

Les images





Une image n'est pas un conteneur

Ce que le TP a montré

- La page d'accueil modifiée disparaît avec le conteneur
- `docker container prune` n'a touché à aucune image
- Les images survivent, les conteneurs passent

Deux objets distincts

- **Image** : un modèle figé, en lecture seule
- **Conteneur** : une exécution de ce modèle
- Au démarrage, Docker ajoute une **couche inscriptible** par-dessus

Une image, plusieurs conteneurs

- Dix conteneurs peuvent partager la même image
- L'image n'est **jamaïs** modifiée par un conteneur
- Chaque conteneur écrit dans sa propre couche



Comment le conteneur écrit

Comment ça marche ?

Florian Baune - 2026

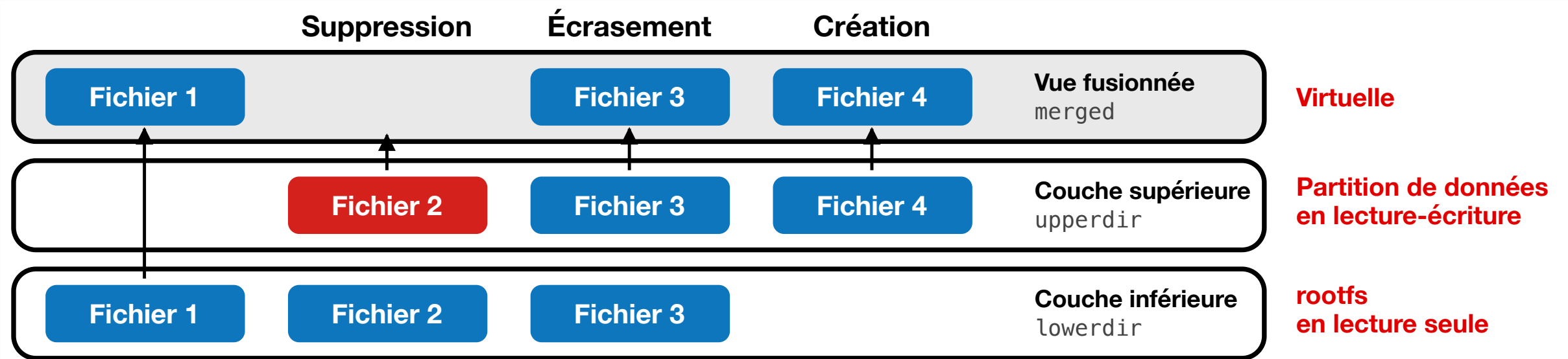
Le problème

- Une image est en lecture seule, par construction
- Un conteneur, lui, écrit en permanence : fichiers temporaires, journaux, données
- Il faut donc lui permettre d'écrire **sans jamais toucher à l'image**

La réponse : OverlayFS

- Le noyau Linux sait fusionner plusieurs répertoires en un seul
- En dessous, l'image, en lecture seule : `lowerdir`
- Au-dessus, une couche vide, en lecture-écriture : `upperdir`
- La fusion des deux : `merged`

“ `merged` est la seule vue que le conteneur connaisse.
Il ignore qu'il y a deux étages. ”



Lire, et créer

- **Lire** un fichier que rien ne masque : il est lu directement dans l'image, sans copie
- **Créer** un fichier : il n'existe que dans la couche inscriptible
- L'image ne bouge pas, dans les deux cas

Écraser : copy-on-write

- Modifier un fichier de l'image est impossible : elle est en lecture seule
- Il est donc d'abord **copié** dans la couche inscriptible, puis modifié

“ /!\ Un fichier de 2 Go modifié d'un octet, c'est 2 Go recopiés. ”

Supprimer : le whiteout

- Supprimer un fichier de l'image est impossible, pour la même raison
- La couche inscriptible pose un **marqueur de suppression**, le whiteout
- Le fichier disparaît de la vue, pas de l'image

Voir la couche inscriptible

- `docker diff` liste ce qu'un conteneur a changé depuis son démarrage
- `A` ajouté, `C` modifié, `D` supprimé
- `docker ps -s` donne son poids, à côté de celui de l'image

“ Un fichier supprimé apparaît en `D` : c'est le whiteout, vu de l'extérieur. ”

Ce que ça change

- L'image n'est **jamais** modifiée par un conteneur
- Dix conteneurs, une seule image, dix couches inscriptibles
- La couche inscriptible meurt avec le conteneur

“ C'est pour ça que la page d'accueil que vous aviez modifiée avait disparu. ”



L'image: un empilement de couche

Un bloc ? Non, une pile

- Jusqu'ici, « l'image » tenait en un seul étage
- En réalité, cet étage est **lui-même une pile ordonnée** de couches
- Toutes en lecture seule, sans exception

“ Le partage lecture seule / lecture-écriture reste le même. Ce qui change, c'est le nombre d'étages en dessous. ”

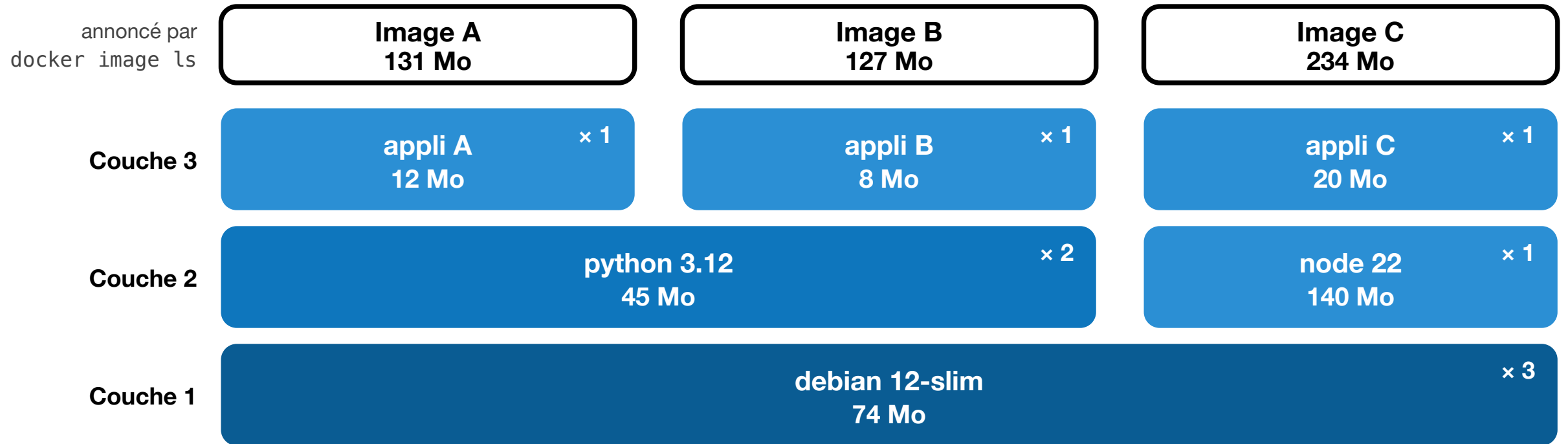
Une couche, c'est quoi

- Un **jeu de différences** : des fichiers ajoutés, modifiés, supprimés
- Produite par une étape de la construction de l'image
- Identifiée par une empreinte, donc reconnaissable d'une image à l'autre

Le même mécanisme, un étage plus bas

- Entre deux couches d'image : même superposition, même whiteout
- Un fichier ajouté puis supprimé pendant la construction **reste** dans la couche du dessous
- La taille de l'image ne diminue jamais

Les couches d'une image



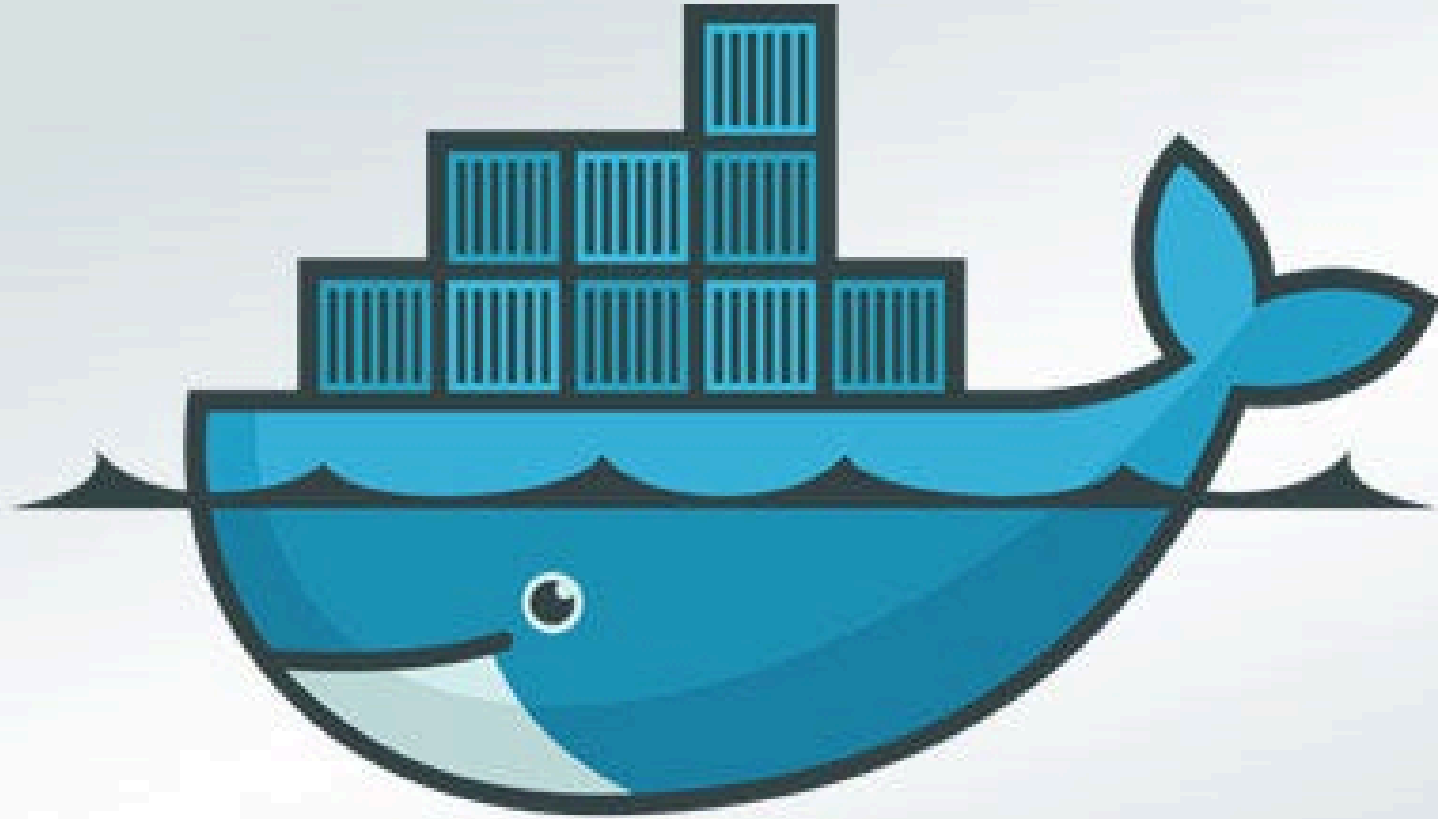
× N : nombre d'images qui s'appuient sur cette couche. Une couche n'est stockée qu'une fois.

Combien ça pèse vraiment ?

- `docker image ls` annonce **131 + 127 + 234 Mo**
- Mais une couche partagée n'est stockée **qu'une seule fois**
- Combien ces trois images occupent-elles réellement sur le disque ?

Ce que le partage permet

- Deux images bâties sur la même base ne la stockent **qu'une fois**
- `pull` ne télécharge que les couches absentes de votre machine
- `push` n'envoie que celles que le registry n'a pas
- `docker system df -v` donne, pour chaque image, sa part **unique**



docker HUB

Les registries...

- Un registry ne stocke pas des images, mais des **couches** et des manifestes
- `push` n'envoie que les couches qu'il n'a pas déjà
- `pull` ne récupère que celles qui manquent localement

Où vivent les images

- **Docker Hub** : le registry par défaut, le plus peuplé
- **GHCR** : le registry de GitHub, adossé aux dépôts
- **Registries privés** : Harbor, GitLab, `registry` auto-hébergé, offres cloud

Sur Docker Hub, tout n'est pas égal

- **Officielles** : maintenues avec Docker, dans `library`
- **Éditeurs vérifiés** : publiées par l'éditeur du logiciel
- **Communautaires** : tout le reste, sans aucune garantie

“ N'importe qui peut publier n'importe quoi sous n'importe quel nom. ”

Deux frictions bien réelles

- `docker login` est nécessaire pour les dépôts privés et pour pousser
- Les `pull` anonymes sont **limités en nombre** sur Docker Hub

“ En salle, trente machines derrière une seule IP publique épuisent vite le quota. ”



Nommer une image

Nommer une image

Florian Baune - 2026

Une référence complète

```
mcr.microsoft.com/dotnet/aspnet:9.0
```

- Quatre éléments, séparés par des `/` et un `:`
- Registry, espace de noms, dépôt, tag

Les quatre éléments

- `mcr.microsoft.com` — le **registry**, le serveur qui héberge
- `dotnet` — l'**espace de noms**, qui publie l'image
- `aspnet` — le **dépôt**, le logiciel lui-même
- `9.0` — le **tag**, la version voulue

Et pour **nginx**, alors ?

```
docker run nginx
```

- Ni registry, ni espace de noms, ni tag
- Un seul mot, là où il en fallait quatre
- Docker a pourtant trouvé l'image. Où est-il allé la chercher ?

Trois valeurs par défaut

- Registry absent → `docker.io`
- Espace de noms absent → `library`, réservé aux images officielles
- Tag absent → `latest`

“

Vous tapez `nginx`, Docker lit
`docker.io/library/nginx:latest`.

Trois raccourcis commodes, mais attention aux malentendus.

”

latest n'est pas « la dernière »

- C'est un tag comme un autre, juste celui pris par défaut
- Rien n'oblige un éditeur à le mettre à jour
- Rien ne garantit qu'il existe
- Deux **pull** à six mois d'écart peuvent donner deux images différentes

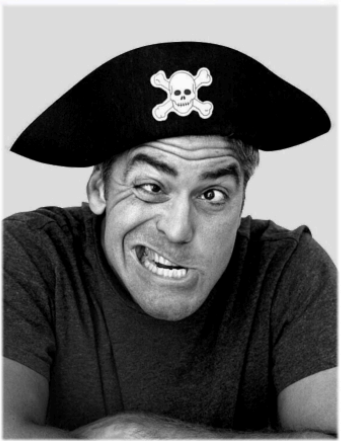
Tag ou digest

- **Tag** : une étiquette, déplaçable à volonté
- **Digest** : l'empreinte `sha256` du contenu, immuable
- Un tag peut changer de cible, un digest jamais

Viser une image précise

- Par tag précis : `nginx:1.27.3`
- Par empreinte : `nginx@sha256:...`
- Seule la seconde forme garantit le **bit à bit**

Choisir son image de base



Photomathon - © copyright

Choisir son image de base

Florian Baune - 2026

Quelques exemples...

- **Distribution complète** : `debian` , `ubuntu` , `fedora`
- **Variante `slim`** : la même, sans la documentation ni l'accessoire
- **`alpine`** : une distribution taillée pour la petitesse
- **`distroless`** : l'application et ses dépendances, rien d'autre
- **`scratch`** : l'image vide, littéralement

Le compromis

Plus petite

Moins de surface d'attaque

Téléchargement rapide

Plus complète

Débogage confortable

Outils déjà là

“ L'écart de taille entre ces familles, vous allez le mesurer en TP. ”

distroless et scratch

- Pas de shell, pas de gestionnaire de paquets
- Donc : pas de `docker exec sh`, pas d' `apt install` pour déboguer
- Très peu de surface d'attaque, très peu de confort

“ Excellentes en production, pénibles pendant l'apprentissage. ”

Une image, plusieurs architectures



Un nom, plusieurs images

- `amd64` sur un PC, `arm64` sur un Mac récent ou un Raspberry Pi
- Un même nom d'image peut regrouper plusieurs variantes
- Docker choisit **tout seul** celle qui correspond à l'hôte

Forcer la main : `--platform`

- `docker pull --platform linux/amd64 nginx`
- Utile pour construire ou tester pour une autre cible
- L'image ne correspond plus au processeur : il faut **traduire**

Ce que coûte la traduction

- Docker Desktop fait appel à une **émulation** du jeu d'instructions
- C'est exactement l'émulation du premier chapitre, avec son prix
- Le conteneur fonctionne, mais lentement

“

On traduit, et ça se paie.

”

Quand ça casse

- Image sans variante pour votre architecture
- Message `exec format error` au démarrage
- Base multi-arch, binaire téléchargé mono-arch : l'erreur arrive plus tard

À retenir (1/2)

- L'image survit au conteneur, la couche inscriptible meurt avec lui
- Une image est une succession de couches
- Supprimer un fichier n'allège pas une image

À retenir (2/2)

- latest ne veut pas dire « la dernière »
- Plus petite veut dire moins de confort pour déboguer
- Une image forcée hors de son architecture tourne émulée

Les images

