Engineer

Docker

livrer un programme avec tout ce dont il a besoin

Ce que c’est

Docker est un ensemble d’outils qui rassemble un programme, ses bibliothèques, son environnement d’exécution du langage et ses fichiers dans un artefact appelé image, puis l’exécute comme un processus isolé appelé conteneur. L’image est construite une fois selon une recette, le Dockerfile. La même image démarre ensuite sur un portable, sur un serveur de construction et en production. Toute la promesse tient en une phrase : ce qui a été testé est, octet pour octet, ce qui s’exécute.

Quatre termes constituent l’essentiel du vocabulaire. Un Dockerfile est un fichier texte d’instructions comme `FROM`, `COPY` et `RUN`. Une image est un modèle en lecture seule construit à partir de ces instructions, stocké en couches superposées, une par instruction. Après un petit changement, la reconstruction ne refait ainsi que les couches modifiées. Un conteneur est une instance active d’une image avec une fine couche inscriptible au-dessus. Un registre, comme Docker Hub ou celui d’un fournisseur cloud, est l’endroit où les images sont publiées et téléchargées.

Sous la ligne de commande se trouvent un client et un serveur. La commande `docker` est un client ; elle envoie des requêtes par une API REST, généralement via un socket Unix, à un démon permanent nommé `dockerd`. Ce démon gère images, conteneurs, réseaux et volumes. Pour le cycle de vie du conteneur lui-même, il délègue à containerd, qui utilise runc sans configuration contraire. Ce petit programme demande au noyau Linux des espaces de noms et des groupes de contrôle, puis y démarre le processus. Un conteneur est donc un processus ordinaire sur le noyau de l’hôte, avec une vue restreinte du système. Ce n’est pas une machine virtuelle. Sous macOS ou Windows, Docker Desktop exécute une petite machine virtuelle Linux pour disposer d’un noyau Linux partagé.

Le format d'image a dépassé l'entreprise qui l'avait créé. En juin 2015, Docker, CoreOS et d'autres ont lancé l'Open Container Initiative. Docker lui a donné son format d'image et runc. Les spécifications qui en résultent permettent à une image construite avec Docker de fonctionner sans changement sous Podman, containerd ou un service cloud géré.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Docker ? » 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

La question utile n'est jamais « Docker ou non » dans l'abstrait. Il faut savoir si une charge de travail justifie une image, ou si un simple service sur l'hôte ferait le même travail avec moins d'éléments à gérer.

Une image se justifie lorsque le programme traîne un environnement avec lui. Une application Python ou R exigeant un ensemble exact de paquets, un service ancien fixé à un vieux moteur d'exécution, deux programmes demandant des versions incompatibles d'une même bibliothèque : chacun est pénible à reproduire à la main et très simple à reproduire depuis une image. Revenir en arrière consiste à démarrer l'image précédente. Une nouvelle personne dans l'équipe de développement exécute une commande au lieu de suivre une page de notes d'installation. La comparaison avec une machine virtuelle présente l'autre face du même arbitrage : les conteneurs démarrent en quelques instants et partagent un noyau. Des dizaines tiennent donc sur un hôte qui ne recevrait que quelques machines virtuelles, au prix d'une isolation plus faible.

La contrepartie est une véritable infrastructure. Par défaut, le démon s'exécute comme root et gère le réseau ; Docker propose aussi un mode sans privilèges root. Il faut un registre à exploiter ou à payer, des images de base qui reçoivent des correctifs de sécurité et qu'il faut reconstruire pour en bénéficier, et une

couche de stockage dont les règles diffèrent de celles de l'hôte. Un unique binaire statique sans dépendances n'y gagne presque rien ; une unité systemd confinée le fait fonctionner tout aussi bien.

La différence avec Podman compte aussi. Tous deux exécutent les mêmes images et acceptent la plupart des mêmes commandes. Docker utilise un démon, privilégié par défaut. Podman démarre directement les conteneurs comme enfants de la commande ou de systemd. Sur un portable, la différence est faible. Sur un serveur dont le pare-feu et la supervision des processus ont déjà été conçus, elle peut décider du choix.

À quoi cela ressemble en pratique

Docker apparaît dans plusieurs travaux de ce site, et l'un des projets l'écarte délibérément.

Entre 2022 et 2024, vingt applications conteneurisées fonctionnaient sur Google Cloud et AWS, dont des applications d'analyse écrites en R, pour lesquelles reproduire l'ensemble exact des paquets est la partie difficile. Les vingt ont été déployées et maintenues avec une configuration et une supervision cohérentes. Elles sont ainsi restées stables et reproductibles, et les problèmes étaient plus rapides à diagnostiquer entre les clouds.

À plus grande échelle, cent applications Docker résidaient sur des hôtes Ubuntu derrière un ensemble d'Apache et de Nginx. Chacune avait besoin d'un certificat TLS valide. Un processus fondé sur ACME a automatisé l'émission, le renouvellement et le déploiement de ces certificats. Les cent conservaient donc des certificats valides, et le travail manuel sur les certificats a disparu.

Le troisième cas va dans l'autre sens. Sur un hôte dont les règles de pare-feu proviennent d'un rôle de durcissement, la couche de conteneurs a utilisé Podman et Quadlet plutôt que Docker. Le réseau de Docker, dans sa configuration initiale, réécrit en effet les règles de filtrage des paquets du noyau et publie les ports des conteneurs en amont des règles du pare-feu. La raison accompagne le choix dans la documentation. La prochaine personne à intervenir sait donc que la configuration initiale a été évaluée, pas oubliée.

Où les problèmes surviennent

Docker facilite le lancement du premier conteneur. Les problèmes arrivent ensuite, généralement sous l'une de ces quatre formes.

Un port bloqué par le pare-feu est accessible depuis Internet. Sans configuration contraire, un port publié écoute sur toutes les adresses de l'hôte. Les règles de filtrage écrites par Docker envoient ce trafic au conteneur avant qu'un outil comme ufw le voie. Le pare-feu annonce un port fermé alors qu'il est ouvert. Si seul un proxy inverse local a besoin du conteneur, publiez sur l'adresse de bouclage :

```
# reachable from anywhere, whatever ufw reports
docker run -d -p 8080:80 nginx:1.27

# reachable only from the host itself
docker run -d -p 127.0.0.1:8080:80 nginx:1.27
```

Les données disparaissent avec le conteneur. Tout ce qui est écrit dans sa propre couche inscriptible est supprimé avec lui, exactement ce que fait une mise à niveau. Les bases, les fichiers téléversés et tout ce qui doit survivre vont sur un volume nommé ou un montage bind. Ce volume a ensuite besoin de sa propre sauvegarde.

Une reconstruction produit autre chose que la semaine passée. Une étiquette comme `python:3` ou `latest` est un pointeur mouvant. Le même Dockerfile peut donc construire une autre image un autre jour. Référencez les images de base par une version précise, ou par leur empreinte si la construction doit être exactement reproductible. Reconstruisez délibérément lorsque l'image de base publie des correctifs de sécurité.

Un identifiant secret est livré dans l'image. Les arguments de construction et les variables d'environnement persistent dans l'image finale. Un fichier copié dans une couche y reste même si une instruction ultérieure le supprime. Toute personne capable de télécharger l'image peut le lire. Transmettez les secrets à la construction par un montage secret qui ne les expose qu'à une étape `RUN`, sans les écrire dans aucune couche.

Pour aller plus loin

- Documentation Docker — les manuels, guides et références officiels du moteur, de la ligne de commande et de Compose.
- Présentation de Docker — client, démon, images et conteneurs expliqués par le projet.
- Publication et correspondance des ports — pourquoi les ports publiés sont ouverts à l'extérieur sans configuration contraire et comment les lier à localhost.
- Filtrage des paquets et pare-feu — comment Docker écrit des règles iptables ou nftables et où ufw cesse de voir le trafic.
- Secrets de construction — transmettre des identifiants secrets à une construction sans les laisser dans l'image.
- Spécification du format d'image OCI — la norme ouverte qui permet à une même image de fonctionner sous Docker, Podman et containerd.



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 Docker ? » 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/docker/