microVMs avec Firecracker

12 septembre 2024

Journée 2RM / MIn2RIEN

Mickaël MASQUELIN

Mickaël Masquelin

Ingénieur de Recherche CNRS @ CRIStAL

https://www.cristal.univ-lille.fr

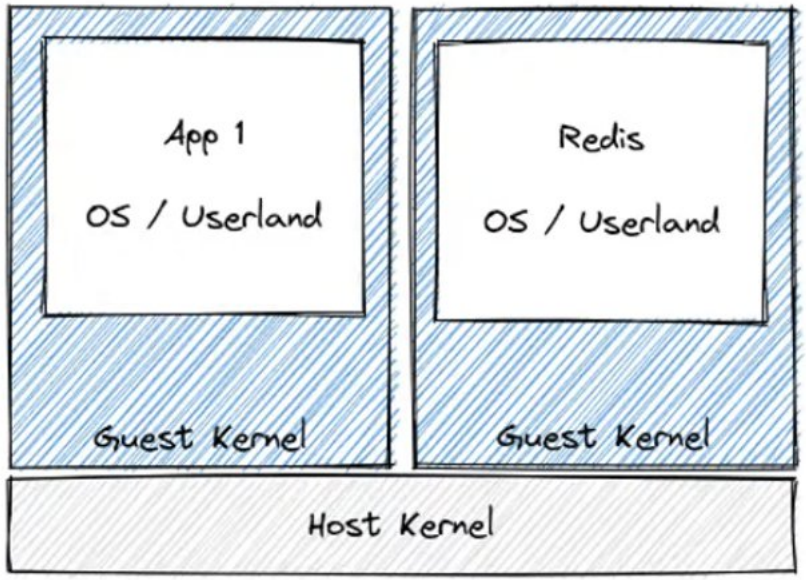
mickael.masquelin _at_ cnrs.fr

Agenda

- La virtualisation
- Les origines du projet
- Firecracker ?
- Mise en œuvre de la solution
- Démo / conclusion

La virtualisation

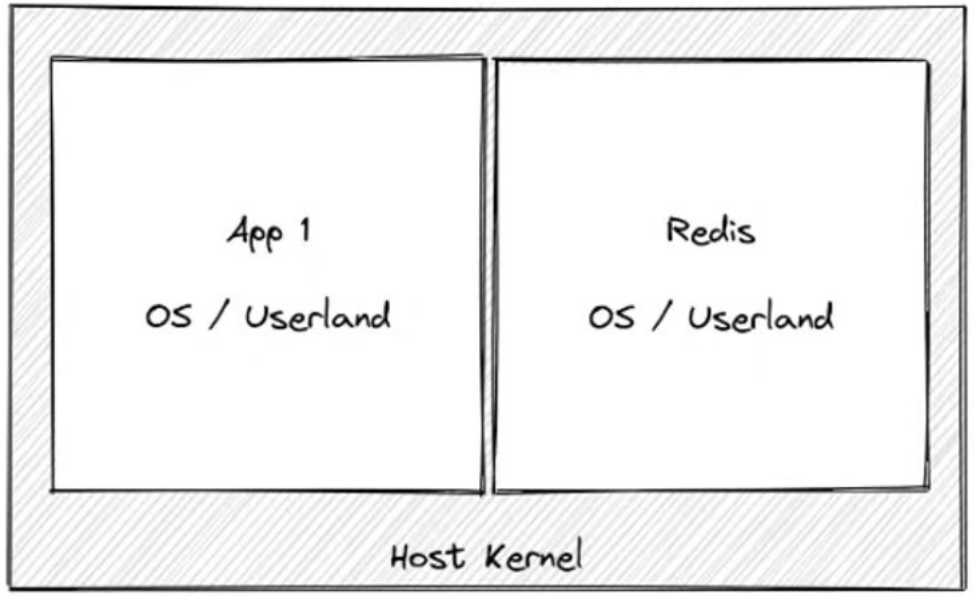
Virtualisation avec hyperviseur



Cas de KVM :

- Démarrage « lent » ;
- Images assez grosses en taille ;
- Nécessite d'importantes ressources ;
- Embarque parfois trop de choses …

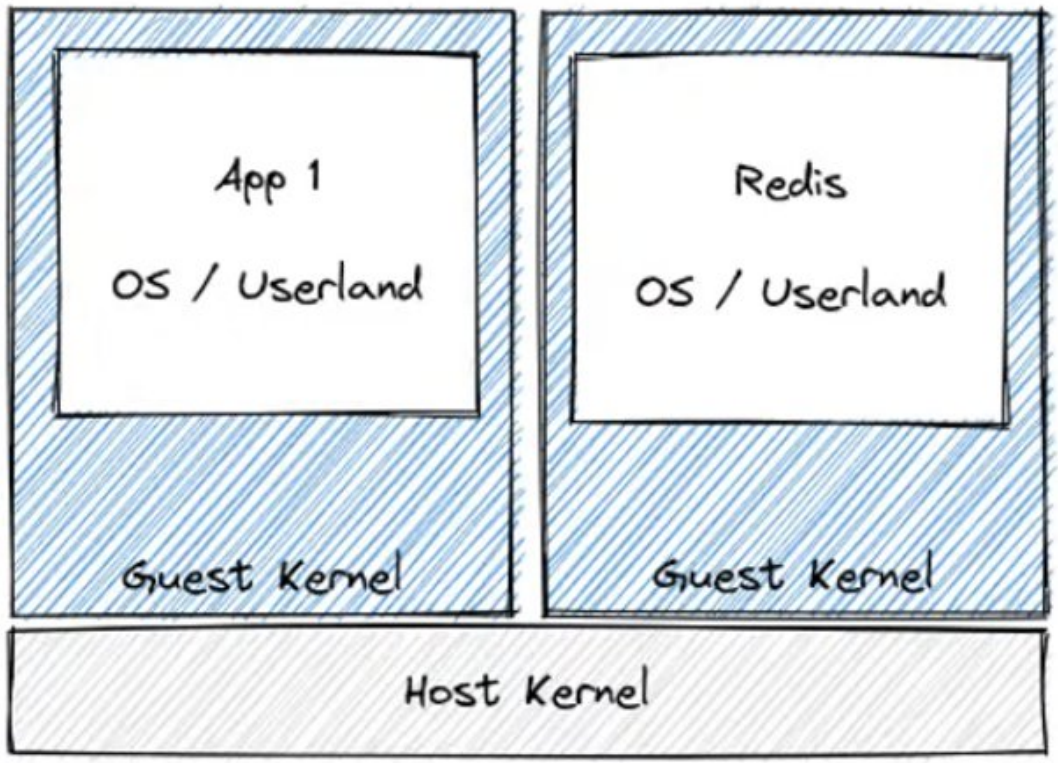
Virtualisation avec des conteneurs



Cas de Docker :

- Image construite exécutable « n'importe où » (ou presque)
- Docker Hub
- Démarrage rapide
- Système de couches pour faciliter la ré-utilisation
- Isolation au niveau de l'OS hôte

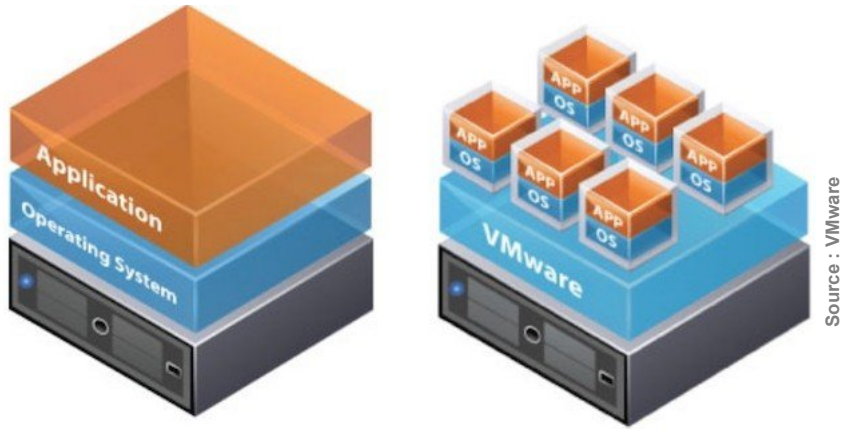
Virtualisation avec microVM



Aux origines du projet …

Pourquoi ce talk aujourd'hui ?

Notre infrastructure … avant …



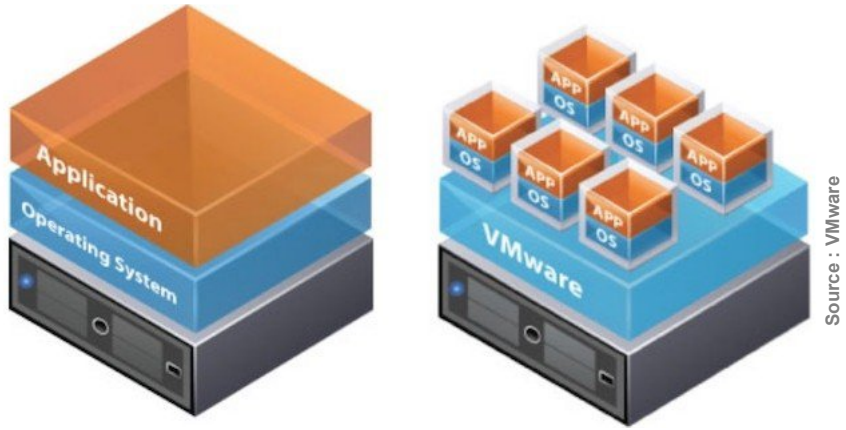
Architecture traditionnelle — **Approche via virtualisation**

Hyperviseurs bare-metal VMware / Proxmox

(VMs **ET** conteneurs LXC)

Pourquoi ce talk aujourd'hui ?

Notre infrastructure … avant …

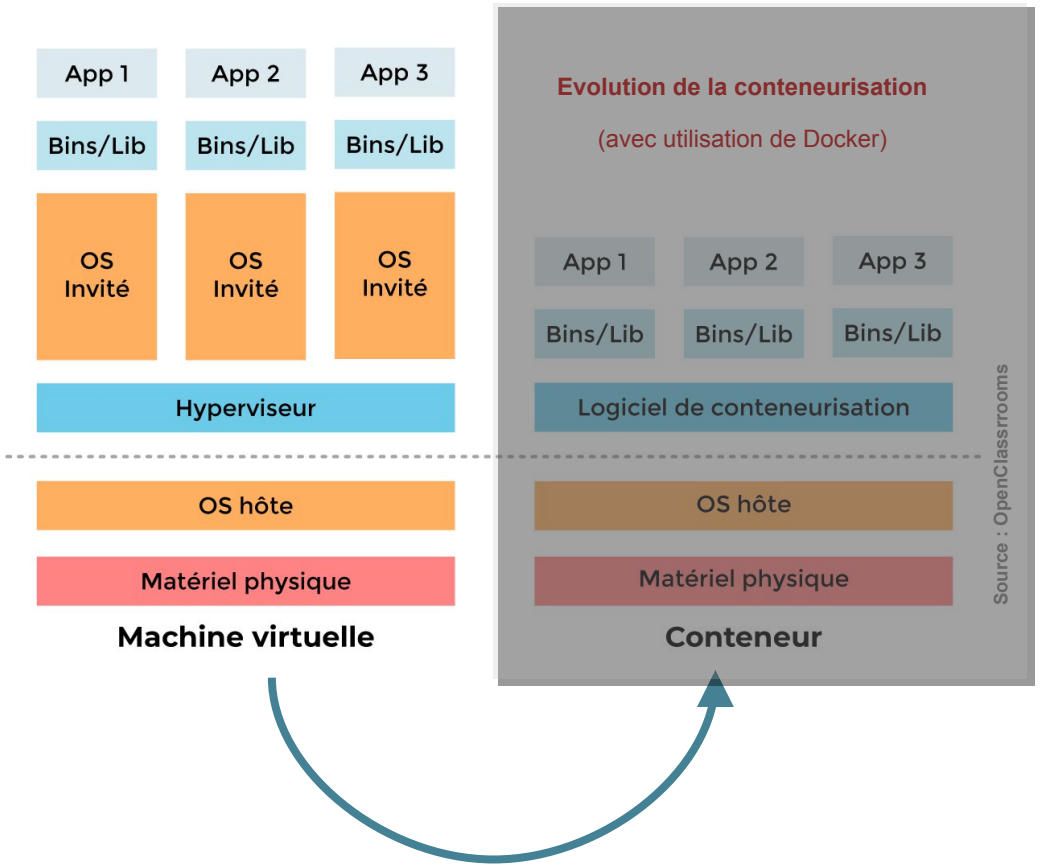


Architecture traditionnelle — **Approche via virtualisation**

Hyperviseurs bare-metal VMware / Proxmox

(VMs **ET** conteneurs LXC)

… puis progressivement …



Vers l'utilisation progressive de Docker

- Recours à des conteneurs Docker pour remplacer LXC ;
- Gérer des « fermes » de conteneurs Docker unitairement est complexe …

- Comment faire fonctionner plusieurs conteneurs différents côte à côte ?
- Comment gérer la création / suppression des conteneurs de manière automatique selon la charge ?

▶ Nécessité de déployer et d'utiliser **une solution technique plus adaptée** pour mieux **gérer nos flux de travail** !

Pistes envisagées

Nomad

Consul

Vault

Terraform

kubernetes

Nomad

Orchestrer ses conteneurs

- Développement démarré en 2015 ;
- Alternative à Kubernetes en tant qu'orchestrateur si combiné à Consul ;
- Permet d'utiliser d'autres ressources que des conteneurs : VMs QEMU, Firecracker, JAVA, GPU Nvidia, etc. ;
- Peut être déployé sur plusieurs régions et centres de données afin d'être hautement disponible et tolérant aux pannes ;
- Fourni sous la forme d'un seul binaire pour configurer des nœuds master ou piloter des minions.

K8s vs Nomad (en tant qu'orchestrateur)

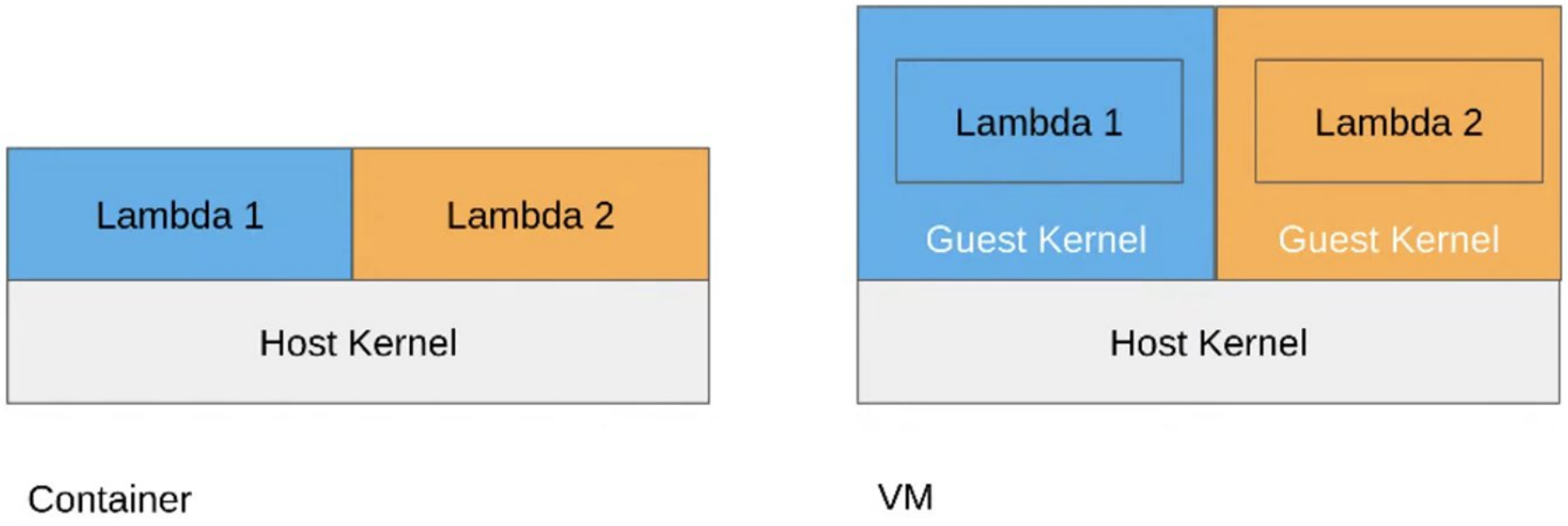
Kubernetes		Nomad	
Avantages	**Inconvénients**	**Avantages**	**Inconvénients**
Support communautaire important	Ecosystème assez « compliqué »	Facile à comprendre et à utiliser	Manque de maturité pour Nomad en comparaison avec Kubernetes
Plusieurs distributions disponibles	Peut entraîner des coûts « cachés »	Nécessite peu d'efforts en terme de configuration et est plutôt appropriée pour de petites équipes	Choix d'outils limités
Portabilité et flexibilité de la solution assurée	Nécessite la configuration de plusieurs composants interopérables qu'il faut gérer et installer	Fonctionne avec des modèles réseaux « courants »	Manque de contributeurs et d'un support communautaire large
Compatibilité multi-clouds	Nécessite une compréhension approfondie des concepts réseaux mis en œuvre dans K8s (MetalLB, Calico, eBPF par exemple)	Possibilité de scheduler, déployer et de gérer plusieurs types de charges serveurs (Docker, QEMU, Java, LXC, etc.)	
Solution open source		Agnostique : supporte Windows, Linux et macOS	

Firecracker ?

Généralités sur Firecracker

- https://github.com/firecracker-microvm/firecracker
- Technologie de virtualisation open-source
- Ecrite en Rust, compilée en binaire, pour les architectures x86_64 et aarch64 (dernière version : 1.9.0)
- Portée et développée par Amazon Web Services (AWS)
- Conçue pour exécuter des charges de travail de type `serverless` ou des applications conteneurisées
- Possibilité d'exécuter des microVM en tant qu'utilisateur non privilégié (`rootless`)

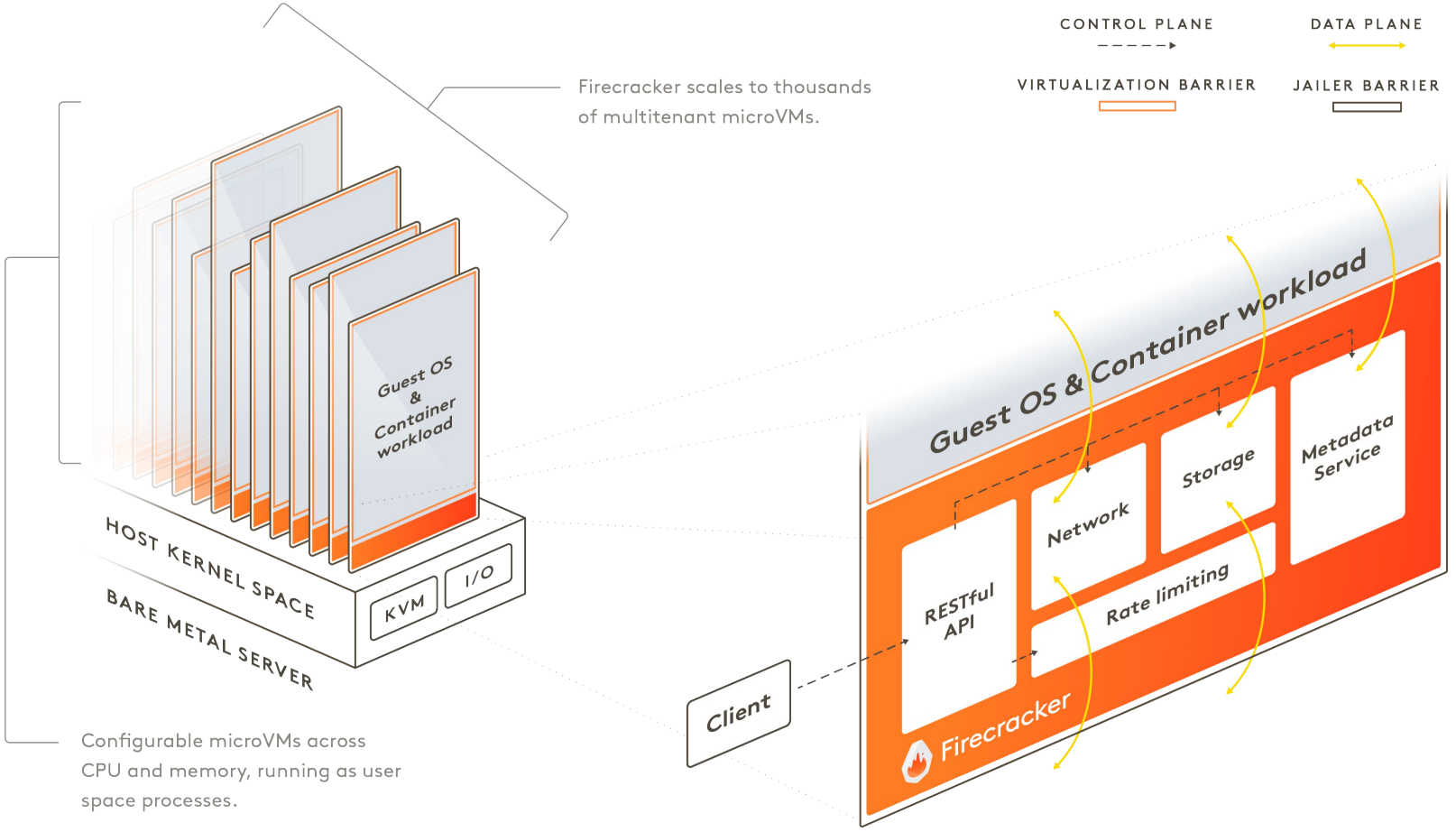
Point de vue d'AWS sur l'isolation



Caractéristiques principales de Firecracker

- **Minimaliste** car seules les fonctionnalités essentielles sont embarquées dans la VM (élimination des choses superflues) ;
- **Legereté** car overhead en mémoire : pas plus de 5 MB par microVM ;
- Temps de démarrage **rapide** des VMs : moins de 125 millisecondes pour booter ;
- Incorpore des mécanismes d'**isolation** forts : repose sur KVM pour les aspects sécurité dans la virtualisation ;

Architecture (1/2)



Architecture (2/2)

- Extrêmement rationnalisée pour atteindre les objectifs de minimalisme et d'efficacité affichés
- En termes de « composants » :
  - **microVMs** : exécutent une seule appli ou un ensemble de processus qui concourent à la réalisation d'une fonction précise
  - **Jailer** : contrôle le fait que les microVMs soient effectivement isolées et « sécurisées »
  - **KVM** : fournit les éléments nécessaires à la réalisation de l'opération de virtualisation
  - **Server API** : comprend une API REST pour gérer le cycle de vie de ses microVMs (création, configuration, démarrage, arrêt)

Comparaison avec Docker (1/3)

- **Isolation :**
  - Côté Firecracker : isolation au niveau du matériel à l'aide de KVM (plus sûr et plus robuste contre les attaques qui exploitent un noyau partagé) ;
  - Côté Docker : isolation au niveau de l'OS (les conteneurs partageant le noyau de l'hôte) … ce qui le rend léger mais potentiellement moins sûr !

- **Performances :**
  - Côté Firecracker : léger overhead en raison de la couche de virtualisation mais les microVMs restent légères ;
  - Côté Docker : très faible overhead car partage direct des ressources de l'hôte.

Comparaison avec Docker (2/3)

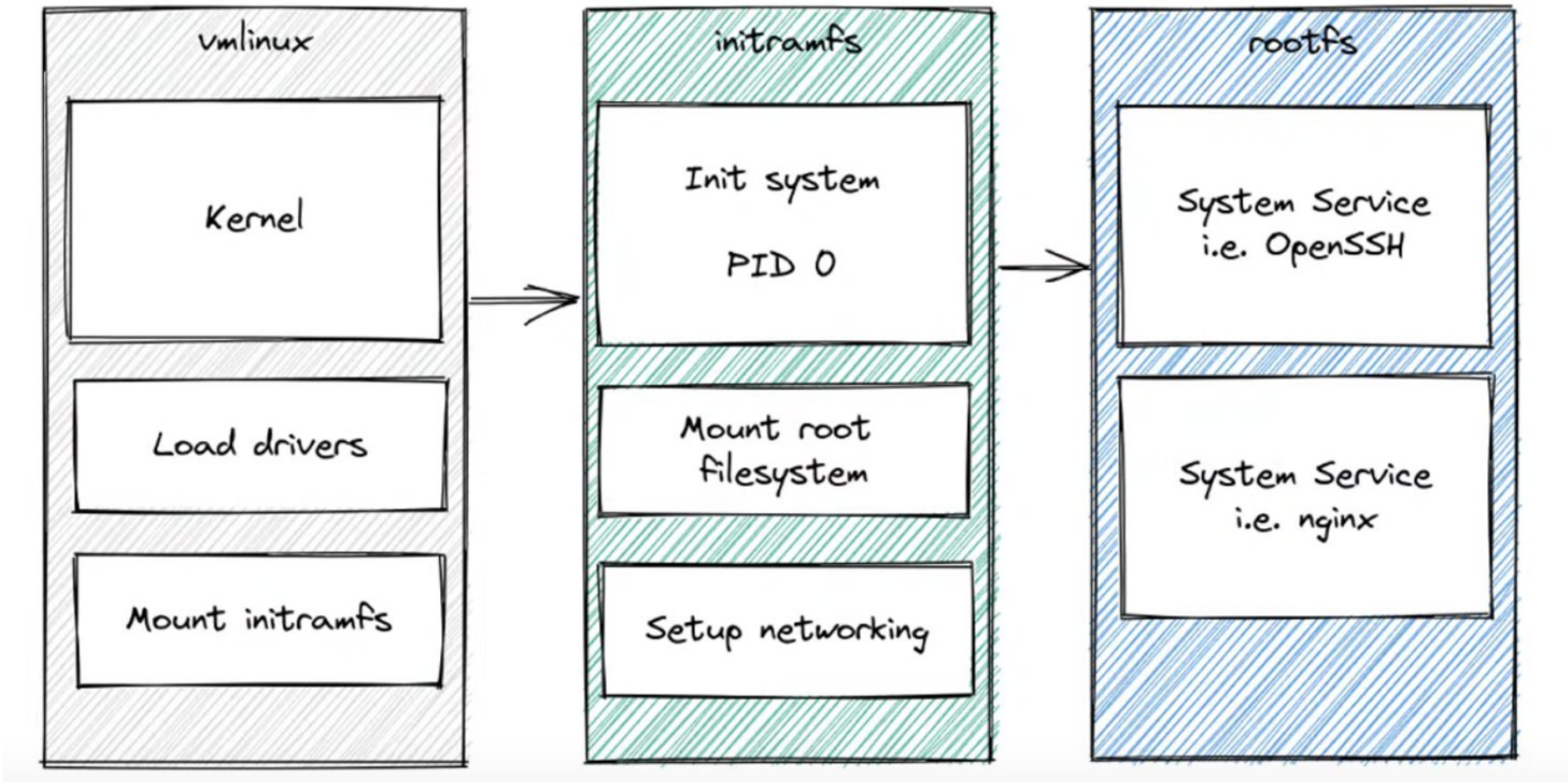
- **Temps de démarrage :**
  - Côté Firecracker : démarrage en moins de 125ms, idéal pour des tâches brèves ;
  - Côté Docker : démarrage encore plus rapide que les microVMs (quasi instantané).

- **Cas d'utilisation :**
  - Côté Firecracker : idéal pour une charge de travail de type serverless ou dans un contexte nécessitant des besoins d'isolations forts … ou des scénarios où la sécurité est essentielle ;
  - Côté Docker : Mieux adapté aux microservices, aux pipelines de CI/CD et aux environnements de développement où la vitesse et l'efficacité des ressources sont importantes.

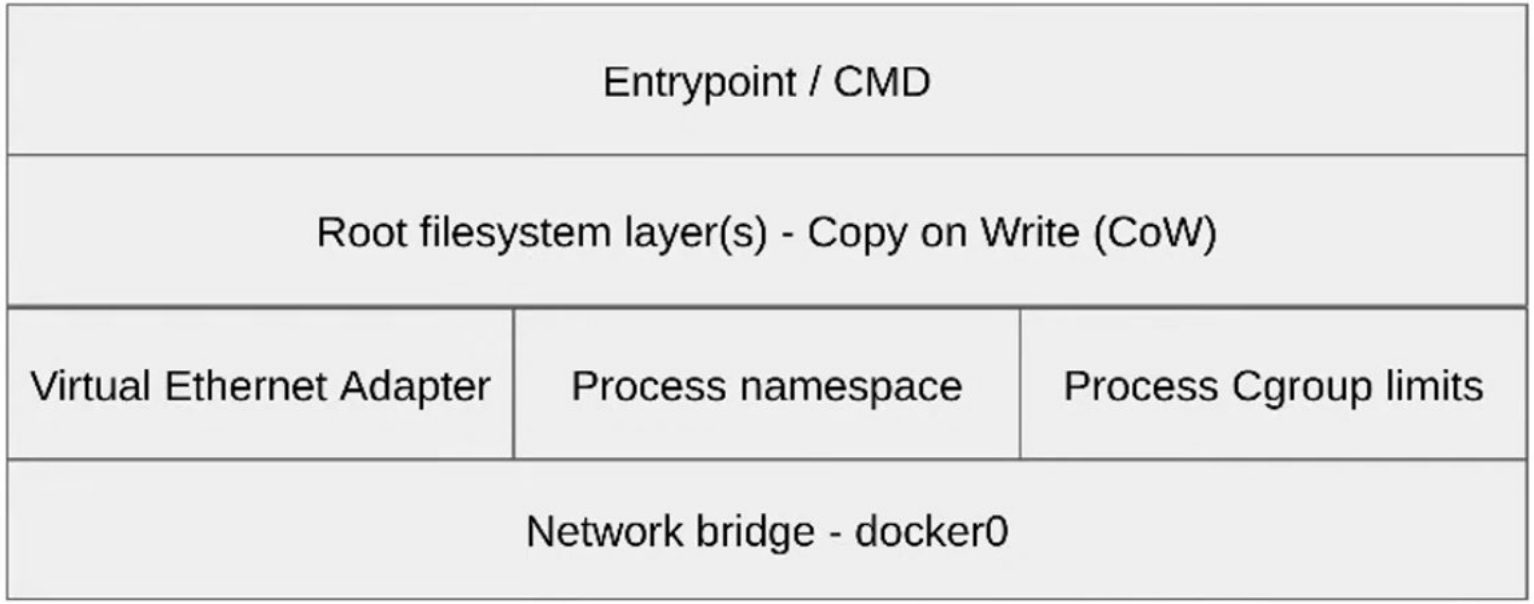
Comparaison avec Docker (3/3)

- **Gestion :**
  - Côté Firecracker : via une API REST très simple, conçu pour fonctionner avec des outils d'orchestration de haut niveau (type Nomad de chez HashiCorp, OpenNebula, etc.) ;
  - Côté Docker : via le client Docker en CLI et l'API, avec le support des outils d'orchestration (tels que K8s ou Nomad).

La séquence de boot en détail …



Architecture avec Docker



Mise en œuvre de Firecracker

Une mise en œuvre en 3 étapes simples

1. Récupérer/installer le binaire Firecracker (depuis GH) ;
2. Télécharger une image du noyau Linux et un disque rootfs minimaliste ;
3. Lancer Firecracker et commencer à interagir avec l'outil via l'API RESTful afin de configurer sa première microVM

Connexion ensuite possible à sa microVM en utilisant la socket unix exposée par défaut (avec `socat`) ou une pseudo console série (via `screen`)

Exemple d'appel à l'API

```
POST /vmm/v1/microvm
{
  "kernel_image_path": "/chemin/vers/vmlinux",
  "root_drive": {
    "path": "/chemin/vers/rootfs",
    "is_root_device": true,
    "is_read_only": false
  },
  "cpus": {
    "count": 2
  },
  "mem_size_mib": 256
}
```

Conclusion

Conclusion

- Evolution intéressante de la virtualisation ;
- Offre un équilibre convaincant entre performance, sécurité et efficacité ;
- Solution idéale si vous recherchez une isolation rigoureuse et un overhead limité !

Questions ?