CloudApps

MASTER

3.2 – Contenedores y Orquestadores

Tema 3 – Seguridad en Docker

Universidad Rey Juan Carlos

Micael Gallego
micael.gallego@urjc.es
@micael_gallego

code URJC

Índice

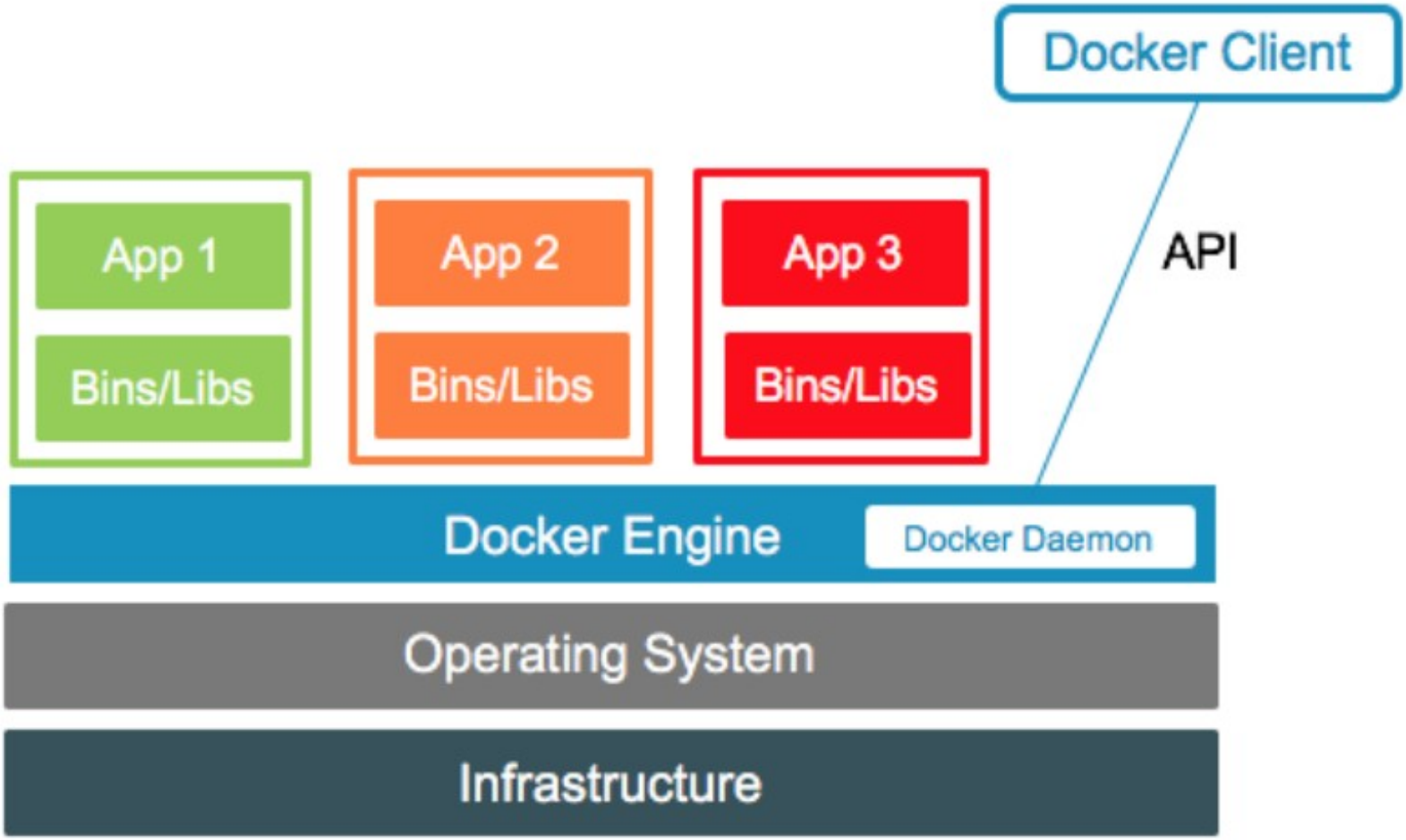
Seguridad en Docker

- Docker permite la **ejecución segura de aplicaciones**
  - Aislamiento entre varias aplicaciones
  - Aislamiento entre cada aplicación y el host
  - Se restringen sus capacidades (recursos, sistema de ficheros, red…)
  - Se recomienda el principio de menor privilegio (*principle of least privilege*)

Seguridad en Docker

- Docker puede ejecutarse sobre una **máquina virtual** para ofrecer un **grado mayor de aislamiento** y protección
- Es el enfoque usado habitualmente en las plataformas ***cloud computing* públicas o privadas**
- Recientemente han aparecido **tecnologías de aislamiento** de contenedores basadas en VMs ligeras o kernel en espacio de usario

Arquitectura Docker



Arquitectura Docker

- El cliente docker se puede conectar al **Docker engine localmente** (unix pipe) si están en la misma máquina
- También puede controlarse de forma **remota vía REST API** (se puede cifrar por **TLS**)
- Ideal para **desarrollo** en sistemas operativos Windows o Mac, pero **no recomendable en producción** por exponer el host

Aislamiento de aplicaciones

code URJC

- Docker facilita la recomendación de **enjaular** (*chroot*) aplicaciones para acceder únicamente a los recursos que necesitan (**principio de menor privilegio**)
- Para ello se usan los **namespaces** del kernel de linux, que permiten mostrar recursos a un proceso que no son visibles a otros procesos

http://man7.org/linux/man-pages/man7/namespaces.7.html

Aislamiento de aplicaciones

code URJC

- **Namespaces** usados por Docker
  - **PID:** Espacio de PIDs propio
  - **MNT:** Puntos de montaje propios
  - **NET:** Stack de red propio
  - **UTS:** Aislamiento entre identificadores del sistema como hostname y NIS domain name
  - **IPC:** Inter-process communication (IPC) para los procesos del mismo namespace

http://man7.org/linux/man-pages/man7/namespaces.7.html

Aislamiento de aplicaciones

code URJC

- **Cgroups (Control Groups)**
  - Funcionalidad del kernel de linux para **limitar los recursos** usados por un proceso (memoria, CPU)
  - Por **defecto no se limitan** los recursos usados por los contenedores
  - Se **recomienda limitar los recursos** para que un contenedor atacado no pueda provocar un ataque de denegación de servicio en el host por usar todos los recursos disponibles

https://en.wikipedia.org/wiki/Cgroups

https://www.kernel.org/doc/Documentation/cgroup-v2.txt

https://docs.docker.com/engine/reference/run/#runtime-constraints-on-resources

Permisos de las aplicaciones

- **Seccomp (secure computing mode)**
  - Se puede definir las llamadas al sistema permitidas por cada contenedor
  - Por defecto se bloquean unas 44 llamadas al sistema
    - Cambio de hora del host
    - Cambio de privilegios de E/S del Kernel
    - Montaje de volúmenes
    - Reinicio del host
    - …

https://en.wikipedia.org/wiki/Seccomp
https://docs.docker.com/engine/security/seccomp/
https://github.com/moby/moby/blob/master/profiles/seccomp/default.json

Permisos de las aplicaciones

- **Linux Capabilities**
  - Se puede definir las capacidades de un contenedor docker
  - En modo privilegiado (--priviledged) con acceso total al host
  - Control de grano fino
    - Se pueden eliminar algunas que se permiten por defecto (Envío de señales, cambio del GID y UID, …)
    - Se pueden añadir algunas que no se permiten (Modificar los límites de los recursos, administrar la red…)

http://man7.org/linux/man-pages/man7/capabilities.7.html

https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities

Permisos de las aplicaciones

- **Dispositivos (Devices)**
  - El acceso a los dispositivos físicos del host está restringido para contenedores
  - Se les puede dar acceso a dispositivos específicos en diferentes modos (read, write, mknod)

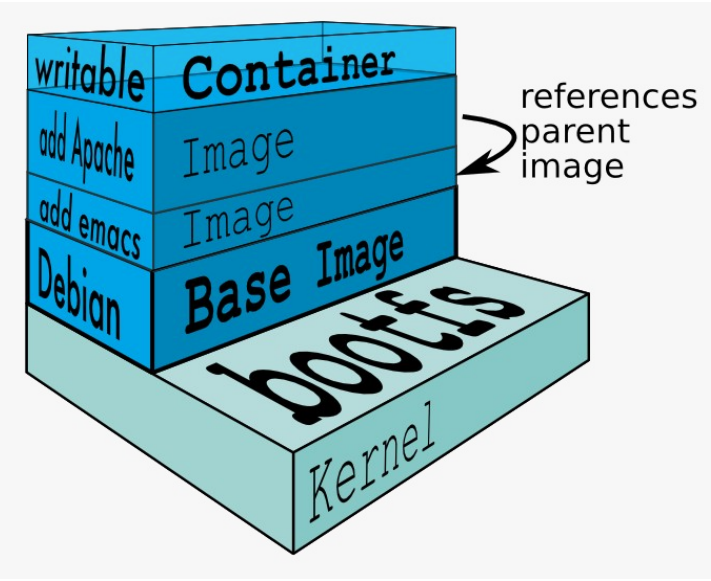
```
$ docker run --device=/dev/sda:/dev/xvdc:w --rm -it ubuntu:22.04 fdisk /dev/xvdc
```

```
$ docker run -it --privileged alpine:3.17 /bin/sh
```

https://docs.docker.com/engine/reference/commandline/run/#add-host-device-to-container---device

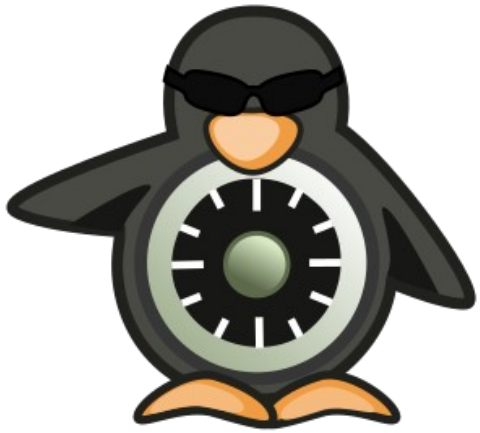
Permisos de las aplicaciones

- **Ficheros**
  - Sistema de ficheros Copy-on-write
  - Permite ejecutar varios contenedores desde la misma imagen
  - Los contenedores están aislados y la creación de un fichero en uno no es visible desde otro
  - Los ficheros del kernel del host que tienen que ser accesibles, se montan de sólo lectura



Seguridad linux

- Aunque Docker proporciona mecanismos de seguridad para los contenedores, se recomienda usar otras herramientas linux
- AppArmor, SELinux, TOMOYO, GRSec…

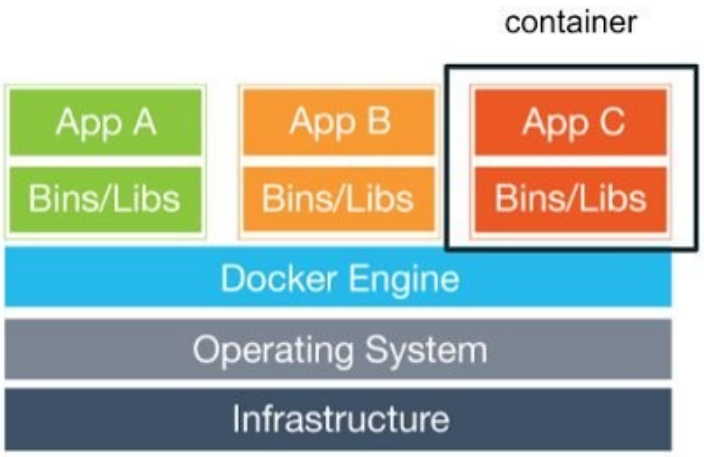


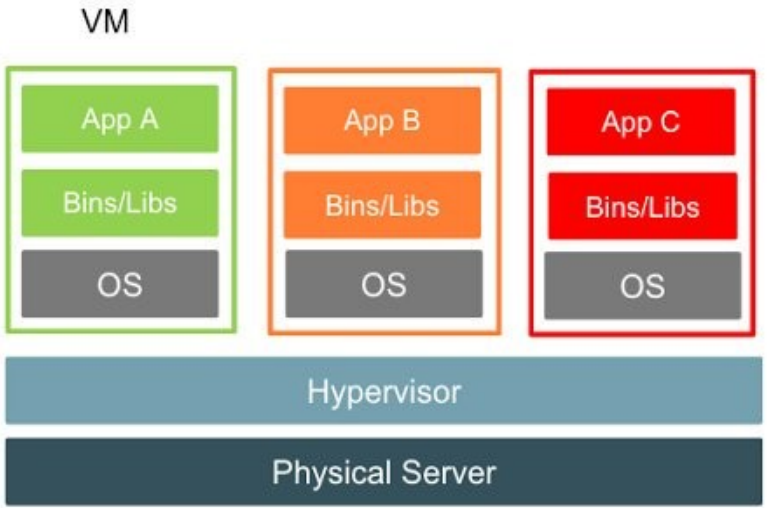
https://en.wikipedia.org/wiki/AppArmor

https://en.wikipedia.org/wiki/Security-Enhanced_Linux

VMs vs Contenedores

- Entornos para la **ejecución aislada de aplicaciones** en un host compartido
  - **Contenedores** docker comparten el mismo kernel (son más ligeros)
  - **VMs** tienen un SO completo sobre un hypervisor (son más pesados)





VMs vs Contenedores

- Las **VMs están más aisladas** del host que los contenedores, pero con mayor uso de recursos
- ¿Podemos **fiarnos** del aislamiento de los contenedores?
- ¿Es **seguro** ejecutar varios contenedores "potencialmente maliciosos" en el mismo host?
- ¿Qué política sigue la **industria**?

VMs vs Contenedores

Vulnerabilidad en Docker

runc - Malicious container escape

CVE-2019-5736

Si el contenedor se ejecuta como root (lo más habitual) puede ganar privilegios de root en el host

https://www.openwall.com/lists/oss-security/2019/02/11/2
https://access.redhat.com/security/vulnerabilities/runcescape

VMs vs Contenedores

- Los proveedores cloud principales **no ejecutan contenedores de diferentes usuarios en el mismo host**
- Cada usuario gestiona **una o varias VMs** (instancias) en las que se **ejecutan contenedores** (ECS, EKS…)
- Se han creado **VMs ligeras** para ejecutar un único contenedor en ellas

https://katacontainers.io/

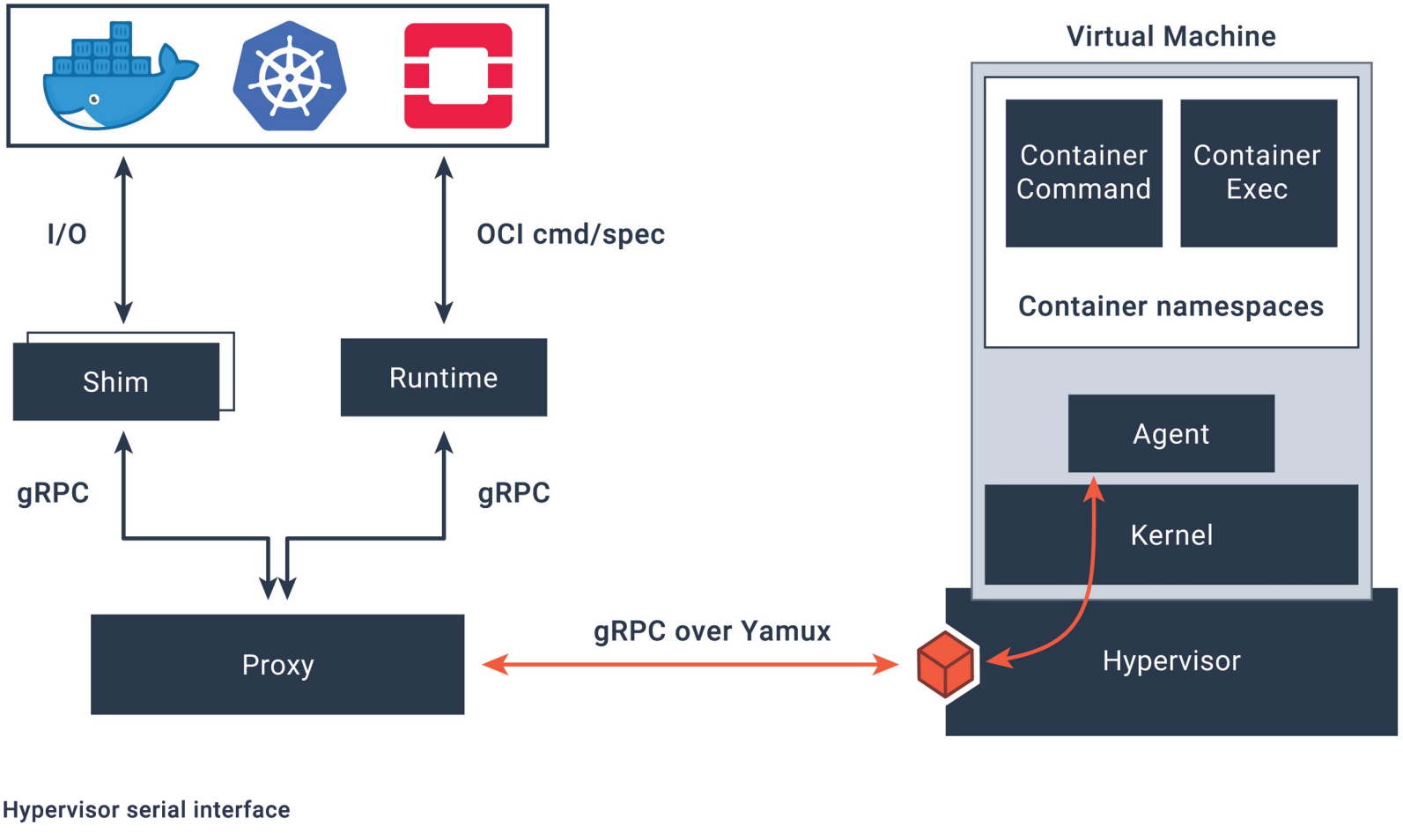
https://firecracker-microvm.github.io/

Katacontainers

- La **velocidad** de los contenedores, la **seguridad** de las máquinas virtuales
- Se soportan múltiples hypervisors incluyendo **QEMU**, **NEMU** and **Firecracker**

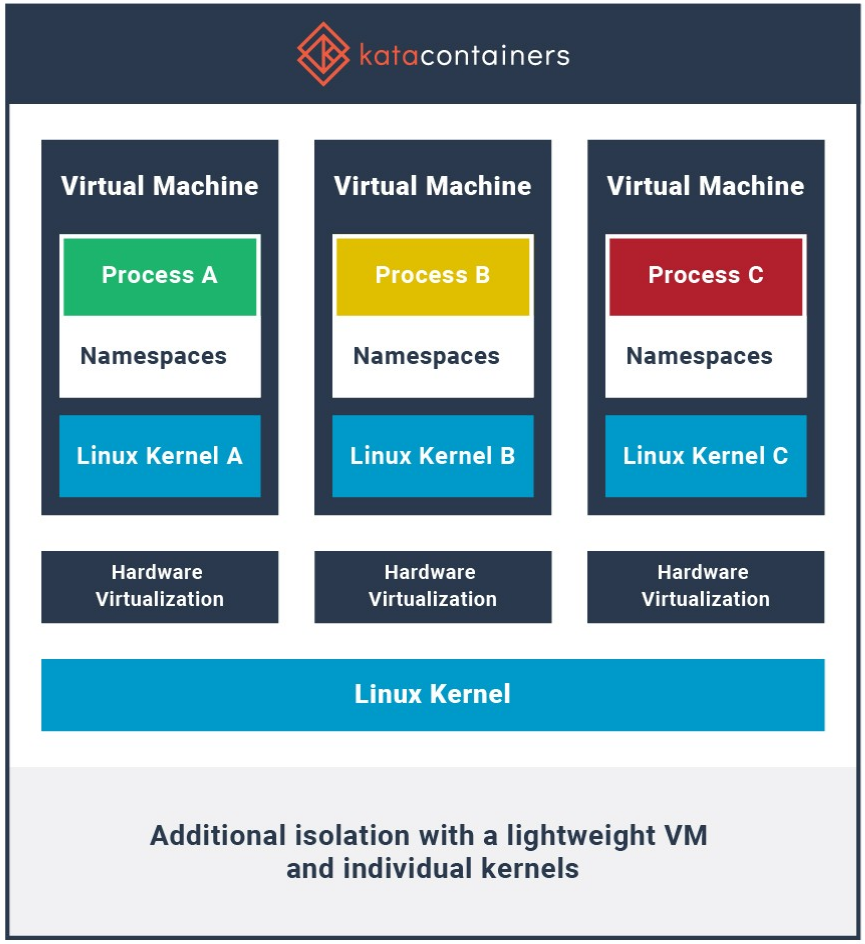
https://katacontainers.io/

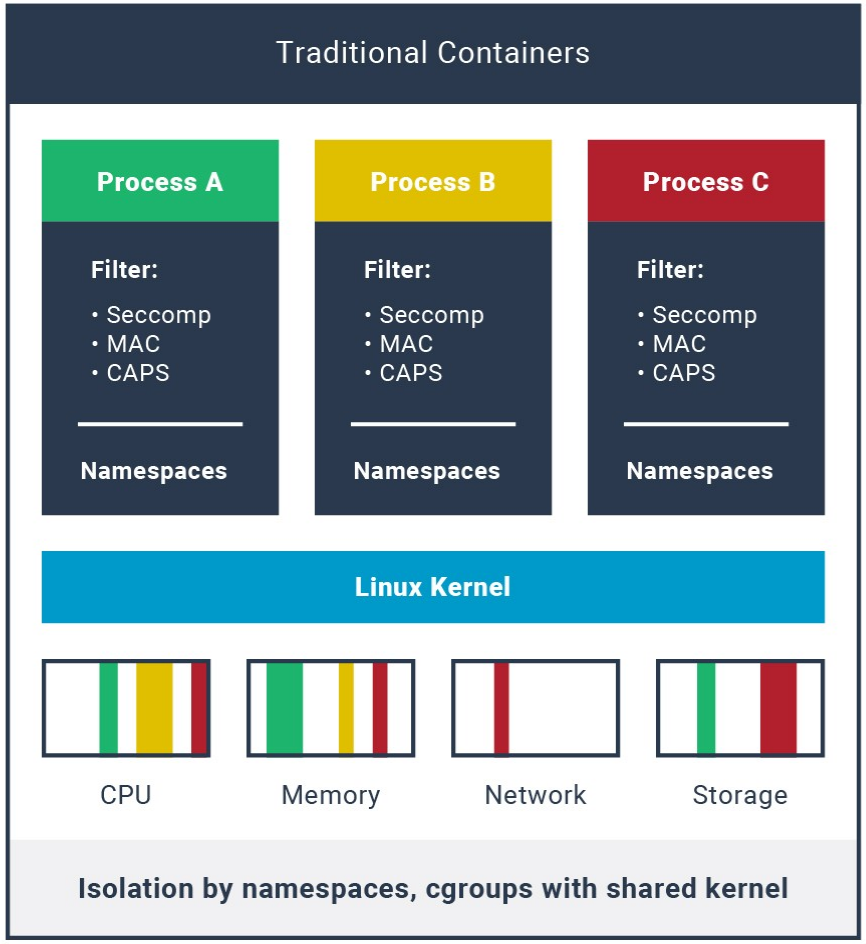
Katacontainers



https://katacontainers.io/

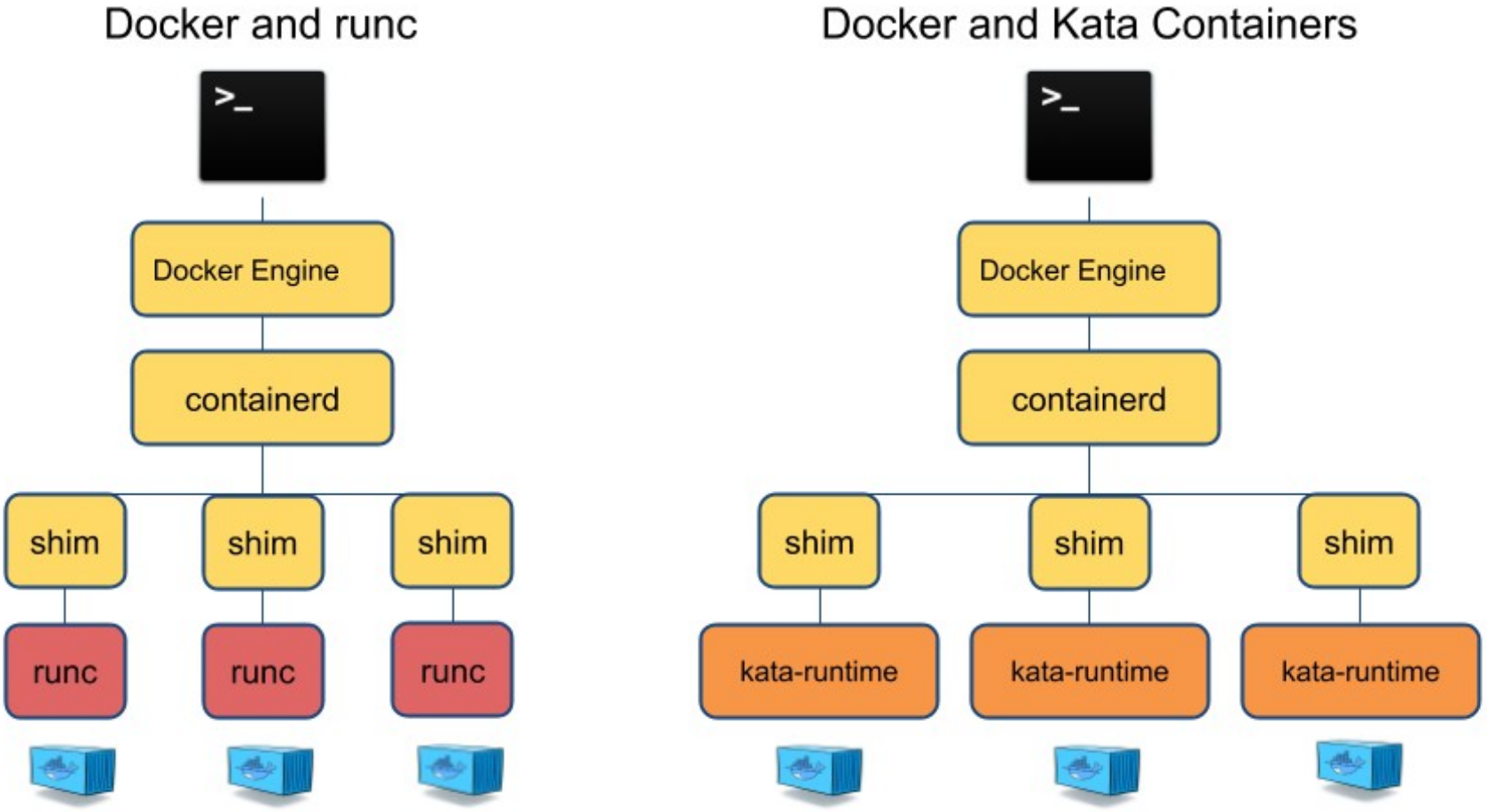
Katacontainers





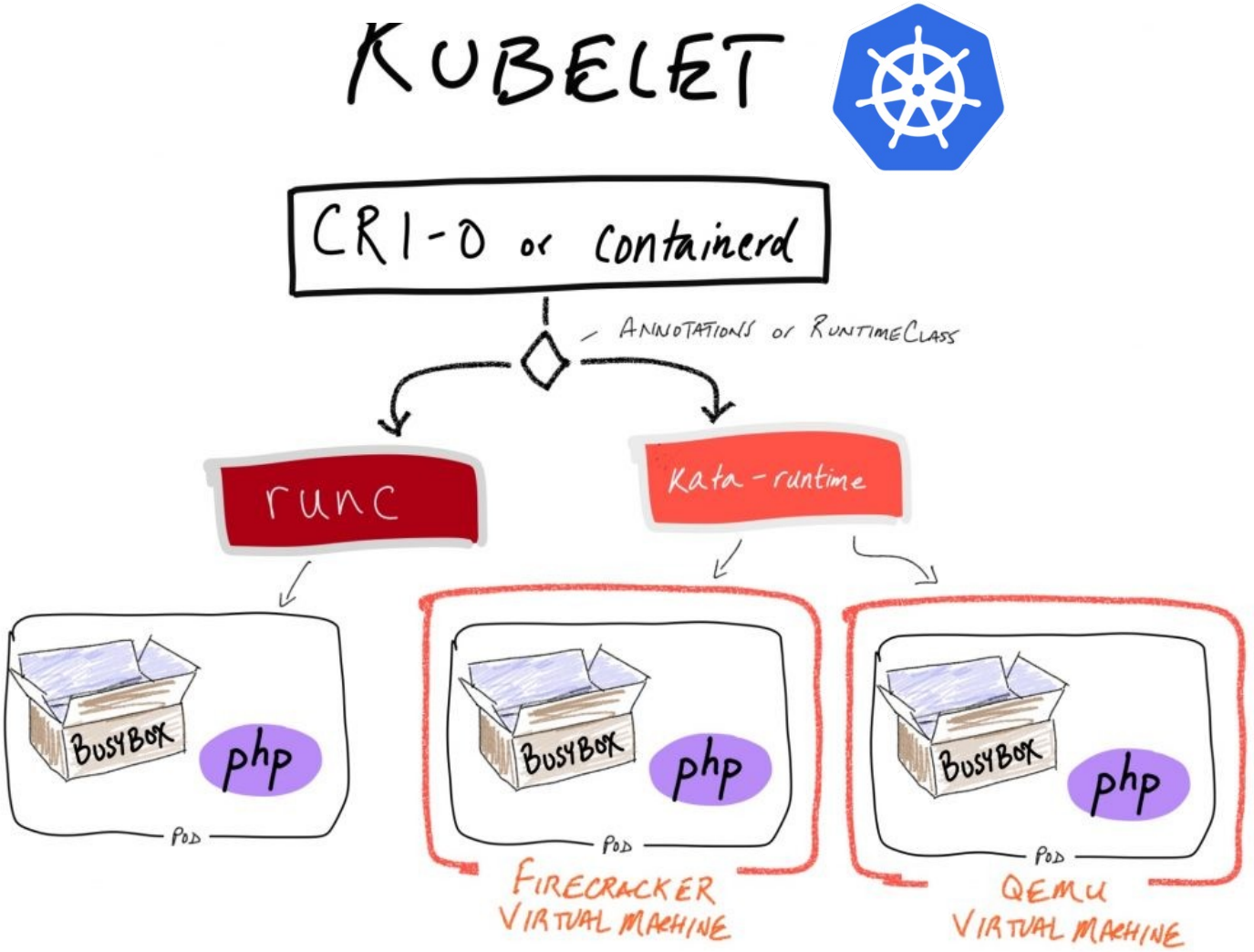
https://katacontainers.io/

Katacontainers



https://katacontainers.io/

Katacontainers



https://aws.amazon.com/es/blogs/opensource/kata-containers-1-5-firecracker-support/

Firecracker

Firecracker

aws

- MicroVMs **seguras** y **rápidas** para computación *serverless*
- Máquinas virtuales ligeras con diseño minimalista
- **Velocidad** (125ms) y **consumo reducido** (5Mb RAM)
- Usado por AWS en Lambda y Fargate

https://firecracker-microvm.github.io/

Firecracker

Firecracker

aws

- Integrado en

Weave Firekube

Firekube is an open source Kubernetes distribution that enables secure clouds anywhere.

https://www.weave.works/oss/firekube/

firecracker-microvm / **firecracker-containerd**

https://github.com/firecracker-microvm/firecracker-containerd

https://firecracker-microvm.github.io/

gVisor

- Google **no usa VMs** para para aumentar el **aislamiento** de los contenedores
- Implementa un **kernel linux en espacio de usuario** que aisla el contenedor del kernel del host

https://gvisor.dev/

gVisor

[…running untrusted or potentially malicious code without additional isolation is not a good idea. The efficiency and performance gains from using a single, shared kernel also mean that container escape is possible with a single vulnerability.]

https://gvisor.dev/

gVisor

- Un ***container sandbox runtime*** focalizado en seguridad, eficiencia y facilidad de uso
- Kernel en ***user-space*** para contenedores
- Limita el **acceso al kernel del host** al ejecutar las funcionalidades en espacio de usuario
- Soporta **diferentes mecanismos de interceptar llamadas** al kernel para máxima compatibilidad

```
$ docker run --runtime=runsc --rm hello-world
```

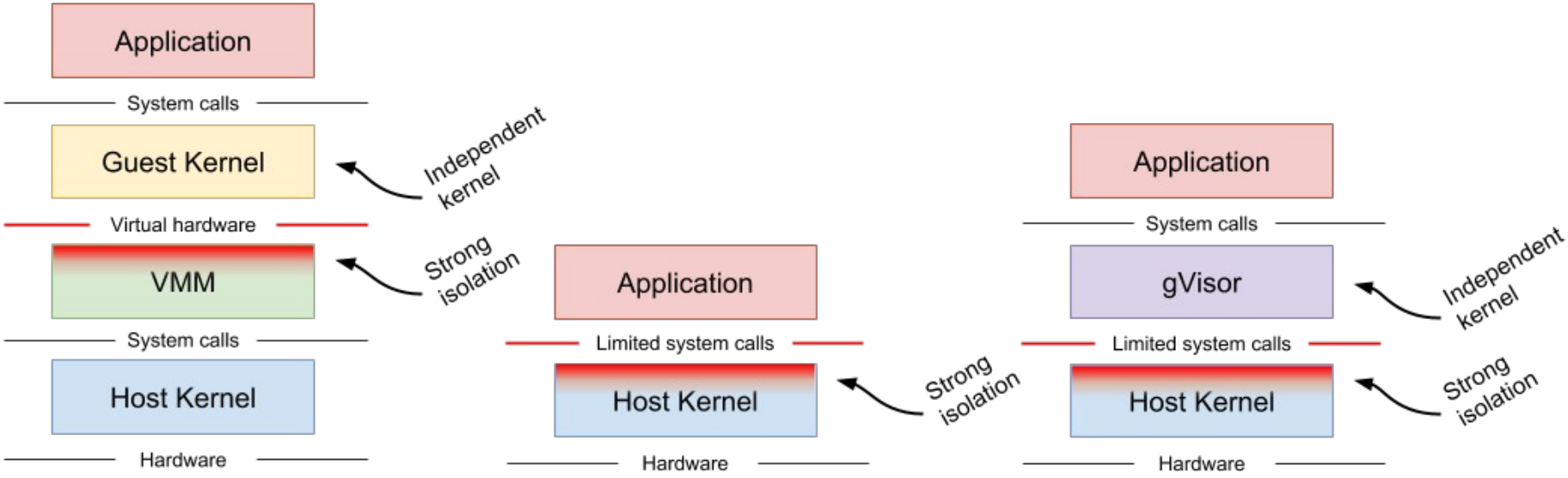
https://gvisor.dev/

gVisor

Virtual Machine

Seccomp, SELinux, AppArmor

gVisor



https://gvisor.dev/

Índice

1. Seguridad en Docker
2. **CIS Docker Benchmark**
3. Dockerfile linters

CIS Docker Benchmark

Center for Internet Security

- Entidad sin ánimo de lucro que aprovecha el poder de una **comunidad global de TI** para proteger a las organizaciones públicas y privadas contra las amenazas cibernéticas
- Elabora ***benchmarks*** **con buenas prácticas de seguridad** sobre tecnologías, herramientas y sistemas IT

https://www.cisecurity.org/

CIS Docker Benchmark

- Guía prescriptiva para una configuración segura de **Docker versión 1.5.0**
- Ha sido elaborada con **Docker Engine 20.10.20** sobre **RHEL 7** y **Ubuntu 20.04**

https://www.cisecurity.org/benchmark/docker

CIS Docker Benchmark

- **Recomendaciones en varios niveles**
  - **Nivel 1**
    - Práctico y prudente
    - Beneficio en seguridad
    - No limitar la utilidad de la tecnología
    - Aplicado a Docker y Linux Host
  - **Nivel 2**
    - Casos en los que la seguiridad es muy importante
    - Mecanismos de defensa en profundidad
    - Puede afectar a la funcionalidad y al rendimiento
    - Aplicado a Docker y Linux Host

CIS Docker Benchmark

- Se denomina ***benchmark*** porque la mayoría de las **recomendaciones** tienen una **puntuación** (***score***)
- El score global se calcula sumando el número de las recomendaciones que se cumplen

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Host Configuration**		
1.1	**Linux Hosts Specific Configuration**		
1.1.1	Ensure a separate partition for containers has been created (Automated)	☐	☐
1.1.2	Ensure only trusted users are allowed to control Docker daemon (Automated)	☐	☐
1.1.3	Ensure auditing is configured for the Docker daemon (Automated)	☐	☐
1.1.4	Ensure auditing is configured for Docker files and directories - /run/containerd (Automated)	☐	☐
1.1.5	Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)	☐	☐

CIS Docker Benchmark

- **1 Host Configuration**
  - Recomendaciones de seguridad para preparar los hosts **Linux** en el que se ejecutará Docker
- **2 Docker daemon configuration**
  - Recomendaciones que hacen más seguro el Docker daemon. Estas recomendaciones afectan a todos los contenedores
- **3 Docker daemon configuration files**
  - Permisos y propiedad de los ficheros y directorios relacionados con Docker.

CIS Docker Benchmark

- **4 Container Images and Build File**
  - Las imágenes base y los ficheros de construcción (Dockerfile) gobiernan los fundamentos sobre cómo se ejecutará un contenedor basado en una imagen determinada.
  - Asegurar que estás usando la imagen base correcta y los ficheros de construcción apropiados es muy importante para la seguridad
- **5 Container Runtime**
  - La forma en que se inicia un contenedor regula muchas de las implicaciones de seguridad.
  - Ciertos parámetros en tiempo de ejecución podrían ser peligrosos y comprometer al host y a otros contenedores.

CIS Docker Benchmark

- **6 Docker Security Operations**
  - Aspectos de seguridad relacionados con las operaciones
  - Recordatorios a las organizaciones que deberían extender sus buenas prácticas para incluir a los contenedores
- **7 Docker Swarm Configuration**
  - Aspectos de seguridad relacionados con Docker Swarm

1 Host Configuration

- 1.1.1 Ensure a separate partition for containers has been created
- 1.1.2 Ensure only trusted users are allowed to control Docker daemon
- 1.1.3 Ensure auditing is configured for the Docker daemon
- 1.1.4 Ensure auditing is configured for Docker files and directories - /run/containerd
- 1.1.5 Ensure auditing is configured for Docker files and directories - /var/lib/docker
- 1.1.6 Ensure auditing is configured for Docker files and directories - /etc/docker
- 1.1.7 Ensure auditing is configured for Docker files and directories – docker.service
- 1.1.8 Ensure auditing is configured for Docker files and directories - containerd.sock
- …

1 Host Configuration

- ...
- 1.1.9 Ensure auditing is configured for Docker files and directories – docker.socket
- 1.1.10 Ensure auditing is configured for Docker files and directories - /etc/default/docker
- 1.1.11 Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
- 1.1.12 Ensure auditing is configured for Docker files and directories - /etc/containerd/config.toml
- 1.1.13 Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker
- 1.1.14 Ensure auditing is configured for Docker files and directories - /usr/bin/containerd
- ...

1 Host Configuration

- ...
- 1.1.15 Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim
- 1.1.16 Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v1
- 1.1.17 Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v2
- 1.1.18 Ensure auditing is configured for Docker files and directories - /usr/bin/runc

1 Host Configuration

- **1.1.1 Ensure a separate partition for containers has been created**
  - **Profile Applicability:** Level 1 - Linux Host OS
  - **Description:** All Docker containers and their data and metadata is stored under /var/lib/docker directory. By default, /var/lib/docker should be mounted under either the / or /var partitions dependent on how the Linux operating system in use is configured.
  - **Rationale:** Docker depends on /var/lib/docker as the default directory where all Docker related files, including the images, are stored. This directory could fill up quickly causing both Docker and the host to become unusable. For this reason, you should create a separate partition (logical volume) for storing Docker files.

1 Host Configuration

- **1.1.1 Ensure a separate partition for containers has been created**
  - **Audit:** At the Docker host execute one of the below commands:

```
$ grep '/var/lib/docker\s' /proc/mounts
```

This should return the partition details for the /var/lib/docker mountpoint.

```
$ mountpoint -- "$(docker info -f '{{ .DockerRootDir }}')"
```

https://www.projectatomic.io/docs/docker-storage-recommendation/

https://docs.docker.com/storage/

1 Host Configuration

- **1.1.1 Ensure a separate partition for containers has been created**
  - **Remediation:** For new installations, you should create a separate partition for the /var/lib/docker mount point. For systems which have already been installed, you should use the Logical Volume Manager (LVM) within Linux to create a new partition.
  - **Default Value:** By default, /var/lib/docker is mounted under the / or /var partitions dependent on how the OS is configured.

https://www.projectatomic.io/docs/docker-storage-recommendation/

https://docs.docker.com/storage/

1 Host Configuration

- 1.2.1 Ensure the container host has been Hardened
- 1.2.2 Ensure that the version of Docker is up to date

1 Host Configuration

- **1.2.1 Ensure the container host has been Hardened**
  - **Profile Applicability:** Level 1 - Linux Host OS
  - **Description:** A container host is able to run one or more containers. It is of utmost importance to harden the host to mitigate host security misconfiguration.
  - **Rationale:** You should follow infrastructure security best practices and harden your host OS. Keeping the host system hardened will ensure that host vulnerabilities are mitigated. Not hardening the host system could lead to security exposures and breaches.

1 Host Configuration

- **1.2.1 Ensure the container host has been Hardened**
  - **Audit:** Ensure that the host specific security guidelines are followed. Ask the system administrators which security benchmark the current host system should currently be compliant with and check that security standards associated with this standard are currently in place.
  - **Remediation:** You may consider various CIS Security Benchmarks for your container host. If you have other security guidelines or regulatory requirements to adhere to, please follow them as suitable in your environment.

1 Host Configuration

- **1.2.1 Ensure the container host has been Hardened**
  - **Impact:** None.
  - **Default Value:** By default, the host has factory setting and is not hardened.
  - **References:**

https://docs.docker.com/engine/security/

https://www.cisecurity.org/cis-benchmarks/

2 Docker daemon configuration

- 2.1 Run the Docker daemon as a non-root user, if possible
- 2.2 Ensure network traffic is restricted between containers on the default bridge
- 2.3 Ensure the logging level is set to 'info'
- 2.4 Ensure Docker is allowed to make changes to iptables
- 2.5 Ensure insecure registries are not used
- 2.6 Ensure aufs storage driver is not used
- 2.7 Ensure TLS authentication for Docker daemon is configured
- 2.8 Ensure the default ulimit is configured appropriately
- 2.9 Enable user namespace support
- 2.10 Ensure the default cgroup usage has been confirmed
- 2.11 Ensure base device size is not changed until needed
- 2.12 Ensure that authorization for Docker client commands is enabled
- …

2 Docker daemon configuration

- ...
- 2.13 Ensure centralized and remote logging is configured
- 2.14 Ensure containers are restricted from acquiring new privileges
- 2.15 Ensure live restore is enabled
- 2.16 Ensure Userland Proxy is Disabled
- 2.17 Ensure that a daemon-wide custom seccomp profile is applied if appropriate
- 2.18 Ensure that experimental features are not implemented in production

2 Docker daemon configuration

- **2.1 Run the Docker daemon as a non-root user, if possible**
  - Por defecto el demonio de Docker arranca con permisos de root
  - Utilizar el modo Rootless ejecutará el demonio de Docker y los contenedores en modo non-root, mitigando potenciales vulnerabilidades en el demonio de Docker y en la ejecución de contenedores
  - Existen limitaciones conocidas a nivel de red y recursos al utilizar este modo

https://docs.docker.com/engine/security/rootless/

2 Docker daemon configuration

- **2.2 Ensure network traffic is restricted between containers on the default bridge**
- Por defecto todos los contenedores pueden comunicarse por red entre sí y eso podría aumentar el impacto (*blast radius*) de un ataque
- **2.3 Ensure the logging level is set to 'info'**
  - Un nivel de log INFO es recomendable para auditar el servicio
- **2.4 Ensure Docker is allowed to make changes to iptables**
  - Permite a Docker mantener una correcta gestión de IPTables

2 Docker daemon configuration

- **2.5 Ensure insecure registries are not used**
  - Deberías usar repositorios cifrados por TLS (considerados seguros)
  - El Cerficado CA debería estar configurador en el servicio Docker
  - Docker por defecto permite el uso de registros no seguros. La comunicación con ellos podría sufrir ataques.

https://docs.docker.com/registry/insecure/

2 Docker daemon configuration

- **2.7 Ensure TLS authentication for Docker daemon is configured**
  - Si expones el servicio Docker por red, debería tener autenticación TLS
  - Si no es necesario, no expongas el servicio por red
  - Deja su valor por defecto a socket UNIX local

https://docs.docker.com/engine/security/protect-access/

2 Docker daemon configuration

- **2.9 Enable user namespace support**
  - Si un contenedor se ejecuta con usuario root, el user namespace cambia ese usuario en el host
  - Es una funcionalidad experimental que puede tener impacto en rendimiento y funcionalidad
  - No es necesaria si el contenedor se ejecuta con un usuario no root
  - OpenShift no permite la ejecución de contenedores root (*Support Arbitrary User IDs*)

https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-user-namespace-options

https://docs.openshift.com/container-platform/4.12/openshift_images/create-images.html

2 Docker daemon configuration

- **2.12 Ensure that authorization for Docker client commands is enabled**
  - Si varios usuarios van a necesitar acceso al servicio Docker, se puede configurar un authorization plugin para controlar sus permisos individualmente

    https://docs.docker.com/engine/extend/plugins_authorization/

- **2.13 Ensure centralized and remote logging is configured**
  - Es recomendable gestionar los logs de los contenedores para auditar su comportamiento

    https://docs.docker.com/config/containers/logging/configure/

2 Docker daemon configuration

- **2.16 Ensure Userland Proxy is Disabled**
  - La funcionalidad preferida basada en IPTables funciona en la mayoría de los sistemas

https://docs.docker.com/config/containers/container-networking/

- **2.17 Ensure that a daemon-wide custom seccomp profile is applied if appropriate**
  - Puedes limitar aún más el perfil seccomp del engine Docker

https://docs.docker.com/engine/security/seccomp/

3 Docker daemon config files

- 3.1 Ensure that the docker.service file ownership is set to root:root
- 3.2 Ensure that docker.service file permissions are appropriately set
- 3.3 Ensure that docker.socket file ownership is set to root:root
- 3.4 Ensure that docker.socket file permissions are set to 644 or more restrictive
- 3.5 Ensure that the /etc/docker directory ownership is set to root:root
- 3.6 Ensure that /etc/docker directory permissions are set to 755 or more restrictively
- 3.7 Ensure that registry certificate file ownership is set to root:root
- 3.8 Ensure that registry certificate file permissions are set to 444 or more restrictively
- 3.9 Ensure that TLS CA certificate file ownership is set to root:root
- 3.10 Ensure that TLS CA certificate file permissions are set to 444 or more restrictively
- 3.11 Ensure that Docker server certificate file ownership is set to root:root
- 3.12 Ensure that the Docker server certificate file permissions are set to 444 or more restrictively
- ...

3 Docker daemon config files

- ...
- 3.13 Ensure that the Docker server certificate key file ownership is set to root:root
- 3.14 Ensure that the Docker server certificate key file permissions are set to 400
- 3.15 Ensure that the Docker socket file ownership is set to root:docker
- 3.16 Ensure that the Docker socket file permissions are set to 660 or more restrictively
- 3.17 Ensure that the daemon.json file ownership is set to root:root
- 3.18 Ensure that daemon.json file permissions are set to 644 or more restrictive
- 3.19 Ensure that the /etc/default/docker file ownership is set to root:rootç
- 3.20 Ensure that the /etc/sysconfig/docker file permissions are set to 644 or more restrictively
- 3.21 Ensure that the /etc/sysconfig/docker file ownership is set to root:root
- ...

3 Docker daemon config files

- ...
- 3.22 Ensure that the /etc/default/docker file permissions are set to 644 or more restrictively
- 3.23 Ensure that the Containerd socket file ownership is set to root:root
- 3.24 Ensure that the Containerd socket file permissions are set to 660 or more restrictively

4 Container Images and build file

- 4.1 Ensure that a user for the container has been created
- 4.2 Ensure that containers use only trusted base images
- 4.3 Ensure that unnecessary packages are not installed in the container
- 4.4 Ensure images are scanned and rebuilt to include security patches
- 4.5 Ensure Content trust for Docker is Enabled
- 4.6 Ensure that HEALTHCHECK instructions have been added to container images
- 4.7 Ensure update instructions are not used alone in Dockerfiles
- 4.8 Ensure setuid and setgid permissions are removed
- 4.9 Ensure that COPY is used instead of ADD in Dockerfiles
- 4.10 Ensure secrets are not stored in Dockerfiles
- 4.11 Ensure only verified packages are installed
- 4.12 Ensure all signed artifacts are validated

4 Container Images and build file

- **4.1 Ensure that a user for the container has been created**
  - Crea un usuaro no root para ejecutar el contenedor
  - Dockerfile

```
RUN useradd -d /home/username -m -s /bin/bash username
USER username
```

https://docs.docker.com/engine/reference/builder/#user

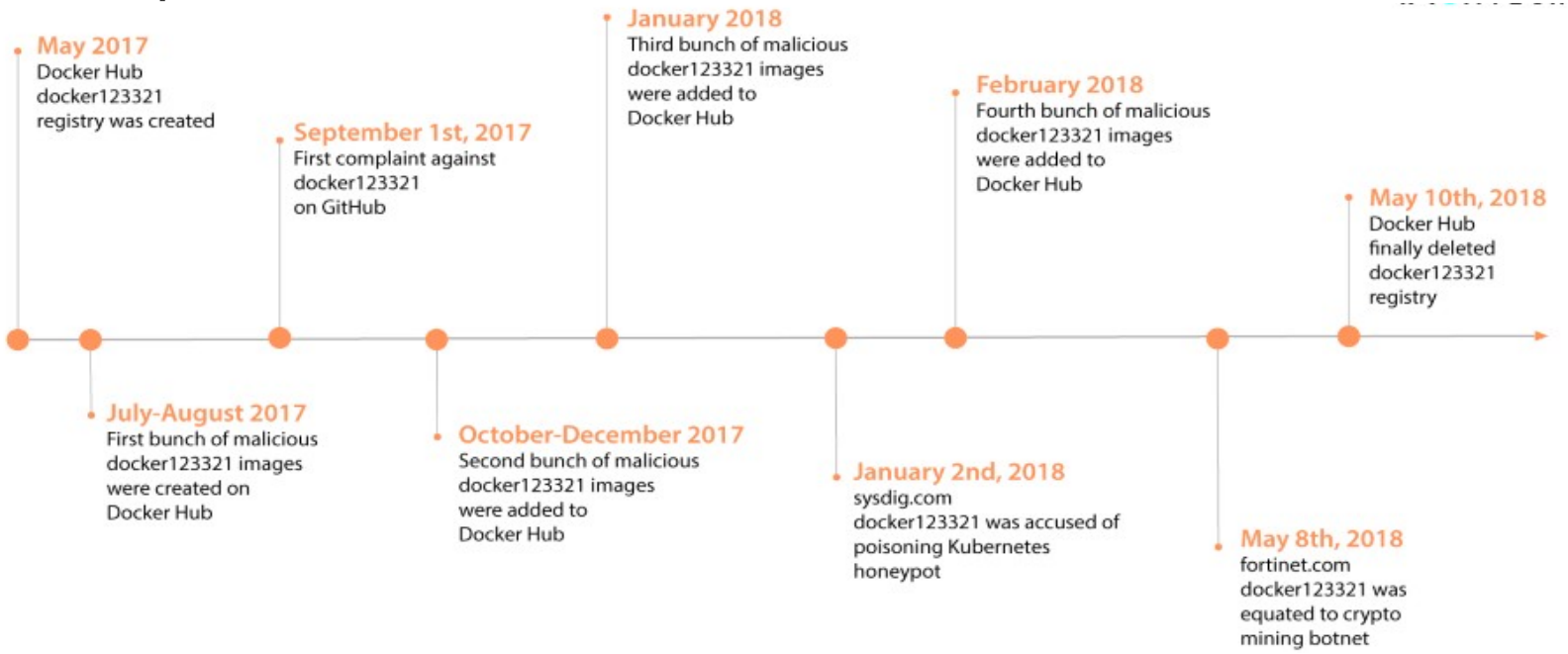
https://docs.docker.com/engine/reference/run/#user

4 Container Images and build file

- **4.2 Ensure that containers use only trusted base images**
  - Verifica el origen de las imágenes que usas como base
  - Siempre tiene que estar disponible el Dockerfile para poder auditarlo y reconstruir la imagen
  - No uses el tag default porque la imagen podría cambiar de forma automática (en una actualización)
  - Cuidado con usar comandos en el Dockerfile que instalen la última versión de paquetes, reconstrucciones de la imagen pueden ser diferentes (actualización de paquetes)

4 Container Images and build file

- **4.2 Ensure that containers use only trusted base images**
  - En el pasado se detectaron 17 imágenes en DockerHub con puerta trasera



https://www.bleepingcomputer.com/news/security/17-backdoored-docker-images-removed-from-docker-hub/

4 Container Images and build file

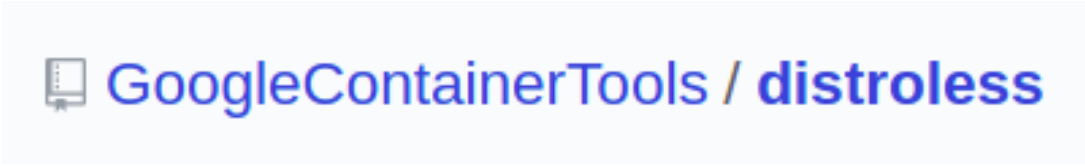
- **4.3 Ensure that unnecessary packages are not installed in the container**
  - Reduce la superficie de ataque a los contenedores lo más posible
  - Sólo instala los paquetes que necesitas
  - Considera usar imágenes base reducidas

4 Container Images and build file

- **Imágenes base con lo mínimo para ejecutar las aplicaciones**

https://alpinelinux.org/

https://hub.docker.com/_/alpine



https://github.com/GoogleContainerTools/distroless

SLIM

https://hub.docker.com/_/debian

MINIMAL

https://blog.ubuntu.com/2018/07/09/minimal-ubuntu-released

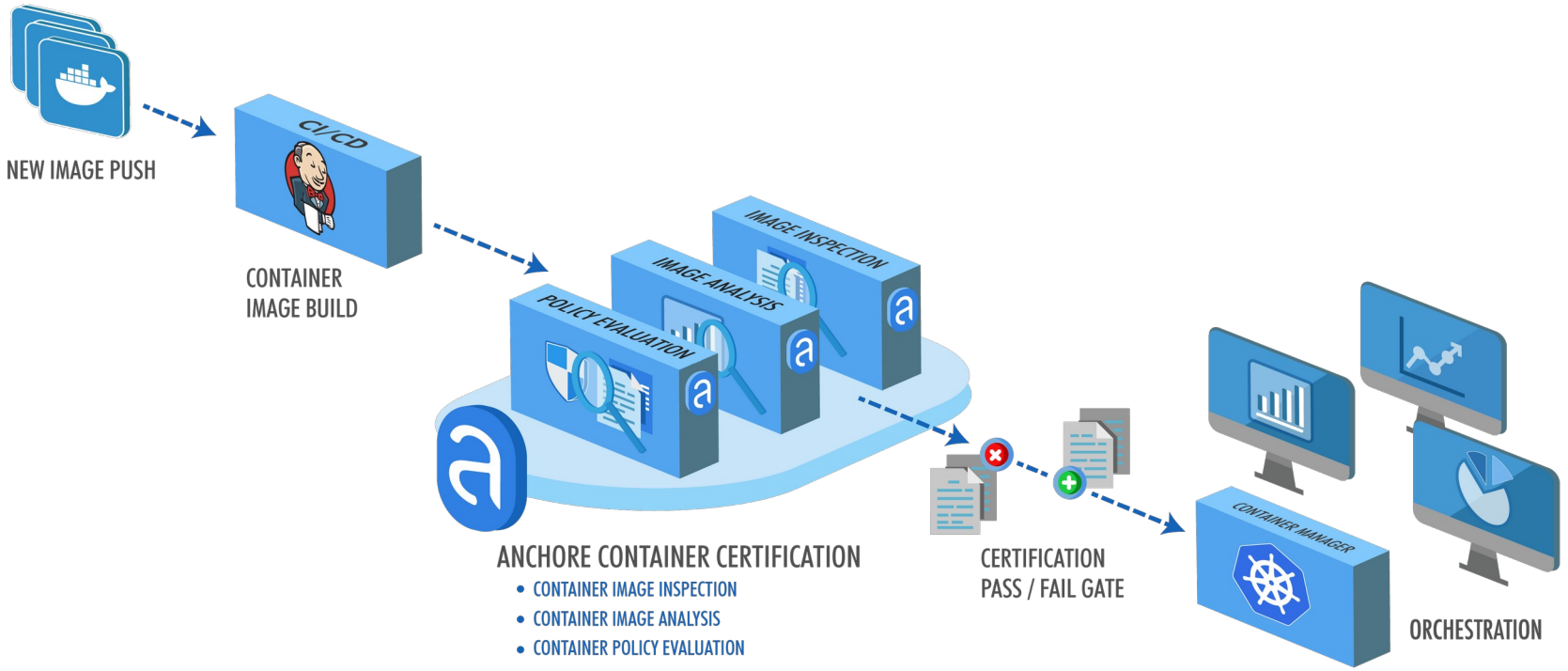
https://hub.docker.com/_/ubuntu

4 Container Images and build file

- **4.4 Ensure images are scanned and rebuilt to include security patches**
  - Constantemente se publican CVEs anunciando de nuevas vulnerabilidades en imágenes base, paquetes, runtimes de ejecución (Java, Python…)
  - Actualizar las versiones de estos elementos y regenerar la imagen habitualmente es una buena medida de permanecer seguro
  - No uses la caché de construcción **"docker build --no-cache"**

4 Container Images and build file

- Análisis estático de vulnerabilidades SAST en imágenes Docker



4 Container Images and build file

- **Análisis estático de vulnerabilidades SAST en imágenes Docker**

https://snyk.io/

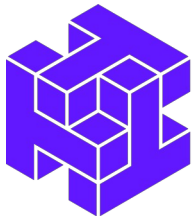
https://github.com/coreos/clair

https://anchore.com/

https://github.com/eliasgranderubio/dagda

4 Container Images and build file

- Análisis estático de vulnerabilidades SAST en imágenes Docker



Twistlock

https://www.twistlock.com/

aqua

https://www.aquasec.com/

JFrog Xray

https://jfrog.com/xray/

4 Container Images and build file

- Vulnerabilidades encontradas en una imagen

ubuntu 14.04 → Varias vulnerabilidades de las herramientas de ubuntu

vs

GoogleContainerTools / distroless → Sin vulnerabilidades

https://www.abhaybhargav.com/stories-of-my-experiments-with-distroless-containers/

4 Container Images and build file

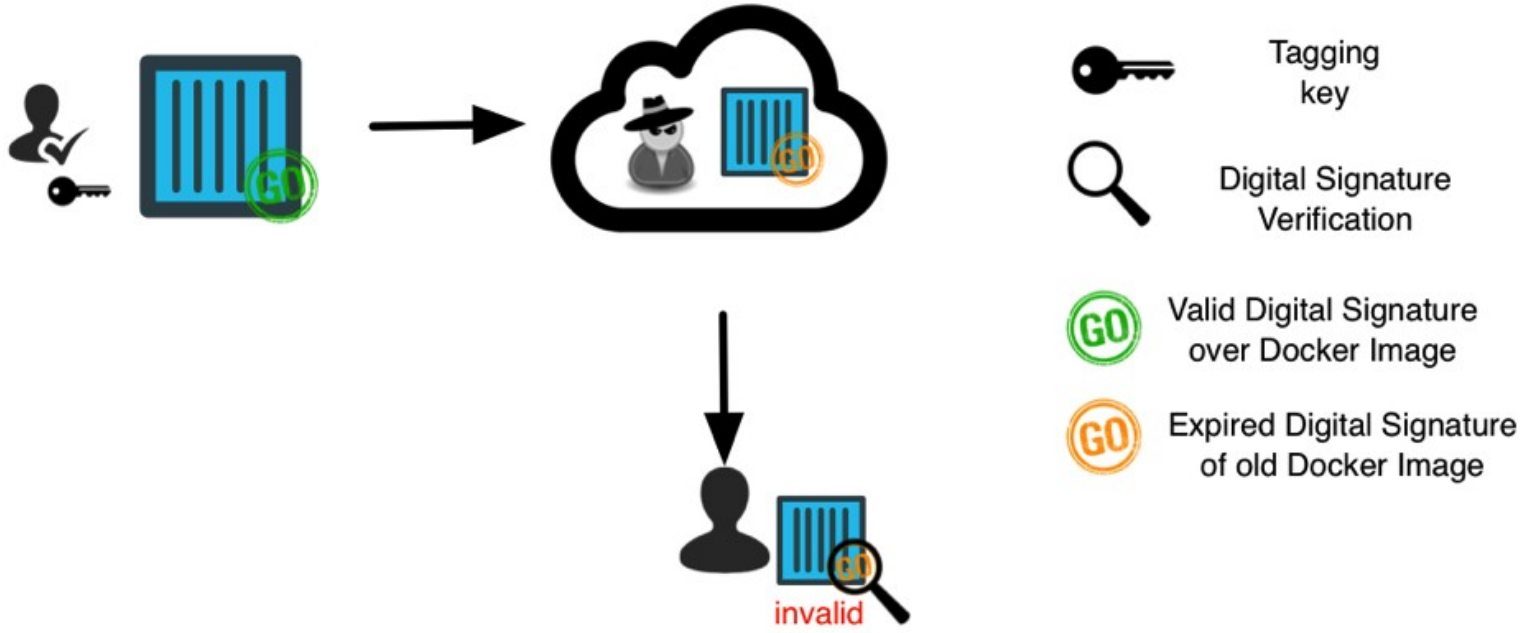
- **4.5 Ensure Content trust for Docker is Enabled**
  - **Docker Content Trust (DCT)** permite a docker verificar la integridad y el publisher de las imágenes que se descarga
  - Eso evita ataques de ***Man in the Middle*** entre el engine Docker y el registro
  - Un cliente con DCT habilitado **sólo puede hacer pull, run o build** de imágenes de confianza
  - Cada **tag** de una repository puede ser firmado (o no) por el publisher

```
export DOCKER_CONTENT_TRUST=1
```

https://docs.docker.com/engine/security/trust/

4 Container Images and build file

- 4.5 Ensure Content trust for Docker is Enabled



4 Container Images and build file

- **4.5 Ensure Content trust for Docker is Enabled**
  - Para usar DCT el registro de imágenes debe tener vinculado un **servidor Notary**

https://hub.docker.com/

https://github.com/notaryproject/notary

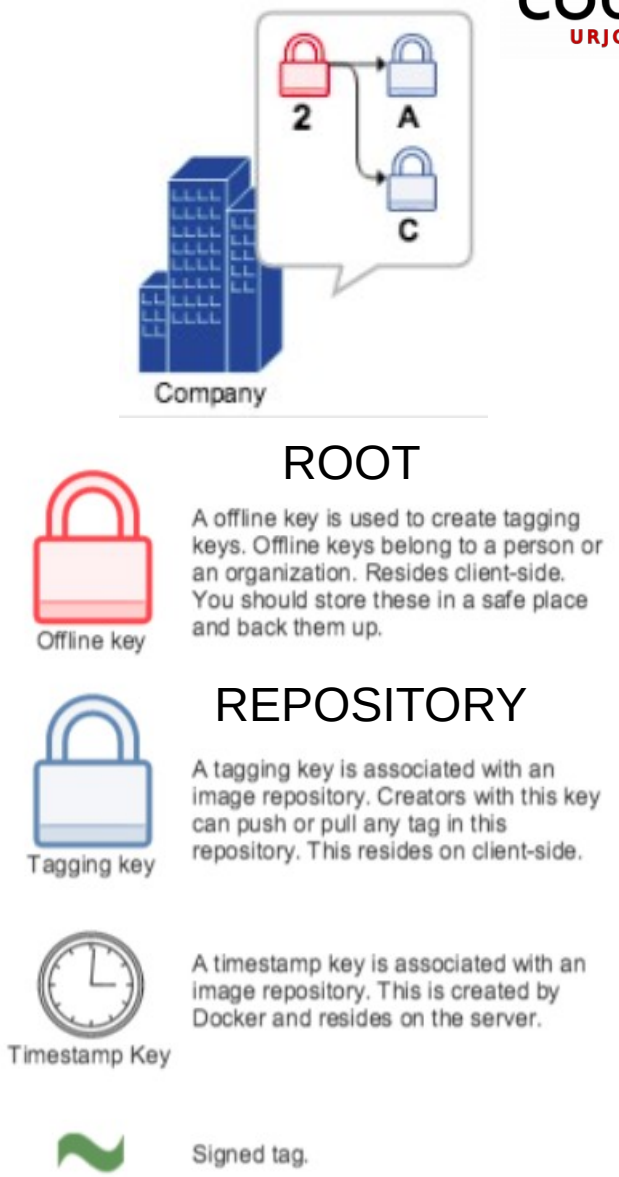
https://github.com/notaryproject/notaryproject

Notary

- Formado por un **cliente** y un **servidor**
- Sirve para **gestionar metadatos de confianza** (verifica que no han sido alterados)
- En los **metadatos** se guardan la lista de ficheros y sus hashes contenidos en las **imágenes Docker**
- El **cliente Docker** integra un cliente notary que **verifica** que los **ficheros** de la imagen concuerdan con los **metadatos** del servidor notary

Notary

- Para garantizar la seguridad, Notary usa **varias claves privadas**
- El esquema de claves permite **delegar** las operaciones de firma sin dar acceso a la clave maestra
- Permite que el servidor firme la **hora de generación**



https://docs.docker.com/engine/security/trust/

4 Container Images and build file

- **4.6 Ensure that HEALTHCHECK instructions have been added to container images**
  - Permite al engine de Docker saber si el servicio del contenedor está en buen estado.
  - Si no lo está, puede reiniciar el contenedor

```
HEALTHCHECK --interval=5m --timeout=3s \
  CMD curl -f http://localhost/ || exit 1
```

https://docs.docker.com/engine/reference/builder/#healthcheck

4 Container Images and build file

- **4.7 Ensure update instructions are not use alone in the Dockerfile**
  - Si usas el comando "apt-get update" o similar en su propio RUN, podría cachearse
  - Eso evita que se vuelva a ejecutar y, por tanto, que se actualicen las nuevas versiones de los paquetes
  - Además, aumenta el tamaño de la imagen innecesariamente

```
RUN apt-get update && apt-get install -y \
    curl \
    ruby1.9.1 \
 && rm -rf /var/lib/apt/lists/*
```

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#apt-get

4 Container Images and build file

- **4.9 Ensure that COPY is used instead of ADD in Dockerfiles**
  - ADD puede usarse con URLs, lo que podría copiar en la imagen contenido malicioso

https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#add-or-copy

- **4.10 Ensure secrets are not stored in Dockerfiles**
  - No guardes secretos en las imágenes, son fácilmente recuperables

https://avicoder.me/2016/07/22/Twitter-Vine-Source-code-dump/

4 Container Images and build file

- **4.11 Ensure only verified packages are are installed**
  - Instala únicamente paquetes verificados. Podrían ser modificados por un atacante

https://www.redhat.com/sysadmin/rpm-gpg-verify-packages

5 Container Runtime

- 5.1 Ensure that, if applicable, an AppArmor Profile is enabled
- 5.2 Ensure that, if applicable, SELinux security options are set
- 5.3 Ensure that Linux kernel capabilities are restricted within containers
- 5.4 Ensure that privileged containers are not used
- 5.5 Ensure sensitive host system directories are not mounted on containers
- 5.6 Ensure sshd is not run within containers
- 5.7 Ensure privileged ports are not mapped within containers
- 5.8 Ensure that only needed ports are open on the container
- 5.9 Ensure that the host's network namespace is not shared
- 5.10 Ensure that the memory usage for containers is limited
- 5.11 Ensure that CPU priority is set appropriately on containers
- 5.12 Ensure that the container's root filesystem is mounted as read only
- 5.13 Ensure that incoming container traffic is bound to a specific host interface
- 5.14 Ensure that the 'on-failure' container restart policy is set to '5'
- 5.15 Ensure that the host's process namespace is not shared
- 5.16 Ensure that the host's IPC namespace is not shared
- 5.17 Ensure that host devices are not directly exposed to containers
- ...

5 Container Runtime

- ...
- 5.18 Ensure that the default ulimit is overwritten at runtime if needed
- 5.19 Ensure mount propagation mode is not set to shared
- 5.20 Ensure that the host's UTS namespace is not shared
- 5.21 Ensure the default seccomp profile is not Disabled
- 5.22 Ensure that docker exec commands are not used with the privileged option
- 5.23 Ensure that docker exec commands are not used with the user=root option
- 5.24 Ensure that cgroup usage is confirmed
- 5.25 Ensure that the container is restricted from acquiring additional privileges
- 5.26 Ensure that container health is checked at runtime
- 5.27 Ensure that Docker commands always make use of the latest version of their
- 5.28 Ensure that the PIDs cgroup limit is used
- 5.29 Ensure that Docker's default bridge "dockero" is not used
- 5.30 Ensure that the host's user namespaces are not shared
- 5.31 Ensure that the Docker socket is not mounted inside any containers

5 Container Runtime

- **5.1 Ensure that, if applicable, an AppArmor Profile is enabled**
  - AppArmor es una forma efectiva de proteger un sistema Linux de amenazas conocidas
  - Usa el perfil por defecto de Docker o crea uno adaptado a tus necesidades

https://docs.docker.com/engine/security/apparmor/

5 Container Runtime

- **5.2 Ensure that, if applicable, SELinux security options are set**
  - Con SELinux se puede incrementar todavía más la seguridad
  - No viene activado por defecto

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html/container_security_guide/docker_selinux_security_policy

5 Container Runtime

- **5.3 Ensure that Linux kernel capabilities are restricted within containers**
- **5.4 Ensure that privileged containers are not used**
- **5.5 Ensure sensitive host system directories are not mounted on containers**
  - Para evitar que información sensible del host pueda ser accesible a un contenedor malicioso
- **5.6 Ensure sshd is not run within containers**
  - No es necesario y aumenta la superficie de ataque

https://jpetazzo.github.io/2014/06/23/docker-ssh-considered-evil/

5 Container Runtime

- **5.7 Ensure privileged ports are not mapped within containers**
  - En general no mapear puertos de los contenedores a puertos privilegiados (por debajo de 1024)
  - Salvo el 80 (HTTP) y el 443 (HTTPS), los demás puertos privilegiados podrían ser usados por un atacante para suplantar servicios del sistema como un demonio SSH y obtener información sensible

https://docs.docker.com/network/

5 Container Runtime

- **5.8 Ensure that only needed ports are open on the container**
  - Indica en el EXPOSE y en la doc sólo aquellos puertos relevantes
- **5.9 Ensure that the host's network namespace is not shared**
  - El contenedor tendrá acceso completo al stack de red del host
  - Salvo que sea estrictamente necesario, no uses --net=host

https://docs.docker.com/network/

5 Container Runtime

- **5.10 Ensure that the memory usage for containers is limited**
  - Por defecto la memoria no está limitada, y un uso desmesurado podría producir un ataque por denegación de servicio

```
$ docker run -d --memory 256m centos sleep 1000
```

- **5.11 Ensure that CPU priority is set appropriately on containers**
  - Por defecto la CPU no está limitada y un uso desmesurado podría provocar denegación de servicio

```
$ docker run -d --cpu-shares 512 centos sleep 1000
```

https://docs.docker.com/config/containers/resource_constraints

5 Container Runtime

- **5.12 Ensure that the container's root filesystem is mounted as read only**
  - Se limitan los cambios que se pueden hacer en un contenedor
  - Para escribir se puede usar un volumen diferente
  - Se reduce la posibilidad de un ataque con sustitución de binarios

```
$ docker run --interactive --tty --read-only \
    -v /opt/app/data:/run/app/data:rw centos /bin/bash
```

https://docs.docker.com/engine/reference/commandline/run/#mount-tmpfs---tmpfs

5 Container Runtime

- **5.25 Ensure that the container is restricted from acquiring additional privileges**
  - Limita la capacidad de un contenedor de adquirir nuevos privilegios

```
$ docker run --rm -it --security-opt=no-new-privileges ubuntu bash
```

- **5.27 Ensure that Docker commands always make use of the latest version of their image**
  - Evita el uso del tag "latest" porque el cliente no se descarga la última versión si ya existe el tag en local

5 Container Runtime

- **5.28 Ensure that the PIDs cgroup limit is used**
  - Limita el número de procesos nuevos que se pueden crear en un contenedor para evitar ataques de denegación de servicio

```
$ docker run -it --pids-limit 100 <Image_ID>
```

- **5.29 Ensure that Docker's default bridge "dockero" is not used**
  - Por defecto los contenedores se conectan al interfaz dockero
  - Este modo de red es vulnerable a ARP spoofing y MAC flooding ya que no se aplican filtros
  - Usa redes definidas por el usuario

https://docs.docker.com/network/bridge/

5 Container Runtime

- **5.30 Ensure that the host's user namespaces are not shared**
  - Por defecto root en el contenedor es root en el host
  - Usando el user namespace se evita y se crea un usuario en el host aunque el contenedor sea root

https://docs.docker.com/engine/security/userns-remap/

- **5.31 Ensure that the Docker socket is not mounted inside any containers**
  - Un contenedor con acceso al Docker engine tiene permisos de root en el host

https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-socket-option

6 Docker Security Operations

- 6.1 Ensure that image sprawl is avoided
- 6.2 Ensure that container sprawl is avoided

6 Docker Security Operations

- **6.1 Ensure that image sprawl is avoided**
- **6.2 Ensure that container sprawl is avoided**
  - No mantengas imágenes con tags antiguos en el host
  - Podrían tener vulnerabildades y ser atacadas
  - Podría llenar el disco y evitar el funcionamiento
  - Prune
    - $ docker image prune > Borra imágenes dangling
    - $ docker image prune -a > Borra imágenes sin contenedor
    - $ docker container prune > Borra contenedores parados
    - $ docker volume prune > Borra contenedores no usados
    - $ docker network prune > Borrar redes no usadas

https://docs.docker.com/config/pruning/

CIS Docker Benchmark

- Herramienta que **verifica de forma automática** múltiples elementos del **CIS Docker Benchmark**
- También incluye **verificaciones propias**
- Para analizar los items referidos a **contenedores**, se tienen que estar **ejecutando** en el momento de la ejecución

https://github.com/docker/docker-bench-security

CIS Docker Benchmark

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.4
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Community Edition Benchmark v1.1.0.
# ------------------------------------------------------------------------------

Initializing Wed May 15 22:14:52 UTC 2019


[INFO] 1 - Host Configuration
[WARN] 1.1  - Ensure a separate partition for containers has been created
[NOTE] 1.2  - Ensure the container host has been Hardened
[INFO] 1.3  - Ensure Docker is up to date
[INFO]      * Using 18.09.6, verify is it up to date as deemed necessary
[INFO]      * Your operating system vendor may provide support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:999:mica
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[INFO] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO]      * File not found
[INFO] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]      * File not found
[WARN] 1.10  - Ensure auditing is configured for Docker files and directories - /etc/default/docker
```

Índice

1. Seguridad en Docker
2. CIS Docker Benchmark
3. **Dockerfile linters**

Dockerfile linters

- Un **Dockerfile** define cómo crear una imagen, con la que se ejecutará un contenedor
- Muchas **buenas prácticas de seguridad** se refieren a cómo debe ser el **Dockerfile** para evitar (o mitigar) potenciales ataques
- Existen herramientas que verifican el fichero Dockerfile y **reportan malas prácticas**

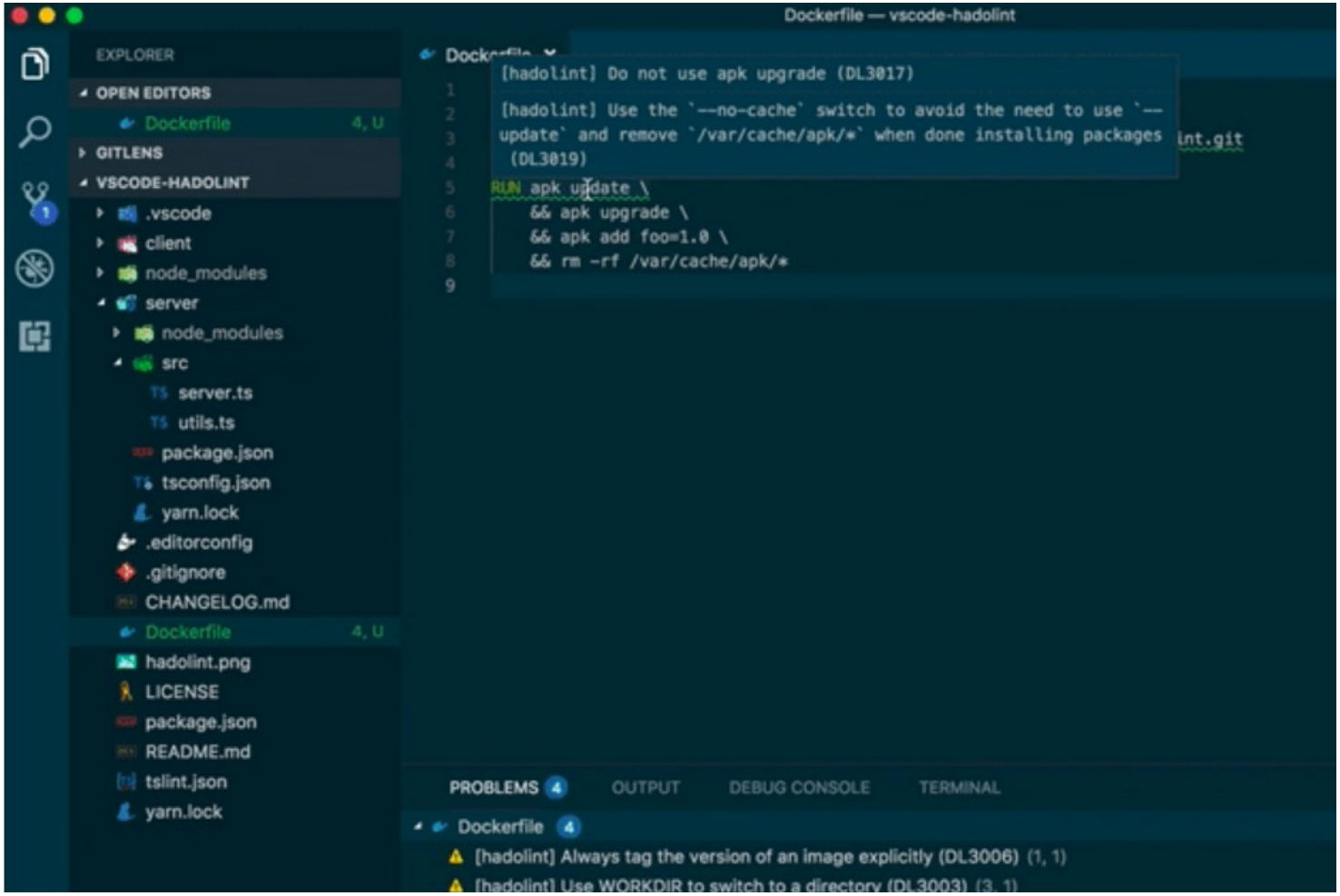
Dockerfile linters

- Hadolint

```
DL4000 Specify a maintainer of the Dockerfile
DL3006 Always tag the version of an image explicitely.
1 FROM debian
SC1007 Remove space after = if trying to assign a value (for empty string, use var='' ... ).
SC2154 node_verson is referenced but not assigned.
DL3009 Delete the apt-get lists after installing something
2 RUN node_version= "0.10" \
3  && apt-get update && apt-get -y install nodejs="$node_verson"
4 COPY package.json usr/src/app
DL3003 Use WORKDIR to switch to a directory
5 RUN cd /usr/src/app \
6  && npm install node-static
7
DL3011 Valid UNIX ports range from 0 to 65535
8 EXPOSE 80000
9 CMD ["npm", "start"]
```

https://github.com/hadolint/hadolint

Hadolint



https://github.com/hadolint/hadolint/blob/master/docs/INTEGRATION.md

Dockerfile linters

Dockerfile Linter

A smarter Dockerfile linter that helps you build best practice Docker images. The linter is parsing the Dockerfile into an AST and performs rules on top of the AST. It additionally is using the famous Shellcheck to lint the Bash code inside RUN instructions. Please help me improve the linter with your suggestions.

```
1 FROM debian
2 RUN export node_version="0.10" \
3 && apt-get update && apt-get -y install nodejs="$node_verion"
4 COPY package.json usr/src/app
5 RUN cd /usr/src/app \
6 && npm install node-static
7
8 EXPOSE 80000
9 CMD ["npm", "start"]
```

Lint Clear

https://hadolint.github.io/hadolint/