Contraintes et Privilèges trouvés pour les runtimes choisies

Runtime	Privilèges Nécessaires	Contraintes
RunC en mode rootful Docker mode Privilégié	**Namespaces** pid , network , ipc , cgroup , mount , uts **Capabilities** Bounding : "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Effective: "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Permitted : "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" **User** uid : 0 gid : 0 **"noNewPrivileges":** true **"root":** "path": "rootfs", "readonly": true **Docker daemon** Cgroup Version: 2 **uid mapping** 0 0 1	**Ressource Limits** "type": "RLIMIT_NOFILE", "hard": 1024, "soft": 1024 **Resources** **Devices** "allow": false, "access": "rwm" **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency_stats", "/proc/timer_list", "/proc/timer_stats", "/proc/sched_debug", "/sys/firmware", "/proc/scsi" **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger" N'est pas sécurisé , si une vulnérabilité affecte le démon , les conteneurs au-dessous sont affectés.
Crun (Le runtime de podman par défault)	**Namespaces** pid , network , user, ipc , cgroup , mount , uts	**Ressource Limits** "type": "RLIMIT_NOFILE",

	Capabilities Bounding : "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Effective: "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Inheritable: Permitted : "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Ambient: "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" **Capabilities** (sur un containeur podman exécuté) Current: cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_sys_chroot,cap_setfcap=ep Bounding set =cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_sys_chroot,cap_setfcap **User** uid : 0 gid : 0 **"noNewPrivileges":** true **"root":** "path": "rootfs", "readonly": true	"hard": 1024, "soft": 1024 **Resources** **Devices** "allow": false, "access": "rwm" **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency_stats", "/proc/timer_list", "/proc/timer_stats", "/proc/sched_debug", "/sys/firmware", "/proc/scsi" **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger" En mode rootless , il utilise fuse-overlay qui est une implémentation d'OverlayFS , elle est lente CgroupsV1 ne peut pas être utilisé car il requiert des privilèges root. MACVLAN, IPVLAN ne sont pas supportés en mode rootless.

	IDmappings gidmap: - container_id: 0 host_id: 1000 size: 1 - container_id: 1 host_id: 524288 size: 65536 uidmap: - container_id: 0 host_id: 1000 size: 1 - container_id: 1 host_id: 524288 size: 65536 cgroupVersion: v2	
RunC en mode rootless Docker mode Rootless	**Namespaces** pid , user , ipc , cgroup , mount , uts **Capabilities** (runc spec) Bounding : "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Effective: "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" Permitted : "CAP_AUDIT_WRITE", "CAP_KILL", "CAP_NET_BIND_SERVICE" **Capabilities** (sur un containeur docker) Current: cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net	**Ressource Limits** "type": "RLIMIT_NOFILE", "hard": 1024, "soft": 1024 **Resources** **Devices** "allow": false, "access": "rwm" **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency_stats", "/proc/timer_list", "/proc/timer_stats", "/proc/sched_debug", "/sys/firmware", "/proc/scsi" **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq",

	_ raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap=ep Bounding set =cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap **User** uid : 0 gid : 0 **"noNewPrivileges":** true **"root":** "path": "rootfs", "readonly": true **Docker daemon** Cgroup Version: 2 **IdMappings** "uidMappings": "containerID": 0, "hostID": 1000, "size": 1 "gidMappings": "containerID": 0, "hostID": 1000, "size": 1	"/proc/sys", "/proc/sysrq-trigger" Les ports au dessous de 1024 ne sont pas accessibles) RunC utilise user namespace qui évite certains Les tables d'IP variables , règles de firewall variable , interfaces réseaux Il n'y a pas les capabilités comme CAP_SYS_ADMIN CAP_DAC_OVERRIDE CAP_NET_OVERRIDE
udocker (en utilisant google collab) Implémenté utilisant runc en mode rootless et crun e	**Capabilities (runc spec)** bounding: CAP_AUDIT_WRITE CAP_KILL CAP_NET_BIND_SERVICE effective: "CAP_AUDIT_WRITE	**Ressource Limits** "type": "RLIMIT_NOFILE", "hard": 1024, "soft": 1024 **Resources** **Devices**

	CAP_KILL CAP_NET_BIND_SERVICE "permitted": CAP_AUDIT_WRITE CAP_KILL CAP_NET_BIND_SERVICE" ambient": CAP_AUDIT_WRITE CAP_KILL CAP_NET_BIND_SERVICE **Capabilities (sur un containeur udocker)** Dépend sur les technologie de PRoot, Fakechroot, runc, crun et Singularity. Est rootless par défaut.	"allow": false, "access": "rwm" **Masked Paths** : "/proc/acpi", "/proc/asound", "/proc/kcore", "/proc/keys", "/proc/latency_stats", "/proc/timer_list", "/proc/timer_stats", "/proc/sched_debug", "/sys/firmware", "/proc/scsi" **Read-only Paths:** "/proc/bus", "/proc/fs", "/proc/irq", "/proc/sys", "/proc/sysrq-trigger" Même limitations d'une technologie rootless car il dépend sur crun. Car il ne dépend pas sur les namespaces niveau kernel comme Docker, il n'offre pas une isolation complète ou un niveau d'intégration complet
Apptainer (anciennement Singularity)	**Besoin d'accès root :** Par défaut non. Besoin d'un accès root dans certains cas (utilisation de cgroups v1 par exemple). **Utilisation de daemon :** Non	**Cgroups :** Supporte l'utilisation des cgroups v1 et v2 (via systemd). Par défaut pas d'utilisation de cgroups, possible avec une option au lancement. Dans la configuration par défaut c'est la version 2 qui est utilisée.