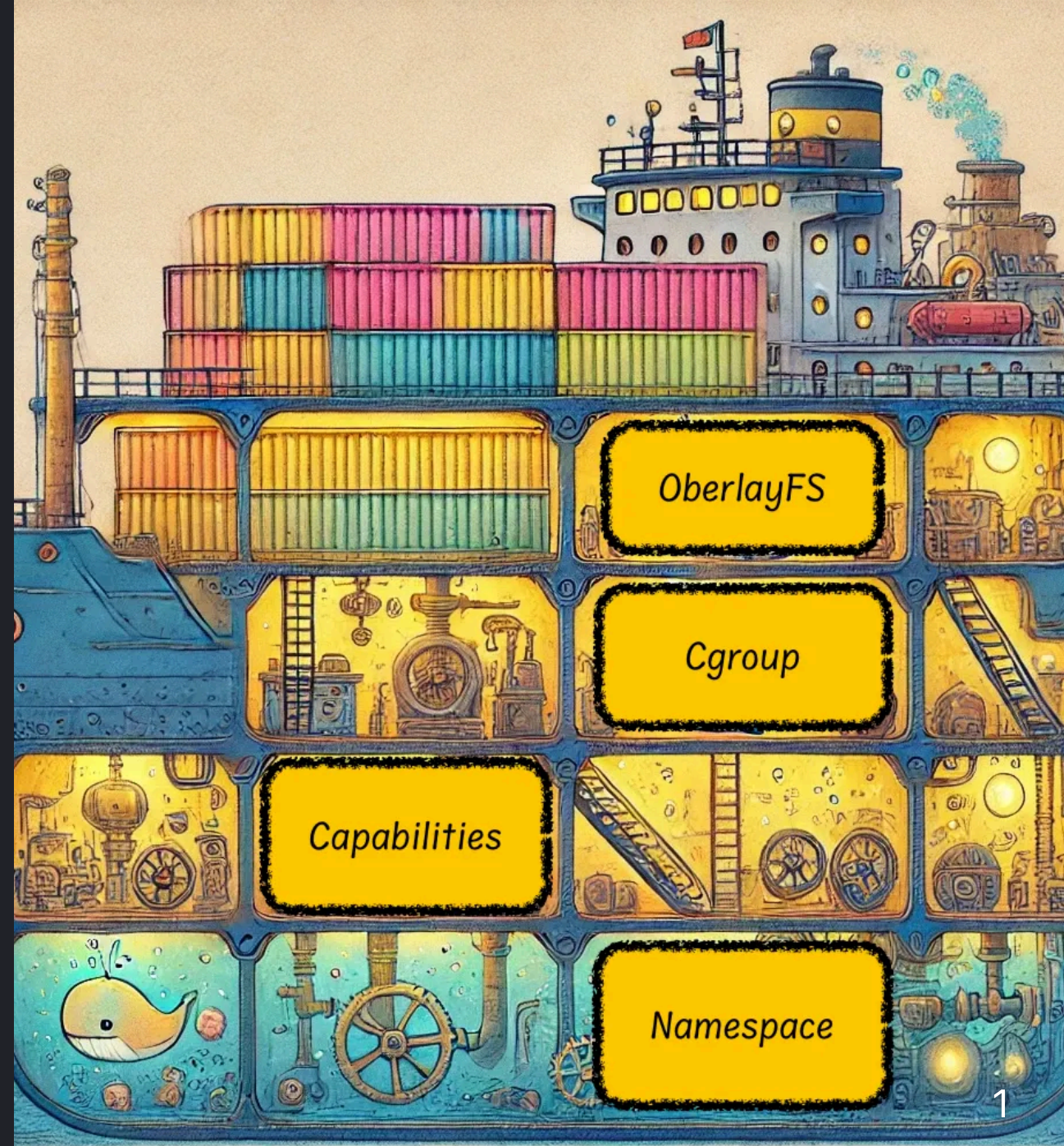
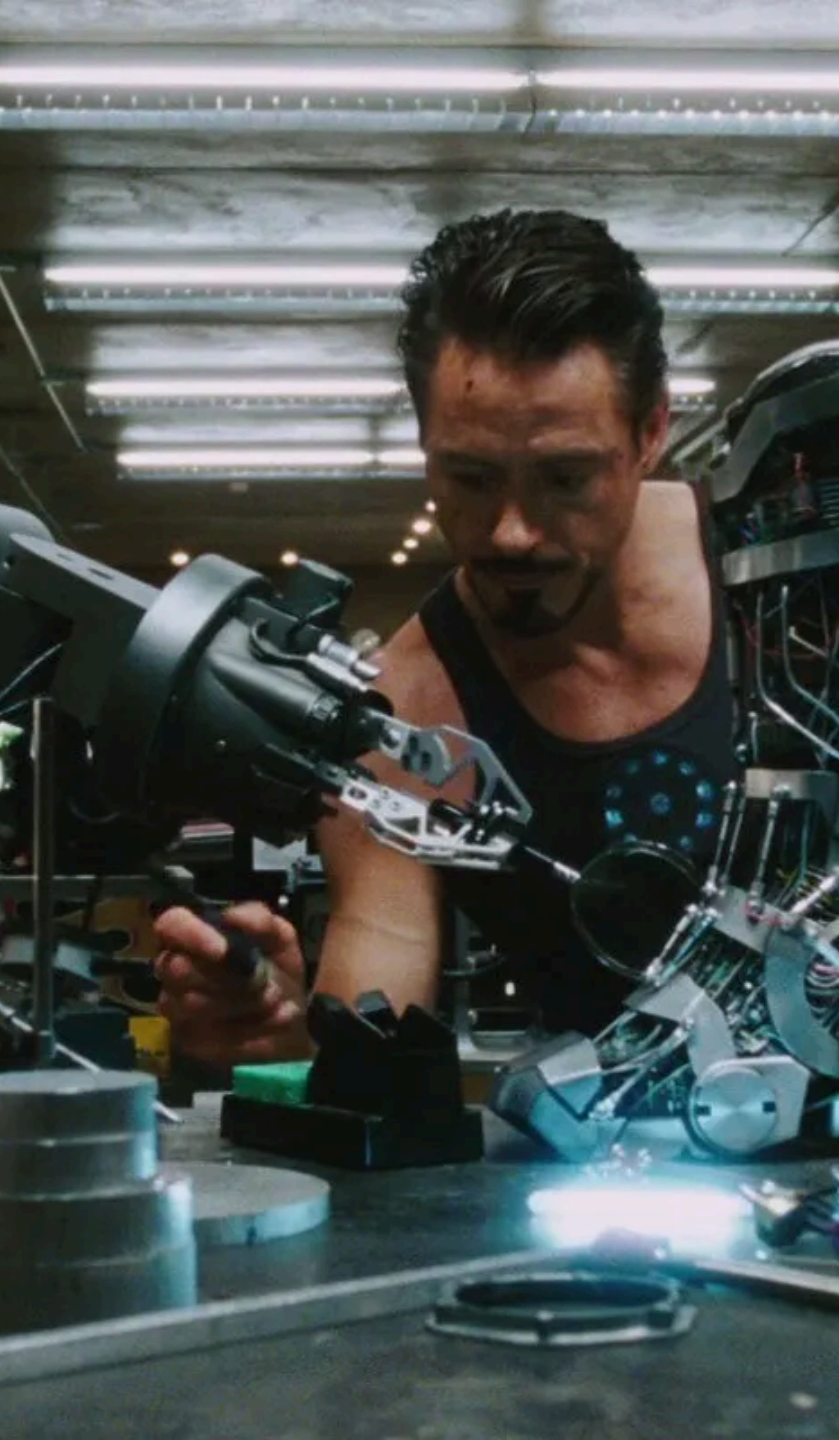


Conteneurisation





Ce que vous venez de faire

Résumé du TP

Quatre briques, empilées

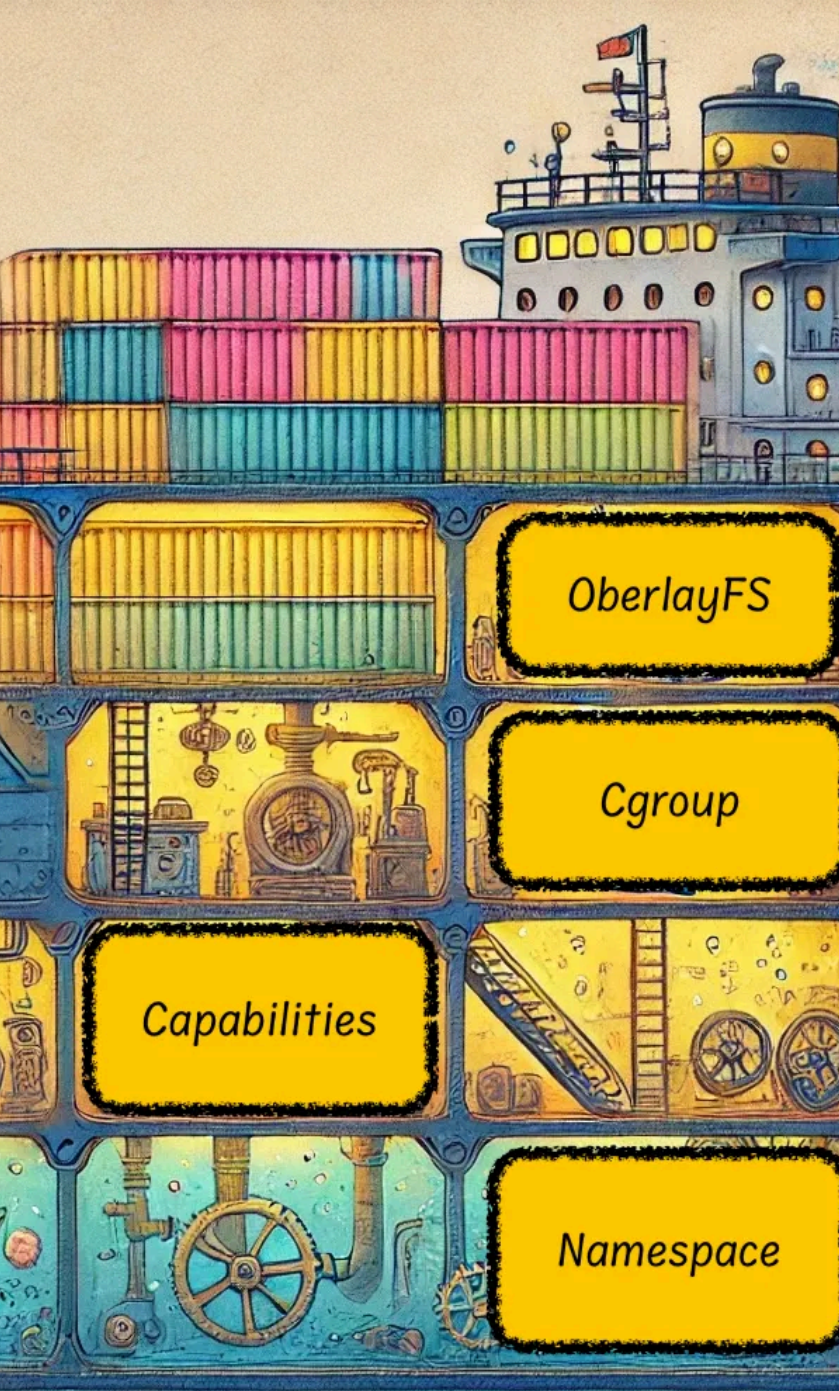
- `chroot` — un autre système de fichiers
- `unshare --uts` — un autre nom de machine
- `unshare --pid` — un autre espace de processus
- `unshare --net` — un autre réseau

“ Aucune de ces commandes n'est propre à Docker.
Elles sont dans le noyau depuis des années. ”

L'idée à retenir

- Un conteneur est **un processus Linux ordinaire**
- Ce qui change, c'est **ce qu'il a le droit de voir**
- Il n'y a ni machine, ni noyau, ni matériel virtualisé

“ Ce n'est pas une mini-VM. C'est un processus à qui on a menti sur son environnement. ”



Les briques

chroot et son successeur

- `chroot` : change la racine du système de fichiers
- Ne touche à rien d'autre — vous l'avez constaté avec `ps`
- `pivot_root` : remplace vraiment l'ancienne racine au lieu de la masquer

“ Les moteurs de conteneurs utilisent `pivot_root`,
parce qu'un `chroot` peut se contourner. ”

Les namespaces du TP

- `mnt` — les points de montage
- `uts` — nom de machine et de domaine
- `pid` — les numéros de processus
- `net` — interfaces, routes, ports

Les quatre autres

- `ipc` — files de messages, mémoire partagée
- `user` — correspondance des UID/GID
- `cgroup` — vue de l'arborescence des cgroups
- `time` — horloges monotones

Capabilities

- Le pouvoir de `root` découpé en une quarantaine de morceaux
- `cap_dac_override` : ignorer toutes les permissions
- `cap_dac_read_search` : ignorer celles en lecture
- `cap_sys_admin` : le fourre-tout dangereux

seccomp

- Filtre les **appels système** que le processus peut émettre
- Granularité plus fine que les capabilities

cgroups

- Limitent ce qu'un processus peut **consommer** : mémoire, CPU, I/O
- Rien à voir avec l'isolation : ils rationnent, ils ne cachent rien
- Namespaces et cgroups répondent à deux questions différentes

“

Qu'est-ce que je vois ?
A combien de ressources ai-je droit ?

”



Conséquences du noyau partagé

Côté performance

- Rien à démarrer : pas de BIOS, pas de noyau, pas de boot
- Rien à traduire : les instructions tournent sur le CPU
- Le coût d'un conteneur, c'est celui d'un processus

Côté sécurité

- Un seul noyau pour tout le monde
- Une faille du noyau est une faille **de tous** les conteneurs
- L'isolation repose sur une configuration, pas sur une frontière matérielle

“ C'est le prix à payer pour la légèreté. Une VM pose une frontière plus épaisse. ”



Chronologie

De **chroot** à Docker

- **1979** — **chroot** arrive dans Unix V7
- **2002** — le premier namespace entre dans le noyau Linux
- **2006** — les cgroups naissent chez Google
- **2008** — les cgroups entrent dans le noyau, LXC réunit les deux
- **2013** — Docker rend tout ça utilisable

En une phrase

“ Un conteneur est un processus Linux dont on a restreint la vue avec des namespaces, les pouvoirs avec des capabilities, et la consommation avec des cgroups. ”

À vous de jouer

