

Veille technologique sur les runtimes

I. Les runtimes de containers choisis et les motifs derrière leur choix

i. runc

Quand Docker exécute une image, runc est le composant qui s'occupe de l'exécution de l'image par défaut. L'image exécutée suit la norme OCI (Open Container Initiative), donc une exécution d'images plus facile sans arguments additionnels. En plus, il est considéré le runtime le plus compatible avec Docker, car runc est écrit en Go, en plus il a une performance native car il ne dépend pas sur les boot VM ni l'interception de syscalls, car sa sécurité est réalisée avec seccomp et le choix des capacités Linux. [1]

ii. Youki

L'image exécutée suit la norme OCI (Open Container Initiative), En plus le runtime est développé en Rust, ainsi utilisation sécurisée de la mémoire avec moins d'empreintes et maintenance continue. [2]

iii. crun

Quand Podman exécute une image, crun est le composant qui s'occupe de l'exécution de l'image par défaut. Caractérisé par un temps de démarrage d'image 5 fois plus rapide que runc, et plus de performance. La principale raison vient de la langue de programmation utilisée pour le développement, C en étant compatible OCI [2] (dans le tableau des métriques mesurés)

iv. styrolite [3]

Il assure des fonctionnalités de sécurité en plus à l'image exécutée comme: offrir des sandboxes programmables et contrôler les appels systèmes à travers un hyperviseur de type 1. En plus il est développé en Rust, menant à moins d'empreintes mémoires. [4]

vi. les projets abandonnés

- rkt : un runtime de conteneurs pod-native pour Linux, donc il implémente la fonctionnalité des pods, qui est un composant entrant dans un environnement scalable d'exécution de plusieurs containers, mais avec moins d'overhead que la technologie d'orchestration, Kubernetes. En effet, le projet a terminé il y a 5 ans. Les conférences de CNCF m'ont permis de connaître le runtime qui a connu un grand succès à l'époque. [5]
- nabla : Un runtime se basant sur l'émulation à travers la redirection des appels systèmes vers une LibOS (bibliothèque système utilisateur), ce qui permet de garantir la sécurité sans recourir aux techniques de virtualisation de microVMs, qui sont considérés plus lents à cause de leur overhead. Son support est terminé il y a 5 ans. [6]

II. Comparaison de privilèges

	runc	Youki	crun	styrolite
Mode privilégié	Oui, en ajoutant toutes les capacités au fichier config.json généré par runc spec	Oui, en ajoutant toutes les capacités dans le fichier config.json généré par youki spec	Oui, en ajoutant toutes les capacités dans le fichier config.json généré par crun spec	Oui, en ajoutant les capacités au fichier config.json, puis exécuter l'image avec le nouveau config.json
Isolation par namespace	pid, network, ipc, cgroup, mount,	pid, network, ipc, uts, mount,	pid, network, user, ipc, cgroup,	pid, network, ipc, cgroup, mount,

	uts dans le fichier config.json généré par runc spec	cgroup dans le fichier config.json généré par youki spec	mount, uts dans le fichier config.json généré par crun spec	uts, user, time dans le fichier namespace.rs repo Github
Support de seccomp	Oui, la rubrique "Build Tags" libseccomp [7]	Oui, dans la rubrique Dependencies il intègre libseccomp [2]	Oui, un fichier dédié aux communications avec seccomp [8]	Non, le runtime assure la séc exécute les images dans des zones d'exécution isolées comme des machines virtuelles légères, il n'assure pas la sécurité par seccomp. [9]
AppArmor / SELinux	Non, la rubrique "Build Tags", mais SELinux et AppArmor peuvent changer de contexte et exécuter une image runc [7]	Oui [10], [11]	Oui, le fichier linux.h dans le repo a des fonctions pour ajouter les privilèges aux profils selinux et apparmor (e.g libcrun_set_selinux_image) [12]	Non en cherchant par mot clé sur le repo Github, mais SELinux et AppArmor peuvent changer de contexte et exécuter une image styrolite [12]
Filesystem en lecture seulement	Oui, à la ligne readonly: true dans le fichier config.json généré par runc spec	Oui, à la ligne readonly: true dans le fichier config.json généré par youki spec	Oui, à la ligne readonly: true dans le fichier config.json généré par crun spec	Oui, il supporte les mounts readonly [13]
Mode rootless	Oui, à condition que le user namespaces soit activé et compilé dans le kernel [14]	Oui, dans la rubrique Rootless Container [2]	Oui, avec crun – rootless	Non, il est possible de modifier les mappings dans le fichier config.json mais il n'est pas possible d'exécuter en mode rootless

Les privilèges

- Mode privilégié : il permet d'avoir toutes les capacités du système hôte, ainsi avoir tout le contrôle sur le système hôte. N.B: Le fichier config.json c'est le fichier généré par un runtime qui décrit les privilèges d'un conteneur exécuté.

- Support seccomp : un appel système qui filtre les syscalls ainsi contribuant à moins de droits d'utilisateur de communication avec le kernel [15]
- AppArmor / SELinux: définir les ressources ou définir les privilèges d'un container. [16]
- Filesystem en readonly: éviter la modification du filesystem par un utilisateur non-privilégué. [17]
- Mode rootless: Le remappage pour le user namespace afin d'exécuter les conteneurs en étant non-root ce qui limite à l'utilisateur l'accès à certaines capacités, limitant l'accès au filesystem. [18]

Capabilités [19]

	runc mode non-privilégué	youki mode non-privilégué	crun mode non-privilégué	styrolite mode non-privilégué
SYS_ADMIN	Non	Non	Non	Non
NET_ADMIN	Non	Non	Non	Non
SYS_PTRACE	Non	Non	Non	Non
SYS_MODULE	Non	Non	Non	Non
SYS_TIME	Non	Non	Non	Non
AUDIT_WRITE	Oui / P-E-B	Oui	Oui / P-E-B-A	Oui / P-E-B-A-I
NET_BIND_SERVICE	Oui / P-E-B	Oui	Oui / P-E-B-A	Oui / P-E-B-A-I
KILL	Oui / P-E-B	Oui	Oui / P-E-B-A	Oui / P-E-B-A-I

Les capabilités étaient récupérés respectivement par:

- runc spec
- youki spec
- crun spec
- par un des trois fichiers config.json générés par l'un des 3 commandes précédentes, car styrolite n'a pas une commande de génération d'un fichier config.json.

Les types de capabilités

Permitted (P): Les capabilités sont automatiquement permises au thread sans regarder les capabilités du master thread.

Effectifs (E): les capabilités utilisés dans les checks de permission du kernel.

Bounding (B): les capabilités utilisés durant l'appel system execve2 appelé avec le chemin de la commande.

Ambient (A): Les capabilités restent activé avec le processus appelé sans avoir besoin d'être privilégié.

Inheritable (I): Les capabilités sont gardés au runtime lors de l'appel de execve(2)

– En premier lieu, les capabilités qui doivent être désactivés par défaut

- CAP_SYS_ADMIN : Elle offre un nombre de privilèges d'administration système tel que l'accès aux appels systems (mount, unmount, pivot_root), de surpasser le filesystem, créer de nouveaux namespaces avec clone(2), et bien d'autres dans la page manuel de Linux.
- CAP_NET_ADMIN: qui permet de configurer les adresses IP , les tables de routages , il doit activé dans les containers de réseaux pas dans les environnements partagés.

iii. CAP_SYS_PTRACE: Avec cette capacité , un processus peut lire la mémoire d'un autre processus pour débogage, il doit être activé dans les environnements de débogage sécurisés, sinon trop risqué.

iv. CAP_SYS_MODULE: Charger des modules kernels, peut entraîner à des vulnérabilités dans le conteneur pour entrer en mode kernel privilégié.

v. CAP_SYS_TIME : Changer le temps système et affecter le résultat des logs.

– En deuxième lieu, les capacités qui sont activés par défaut.

ii. CAP_AUDIT_WRITE: Écrire dans les infos d'audit existants dans le kernel.

iii. CAP_NET_BIND_SERVICE: Permet l'accès à numéro port plus petit que 1024 même si l'utilisateur n'a pas les privilèges de root.

iv. CAP_KILL: Envoyer des signaux aux processus comme SIG_KILL , l'utilisateur ne peut pas tuer d'autres process que les siens.

III. Limites

- Incompatibilité avec des kernels Linux

les versions les plus modernes des runtimes nécessitent des kernels Linux de 4.5 ou plus pour supporter la version la plus récente de cgroup, une fonctionnalité Linux qui isole les ressources utilisés par un conteneur (e.g RAM , CPU ..) Cgroupv2 est la version la plus moderne pour cgroup et elle isole mieux les ressources que la premières. [20]

- Incompatibilité avec overlayFS

OverlayFS est un système de fichiers pour linux qui est plus optimal que d'autres système de fichiers comme vfs [21] en montage de filesystem car il :

1. Permet aux containers de partager même système de fichiers linux de base, permettant de réduire l'espace utilisée.
2. Un démarrage plus rapide des containers.
3. Copy-on-write (CoW) Les fichiers du système de fichier de base qui sont copiés sont justes ceux modifiés.

- Incompatibilité avec Macvlan et ipvlan

Macvlan est un driver Linux de réseau qui permet aux containers d'être connecté à un réseau physique pour leur offrir des fonctionnalités avancés de réseau en permettant une direct visibilité de couche Ethernet dans le modèle OSI. Son utilisation garantit plus d'isolation en évitant les microVMs, qui représentent plus d'overhead. [22], [23]

	runc	Youki	crun	styrolite
Incompatibilité du runtime à cause de cgroupv2	runc requiert un kernel qui supporte cgroupsv2, donc un Kernel plus petit que 4.15 n'est pas compatible avec cgroupv2 sauf si	requiert un kernel >= 5.3	Si, car dans leur repo Github, du support pour cgroupv1	Oui, le développement de styrolite a démarré en 2025, il utilise la version du kernel Linux la plus récente (kernel 6) qui

	le user namespace est activé [24]			supporte cgroupv2
Incompatibilité avec overlay FS	Non, overlayFS est supporté en cherchant dans le repo Github	Oui,dans le repo, des mounts tels que sysfs, btrfs sont suppo	Non,en cherchant overlay dans le repo Github, le driver est initialisé à overlay [25]	Oui, juste vfs est supporté [26]
Incompatibilité avec Macvlan	Non [27]	Non, car il peut être rootful mais il n'est pas compatible avec Macvlan en mode rootless [28]	Oui, car il est rootless [29]	Non, il n'est pas rootless

Bibliography

- [1] [Online]. Available: <https://jameshunt.us/writings/dockerless-docker-with-runc/>
- [2] [Online]. Available: <https://github.com/youki-dev/youki>
- [3] [Online]. Available: <https://github.com/edera-dev/styrolite?tab=readme-ov-file>
- [4] [Online]. Available: <https://www.lemondeinformatique.fr/actualites/lire-le-projet-open-source-styrolite-muscle-la-securite-du-runtime-des-conteneurs-96465.html>
- [5] [Online]. Available: https://www.youtube.com/watch?v=lHv0LVEIPk8&list=PLUVcZgyBBsZGrtrtXGinRm7AaNHqJ_prCJ&index=3
- [6] [Online]. Available: <https://github.com/nabla-containers/runnc#limitations>
- [7] [Online]. Available: <https://github.com/opencontainers/runc>
- [8] [Online]. Available: <https://github.com/containers/crun/blob/main/src/libcrun/seccomp.c>
- [9] [Online]. Available: <https://arxiv.org/pdf/2501.04580>
- [10] [Online]. Available: <https://github.com/youki-dev/youki/blob/main/crates/libcontainer/src/apparmor.rs>
- [11] [Online]. Available: <https://github.com/youki-dev/youki/tree/main/experiment/selinux/src>
- [12] [Online]. Available: <https://www.pearsonitcertification.com/articles/article.aspx?p=2992608&seqNum=4>
- [13] [Online]. Available: <https://www.infoq.com/news/2025/04/styrolite-low-level-runtime/?topicPageSponsorship=ce1390d7-0b6f-4b95-823c-b4b1f12acc26>
- [14] [Online]. Available: <https://github.com/opencontainers/runc/pull/774>
- [15] [Online]. Available: <https://man7.org/linux/man-pages/man2/seccomp.2.html>
- [16] [Online]. Available: <https://doc.ubuntu-fr.org/apparmor>

- [17] [Online]. Available: https://docs.datadoghq.com/security/default_rules/byb-wyq-f3q/
- [18] [Online]. Available: https://man7.org/linux/man-pages/man7/user_namespaces.7.html
- [19] [Online]. Available: <https://man7.org/linux/man-pages/man7/capabilities.7.html>
- [20] [Online]. Available: <https://docs.kernel.org/admin-guide/cgroup-v2.html>
- [21] [Online]. Available: <https://docs.docker.com/engine/storage/drivers/select-storage-driver/>
- [22] [Online]. Available: <https://docs.docker.com/engine/network/drivers/macvlan/#create-a-macvlan-network>
- [23] [Online]. Available: <https://dev.to/rimelek/comparing-3-docker-container-runtimes-runc-gvisor-and-kata-containers-16j>
- [24] [Online]. Available: <https://github.com/opencontainers/runc/blob/cdf953070162f400d75d2e0e83b506b3fb8cee5e/docs/cgroup-v2.md#L4>
- [25] [Online]. Available: <https://github.com/containers/crun/blob/9489b9962b2fa40e7c6b5d7260f40b3baa3cbd89/tests/podman/run-tests.sh#L22>
- [26] [Online]. Available: <https://github.com/search?q=org%3Aedera-dev+vfs&type=code>
- [27] [Online]. Available: <https://github.com/opencontainers/runc/discussions/3411>
- [28] [Online]. Available: <https://github.com/search?q=org%3Ayouki-dev+rootful&type=code>
- [29] [Online]. Available: <https://github.com/containers/podman/issues/21867>