

Version : **2022.01**

Dernière mise-à-jour : 2021/06/08 13:51

DOF204 - Gestion de la Sécurité de Docker

Contenu du Module

- **DOF204 - Gestion de la Sécurité de Docker**

- Contenu du Module
- LAB #1 - Travailler avec les CGroups
 - 1.1 - Présentation des Namespaces
 - 1.2 - Présentation des CGroups
 - 1.3 - Limitation de la Mémoire
 - 1.4 - Le Paquet cgroup-tools
 - La commande cgcreate
 - La Commande cgexec
 - La Commande cgdelete
 - Le Fichier /etc/cgconfig.conf
- LAB #2 - Utilisation des Docker Secrets
- LAB #3 - Création d'un Utilisateur de Confiance pour Contrôler le Daemon Docker
- LAB #4 - Le Script docker-bench-security.sh
- LAB #5 - Sécurisation de la Configuration de l'Hôte Docker
 - 5.1 - [WARN] 1.2.1 - Ensure a separate partition for containers has been created
 - 5.2 - [WARN] 1.2.3 - Ensure auditing is configured for the Docker daemon
- LAB #6 - Sécurisation de la Configuration du daemon Docker
 - 6.1 - [WARN] 2.1 - Ensure network traffic is restricted between containers on the default bridge
 - 6.2 - [WARN] 2.8 - Enable user namespace support
 - 6.3 - [WARN] 2.11 - Ensure that authorization for Docker client commands is enabled
 - 6.4 - [WARN] 2.12 - Ensure centralized and remote logging is configured
 - 6.5 - [WARN] 2.14 - Ensure Userland Proxy is Disabled

- 6.6 - [WARN] 2.17 - Ensure containers are restricted from acquiring new privileges
- 6.7 - Le Fichier /etc/docker/daemon.json
- LAB #7 - Sécurisation des Images et les Fichiers de Construction
 - 7.1 - [WARN] 4.1 - Ensure a user for the container has been created
 - 7.2 - [WARN] 4.5 - Ensure Content trust for Docker is Enabled
 - 7.3 - [WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images
- LAB #8 - Sécurisation du Container Runtime
 - 8.1 - [WARN] 5.1 - Ensure AppArmor Profile is Enabled
 - 8.2 - [WARN] 5.2 - Ensure SELinux security options are set, if applicable
 - 8.3 - [WARN] 5.10 - Ensure memory usage for container is limited
 - 8.4 - [WARN] 5.11 - Ensure CPU priority is set appropriately on the container
 - 8.5 - [WARN] 5.12 - Ensure the container's root filesystem is mounted as read only
 - 8.6 - [WARN] 5.14 - Ensure 'on-failure' container restart policy is set to '5'
 - 8.7 - [WARN] 5.25 - Ensure the container is restricted from acquiring additional privileges
 - 8.8 - [WARN] 5.26 - Ensure container health is checked at runtime
 - 8.9 - [WARN] 5.28 - Ensure PIDs cgroup limit is used
- LAB #9 - Sécurisation des Images avec Docker Content Trust
 - 9.1 - DOCKER_CONTENT_TRUST
 - 9.2 - DCT et la commande docker pull
 - L'option disable-content-trust
 - 9.3 - DCT et la commande docker push
 - 9.4 - DCT et la commande docker build
 - Créer un deuxième Repository
 - Supprimer une Signature
- LAB #10 - Sécurisation du Socket du Daemon Docker
 - 10.1 - Création du Certificat de l'Autorité de Certification
 - 10.2 - Création du Certificat du Serveur Hôte du Daemon Docker
 - 10.3 - Création du Certificat du Client
 - 10.4 - Démarrage du Daemon Docker avec une Invocation Directe
 - 10.5 - Configuration du Client

LAB #1 - Travailler avec les CGroups

1.1 - Présentation des Namespaces

Les espaces de noms permettent de regrouper des processus dans un même espace et d'attribuer des droits sur des ressources par espace. Ceci permet l'exécution de plusieurs init, chacun dans un namespace, afin de recréer un environnement pour les processus qui doivent être isolés.

1.2 - Présentation des CGroups

Les Groupes de Contrôles (Control Groups) aussi appelés **CGroups**, sont une nouvelle façon de contrôler et de limiter des ressources. Les groupes de contrôle permettent l'allocation de ressources, même d'une manière dynamique pendant que le système fonctionne, telles le temps processeur, la mémoire système, la bande réseau, ou une combinaison de ces ressources parmi des groupes de tâches (processus) définis par l'utilisateur et exécutés sur un système.

Les CGroups sont organisés de manière hiérarchique, comme des processus. Par contre, la comparaison entre les deux démontre que tandis que les processus se trouvent dans une arborescence unique descendant tous du processus init et héritant de l'environnement de leurs parents, les CGroups peuvent être multiples donnant lieu à des arborescences ou **hiérarchies** multiples qui héritent de certains attributs de leurs groupes de contrôle parents.

Ces hiérarchies multiples et séparés sont nécessaires parce que chaque hiérarchie est attaché à un ou plusieurs sous-système(s) aussi appelés des **Contrôleurs de Ressources** ou simplement des **Contrôleurs**. Les contrôleurs disponibles sont :

- **blkio** - utilisé pour établir des limites sur l'accès des entrées/sorties à partir et depuis des périphériques blocs,
- **cpu** - utilisé pour fournir aux tâches des groupes de contrôle accès au CPU grâce au planificateur,
- **cpuacct** - utilisé pour produire des rapports automatiques sur les ressources CPU utilisées par les tâches dans un groupe de contrôle,
- **cpuset** - utilisé pour assigner des CPU individuels sur un système multicœur et des nœuds de mémoire à des tâches dans un groupe de contrôle,
- **devices** - utilisé pour autoriser ou pour refuser l'accès des tâches aux périphériques dans un groupe de contrôle,
- **freezer** - utilisé pour suspendre ou pour réactiver les tâches dans un groupe de contrôle,
- **memory** - utilisé pour établir les limites d'utilisation de la mémoire par les tâches d'un groupe de contrôle et pour générer des rapports automatiques sur les ressources mémoire utilisées par ces tâches,
- **net_cls** - utilisé pour repérer les paquets réseau avec un identifiant de classe (classid) afin de permettre au contrôleur de trafic Linux, **tc**,

d'identifier les paquets provenant d'une tâche particulière d'un groupe de contrôle.

- **perf_event** - utilisé pour permettre le monitoring des CGroups avec l'outil perf,
- **hugetlb** - utilisé pour limiter des ressources sur des pages de mémoire virtuelle de grande taille.

Les hiérarchies ont des points de montage dans le répertoire **/sys/fs/cgroup** :

```
trainee@debian9:~$ su -  
Mot de passe : fenestros  
root@debian9:~# ls -l /sys/fs/cgroup/  
total 0  
dr-xr-xr-x 2 root root 0 mai 26 08:52 blkio  
lrwxrwxrwx 1 root root 11 mai 26 08:52 cpu -> cpu,cpuacct  
lrwxrwxrwx 1 root root 11 mai 26 08:52 cpuacct -> cpu,cpuacct  
dr-xr-xr-x 2 root root 0 mai 26 08:52 cpu,cpuacct  
dr-xr-xr-x 2 root root 0 mai 26 08:52 cpuset  
dr-xr-xr-x 5 root root 0 mai 26 08:52 devices  
dr-xr-xr-x 2 root root 0 mai 26 08:52 freezer  
dr-xr-xr-x 2 root root 0 mai 26 08:52 memory  
lrwxrwxrwx 1 root root 16 mai 26 08:52 net_cls -> net_cls,net_prio  
dr-xr-xr-x 2 root root 0 mai 26 08:52 net_cls,net_prio  
lrwxrwxrwx 1 root root 16 mai 26 08:52 net_prio -> net_cls,net_prio  
dr-xr-xr-x 2 root root 0 mai 26 08:52 perf_event  
dr-xr-xr-x 5 root root 0 mai 26 08:52 pids  
dr-xr-xr-x 5 root root 0 mai 26 08:52 systemd
```

Systemd organise les processus dans chaque CGroup. Par exemple tous les processus démarrés par le serveur Apache se trouveront dans le même CGroup, y compris les scripts CGI. Ceci implique que la gestion des ressources en utilisant des hiérarchies est couplé avec l'arborescence des unités de Systemd.

En haut de l'arborescence des unités de Systemd se trouve la tranche root - **-.slice**, dont dépend :

- le **system.slice** - l'emplacement des services système,
- le **user.slice** - l'emplacement des sessions des utilisateurs,
- le **machine.slice** - l'emplacement des machines virtuelles et conteneurs.

En dessous des tranches peuvent se trouver :

- des **scopes** - des processus créés par **fork**,
- des **services** - des processus créés par une **Unité**.

Les slices peuvent être visualisés avec la commande suivante :

```
root@debian9:~# systemctl list-units --type=slice
UNIT                                LOAD    ACTIVE SUB    DESCRIPTION
-.slice                            loaded active active Root Slice
system-getty.slice                loaded active active system-getty.slice
system.slice                      loaded active active System Slice
user-1000.slice                   loaded active active User Slice of trainee
user-112.slice                    loaded active active User Slice of lightdm
user.slice                        loaded active active User and Session Slice
```

LOAD = Reflects whether the unit definition was properly loaded.
 ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
 SUB = The low-level unit activation state, values depend on unit type.

6 loaded units listed. Pass --all to see loaded but inactive units, too.
 To show all installed unit files use 'systemctl list-unit-files'.

L'arborescence des unités de Systemd est la suivante :

```
root@debian9:~# systemd-cgls
Control group /:
-.slice
├─user.slice
│   └─user-112.slice
│       └─user@112.service
│           └─dbus.service
│               └─539 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation
└─init.scope
```

```
| | | | | 527 /lib/systemd/systemd --user
| | | | | 528 (sd-pam)
| | | | | gvfs-daemon.service
| | | | | 547 /usr/lib/gvfs/gvfsd
| | | | | session-c1.scope
| | | | | 524 lightdm --session-child 18 21
| | | | | 532 /usr/sbin/lightdm-gtk-greeter
| | | | | 534 /usr/lib/at-spi2-core/at-spi-bus-launcher --launch-immediately
| | | | | 541 /usr/bin/dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibility.conf --nofork --print-
address 3
| | | | | 543 /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-session
| | | | | user-1000.slice
| | | | | | session-2.scope
| | | | | | | 668 sshd: trainee [priv]
| | | | | | | 679 sshd: trainee@pts/0
| | | | | | | 680 -bash
| | | | | | | 689 su -
| | | | | | | 690 -su
| | | | | | | 708 systemd-cgls
| | | | | | | 709 systemd-cgls
| | | | | | user@1000.service
| | | | | | | init.scope
| | | | | | | | 670 /lib/systemd/systemd --user
| | | | | | | | 671 (sd-pam)
| | | | | init.scope
| | | | | | 1 /sbin/init
| | | | | system.slice
| | | | | | lightdm.service
| | | | | | | 410 /usr/sbin/lightdm
| | | | | | | 425 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
| | | | | | | 588 lightdm --session-child 14 21
| | | | | | anacron.service
lines 1-39
```

En utilisant Systemd, plusieurs ressources peuvent être limitées :

- **CPUShares** - par défaut 1024,
- **MemoryLimit** - limite exprimée en Mo ou en Go. Pas de valeur par défaut,
- **BlockIOWeight** - valeur entre 10 et 1000. Pas de valeur par défaut,
- **StartupCPUShares** - comme CPUShares mais uniquement appliqué pendant le démarrage,
- **StartupBlockIOWeight** - comme BlockIOWeight mais uniquement appliqué pendant le démarrage,
- **CPUQuota** - utilisé pour limiter le temps CPU, même quand le système ne fait rien.



Important : Consultez le manuel `systemd.resource-control(5)` pour voir les paramètres CGroup qui peuvent être passés à `systemctl`.

1.3 - Limitation de la Mémoire

Pour travailler avec les CGroups dans Debian 9, il convient d'installer les outils nécessaires :

```
root@debian9:~# apt-get install libcgroup1 cgroup-tools
```

Commencez ensuite par créer le script **hello-world.sh** qui servira à générer un processus pour travailler avec les CGroups :

```
root@debian9:~# vi hello-world.sh
root@debian9:~# cat hello-world.sh
#!/bin/bash
while [ 1 ]; do
    echo "hello world"
    sleep 60
done
```

Rendez le script exécutable et testez-le :

```
root@debian9:~# chmod u+x hello-world.sh
```

```
root@debian9:~# ./hello-world.sh
hello world
hello world
^C
```

Créez maintenant un CGroup dans le sous-système **memory** appelé **helloworld** :

```
root@debian9:~# mkdir /sys/fs/cgroup/memory/helloworld
```

Par défaut, ce CGroup héritera de l'ensemble de la mémoire disponible. Pour éviter cela, créez maintenant une limite de **40000000** octets pour ce CGroup :

```
root@debian9:~# echo 40000000 > /sys/fs/cgroup/memory/helloworld/memory.limit_in_bytes
root@debian9:~# cat /sys/fs/cgroup/memory/helloworld/memory.limit_in_bytes
39997440
```



Important - Notez que les 40 000 000 demandés sont devenus 39 997 440 ce qui correspond à un nombre entier de pages mémoire du noyau de 4Ko. ($39\,997\,440 / 4096 = 9\,765$).

Lancez maintenant le script **helloworld.sh** :

```
root@debian9:~# ./hello-world.sh &
[1] 1012
root@debian9:~# hello world

root@debian9:~# ps aux | grep hello-world
root      1012  0.0  0.1 11172 2868 pts/0    S   10:24   0:00 /bin/bash ./hello-world.sh
root      1015  0.0  0.0 12784  968 pts/0    S+  10:25   0:00 grep hello-world
```

Notez l'héritage par défaut :


```
root@debian9:~# ps -ww -o cgroup 1012
CGROUP
6:devices:/user.slice,3:pids:/user.slice/user-1000.slice/session-2.scope,1:name=systemd:/user.slice/user-1000.slice/session-2.scope
```

Insérer le PID de notre script dans le CGroup **helloworld** :

```
root@debian9:~# echo 1012 > /sys/fs/cgroup/memory/helloworld/cgroup.procs
```

Notez maintenant l'héritage de la limitation de la mémoire - **2:memory:/helloworld** :

```
root@debian9:~# ps -ww -o cgroup 1012
CGROUP
6:devices:/user.slice,3:pids:/user.slice/user-1000.slice/session-2.scope,2:memory:/helloworld,1:name=systemd:/user.slice/user-1000.slice/session-2.scope
```

Constatez ensuite l'occupation mémoire réelle :

```
root@debian9:~# cat /sys/fs/cgroup/memory/helloworld/memory.usage_in_bytes
319488
```

Tuez le script **hello-world.sh** :

```
root@debian9:~# kill 1012
root@debian9:~# ps aux | grep hello-world
root      1038  0.0  0.0  12784   936 pts/0    S+   10:33   0:00 grep hello-world
[1]+  Complété                  ./hello-world.sh
```

Créez un second CGroup beaucoup plus restrictif :

```
root@debian9:~# mkdir /sys/fs/cgroup/memory/helloworld1
root@debian9:~# echo 6000 > /sys/fs/cgroup/memory/helloworld1/memory.limit_in_bytes
root@debian9:~# cat /sys/fs/cgroup/memory/helloworld1/memory.limit_in_bytes
```

4096

Relancez le script **hello-world.sh** et insérez-le dans le nouveau CGroup :

```
root@debian9:~# ./hello-world.sh &
[1] 1042
root@debian9:~# hello world

root@debian9:~# echo 1042 > /sys/fs/cgroup/memory/helloworld1/cgroup.procs
```

Attendez la prochaine sortie de **hello world** sur le canal standard puis constatez que le script s'arrête :

```
root@debian9:~# hello world

[1]+  Processus arrêté      ./hello-world.sh
```

Consultez en suite la fin du fichier **/var/log/messages** :

```
root@debian9:~# tail /var/log/messages
Jul 28 10:39:26 debian9 kernel: [ 1501.775169] [<fffffffffab6253d>] ? __do_page_fault+0x4bd/0x4f0
Jul 28 10:39:26 debian9 kernel: [ 1501.775171] [<fffffffffabfc2950>] ? SyS_brk+0x160/0x180
Jul 28 10:39:26 debian9 kernel: [ 1501.775175] [<fffffffffac41a358>] ? page_fault+0x28/0x30
Jul 28 10:39:26 debian9 kernel: [ 1501.775176] Task in /helloworld1 killed as a result of limit of /helloworld1
Jul 28 10:39:26 debian9 kernel: [ 1501.775180] memory: usage 4kB, limit 4kB, failcnt 17
Jul 28 10:39:26 debian9 kernel: [ 1501.775180] memory+swap: usage 0kB, limit 9007199254740988kB, failcnt 0
Jul 28 10:39:26 debian9 kernel: [ 1501.775181] kmem: usage 0kB, limit 9007199254740988kB, failcnt 0
Jul 28 10:39:26 debian9 kernel: [ 1501.775181] Memory cgroup stats for /helloworld1: cache:0KB rss:4KB
rss_huge:0KB mapped_file:0KB dirty:0KB writeback:0KB inactive_anon:0KB active_anon:0KB inactive_file:0KB
active_file:0KB unevictable:0KB
Jul 28 10:39:26 debian9 kernel: [ 1501.775188] [ pid ]    uid  tgid total_vm      rss nr_ptes nr_pmds swapents
oom_score_adj name
Jul 28 10:39:26 debian9 kernel: [ 1501.775219] [ 1042]        0  1042     2795      732      10        3         0
0 hello-world.sh
```



Important - Notez la trace **Task in /helloworld1 killed as a result of limit of /helloworld1.**

1.4 - Le Paquet cgroup-tools

Le paquet **cgroup-tools** installe des commandes dites *de facilité* dont :

La commande cgcreate

Cette commande permet la création d'un CGroup :

```
root@debian9:~# cgcreate -g memory:helloworld2
root@debian9:~# ls -l /sys/fs/cgroup/memory/helloworld2/
total 0
-rw-r--r-- 1 root root 0 juil. 28 11:09 cgroup.clone_children
--w--w--w- 1 root root 0 juil. 28 11:09 cgroup.event_control
-rw-r--r-- 1 root root 0 juil. 28 11:09 cgroup.procs
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.failcnt
--w----- 1 root root 0 juil. 28 11:09 memory.force_empty
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.failcnt
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.limit_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.max_usage_in_bytes
-r--r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.slabinfo
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.tcp.failcnt
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.tcp.limit_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.tcp.max_usage_in_bytes
-r--r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.tcp.usage_in_bytes
-r--r--r-- 1 root root 0 juil. 28 11:09 memory.kmem.usage_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.limit_in_bytes
```

```
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.max_usage_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.move_charge_at_immigrate
-r--r--r-- 1 root root 0 juil. 28 11:09 memory.numa_stat
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.oom_control
----- 1 root root 0 juil. 28 11:09 memory.pressure_level
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.soft_limit_in_bytes
-r--r--r-- 1 root root 0 juil. 28 11:09 memory.stat
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.swappiness
-r--r--r-- 1 root root 0 juil. 28 11:09 memory.usage_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 11:09 memory.use_hierarchy
-rw-r--r-- 1 root root 0 juil. 28 11:09 notify_on_release
-rw-r--r-- 1 root root 0 juil. 28 11:09 tasks
```

Il n'existe cependant pas de commande pour affecter une limitation de la mémoire :

```
root@debian9:~# echo 40000000 > /sys/fs/cgroup/memory/helloworld2/memory.limit_in_bytes
```

La Commande cgexec

Cette commande permet d'insérer la limitation dans le CGroup **et** de lancer le script en une seule ligne :

```
root@debian9:~# cgexec -g memory:helloworld2 ./hello-world.sh &
[2] 1860
root@debian9:~# hello world

root@debian9:~#
```

La Commande cgdelete

Une fois le script terminé, cette commande permet de supprimer le cgroup :

```
root@debian9:~# ps aux | grep *.sh
root      1073  0.0  0.1 11172 2868 pts/0    S   11:18   0:00 /bin/bash ./hello-world.sh
root      1076  0.0  0.0   528    4 pts/0    R+  11:18   0:00 grep hello-world.sh
root@debian9:~# kill 1073
root@debian9:~# ps aux | grep *.sh
root      1078  0.0  0.0 12784  920 pts/0    S+  11:18   0:00 grep hello-world.sh
[1]+  Complété                  cgexec -g memory:helloworld2 ./hello-world.sh
root@debian9:~# cgdelete memory:helloworld2
root@debian9:~# ls -l /sys/fs/cgroup/memory/helloworld2/
ls: impossible d'accéder à '/sys/fs/cgroup/memory/helloworld2/': Aucun fichier ou dossier de ce type
```

Le Fichier `/etc/cgconfig.conf`

Afin de les rendre persistants, il convient de créer les CGroups dans le fichier `/etc/cgconfig.conf` :

```
root@debian9:~# vi /etc/cgconfig.conf
root@debian9:~# cat /etc/cgconfig.conf
group helloworld2 {
    cpu {
        cpu.shares = 100;
    }
    memory {
        memory.limit_in_bytes = 40000;
    }
}
```



Important - Notez la création de **deux** limitations, une de 40 000 octets de mémoire et l'autre de **100 cpu.shares**. Cette dernière est une valeur exprimée sur 1 024, où 1 024 représente 100% du temps CPU. La limite fixée est donc équivalente à 9,77% du temps CPU.

Créez donc les deux CGroups concernés :

```
root@debian9:~# cgcreate -g memory:helloworld2
root@debian9:~# ls -l /sys/fs/cgroup/memory/helloworld2/
total 0
-rw-r--r-- 1 root root 0 juil. 28 12:47 cgroup.clone_children
--w--w--w- 1 root root 0 juil. 28 12:47 cgroup.event_control
-rw-r--r-- 1 root root 0 juil. 28 12:47 cgroup.procs
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.failcnt
--w----- 1 root root 0 juil. 28 12:47 memory.force_empty
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.failcnt
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.limit_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.max_usage_in_bytes
-r--r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.slabinfo
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.tcp.failcnt
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.tcp.limit_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.tcp.max_usage_in_bytes
-r--r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.tcp.usage_in_bytes
-r--r--r-- 1 root root 0 juil. 28 12:47 memory.kmem.usage_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.limit_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.max_usage_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.move_charge_at_immigrate
-r--r--r-- 1 root root 0 juil. 28 12:47 memory.numa_stat
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.oom_control
----- 1 root root 0 juil. 28 12:47 memory.pressure_level
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.soft_limit_in_bytes
-r--r--r-- 1 root root 0 juil. 28 12:47 memory.stat
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.swappiness
-r--r--r-- 1 root root 0 juil. 28 12:47 memory.usage_in_bytes
-rw-r--r-- 1 root root 0 juil. 28 12:47 memory.use_hierarchy
-rw-r--r-- 1 root root 0 juil. 28 12:47 notify_on_release
-rw-r--r-- 1 root root 0 juil. 28 12:47 tasks
```

```
root@debian9:~# cgcreate -g cpu:helloworld2
root@debian9:~# ls -l /sys/fs/cgroup/cpu/helloworld2/
total 0
```

```
-rw-r--r-- 1 root root 0 juil. 28 12:48 cgroup.clone_children
-rw-r--r-- 1 root root 0 juil. 28 12:48 cgroup.procs
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.stat
-rw-r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage_all
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage_percpu
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage_percpu_sys
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage_percpu_user
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage_sys
-r--r--r-- 1 root root 0 juil. 28 12:48 cpuacct.usage_user
-rw-r--r-- 1 root root 0 juil. 28 12:48 cpu.cfs_period_us
-rw-r--r-- 1 root root 0 juil. 28 12:48 cpu.cfs_quota_us
-rw-r--r-- 1 root root 0 juil. 28 12:48 cpu.shares
-r--r--r-- 1 root root 0 juil. 28 12:48 cpu.stat
-rw-r--r-- 1 root root 0 juil. 28 12:48 notify_on_release
-rw-r--r-- 1 root root 0 juil. 28 12:48 tasks
```

Appliquez le contenu du fichier **/etc/cgconfig.conf** grâce à l'utilisation de la commande **cgconfigparser** :

```
root@debian9:~# cgconfigparser -l /etc/cgconfig.conf
root@debian9:~# cat /sys/fs/cgroup/memory/helloworld2/memory.limit_in_bytes
36864
root@debian9:~# cat /sys/fs/cgroup/cpu/helloworld2/cpu.shares
100
```

LAB #2 - Utilisation des Docker Secrets

Les secrets Docker sont une façon sécurisée de stocker des informations sensibles telles les noms d'utilisateurs, les mots de passe voire les fichiers. Ils ne sont compatibles qu'avec Docker en mode Swarm.

Considérez l'exemple suivant d'un fichier Docker stack servant à créer un conteneur PostgreSQL :

```
version: '3.1'

services:

  db:
    image: postgres
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: database

  adminer:
    image: adminer
    ports:
      - 8080:8080
```

On peut constater dans ce fichier la présence des informations sensibles en non-sécurisées :

- POSTGRES_USER
- POSTGRES_PASSWORD
- POSTGRES_DB

Afin de sécuriser ces informations, commencez par créer le contexte **postgres** :

```
root@manager:~# mkdir postgres
```

Créez ensuite un Docker Secret appelé **pg_user** :

```
root@manager:~# cd postgres
root@manager:~/postgres# echo "postgres" | docker secret create pg_user -
lpk8eq80qvfiqw7z1686fmj5t
```





Important : Notez l'utilisation du caractère - à la fin de la ligne. Celui-ci indique à la commande **docker secret** de lire le contenu du secret `pg_user` à partir de l'entrée standard.

Pour visualiser la liste des secrets, utilisez la commande `docker secrets ls` :

```
root@manager:~/postgres# docker secret ls
```

ID	NAME	DRIVER	CREATED	UPDATED
lpk8eq80qvfiqw7z1686fmj5t	pg_user		About a minute ago	About a minute ago



Important : Notez que la colonne **DRIVER** est vide. Ceci indique que la gestion des secrets est accomplie par Docker lui-même au lieu d'être déléguée à un plugin tiers.

Créez maintenant les secrets **pg_password** et **pg_database** :

```
root@manager:~/postgres# echo "postgres" | docker secret create pg_password -  
h9tsfbfwz6o0sd35roklwpopi
```

```
root@manager:~/postgres# echo "database" | docker secret create pg_database -  
5lx4zydpfocwgpdto0yy1jod9
```



Important : Notez qu'un secret Docker est immuable.

Vérifiez la prise en compte de vos commandes :

```
root@manager:~/postgres# docker secret ls
```

ID	NAME	DRIVER	CREATED	UPDATED
5lx4zydpfocwgpdto0yy1jod9	pg_database		2 minutes ago	2 minutes ago

h9tsfbfwz6o0sd35roklwpopi	pg_password	3 minutes ago	3 minutes ago
lpk8eq80qvfiqw7z1686fmj5t	pg_user	5 minutes ago	5 minutes ago

Pour obtenir de l'information concernant un secret, il convient d'utiliser la commande docker secret **inspect** :

```
root@manager:~/postgres# docker secret inspect pg_database
[
  {
    "ID": "5lx4zydpfocwgpdto0yy1jod9",
    "Version": {
      "Index": 23
    },
    "CreatedAt": "2021-04-15T03:49:36.344367554Z",
    "UpdatedAt": "2021-04-15T03:49:36.344367554Z",
    "Spec": {
      "Name": "pg_database",
      "Labels": {}
    }
  }
]
```



Important : On peut constater dans la sortie de cette commande la valeur **CreatedAt** qui correspond à la date de création du secret ainsi que **UpdatedAt** qui correspond à la date de modification du secret.

L'option **-pretty** de la commande fait apparaître ces informations plus clairement :

```
root@manager:~/postgres# docker secret inspect --pretty pg_database
ID:          5lx4zydpfocwgpdto0yy1jod9
Name:        pg_database
Driver:
```

```
Created at:      2021-04-15 03:49:36.344367554 +0000 utc
Updated at:      2021-04-15 03:49:36.344367554 +0000 utc
```

Créez maintenant le fichier compose **postgres-secrets.yaml** :

```
root@manager:~/postgres# vi postgres-secrets.yaml
root@manager:~/postgres# cat postgres-secrets.yaml
version: '3.1'

services:

  db:
    image: postgres
    restart: always
    environment:
      POSTGRES_USER_FILE: /run/secrets/pg_user
      POSTGRES_PASSWORD_FILE: /run/secrets/pg_password
      POSTGRES_DB_FILE: /run/secrets/pg_database
    secrets:
      - pg_password
      - pg_user
      - pg_database

  adminer:
    image: adminer
    ports:
      - 8080:8080

secrets:
  pg_user:
    external: true
  pg_password:
    external: true
  pg_database:
```

```
external: true
```

Notez que dans ce fichier les trois variables **POSTGRES_USER**, **POSTGRES_PASSWORD** et **POSTGRES_DB** ont un suffixe **_FILE** car elles référencent des **fichiers** contenant des informations plutôt que des informations elles-mêmes. Ces fichiers se trouvent dans le répertoire **/run/secrets** du **conteneur**.

Deuxièmement la section suivantes spécifie les noms des secrets à utiliser avec le service :

```
secrets:  
  - pg_password  
  - pg_user  
  - pg_database
```

La dernière section spécifie que les secrets sont **externes** :

```
secrets:  
  pg_user:  
    external: true  
  pg_password:  
    external: true  
  pg_database:  
    external: true
```



Important : Le terme **externe** indique que les secrets ne seront pas stockés dans l'image construite mais **uniquement** dans le conteneur créé.

Déployez maintenant le service en utilisant la commande **docker stack** :

```
root@manager:~/postgres# docker stack deploy -c postgres-secrets.yaml postgres  
Ignoring unsupported options: restart
```

```
Creating network postgres_default
Creating service postgres_db
Creating service postgres_adminer
```



Important : Notez la présence de l'erreur **Ignoring unsupported options: restart**. Celle-ci est due au fait que la directive **restart** est compatible avec la commande **docker-compose** mais pas avec la commande **docker stack**. La directive qui aurait dû être utilisée dans le fichier est **restart_policy**.

Connectez-vous maintenant à Apache Guacamole et ouvrez un navigateur web dans la machine virtuelle. Naviguez ensuite à l'adresse du Manager sur le port **8080** et renseignez les valeurs des secrets :

Authentification - Adminer - Mozilla Firefox

10.0.2.62:8080

Langue: français

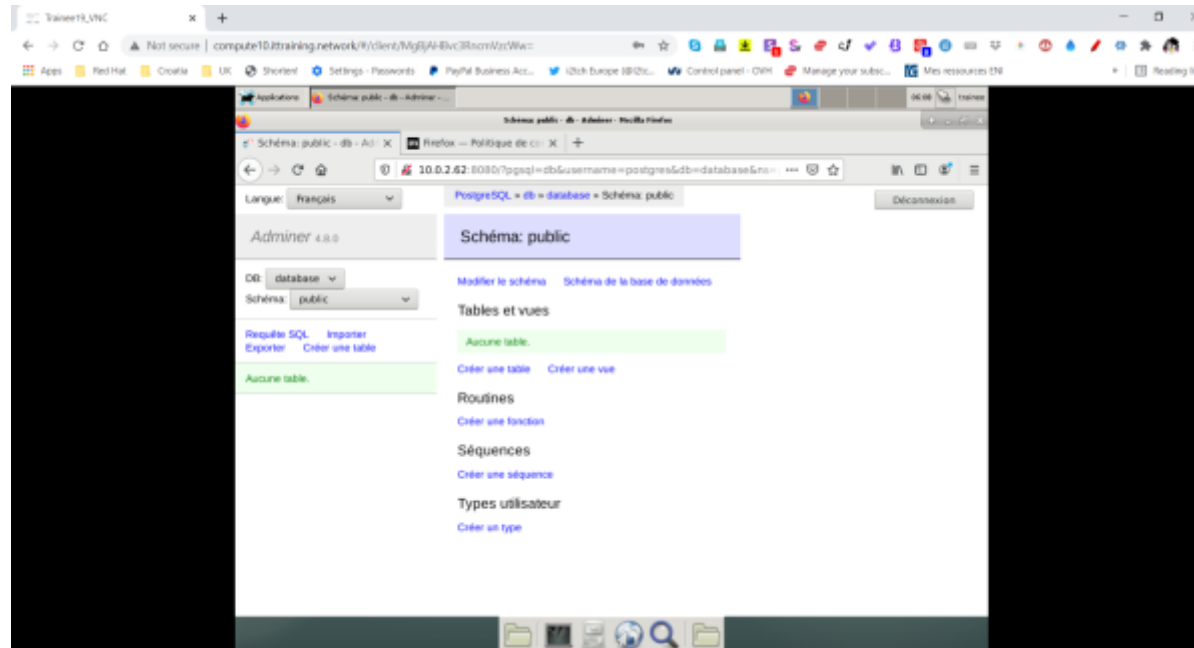
Adminer 4.8.0

Authentification

Système	PostgreSQL
Serveur	db
Utilisateur	postgres
Mot de passe	••••••••
Base de données	database

Authentification ☐ Authentification permanente

Validez le formulaire et vérifiez que les secrets ont été pris en compte :



Dernièrement, supprimez le service :

```
root@manager:~/postgres# docker stack ls
NAME                SERVICES                ORCHESTRATOR
postgres            2                        Swarm
root@manager:~/postgres# docker stack rm postgres
Removing service postgres_adminer
Removing service postgres_db
Removing network postgres_default
```

LAB #3 - Création d'un Utilisateur de Confiance pour Contrôler le Daemon Docker

Au contraire des solutions classiques de gestion de machines virtuelles où l'accès est souvent conditionné à l'attribution de rôles, Docker ne possède pas ce type de mécanisme. De ce fait toute personne ayant accès à l'hôte soit par **sudo** soit en étant membre du groupe **docker** peut accéder à tous les conteneurs voire les arrêter, supprimer et en créer d'autres.

```
root@manager:~# cat /etc/group | grep docker
docker:x:999:
root@manager:~# usermod -aG docker trainee
root@manager:~# exit
déconnexion
trainee@manager:~$ docker ps
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get
http://%2Fvar%2Frun%2Fdocker.sock/v1.40/containers/json: dial unix /var/run/docker.sock: connect: permission
denied
trainee@manager:~$ newgrp docker
trainee@manager:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
d02c6115724c	alpine	"/bin/sh"	6 days ago	Exited (0) 6 days ago

```
alpine1
trainee@manager:~$ docker rm alpine1
alpine1
trainee@manager:~$ docker run -d --name alpine1 alpine sleep 99999
a214e2df0499c97e8da25a6c9ea751ac75344c9bcd7d238f8cb8d5c777510ab9
trainee@manager:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
a214e2df0499	alpine	"/bin/sh"	6 seconds ago	Up 5 seconds	

```
alpine1
```

LAB #4 - Le Script docker-bench-security.sh

Le **Center for Internet Security (CIS)** est une organisation indépendante à but non-lucratif qui publie des best practices dans de nombreux domaines de l'informatique. Le guide pour Docker peut être téléchargé à partir de l'adresse <https://www.cisecurity.org/benchmark/docker/>.

Le guide est divisé en plusieurs sections :

- La configuration de l'hôte Docker,
- La configuration du daemon Docker,
- Les fichiers de configuration du daemon Docker,
- Les images ainsi que les fichiers servant à la construction des images,
- Le container runtime,
- Les opérations sécuritaires relatives à Docker,
- La configuration de Docker Swarm.

Ce guide est à utiliser avec le script **Docker Benchmark Security**.

Clonez le script **docker-bench-security.sh** en utilisant **git** :

```
trainee@manager:~$ su -  
Mot de passe : fenestros  
root@manager:~# git clone https://github.com/docker/docker-bench-security.git  
Clonage dans 'docker-bench-security'...  
remote: Enumerating objects: 18, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (16/16), done.  
remote: Total 1921 (delta 5), reused 6 (delta 2), pack-reused 1903  
Réception d'objets: 100% (1921/1921), 2.90 MiB | 908.00 KiB/s, fait.  
Résolution des deltas: 100% (1339/1339), fait.
```

Exécutez maintenant le script **Docker Benchmark Security** :

```
root@manager:~# cd docker-bench-security/  
root@manager:~/docker-bench-security# ./docker-bench-security.sh  
# -----  
# Docker Bench for Security v1.3.5  
#  
# Docker, Inc. (c) 2015-  
#  
# Checks for dozens of common best-practices around deploying Docker containers in production.  
# Inspired by the CIS Docker Benchmark v1.2.0.
```



```
# -----  
  
Initializing vendredi 8 novembre 2019, 12:12:03 (UTC+0100)  
  
[INFO] 1 - Host Configuration  
  
[INFO] 1.1 - General Configuration  
[NOTE] 1.1.1 - Ensure the container host has been Hardened  
[INFO] 1.1.2 - Ensure Docker is up to date  
[INFO]      * Using 19.03.4, verify is it up to date as deemed necessary  
[INFO]      * Your operating system vendor may provide support and security maintenance for Docker  
  
[INFO] 1.2 - Linux Hosts Specific Configuration  
[WARN] 1.2.1 - Ensure a separate partition for containers has been created  
...
```

Ce script sert à automatiser le contrôle des points précédemment cités et produit un rapport contenant des annotations :

- **[PASS]** : Concerne les points qui n'ont pas besoin d'être modifiés,
- **[WARN]** : Concerne les points qui **doivent** être modifiés,
- **[INFO]** : Concerne les points qui doivent être passés en revue selon les besoins de votre configuration,
- **[NOTE]** : Vous informe d'un **best practice**.

LAB #5 - Sécurisation de la Configuration de l'Hôte Docker

Lors de l'exécution du script, vous devez obtenir un résultat similaire à ceci en ce qui concerne la Sécurité de la Configuration de l'hôte Docker :

```
...  
[INFO] 1 - Host Configuration  
  
[INFO] 1.1 - General Configuration
```

```
[NOTE] 1.1.1 - Ensure the container host has been Hardened
[INFO] 1.1.2 - Ensure Docker is up to date
[INFO]      * Using 19.03.4, verify is it up to date as deemed necessary
[INFO]      * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created
[INFO] 1.2.2 - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:999:trainee
[WARN] 1.2.3 - Ensure auditing is configured for the Docker daemon
[WARN] 1.2.4 - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5 - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.2.6 - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.2.7 - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.2.8 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO] 1.2.9 - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
[INFO]      * File not found
[INFO] 1.2.10 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.2.11 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
[INFO] 1.2.12 - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc
[INFO]      * File not found
...
```

Les problèmes de sécurité qu'il convient à résoudre sont indiqués par les annotations **[WARN]**.

5.1 - **[WARN] 1.2.1 - Ensure a separate partition for containers has been created**

Par défaut, tous les fichiers de Docker sont stockés dans le répertoire **/var/lib/docker**, y compris toutes les images, tous les conteneurs et tous les volumes. Sur un système hôte n'ayant qu'une seule partition il y a un risque, tous comme le risque lié au répertoire **/var/log/**, que le disque devient saturé.

5.2 - [WARN] 1.2.3 - Ensure auditing is configured for the Docker daemon

```
[WARN] 1.2.4 - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.2.5 - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.2.6 - Ensure auditing is configured for Docker files and directories - docker.service
[WARN] 1.2.7 - Ensure auditing is configured for Docker files and directories - docker.socket
[WARN] 1.2.8 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[WARN] 1.2.11 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
```

Ces avertissements sont présents parce que **auditd** n'est pas installé et parce qu'il n'y a pas de règles spécifiques au daemon Docker et ses répertoires et fichiers associés.

Pour installer auditd, utilisez **apt-get** :

```
root@manager:~/docker-bench-security# apt-get install auditd
```

Modifiez ensuite le fichier **/etc/audit/rules.d/audit.rules** :

```
root@manager:~/docker-bench-security# vi /etc/audit/rules.d/audit.rules
root@manager:~/docker-bench-security# cat /etc/audit/rules.d/audit.rules
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 0

## Set failure mode to syslog
-f 1
```

##Docker

```
-w /usr/bin/docker -p wa
-w /var/lib/docker -p wa
-w /etc/docker -p wa
-w /lib/systemd/system/docker.service -p wa
-w /lib/systemd/system/docker.socket -p wa
-w /etc/default/docker -p wa
-w /etc/docker/daemon.json -p wa
-w /usr/bin/docker-containerd -p wa
-w /usr/bin/docker-runc -p wa
-w /usr/bin/containerd -p wa
```



Important : L'option **-w** indique **watch** et concerne le fichier qui suit. L'option **-p** journalise les modifications éventuelles.

Re-démarrez ensuite auditd :

```
root@manager:~/docker-bench-security# systemctl restart auditd
```

Vérifiez ensuite la prise en charge des règles :

```
root@manager:~/docker-bench-security# cat /etc/audit/audit.rules
## This file is automatically generated from /etc/audit/rules.d
-D
-b 8192
-f 1
--backlog_wait_time 0
-w /usr/bin/docker -p wa
-w /var/lib/docker -p wa
-w /etc/docker -p wa
-w /lib/systemd/system/docker.service -p wa
```

```
-w /lib/systemd/system/docker.socket -p wa
-w /etc/default/docker -p wa
-w /etc/docker/daemon.json -p wa
-w /usr/bin/docker-containerd -p wa
-w /usr/bin/docker-runc -p wa
-w /usr/bin/containerd -p wa
```



Important - Pour plus d'information concernant la création de règles personnalisées avec auditd, consultez cette [page](#).

Ré-exécutez le script **Docker Benchmark Security** :

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[PASS] 1.2.4 - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[PASS] 1.2.5 - Ensure auditing is configured for Docker files and directories - /etc/docker
[PASS] 1.2.6 - Ensure auditing is configured for Docker files and directories - docker.service
[PASS] 1.2.7 - Ensure auditing is configured for Docker files and directories - docker.socket
[PASS] 1.2.8 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
...
[PASS] 1.2.11 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
...
```

LAB #6 - Sécurisation de la Configuration du daemon Docker

Exécutez de nouveau le script **docker-bench-security.sh**. Vous devez obtenir un résultat similaire à ceci en ce qui concerne la sécurité de la configuration du daemon Docker :

```
...
```

```
[INFO] 2 - Docker daemon configuration
[WARN] 2.1 - Ensure network traffic is restricted between containers on the default bridge
[PASS] 2.2 - Ensure the logging level is set to 'info'
[PASS] 2.3 - Ensure Docker is allowed to make changes to iptables
[PASS] 2.4 - Ensure insecure registries are not used
[PASS] 2.5 - Ensure aufs storage driver is not used
[INFO] 2.6 - Ensure TLS authentication for Docker daemon is configured
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.7 - Ensure the default ulimit is configured appropriately
[INFO]      * Default ulimit doesn't appear to be set
[WARN] 2.8 - Enable user namespace support
[PASS] 2.9 - Ensure the default cgroup usage has been confirmed
[PASS] 2.10 - Ensure base device size is not changed until needed
[WARN] 2.11 - Ensure that authorization for Docker client commands is enabled
[WARN] 2.12 - Ensure centralized and remote logging is configured
[PASS] 2.13 - Ensure live restore is Enabled (Incompatible with swarm mode)
[WARN] 2.14 - Ensure Userland Proxy is Disabled
[PASS] 2.15 - Ensure that a daemon-wide custom seccomp profile is applied if appropriate
[PASS] 2.16 - Ensure that experimental features are not implemented in production
[WARN] 2.17 - Ensure containers are restricted from acquiring new privileges
...
```

Les problèmes de sécurité qu'il convient à résoudre sont indiqués par les annotations **[WARN]**.

6.1 - **[WARN] 2.1 - Ensure network traffic is restricted between containers on the default bridge**

Par défaut Docker permet un trafic réseau sans restrictions entre des conteneurs sur le même hôte. Il est cependant possible de modifier la configuration par défaut. Pour empêcher ceci, il faut fixer la valeur de **icc** à **false**. De cette façon, docker crée des conteneurs qui peuvent communiquer entre eux **uniquement** s'il existe un lien.

Pour plus d'informations, consultez cette [page](#).

6.2 - [WARN] 2.8 - Enable user namespace support

Cet avertissement nous indique que l'utilisation des **user namespaces** n'est pas activée. Le support des **user namespaces** du noyau Linux permet d'attribuer une plage d'UIDs et de GIDs unique à un processus et donc à un conteneur, en dehors de la plage traditionnelle utilisée par l'hôte Docker. L'avantage ici est que les processus ayant l'UID de root dans le conteneur seront mappés à un UID sans privilèges dans l'hôte Docker. Pour utiliser user namespace, il faut fixer la valeur de **userns-remap** à **default**. Dans ce cas précis Docker crée un utilisateur dénommé **dockremap**. Notez qu'il est aussi possible de fixer vos propres valeurs avec **"userns-remap": "user:group"**.

Pour plus d'informations, consultez cette [page](#).

6.3 - [WARN] 2.11 - Ensure that authorization for Docker client commands is enabled

Par défaut, Docker permet un accès sans restrictions aux daemon Docker. Il est possible de restreindre l'accès à des utilisateurs authentifiés en utilisant un plug-in. Cette ligne est sans importance parce que l'accès au socket local Docker est limité aux membres du groupe **docker** (voir DOF202 - La Sécurité de la Configuration de l'Hôte Docker)

Pour plus d'informations, consultez cette [page](#).

6.4 - [WARN] 2.12 - Ensure centralized and remote logging is configured

Cet avertissement indique que la configuration de rsyslog ne permet pas l'envoi des traces vers un serveur de journalisation distant. Elle indique aussi que la valeur de **log-driver** n'a pas été spécifiée. Pour activer cette configuration, il faut fixer la valeur de **log-driver** à **syslog** puis configurer **syslog** ainsi que la valeur de **log-opts** correctement.

Pour plus d'informations, consultez cette [page](#).

6.5 - [WARN] 2.14 - Ensure Userland Proxy is Disabled

Il existe deux méthodes pour qu'un conteneur puisse router vers l'extérieur :

- le mode **Hairpin NAT**,
- **Userland Proxy**.

Il est préférable d'utiliser le mode Hairpin NAT qui peut utiliser Iptables et qui possède de meilleures performances. La plupart des systèmes d'opération modernes peuvent utiliser le mode Hairpin NAT. Pour désactiver Userland Proxy, il faut fixer la valeur de **userland-proxy** à **false**.

Pour plus d'informations, consultez cette [page](#).

6.6 - [WARN] 2.17 - Ensure containers are restricted from acquiring new privileges

Par défaut un conteneur peut obtenir une escalade de privilèges en utilisant les binaires setuid ou setgid. Pour interdire ceci il faut fixer la valeur de **no-new-privileges** à **true**.

Pour plus d'informations, consultez cette [page](#).

6.7 - Le Fichier `/etc/docker/daemon.json`

Créez le fichier `/etc/docker/daemon.json` :

```
root@manager:~/docker-bench-security# vi /etc/docker/daemon.json
root@manager:~/docker-bench-security# cat /etc/docker/daemon.json
{
  "icc": false,
  "userns-remap": "default",
  "log-driver": "syslog",
  "live-restore": true,
  "userland-proxy": false,
  "no-new-privileges": true
}
```

Notez ici que **live-restore** est fixé à **true**. Ceci permet aux conteneurs de continuer à fonctionner même quand le daemon Docker ne fonctionne pas. Ceci est utile pendant la mise-à-jour de Docker.

Re-démarrez le service Docker :

```
root@manager:~/docker-bench-security# systemctl restart docker
```

Vérifiez la présence de l'utilisateur dénommé **dockremap** :

```
root@manager:~/docker-bench-security# id dockremap
uid=116(dockremap) gid=121(dockremap) groupes=121(dockremap)
```

Ré-exécutez le script **Docker Benchmark Security** :

```
root@manager:~# cd docker-bench-security/
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[PASS] 2.1 - Ensure network traffic is restricted between containers on the default bridge
...
[PASS] 2.8 - Enable user namespace support
...
[WARN] 2.11 - Ensure that authorization for Docker client commands is enabled
[PASS] 2.12 - Ensure centralized and remote logging is configured
...
[PASS] 2.14 - Ensure Userland Proxy is Disabled
...
[PASS] 2.17 - Ensure containers are restricted from acquiring new privileges
...
```

Pour plus d'informations, consultez cette [page](#).

LAB #7 - Sécurisation des Images et les Fichiers de Construction

Créez le conteneur mysql :

```
root@manager:~/docker-bench-security# docker container run -d --name mysql -e MYSQL_ROOT_PASSWORD=password mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
80369df48736: Pull complete
e8f52315cb10: Pull complete
cf2189b391fc: Pull complete
cc98f645c682: Pull complete
27a27ac83f74: Pull complete
fa1f04453414: Pull complete
d45bf7d22d33: Pull complete
3dbac26e409c: Pull complete
9017140fb8c1: Pull complete
b76dda2673ae: Pull complete
bea9eb46d12a: Pull complete
elf050a38d0f: Pull complete
Digest: sha256:7345ce4ce6f0c1771d01fa333b8edb2c606ca59d385f69575f8e3e2ec6695eee
Status: Downloaded newer image for mysql:latest
54606c03c52c5e3ec0328029d69b869d4b285fb433015576dedc8b8dd4ad0494
root@manager:~/docker-bench-security# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
54606c03c52c	mysql	"docker-entrypoint.s..."	5 seconds ago	Up 4 seconds
3306/tcp	mysql			

Exécutez de nouveau le script **docker-bench-security.sh**. Vous devez obtenir un résultat similaire à ceci en ce qui concerne la sécurité des images et les fichiers de leur construction :

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[INFO] 4 - Container Images and Build File
[WARN] 4.1 - Ensure a user for the container has been created
[WARN]      * Running as root: mysql
[NOTE] 4.2 - Ensure that containers use only trusted base images
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container
```

```
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches
[WARN] 4.5 - Ensure Content trust for Docker is Enabled
[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images
[WARN]      * No Healthcheck found: [alpine:latest]
[WARN]      * No Healthcheck found: [mysql:latest]
[INFO] 4.7 - Ensure update instructions are not use alone in the Dockerfile
[INFO]      * Update instruction found: [mysql:latest]
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed
[PASS] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles
[NOTE] 4.11 - Ensure only verified packages are installed
...
```

7.1 - [WARN] 4.1 - Ensure a user for the container has been created

Les processus dans le conteneur **mysql** tourne sous l'UID de root. Ceci est l'action par défaut de Docker.

Pour plus d'informations, consultez cette [page](#).

7.2 - [WARN] 4.5 - Ensure Content trust for Docker is Enabled

Cette ligne indique que le support de Content trust n'a pas été activé. Content trust permet de s'assurer de la provenance des images utilisées car celles-ci sont signées.

Pour activer le Content trust, il faut positionner la valeur de la variable **DOCKER_CONTENT_TRUST** à **1** :

```
root@manager:~/docker-bench-security# echo "DOCKER_CONTENT_TRUST=1" | sudo tee -a /etc/environment
DOCKER_CONTENT_TRUST=1
root@manager:~/docker-bench-security# source /etc/environment
```

Re-démarrez la machine virtuelle **Manager** et démarrez le conteneur **mysql** :

```
root@manager:~/docker-bench-security# docker container start mysql
mysql
root@manager:~/docker-bench-security# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
54606c03c52c	mysql	"docker-entrypoint.s..."	10 minutes ago	Up 2 seconds	
3306/tcp, 33060/tcp	mysql				

Exécutez de nouveau le script et notez le contenu de la section 4 :

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[INFO] 4 - Container Images and Build File
[WARN] 4.1 - Ensure a user for the container has been created
[WARN]      * Running as root: mysql
[NOTE] 4.2 - Ensure that containers use only trusted base images
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches
[PASS] 4.5 - Ensure Content trust for Docker is Enabled
[WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images
[WARN]      * No Healthcheck found: [alpine:latest]
[WARN]      * No Healthcheck found: [mysql:latest]
[INFO] 4.7 - Ensure update instructions are not use alone in the Dockerfile
[INFO]      * Update instruction found: [mysql:latest]
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed
[PASS] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles
[NOTE] 4.11 - Ensure only verified packages are installed
...
```

Pour plus d'informations, consultez cette [page](#).

7.3 - [WARN] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images

Quand une image est construite il est possible d'y mettre un **HEALTHCHECK** dont le statut peut être vérifié par Docker afin de relancer le conteneur si nécessaire.

Pour mettre en place un HEALTHCHECK, il conviendrait, par exemple, d'inclure la ligne suivante dans le fichier DOCKERFILE servant à construire l'image :

```
HEALTHCHECK --interval=20s --timeout=3s CMD curl -f http://localhost:8000/ || exit 1
```

Ce test permet de vérifier que le conteneur peut atteindre l'URL indiqué tous les 20 secondes et produit une erreur au bout de 3 secondes.

Pour plus d'informations, consultez cette [page](#).

LAB #8 - Sécurisation du Container Runtime

Exécutez de nouveau le script **docker-bench-security.sh**, vous devez obtenir un résultat similaire à ceci en ce qui concerne la sécurité du Container Runtime :

```
root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[INFO] 5 - Container Runtime
[WARN] 5.1 - Ensure that, if applicable, an AppArmor Profile is enabled
[WARN]      * No AppArmorProfile Found: mysql
[WARN] 5.2 - Ensure that, if applicable, SELinux security options are set
[WARN]      * No SecurityOptions Found: mysql
[PASS] 5.3 - Ensure Linux Kernel Capabilities are restricted within containers
[PASS] 5.4 - Ensure that privileged containers are not used
[PASS] 5.5 - Ensure sensitive host system directories are not mounted on containers
[PASS] 5.6 - Ensure sshd is not run within containers
[PASS] 5.7 - Ensure privileged ports are not mapped within containers
[NOTE] 5.8 - Ensure that only needed ports are open on the container
```

```
[PASS] 5.9 - Ensure the host's network namespace is not shared
[WARN] 5.10 - Ensure that the memory usage for containers is limited
[WARN] * Container running without memory restrictions: mysql
[WARN] 5.11 - Ensure CPU priority is set appropriately on the container
[WARN] * Container running without CPU restrictions: mysql
[WARN] 5.12 - Ensure that the container's root filesystem is mounted as read only
[WARN] * Container running with root FS mounted R/W: mysql
[PASS] 5.13 - Ensure that incoming container traffic is bound to a specific host interface
[WARN] 5.14 - Ensure that the 'on-failure' container restart policy is set to '5'
[WARN] * MaximumRetryCount is not set to 5: mysql
[PASS] 5.15 - Ensure the host's process namespace is not shared
[PASS] 5.16 - Ensure the host's IPC namespace is not shared
[PASS] 5.17 - Ensure that host devices are not directly exposed to containers
[INFO] 5.18 - Ensure that the default ulimit is overwritten at runtime if needed
[INFO] * Container no default ulimit override: mysql
[PASS] 5.19 - Ensure mount propagation mode is not set to shared
[PASS] 5.20 - Ensure the host's UTS namespace is not shared
[PASS] 5.21 - Ensure the default seccomp profile is not Disabled
[NOTE] 5.22 - Ensure docker exec commands are not used with privileged option
[NOTE] 5.23 - Ensure that docker exec commands are not used with the user=root option
[PASS] 5.24 - Ensure that cgroup usage is confirmed
[WARN] 5.25 - Ensure that the container is restricted from acquiring additional privileges
[WARN] * Privileges not restricted: mysql
[WARN] 5.26 - Ensure that container health is checked at runtime
[WARN] * Health check not set: mysql
[INFO] 5.27 - Ensure that Docker commands always make use of the latest version of their image
[WARN] 5.28 - Ensure that the PIDs cgroup limit is used
[WARN] * PIDs limit not set: mysql
[INFO] 5.29 - Ensure that Docker's default bridge 'docker0' is not used
[INFO] * Container in docker0 network: mysql
[PASS] 5.30 - Ensure that the host's user namespaces are not shared
[PASS] 5.31 - Ensure that the Docker socket is not mounted inside any containers
...
```

Les problèmes de sécurité qu'il convient à résoudre sont indiqués par les annotations **[WARN]**.

8.1 - [WARN] 5.1 - Ensure AppArmor Profile is Enabled

Cet avertissement est présent parce que le conteneur n'utilise pas AppArmor.

Pour plus d'informations, consultez cette [page](#).

8.2 - [WARN] 5.2 - Ensure SELinux security options are set, if applicable

Cet avertissement est présent parce que le conteneur n'utilise pas SELinux.

Pour plus d'informations, consultez cette [page](#).

8.3 - [WARN] 5.10 - Ensure memory usage for container is limited

Cet avertissement est du au fait que les conteneurs ont automatiquement accès à la totalité de la RAM de l'hôte Docker :

```
root@manager:~# docker run -d -p 8081:80 nginx
b04b2a6f0dd93da21a8b7640afc319406e42868a141f90936dbcf52ab5bffb0d
root@manager:~# docker stats
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O
BLOCK I/O	PIDS				
b04b2a6f0dd9	dazzling_blackburn	0.00%	1.789MiB / 1.957GiB	0.09%	2.38kB /
0B	0B / 0B	2			
^C					

Supprimez le conteneur et re-créez le avec une limite de mémoire :

```
root@manager:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	
PORTS	NAMES				
b04b2a6f0dd9	nginx	"nginx -g 'daemon of..."	About a minute ago	Up About a minute	
0.0.0.0:8081->80/tcp	dazzling_blackburn				
5b31fe1e13bc	ubuntu	"bash -c ':() { : ..."	14 minutes ago	Exited (254) 13 minutes ago	
pensive_fermat					
7788c67c3b69	mysql	"docker-entrypoint.s..."	About an hour ago	Exited (255) 18 minutes ago	
3306/tcp, 33060/tcp	mysql				
root@manager:~# docker rm -f b0					
b0					
root@manager:~# docker run -d -p 8081:80 --memory="256m" nginx					
095472e5096a57277230ff94822d9bd0ad479ad26a33cbf83ec381cdb02910e1					
root@manager:~# docker stats					
CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O
BLOCK I/O	PIDS				
095472e5096a	affectionate_goldstine	0.00%	1.805MiB / 256MiB	0.70%	2.38kB /
0B	0B / 0B	2			
^C					

Pour plus d'informations, consultez cette [page](#).

8.4 - [WARN] 5.11 - Ensure CPU priority is set appropriately on the container

Cet avertissement est du au fait que les conteneurs ont automatiquement accès à tous les CPU de l'hôte Docker. Pour limiter cet accès, plusieurs options sont possibles dont le plus couramment utilisée est **-cpu-shares**.

La valeur de cpu-shares est relative à la valeur par défaut de **1024**. Une valeur de 512 permet au conteneur d'accéder à 50% des cycles du CPU mais uniquement quand les cycles sont limités. Quand les cycles de CPU ne sont pas restreints, chaque conteneur utilise autant qu'il en a besoin.

Pour plus d'informations, consultez cette [page](#).

8.5 - [WARN] 5.12 - Ensure the container's root filesystem is mounted as read only

Afin de minimiser le risque de compromettre un conteneur par la présence de code malicieux, il est conseillé de démarrer les conteneurs en lecture seule, sauf pour les volumes qui nécessitent un accès en écriture/lecture.

Créez le fichier **write_a_file** dans le conteneur **mysql** :

```
root@manager:~/docker-bench-security# docker container exec mysql touch /write_a_file
```

La Commande **docker container diff** indique les différences apportées au conteneur par rapport à l'image dont il est issu :

```
root@manager:~/docker-bench-security# docker container diff mysql
A /write_a_file
C /run
C /run/mysqld
A /run/mysqld/mysqld.sock
A /run/mysqld/mysqld.sock.lock
A /run/mysqld/mysqldx.sock
A /run/mysqld/mysqldx.sock.lock
A /run/mysqld/mysqld.pid
```



Important : Notez que la sortie indique les changements apportés au conteneur.

Arrêtez et supprimez le conteneur :

```
root@manager:~/docker-bench-security# docker container stop mysql
mysql
root@manager:~/docker-bench-security# docker container rm mysql
mysql
```

Lancez un conteneur mysql en lecture seule :

```
root@manager:~/docker-bench-security# docker container run -d --name mysql --read-only -v /var/lib/mysql -v /tmp
-v /var/run/mysqld -e MYSQL_ROOT_PASSWORD=password mysql
7788c67c3b692515f63f4659a8f40af397bfbde97485e2e40c500c16b158045b
root@manager:~/docker-bench-security# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
7788c67c3b69	mysql	"docker-entrypoint.s..."	5 seconds ago	Up 5 seconds	
3306/tcp	mysql				

Créez le fichier **write_a_file** dans le conteneur **mysql** :

```
root@manager:~/docker-bench-security# docker container exec mysql touch /write_a_file
touch: cannot touch '/write_a_file': Read-only file system
```



Important : Notez l'erreur **touch: cannot touch '/write_a_file': Read-only file system.**

Exécutez la commande **docker container diff** :

```
root@manager:~/docker-bench-security# docker container diff mysql
root@manager:~/docker-bench-security#
```



Important : Notez que la commande ne retourne aucune sortie. En effet le conteneur étant en lecture seule, aucun changement ne peut intervenir.

8.6 - [WARN] 5.14 - Ensure 'on-failure' container restart policy is set to '5'

Cet avertissement concerne la politique de re-démarrage du conteneur. La politique **on-failure[:max-retries]** implique que le conteneur est re-démarré en cas d'arrêt du à une erreur qui se manifeste en tant que code de retour autre que zéro. La valeur de **max-retries** est le nombre de fois que Docker va essayer de re-démarrer le conteneur. Cette politique peut être mise en place au démarrage du conteneur, par exemple :

```
# docker container run -d --name mysql --read-only --restart on-failure:5 -v /var/lib/mysql -v /tmp -v /var/run/mysqld -e MYSQL_ROOT_PASSWORD=password mysql
```

Pour plus d'informations, consultez cette [page](#).

8.7 - [WARN] 5.25 - Ensure the container is restricted from acquiring additional privileges

Pour compléter la configuration précédemment mise en place, il convient de lancer le conteneur en utilisant l'option **-security-opt** :

```
# docker container run -d --name mysql --read-only --restart on-failure:5 --security-opt="no-new-privileges:true" -v /var/lib/mysql -v /tmp -v /var/run/mysqld -e MYSQL_ROOT_PASSWORD=password mysql
```

Pour plus d'informations, consultez cette [page](#).

8.8 - [WARN] 5.26 - Ensure container health is checked at runtime

Voir l'avertissement 4.6.

8.9 - [WARN] 5.28 - Ensure PIDs cgroup limit is used

Sans l'utilisation de l'option **-pids-limit** un conteneur pourrait être victime d'une attaque de type **Fork Bomb**, un type spécifique de déni de service. Ce type d'attaque peut faire crasher l'hôte Docker et le seul remède est de re-démarrer l'hôte. Voici un exemple d'un Fork Bomb :

```
root@manager:~/docker-bench-security# docker run -u 1000 ubuntu bash -c ":( ) { : | : & }; ;; while [[ true ]]; do sleep 1; done"
```

L'hôte Docker **manager** crash. Après avoir re-démarrer la machine virtuelle, créez de nouveau le conteneur en utilisant l'option **-pids-limit** :

```
root@manager:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS
PORTS              NAMES
05b11e44e595        ubuntu             "bash -c ':( ) { : | ..." 6 minutes ago      Exited (255) 3 minutes ago
upbeat_turing
...
root@manager:~# docker rm 05
05
root@manager:~# docker run -u 1000 --pids-limit 100 ubuntu bash -c ":( ) { : | : & }; ;; while [[ true ]]; do sleep 1; done"
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
environment: fork: retry: Resource temporarily unavailable
^C
```

Pour plus d'informations, consultez cette [page](#).

Supprimez maintenant tous les conteneurs déjà créés :

```
root@manager:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS
PORTS              NAMES
095472e5096a        nginx              "nginx -g 'daemon of..." 13 minutes ago      Up 13 minutes
0.0.0.0:8081->80/tcp  affectionate_goldstine
5b31fe1e13bc        ubuntu             "bash -c ':( ) { : | ..." 28 minutes ago      Exited (254) 28 minutes ago
pensive_fermat
```

```

7788c67c3b69      mysql      "docker-entrypoint.s..." About an hour ago   Exited (255) 33 minutes ago
3306/tcp, 33060/tcp      mysql
root@manager:~# docker stop 095
095
root@manager:~# docker rm `docker ps -aq`
5b31fe1e13bc
7788c67c3b69
095472e5096a
root@manager:~# docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					

Re-créez le conteneur mysql en intégrant les points vus ci-dessus :

```

root@manager:~# cd docker-bench-security/
root@manager:~/docker-bench-security# docker container run -d --name mysql --read-only --restart on-failure:5 --
security-opt="no-new-privileges:true" --pids-limit 100 --memory="256m" --cpu-shares 512 -v /var/lib/mysql -v /tmp
-v /var/run/mysqld -e MYSQL_ROOT_PASSWORD=password mysql
df54974ebc11fe357f6e8e9b0f8499aee2658af435e32a45058a1e49fcd3dc24
root@manager:~/docker-bench-security# docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
NAMES					
df54974ebc11	mysql	"docker-entrypoint.s..."	5 seconds ago	Up 4 seconds	
3306/tcp, 33060/tcp	mysql				

Exécutez de nouveau le script **docker-bench-security.sh**, vous devez obtenir un résultat similaire à ceci en ce qui concerne la sécurité du Container Runtime :

```

root@manager:~/docker-bench-security# ./docker-bench-security.sh
...
[INFO] 5 - Container Runtime
[WARN] 5.1 - Ensure that, if applicable, an AppArmor Profile is enabled
[WARN]      * No AppArmorProfile Found: mysql
[PASS] 5.2 - Ensure that, if applicable, SELinux security options are set

```

```
[PASS] 5.3 - Ensure Linux Kernel Capabilities are restricted within containers
[PASS] 5.4 - Ensure that privileged containers are not used
[PASS] 5.5 - Ensure sensitive host system directories are not mounted on containers
[PASS] 5.6 - Ensure sshd is not run within containers
[PASS] 5.7 - Ensure privileged ports are not mapped within containers
[NOTE] 5.8 - Ensure that only needed ports are open on the container
[PASS] 5.9 - Ensure the host's network namespace is not shared
[PASS] 5.10 - Ensure that the memory usage for containers is limited
[PASS] 5.11 - Ensure CPU priority is set appropriately on the container
[PASS] 5.12 - Ensure that the container's root filesystem is mounted as read only
[PASS] 5.13 - Ensure that incoming container traffic is bound to a specific host interface
[PASS] 5.14 - Ensure that the 'on-failure' container restart policy is set to '5'
[PASS] 5.15 - Ensure the host's process namespace is not shared
[PASS] 5.16 - Ensure the host's IPC namespace is not shared
[PASS] 5.17 - Ensure that host devices are not directly exposed to containers
[INFO] 5.18 - Ensure that the default ulimit is overwritten at runtime if needed
[INFO]      * Container no default ulimit override: mysql
[PASS] 5.19 - Ensure mount propagation mode is not set to shared
[PASS] 5.20 - Ensure the host's UTS namespace is not shared
[PASS] 5.21 - Ensure the default seccomp profile is not Disabled
[NOTE] 5.22 - Ensure docker exec commands are not used with privileged option
[NOTE] 5.23 - Ensure that docker exec commands are not used with the user=root option
[PASS] 5.24 - Ensure that cgroup usage is confirmed
[PASS] 5.25 - Ensure that the container is restricted from acquiring additional privileges
[WARN] 5.26 - Ensure that container health is checked at runtime
[WARN]      * Health check not set: mysql
[INFO] 5.27 - Ensure that Docker commands always make use of the latest version of their image
[PASS] 5.28 - Ensure that the PIDs cgroup limit is used
[INFO] 5.29 - Ensure that Docker's default bridge 'docker0' is not used
[INFO]      * Container in docker0 network: mysql
[PASS] 5.30 - Ensure that the host's user namespaces are not shared
[PASS] 5.31 - Ensure that the Docker socket is not mounted inside any containers
...
```

LAB #9 - Sécurisation des Images avec Docker Content Trust

Docker Content Trust (DCT) a été introduit avec Docker Engine 1.8 et Docker CS Engine 1.9.0. DCT permet la vérification de l'authenticité, de l'intégrité et la date de publication d'une image Docker dans un registry. Par défaut, DCT est **désactivé**.

DCT est utilisé par le **Docker Hub Registry** mais peut aussi être mis en place dans des Registry privés, notamment grâce à la mise en place du **Docker Container Registry** qui est inclus avec **Docker Enterprise**.

DCT est basé sur l'utilisation de l'outil **Docker Notary** pour publier et gérer du contenu ainsi que **The Update Framework (TUF)**.

Pour plus d'information concernant DCT, consultez cette [page](#).

9.1 - DOCKER_CONTENT_TRUST

Pour utiliser **Docker Content Trust (DCT)**, il convient de vérifier que la valeur de la variable **DOCKER_CONTENT_TRUST** est **1** :

```
root@manager:~# echo $DOCKER_CONTENT_TRUST
1
```

Dans le cas contraire, il faut fixer la valeur de la variable à 1 :

```
root@manager:~# export DOCKER_CONTENT_TRUST=1
root@manager:~# echo $DOCKER_CONTENT_TRUST
1
```

9.2 - DCT et la commande docker pull

Afin d'utiliser un registry privé du Docker Hub, il est nécessaire de se connecter :

```
root@manager:~# docker login
```

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to <https://hub.docker.com> to create one.

Username: <votre_compte>

Password: <votre_mot_de_passe>

WARNING! Your password will be stored unencrypted in /root/.docker/config.json.

Configure a credential helper to remove this warning. See

<https://docs.docker.com/engine/reference/commandline/login/#credentials-store>

Login Succeeded

Pour constater l'impact de l'utilisation de DCT, il convient simplement de faire un **pull** d'une image non-signée :

```
root@manager:~# docker image pull i2tch/docker:unsigned
```

```
Error: remote trust data does not exist for docker.io/i2tch/docker: notary.docker.io does not have trust data for docker.io/i2tch/docker
```



Important : Notez l'erreur **Error: remote trust data does not exist for docker.io/i2tch/docker ...** En effet Docker Trust empêche l'utilisation des images non-signées.

Par contre, toutes les images de type **official** sont signées :

```
root@manager:~# docker image pull centos
```

```
Using default tag: latest
```

```
Pull (1 of 1): centos:latest@sha256:f94c1d992c193b3dc09e297ffd54d8a4f1dc946c37cbeceb26d35ce1647f88d9
```

```
sha256:f94c1d992c193b3dc09e297ffd54d8a4f1dc946c37cbeceb26d35ce1647f88d9: Pulling from library/centos
```

```
729ec3a6ada3: Pull complete
```

```
Digest: sha256:f94c1d992c193b3dc09e297ffd54d8a4f1dc946c37cbeceb26d35ce1647f88d9
```

```
Status: Downloaded newer image for centos@sha256:f94c1d992c193b3dc09e297ffd54d8a4f1dc946c37cbeceb26d35ce1647f88d9
```

```
Tagging centos@sha256:f94c1d992c193b3dc09e297ffd54d8a4f1dc946c37cbeceb26d35ce1647f88d9 as centos:latest
```

```
docker.io/library/centos:latest
```


Cette image est maintenant présente sur **manager.i2tch.loc** :

```
root@manager:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	775349758637	9 days ago	64.2MB
nginx	latest	540a289bab6c	2 weeks ago	126MB
alpine	latest	965ea09ff2eb	2 weeks ago	5.55MB
mysql	latest	c8ee894bd2bd	3 weeks ago	456MB
centos	latest	0f3e07c0138f	5 weeks ago	220MB

L'option disable-content-trust

Il est aussi possible d'activer ou de désactiver l'utilisation de DCT avec les options **-disable-content-trust=false/true** lors de l'utilisation des commandes **docker build**, **docker push** et **docker pull**, **docker create** et **docker run** :

```
root@manager:~# docker image pull --disable-content-trust=true i2tch/docker:unsigned
unsigned: Pulling from i2tch/docker
10d70a43a9f9: Pull complete
4f4fb700ef54: Pull complete
8951e3a91277: Pull complete
d1814ff35b8b: Pull complete
ff2a2bbf6141: Pull complete
b7205da5c3c9: Pull complete
458ea241cc75: Pull complete
74d1c0702786: Pull complete
c66f3692932d: Pull complete
9224bd1b9757: Pull complete
Digest: sha256:885fc831cb853700ded04029b4fa70ed502947042f6f154e432395cb35619d11
Status: Downloaded newer image for i2tch/docker:unsigned
docker.io/i2tch/docker:unsigned
```

```
root@manager:~# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

ubuntu	latest	775349758637	9 days ago	64.2MB
nginx	latest	540a289bab6c	2 weeks ago	126MB
alpine	latest	965ea09ff2eb	2 weeks ago	5.55MB
mysql	latest	c8ee894bd2bd	3 weeks ago	456MB
centos	latest	0f3e07c0138f	5 weeks ago	220MB
i2tch/docker	unsigned	9b915a241e29	3 years ago	212MB

```
root@manager:~# docker rmi i2tch/docker:unsigned
```

```
Untagged: i2tch/docker:unsigned
```

```
Untagged: i2tch/docker@sha256:885fc831cb853700ded04029b4fa70ed502947042f6f154e432395cb35619d11
```

```
Deleted: sha256:9b915a241e29dc2767980445e3109412b1905b6f1617aea7098e7ac1e5837ae2
```

```
Deleted: sha256:27eb08aec7b41dbfa2fd49bc2b3fad9b020fe40b0bc8289af7f53770f0843e7d
```

```
Deleted: sha256:7ad0aff4b88909fcff6372fdd26c24d688803b06845426b5a90bcd2f2cae93f4
```

```
Deleted: sha256:b93bcd594116ac8886f2daa0fc8d75a59da00161731dab24ababea853d031908
```

```
Deleted: sha256:54eda0a22e4b2a1b166cf996eb0651a4f53dec7e9dfad3549bbfe6078f2238a4
```

```
Deleted: sha256:36575f1e2764d54fdb92b5296cf4e993499836d6dd9a006f32e173865835070e
```

```
Deleted: sha256:27074774f844bdeba18e786585604c8b6352e925a7bd560deb66252bc8ccb861
```

```
Deleted: sha256:0da68695f8bc66fcea8f09004b5cb078861f5d99748f8b7ed035690e02c41477
```

```
Deleted: sha256:5dbda9873cdda8ff912b0ae5c34790ee06d7117fa27b193610fa2f7063bf55ff
```

```
Deleted: sha256:149690c37bdc8680ec66b0e2cc138f6d63caad74b091acf86a2a18111b90ea79
```

```
Deleted: sha256:2caf8a80130d6e9f4ed22e1ec1c3abd2c3f4330d2df9ec62f3b751300190b9e4
```

```
Deleted: sha256:1445a9131f2b28a12ff6396faebd6b4beb2cccd7af8eae28d5ff659d65de03ad
```

```
Deleted: sha256:4d9799a0754804f5cd623ab744757d16ec81862ee6e5d6986d9d1b0c5e5d5637
```

```
Deleted: sha256:dd833146402e8e6e67c48a6ae79a3c86101123e3d6ab1fc7999685eeea06ccba
```

```
Deleted: sha256:08d8e6ed6c3a5ac1bfee00f7b11f0a870d6bdc4af6d34169fa1e032c241a63a6
```

```
Deleted: sha256:0f3637356bb908638dda037c9c6aa4a2be8a19dbcf452a00cd733a8a456077ac
```

```
Deleted: sha256:aedb1b3b3b6e70ae4a342dfdcea874495b9d095ed6ba8eb4bc08f90ad9e83125
```

```
Deleted: sha256:05903cd969529ea56beec880bbeb7e90f1bdc281882f1cf3755760e41b181409
```

```
Deleted: sha256:d124781fc06a73b05a8644958397994bae668aba2f06f397fe1387c676b0d86f
```

9.3 - DCT et la commande docker push

Pour envoyer l'image dont l'IMAGE ID est **965ea09ff2eb** dans le registry privé, le tag de l'image doit être modifié :

```
root@manager:~# docker image tag alpine:latest <votre_compte>/docker:alpine
```

L'image dont l'IMAGE ID est **965ea09ff2eb** a maintenant deux tags **alpine:latest** et **<votre_compte>/docker:alpine** :

```
root@manager:~# docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	775349758637	9 days ago	64.2MB
nginx	latest	540a289bab6c	2 weeks ago	126MB
<votre_compte>/docker	alpine	965ea09ff2eb	2 weeks ago	5.55MB
alpine	latest	965ea09ff2eb	2 weeks ago	5.55MB
mysql	latest	c8ee894bd2bd	3 weeks ago	456MB
centos	latest	0f3e07c0138f	5 weeks ago	220MB

Lors du push vers le registry privé, il faut créer des passphrases pour **deux** clefs :

- la **root** key aussi connue sous le nom **offline** key (ID 192fc7e), qui est uniquement demandée la **première** fois après la mise en place de DCT lors de la création d'un **repository**,
- la **repository** key aussi connue sous le nom **tagging** key (ID 168c754), utilisée pour signer l'image en y apposant un **tag**. La signature est spécifique au **repository**.

```
root@manager:~# docker push <votre_compte>/docker:alpine
The push refers to repository [docker.io/<votre_compte>/docker]
77cae8ab23bf: Mounted from library/alpine
alpine: digest: sha256:e4355b66995c96b4b468159fc5c7e3540fcef961189ca13fee877798649f531a size: 528
Signing and pushing trust metadata
You are about to create a new root signing key passphrase. This passphrase
will be used to protect the most sensitive key in your signing system. Please
choose a long, complex passphrase and be careful to keep the password and the
key file itself secure and backed up. It is highly recommended that you use a
password manager to generate the passphrase and keep it safe. There will be no
way to recover this key. You can find the key in your config directory.
Enter passphrase for new root key with ID 192fc7e: fenestros
Repeat passphrase for new root key with ID 192fc7e: fenestros
Enter passphrase for new repository key with ID 168c754: fenestros
```

```
Repeat passphrase for new repository key with ID 168c754: fenestros
Finished initializing "docker.io/<votre_compte>/docker"
Successfully signed docker.io/<votre_compte>/docker:alpine
```

Les clefs sont stockées dans le répertoire **~/.docker/trust/private** :

```
root@manager:~# ls -l ~/.docker/trust
total 8
drwx----- 2 root root 4096 nov. 10 14:49 private
drwx----- 3 root root 4096 nov. 8 13:48 tuf

root@manager:~# ls -l ~/.docker/trust/private
total 8
-rw----- 1 root root 447 nov. 10 14:49 168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac.key
-rw----- 1 root root 416 nov. 10 14:49 192fc7ed9543ad4bceec58886ab1d605b7433c35f7462d7343d0780d8fddf1db.key
root@manager:~# cat ~/.docker/trust/private/168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
gun: docker.io/i2tch/docker
role: targets

MIHuMEkGCSqGSIB3DQEFDTA8MBsGCSqGSIB3DQEFDDA0BAhm7HwR0y8FFAICCAAw
HQYJYIZIAWUDBAEqBBC729tU73wKHFQSBmZ1EVZaBIGmGiFSs4lM5tElSGukl1B
HrELT9aFooFgW7oSXNLM8aFfF/vJ+BSjsgfqWLdvuH+DUXXdUidxcoGMEWnVZNIC
3m40g3MywHilW4rUcjoHVTUXABGXUQ3f7h+nI15CXcZ11qRLyWbf2uywE9yYH90
M7GLUcE+pTENJKfZAhRGBEL+LgXNfGI1aAVqaEbBDcDnKKf4Uj1Xu4oLJ7je8+nT
dg==
-----END ENCRYPTED PRIVATE KEY-----

root@manager:~# cat ~/.docker/trust/private/192fc7ed9543ad4bceec58886ab1d605b7433c35f7462d7343d0780d8fddf1db.key
-----BEGIN ENCRYPTED PRIVATE KEY-----
role: root

MIHuMEkGCSqGSIB3DQEFDTA8MBsGCSqGSIB3DQEFDDA0BAiAtCzEar3AhgICCAAw
HQYJYIZIAWUDBAEqBBA07hHWVoq0o6xcETQQDXRdBIGgPUoLzTz07Ajx8K3D8+Vv
```

```
2NUiflMYhH/0I9PL6iA2JJcmD0l+8UelJy+vHRCu7UAIyWXYIHFN5Aab40mk9/Pg
V2BwSlXp7t1Cnqp/ah7g0T40+0nT64JkTS+l3cS0CaCf2E4l6nY8g4cl40hZIFJz
KRE08uEq3v7HcSBBqFm0+TU+92d7hVuDApaj0lZYP+3f7H6AjU0qu6hUoK8Ck/Y
Ig==
-----END ENCRYPTED PRIVATE KEY-----
```

9.4 - DCT et la commande docker build

L'exemple suivant démontre un Dockerfile qui référence une image parente non signée :

```
root@manager:~# mkdir nottrusted
root@manager:~# cd nottrusted/
root@manager:~/nottrusted# vi Dockerfile
root@manager:~/nottrusted# cat Dockerfile
FROM docker/trusttest:latest
RUN echo
```

Lors du build de l'image **<votre_compte>/docker:nottrusted** qui utilise ce Dockerfile, une erreur est retournée car sa création n'est pas conforme à l'utilisation de DCT :

```
root@manager:~/nottrusted# docker build -t <votre_compte>/docker:nottrusted .
Sending build context to Docker daemon

error during connect: Post
http://%2Fvar%2Frun%2Fdocker.sock/v1.40/build?buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquot
a=0&cpusetcpus=&cpusetmems=&pushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=default&rm=1&shmsize=0&t=i2tch%2Fdocker%3Anottrusted&target=&ulimits=null&version=1: Error: remote trust data does not
exist for docker.io/docker/trusttest: notary.docker.io does not have trust data for docker.io/docker/trusttest
```

L'utilisation de l'option **-disable-content-trust** permet la construction de l'image **<votre_compte>/docker:nottrusted** :

```
root@manager:~/nottrusted# docker build --disable-content-trust -t <votre_compte>/docker:nottrusted .
```

```
Sending build context to Docker daemon 2.048kB
Step 1/2 : FROM docker/trusttest:latest
latest: Pulling from docker/trusttest
Image docker.io/docker/trusttest:latest uses outdated schema1 manifest format. Please upgrade to a schema2 image
for better future compatibility. More information at https://docs.docker.com/registry/spec/deprecated-schema-v1/
aac0c133338d: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:50c0cdd0577cc7ab7c78e73a0a89650b222f6ce2b87d10130ecff055981b702f
Status: Downloaded newer image for docker/trusttest:latest
---> cc7629d1331a
Step 2/2 : RUN echo
---> Running in 694e79d3cd88

Removing intermediate container 694e79d3cd88
---> 686e85ee76b8
Successfully built 686e85ee76b8
Successfully tagged <votre_compte>/docker:nottrusted
```

Lors du push de l'image **<votre_compte>/docker:nottrusted** vers le repository distant, celle-ci est **signée** :

```
root@manager:~/nottrusted# docker push <votre_compte>/docker:nottrusted
The push refers to repository [docker.io/<votre_compte>/docker]
5f70bf18a086: Layer already exists
c22f7bc058a9: Mounted from docker/trusttest
nottrusted: digest: sha256:1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc size: 734
Signing and pushing trust metadata
Enter passphrase for repository key with ID 168c754: fenestros
Successfully signed docker.io/<votre_compte>/docker:nottrusted
```



Important : Notez l'utilisation de la même root key que lors du push de l'image **<votre_compte>/docker:alpine** car il s'agit du même repository.

Créer un deuxième Repository

Par contre en modifiant le tag de l'image **<votre_compte>/docker:nottrusted** à **<votre_compte>/otherimage:latest**, un autre repository sera créé lors du push de l'image renommée :

```
root@manager:~/nottrusted# docker tag <votre_compte>/docker:nottrusted <votre_compte>/otherimage:latest
root@manager:~/nottrusted# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
<votre_compte>/docker	nottrusted	686e85ee76b8	9 minutes ago	5.03MB
<votre_compte>/otherimage	latest	686e85ee76b8	9 minutes ago	5.03MB
ubuntu	latest	775349758637	9 days ago	64.2MB
nginx	latest	540a289bab6c	2 weeks ago	126MB
<votre_compte>/docker	alpine	965ea09ff2eb	2 weeks ago	5.55MB
alpine	latest	965ea09ff2eb	2 weeks ago	5.55MB
mysql	latest	c8ee894bd2bd	3 weeks ago	456MB
centos	latest	0f3e07c0138f	5 weeks ago	220MB
docker/trusttest	latest	cc7629d1331a	4 years ago	5.03MB

```
root@manager:~/nottrusted# docker push docker.io/<votre_compte>/otherimage:latest
The push refers to repository [docker.io/<votre_compte>/otherimage]
5f70bf18a086: Mounted from <votre_compte>/docker
c22f7bc058a9: Mounted from <votre_compte>/docker
latest: digest: sha256:1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc size: 734
Signing and pushing trust metadata
Enter passphrase for root key with ID 192fc7e: fenestros
Enter passphrase for new repository key with ID 7b13d02: fenestros
Repeat passphrase for new repository key with ID 7b13d02: fenestros
Finished initializing "docker.io/<votre_compte>/otherimage"
Successfully signed docker.io/<votre_compte>/otherimage:latest
```



Important : Notez la création d'une deuxième repository key (ID 7b13d02 au lieu de ID 168c754) lors du push de l'image **<votre_compte>/otherimage:latest** car il s'agit d'un



autre repository.

La présence de cette deuxième repository key (**7b13d02d74264624fb201e7ae13ae694286b9f761aa86adddefd0408c7234a58.key**) peut être constatée dans le répertoire **~/.docker/trust/private** :

```
root@manager:~/nottrusted# ls -l ~/.docker/trust/private
total 12
-rw----- 1 root root 447 nov. 10 14:49 168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac.key
-rw----- 1 root root 416 nov. 10 14:49 192fc7ed9543ad4bceec58886ab1d605b7433c35f7462d7343d0780d8fddf1db.key
-rw----- 1 root root 451 nov. 10 17:37 7b13d02d74264624fb201e7ae13ae694286b9f761aa86adddefd0408c7234a58.key
```

En inspectant les clefs des images créées, l'utilisation des différentes clefs est démontrées très clairement :

```
root@manager:~/nottrusted# docker trust inspect <votre_compte>/docker:alpine
[
  {
    "Name": "<votre_compte>/docker:alpine",
    "SignedTags": [
      {
        "SignedTag": "alpine",
        "Digest": "e4355b66995c96b4b468159fc5c7e3540fcef961189ca13fee877798649f531a",
        "Signers": [
          "Repo Admin"
        ]
      }
    ],
    "Signers": [],
    "AdministrativeKeys": [
      {
        "Name": "Root",
        "Keys": [
          {
            "ID": "d4074334a4ff5a9a43ebd1320ad77c2df88c990ec812f90eb045c603c01ab698"
```



```
    }
  ],
},
{
  "Name": "Repository",
  "Keys": [
    {
      "ID": "168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac"
    }
  ]
}
]
}
]
}
]
root@manager:~/nottrusted# docker trust inspect <votre_compte>/docker:nottrusted
[
  {
    "Name": "<votre_compte>/docker:nottrusted",
    "SignedTags": [
      {
        "SignedTag": "nottrusted",
        "Digest": "1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc",
        "Signers": [
          "Repo Admin"
        ]
      }
    ],
    "Signers": [],
    "AdministrativeKeys": [
      {
        "Name": "Root",
        "Keys": [
          {
            "ID": "d4074334a4ff5a9a43ebd1320ad77c2df88c990ec812f90eb045c603c01ab698"
          }
        ]
      }
    ]
  }
]
```

```
    }
  ],
  {
    "Name": "Repository",
    "Keys": [
      {
        "ID": "168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac"
      }
    ]
  }
]
```



Important : Notez que les clefs utilisées sont les mêmes pour les deux images.

```
root@manager:~/nottrusted# docker trust inspect <votre_compte>/otherimage:latest
[
  {
    "Name": "<votre_compte>/otherimage:latest",
    "SignedTags": [
      {
        "SignedTag": "latest",
        "Digest": "1183c62a5d31e202b5f5f528e9e7cdc36140aa3212c938e1d471c6b3b59f01bc",
        "Signers": [
          "Repo Admin"
        ]
      }
    ],
    "Signers": [],
  }
]
```

```
    "AdministrativeKeys": [  
      {  
        "Name": "Root",  
        "Keys": [  
          {  
            "ID": "26f00698f51be2824c6fe85a14722c279bbd487125fe8fa18c0fc8f76dd6280d"  
          }  
        ]  
      },  
      {  
        "Name": "Repository",  
        "Keys": [  
          {  
            "ID": "7b13d02d74264624fb201e7ae13ae694286b9f761aa86adddefd0408c7234a58"  
          }  
        ]  
      }  
    ]  
  }  
]
```



Important : Notez que les clefs utilisées sont différentes.

Supprimer une Signature

Dernièrement il est possible de supprimer la signature d'une image avec la commande **docker trust revoke** :

```
root@manager:~# docker trust revoke <votre_compte>/docker:alpine  
Enter passphrase for repository key with ID 168c754:  
Successfully deleted signature for <votre_compte>/docker:alpine
```

```
root@manager:~# docker trust inspect <votre_compte>/docker:alpine
[
  {
    "Name": "<votre_compte>/docker:alpine",
    "SignedTags": [],
    "Signers": [],
    "AdministrativeKeys": [
      {
        "Name": "Root",
        "Keys": [
          {
            "ID": "d4074334a4ff5a9a43ebd1320ad77c2df88c990ec812f90eb045c603c01ab698"
          }
        ]
      },
      {
        "Name": "Repository",
        "Keys": [
          {
            "ID": "168c754ea8f36ce7fbcbe2299b6d91fc0f4d594c9ed9b86916687b618d8438ac"
          }
        ]
      }
    ]
  }
]
```



Important : Il existe un autre mécanisme de signatures cryptographiques qui permet de certifier le contenu des images mises à disposition sur une Registry. Appelé **Notary**, ce système a été développé par la communauté Docker et intègre une partie de la spécification de **The Update Framework** (TUF). Pour plus d'informations, consultez cet [article](#).

LAB #10 - Sécurisation du Socket du Daemon Docker

Par défaut le daemon Docker peut être contacté en utilisant un socket Unix local ce qui implique qu'il faut une connexion SSH vers l'hôte Docker. Docker peut cependant utiliser un socket http.

Pour pouvoir contacter de daemon Docker via le réseau d'une manière sécurisée il faut installer, configurer et activer le support TLS grâce aux options **tlsverify** et **tlscacert**.

La configuration implique que :

- pour le daemon Docker, seules les connections en provenance de clients authentifiés par un certificat signé par l'**autorité de certification** (CA) du serveur seront acceptées,
- pour le client, il ne peut que connecter aux serveurs ayant un certificat signé par le CA du serveur.

La mise en place nécessite **openssl** :

```
root@manager:~# which openssl
/usr/bin/openssl
```

10.1 - Création du Certificat de l'Autorité de Certification

Commencez par créer une clef privée **ca-key.pem** pour le CA :

```
root@manager:~# openssl genrsa -aes256 -out ca-key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for ca-key.pem:fenestros
Verifying - Enter pass phrase for ca-key.pem:fenestros
```

Ensuite, créez le certificat **ca.pem** du CA :

```
root@manager:~# openssl req -new -x509 -days 365 -key ca-key.pem -sha256 -out ca.pem
Enter pass phrase for ca-key.pem:fenestros
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:GB
State or Province Name (full name) [Some-State]:SURREY
Locality Name (eg, city) []:ADDLESTONE
Organization Name (eg, company) [Internet Widgits Pty Ltd]:I2TCH LIMITED
Organizational Unit Name (eg, section) []:TRAINING
Common Name (e.g. server FQDN or YOUR name) []:manager.i2tch.loc
Email Address []:infos@i2tch.co.uk
```

10.2 - Création du Certificat du Serveur Hôte du Daemon Docker

Les clefs du CA ayant été créées, créez une clef **server-key.pem** pour le serveur hôte du daemon Docker :

```
root@manager:~# openssl genrsa -out server-key.pem 4096
Generating RSA private key, 4096 bit long modulus
.....
.....+++++
.....+++++
e is 65537 (0x010001)
```

Créez ensuite un **Certificate Signing Request** (CSR) **server.csr** :

```
root@manager:~# echo $HOSTNAME
manager.i2tch.loc
root@manager:~# openssl req -subj "/CN=`echo $HOSTNAME`" -sha256 -new -key server-key.pem -out server.csr
```

Une connexion TLS peut être effectuée en utilisant un FQDN ou une adresse IP. Pour cette raison, créez le fichier **extfile.cnf** :

```
root@manager:~# echo subjectAltName = DNS:`echo $HOSTNAME`,IP:10.0.2.62,IP:127.0.0.1 >> extfile.cnf
```

Fixez l'attribut étendu de l'utilisation de la clef du daemon à **serverAuth** :

```
root@manager:~# echo extendedKeyUsage = serverAuth >> extfile.cnf
```

Vérifiez que votre fichier a été correctement créé :

```
root@manager:~# cat extfile.cnf
subjectAltName = DNS:manager.i2tch.loc,IP:10.0.2.62,IP:127.0.0.1
extendedKeyUsage = serverAuth
```

Signez maintenant le CSR du serveur **server.csr** avec la clef privée du CA **ca-key.pem** afin de produire le certificat du serveur **server-cert.pem** :

```
root@manager:~# openssl x509 -req -days 365 -sha256 -in server.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -
out server-cert.pem -extfile extfile.cnf
Signature ok
subject=CN = manager.i2tch.loc
Getting CA Private Key
Enter pass phrase for ca-key.pem:fenestros
```

10.3 - Création du Certificat du Client

Créez ensuite la clef privée **key.pem** du client qui se connectera au daemon à partir du réseau :

```
root@manager:~# openssl genrsa -out key.pem 4096
```

```
Generating RSA private key, 4096 bit long modulus
```

```
.....  
.....++++  
.....++++  
e is 65537 (0x010001)
```

Modifiez l'entrée pour l'adresse 10.0.2.61 dans le fichier **/etc/hosts** du manager :

```
root@manager:~# vi /etc/hosts  
root@manager:~# cat /etc/hosts  
127.0.0.1    localhost  
10.0.2.60    debian9.i2tch.loc    debian9  
10.0.2.61    myregistry.i2tch.loc    myregistry  
10.0.2.62    manager.i2tch.loc    manager  
10.0.2.63    worker1.i2tch.loc    worker1  
10.0.2.64    worker2.i2tch.loc    worker2  
  
# The following lines are desirable for IPv6 capable hosts  
::1        localhost ip6-localhost ip6-loopback  
ff02::1    ip6-allnodes  
ff02::2    ip6-allrouters
```

Créez ensuite le CSR du client **client.csr** :

```
root@manager:~# openssl req -subj '/CN=myregistry.i2tch.loc' -new -key key.pem -out client.csr
```

Fixez l'attribut étendu de l'utilisation de la clef du client à **clientAuth** :

```
root@manager:~# echo extendedKeyUsage = clientAuth > extfile-client.cnf
```

Signez le CSR du client **client.csr** avec la clef privée du CA **ca-key.pem** afin de créer le certificat du client **cert.pem** :

```
root@manager:~# openssl x509 -req -days 365 -sha256 -in client.csr -CA ca.pem -CAkey ca-key.pem -CAcreateserial -  
out cert.pem -extfile extfile-client.cnf
```



```
Signature ok
subject=CN = myregistry.i2tch.loc
Getting CA Private Key
Enter pass phrase for ca-key.pem:fenestros
```

Vérifiez la présence des fichiers générés :

```
root@manager:~# ls -l
total 60
-rw----- 1 root root 3326 nov. 11 10:53 ca-key.pem
-rw-r--r-- 1 root root 2163 nov. 11 10:57 ca.pem
-rw-r--r-- 1 root root 17 nov. 11 11:15 ca.srl
-rw-r--r-- 1 root root 1907 nov. 11 11:15 cert.pem
-rw-r--r-- 1 root root 1594 nov. 11 11:12 client.csr
drwxr-xr-x 5 root root 4096 nov. 8 12:58 docker-bench-security
-rw-r--r-- 1 root root 1707 nov. 8 12:35 docker-stack.yml
-rw-r--r-- 1 root root 30 nov. 11 11:13 extfile-client.cnf
-rw-r--r-- 1 root root 95 nov. 11 11:06 extfile.cnf
-rw----- 1 root root 3243 nov. 11 11:10 key.pem
drwxr-xr-x 2 root root 4096 nov. 10 17:21 nottrusted
-rw-r--r-- 1 root root 1964 nov. 11 11:08 server-cert.pem
-rw-r--r-- 1 root root 1594 nov. 11 11:01 server.csr
-rw----- 1 root root 3243 nov. 11 10:59 server-key.pem
-rw-r--r-- 1 root root 882 oct. 27 15:46 stats
```

Supprimez les fichiers ayant déjà été utilisés, à savoir les deux CSR et les deux fichiers des extensions :

```
root@manager:~# rm -v client.csr server.csr extfile.cnf extfile-client.cnf
'client.csr' supprimé
'server.csr' supprimé
'extfile.cnf' supprimé
'extfile-client.cnf' supprimé
```

Modifiez les permissions des clefs privées :

```
root@manager:~# chmod -v 0400 ca-key.pem key.pem server-key.pem
le mode de 'ca-key.pem' a été modifié de 0600 (rw-----) en 0400 (r-----)
le mode de 'key.pem' a été modifié de 0600 (rw-----) en 0400 (r-----)
le mode de 'server-key.pem' a été modifié de 0600 (rw-----) en 0400 (r-----)
```

Ainsi que les permissions des certificats :

```
root@manager:~# chmod -v 0444 ca.pem server-cert.pem cert.pem
le mode de 'ca.pem' a été modifié de 0644 (rw-r--r--) en 0444 (r--r--r--)
le mode de 'server-cert.pem' a été modifié de 0644 (rw-r--r--) en 0444 (r--r--r--)
le mode de 'cert.pem' a été modifié de 0644 (rw-r--r--) en 0444 (r--r--r--)
```

Arrêtez et supprimez le conteneur **mysql** :

```
root@manager:~# docker stop mysql
mysql
root@manager:~# docker rm mysql
mysql
```

10.4 - Démarrage du Daemon Docker avec une Invocation Directe

Arrêtez et désactivez le service Docker :

```
root@manager:~# systemctl stop docker
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
root@manager:~# systemctl disable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable docker
```

Lancez un invocation directe de **dockerd** afin que le daemon n'acceptent que des connexions de clients fournissant un certificat signé par le CA :

```
root@manager:~# dockerd --tlsverify --tlscacert=ca.pem --tlscert=server-cert.pem --tlskey=server-key.pem -
H=0.0.0.0:2376 &
[1] 1868
root@manager:~# INFO[2019-11-11T12:01:43.638540628+01:00] Starting up
INFO[2019-11-11T12:01:43.639187816+01:00] User namespaces: ID ranges will be mapped to subuid/subgid ranges of:
dockremap:dockremap
INFO[2019-11-11T12:01:43.641133642+01:00] User namespaces: ID ranges will be mapped to subuid/subgid ranges of:
dockremap:dockremap
INFO[2019-11-11T12:01:43.646949782+01:00] parsed scheme: "unix" module=grpc
INFO[2019-11-11T12:01:43.648299396+01:00] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2019-11-11T12:01:43.648633123+01:00] ccResolverWrapper: sending update to cc:
[{unix:///run/containerd/containerd.sock 0 <nil>}] <nil> module=grpc
INFO[2019-11-11T12:01:43.650756512+01:00] ClientConn switching balancer to "pick_first" module=grpc
INFO[2019-11-11T12:01:43.656738368+01:00] parsed scheme: "unix" module=grpc
INFO[2019-11-11T12:01:43.657165991+01:00] scheme "unix" not registered, fallback to default scheme module=grpc
INFO[2019-11-11T12:01:43.660938142+01:00] ccResolverWrapper: sending update to cc:
[{unix:///run/containerd/containerd.sock 0 <nil>}] <nil> module=grpc
INFO[2019-11-11T12:01:43.661309281+01:00] ClientConn switching balancer to "pick_first" module=grpc
INFO[2019-11-11T12:01:43.663242717+01:00] [graphdriver] using prior storage driver: overlay2
WARN[2019-11-11T12:01:43.681131788+01:00] Your kernel does not support cgroup rt period
WARN[2019-11-11T12:01:43.681397622+01:00] Your kernel does not support cgroup rt runtime
INFO[2019-11-11T12:01:43.681845166+01:00] Loading containers: start.
INFO[2019-11-11T12:01:43.824330430+01:00] Default bridge (docker0) is assigned with an IP address 172.17.0.0/16.
Daemon option --bip can be used to set a preferred IP address
INFO[2019-11-11T12:01:43.887849374+01:00] Loading containers: done.
INFO[2019-11-11T12:01:43.908567890+01:00] Docker daemon commit=9013bf583a
graphdriver(s)=overlay2 version=19.03.4
INFO[2019-11-11T12:01:43.908851991+01:00] Daemon has completed initialization
INFO[2019-11-11T12:01:43.969272646+01:00] API listen on [::]:2376
[Entrée]
root@manager:~#
```

Vérifiez que le processus tourne :

```
root@manager:~# ps aux | grep docker
root      1868  0.2  4.0 421876 82236 pts/0    Sl   12:01   0:00 dockerd --tlsverify --tlscacert=ca.pem --
tlscert=server-cert.pem --tlskey=server-key.pem -H=0.0.0.0:2376
root      1995  0.0  0.0  12780   964 pts/0    S+   12:02   0:00 grep docker
```

Installez le paquet **net-tools** qui contient le binaire **netstat** :

```
root@manager:~# apt install -y net-tools
```

Vérifiez que le port **2376** est à l'écoute :

```
root@manager:~# netstat -anp | grep 2376
tcp6      0      0 :::2376                 :::*                    LISTEN     1868/dockerd
```

10.5 - Configuration du Client

Transférez ensuite le certificat du CA ainsi que le certificat et la clef privée du client vers la VM **debian91** :

```
root@manager:~# scp ca.pem key.pem cert.pem trainee@10.0.2.61:/home/trainee/
The authenticity of host '10.0.2.61 (10.0.2.61)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjgF/aJgUc9jg56slNaZQdAUcvB0vE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.61' (ECDSA) to the list of known hosts.
trainee@10.0.2.61's password:
ca.pem
100% 2163   917.8KB/s   00:00
key.pem
100% 3243    3.0MB/s   00:00
cert.pem
100% 1907   921.7KB/s   00:00
```

Lancez la commande **docker version** sur la VM **debian91** :

```
trainee@myregistry:~$ docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem --tlskey=key.pem -
H=manager.i2tch.loc:2376 version
Client: Docker Engine - Community
 Version:           19.03.4
 API version:       1.40
 Go version:        go1.12.10
 Git commit:        9013bf583a
 Built:             Fri Oct 18 15:52:34 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:           19.03.4
  API version:       1.40 (minimum version 1.12)
  Go version:        go1.12.10
  Git commit:        9013bf583a
  Built:             Fri Oct 18 15:51:05 2019
  OS/Arch:           linux/amd64
  Experimental:      false
 containerd:
  Version:           1.2.10
  GitCommit:         b34a5c8af56e510852c35414db4c1f4fa6172339
 runc:
  Version:           1.0.0-rc8+dev
  GitCommit:         3e425f80a8c931f88e6d94a8c831b9d5aa481657
 docker-init:
  Version:           0.18.0
  GitCommit:         fec3683
```

Afin de faciliter l'utilisation des commandes sur le serveur à partir de myregistry, créez le répertoire **~/docker** :

```
trainee@myregistry:~$ mkdir -pv ~/.docker
```

```
mkdir: création du répertoire '/home/trainee/.docker'
```

Copiez ensuite les fichiers *.pem dans le répertoire ~/.**docker** :

```
trainee@myregistry:~$ cp -v {ca,cert,key}.pem ~/.docker
'ca.pem' -> '/home/trainee/.docker/ca.pem'
'cert.pem' -> '/home/trainee/.docker/cert.pem'
'key.pem' -> '/home/trainee/.docker/key.pem'
```

Créez les deux variables **DOCKER_HOST** et **DOCKER_TLS_VERIFY** :

```
trainee@myregistry:~$ export DOCKER_HOST=tcp://manager.i2tch.loc:2376 DOCKER_TLS_VERIFY=1
```

Maintenant la connexion est sécurisée par défaut :

```
trainee@myregistry:~$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
i2tch/docker	nottrusted	686e85ee76b8	19 hours ago	5.03MB
i2tch/otherimage	latest	686e85ee76b8	19 hours ago	5.03MB
ubuntu	latest	775349758637	10 days ago	64.2MB
nginx	latest	540a289bab6c	2 weeks ago	126MB
alpine	latest	965ea09ff2eb	2 weeks ago	5.55MB
i2tch/docker	alpine	965ea09ff2eb	2 weeks ago	5.55MB
mysql	latest	c8ee894bd2bd	3 weeks ago	456MB
centos	latest	0f3e07c0138f	5 weeks ago	220MB
docker/trusttest	latest	cc7629d1331a	4 years ago	5.03MB