lui les derniers 4% des conteneurs utilisés [1] . Podman a fait ses premiers débuts fin 2017 même si il est tout à fait utilisable dans son état, cela reste un conteneur qui est toujours entrain de mûrir avec l'arrivée le mois prochain de la version 2.0 et une nouvelle API [5].

## 5 CONCLUSION

Docker a grandement contribué à l'essor des conteneurs LXC, à tel point qu'on entend bien souvent parler de "conteneurs Docker" et non simplement de conteneurs Linux/LXC. Face à cette hégémonie ainsi qu'aux problèmes évidents que pose l'écosystème Docker (notamment en terme de sécurité) Podman fait figure d'alternative efficace. A l'inverse de Docker qui s'est plus centré autour du développement et de la facilité d'utilisation, Podman apporte des fonctionnalités axées vers l'opérationnel. En plus de régler des problèmes fonctionnels, cet outil a tendance à mieux respecter la philosophie Unix avec la séparation de ses composants, la syntaxe proche de celle de Docker et l'utilisation des LXC qui simplifient grandement la migration.

## REFERENCES

[1] [n.d.]. 2018 Docker usage report. https://sysdig.com/blog/2018-docker-usage-report/
[2] [n.d.]. Buildah official website. https://buildah.io/
[3] [n.d.]. Open Containers Initiative official website. https://www.opencontainers.org/
[4] [n.d.]. Podman official documentation. http://docs.podman.io/en/latest/
[5] [n.d.]. Podman official website. https://podman.io/
[6] [n.d.]. runc official code repository. https://github.com/opencontainers/runc
[7] [n.d.]. Sysdig 2019. https://sysdig.com/blog/sysdig-2019-container-usage-report/

## A CADRE DE CET ARTICLE

Cette courte présentation a été réalisée dans le cadre de la première année de cycle ingénieur en Cyberdéfense à l'ENSIBS (Cours de virtualisation).