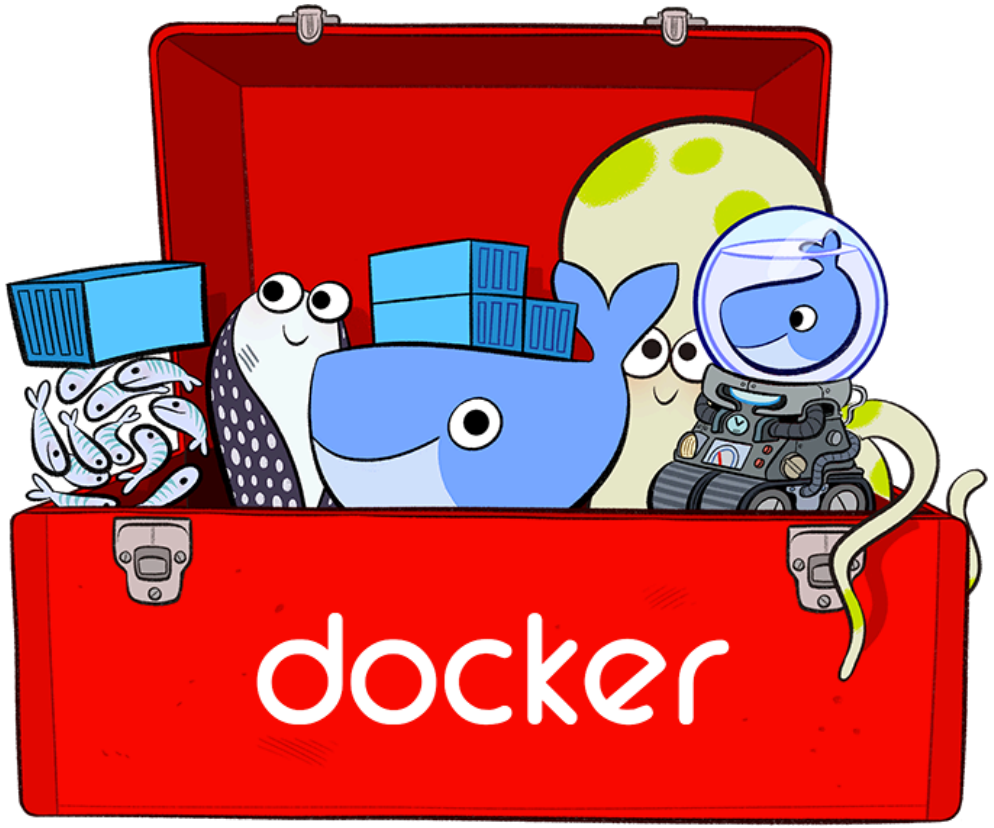
Desarrollo de Código Seguro

Docker (extended)

viu | Universidad Internacional de Valencia

- Repaso
- Docker Compose
- Persistencia
- Redes
- Contexto
- Redes
- Buenas practicas de seguridad



Repaso

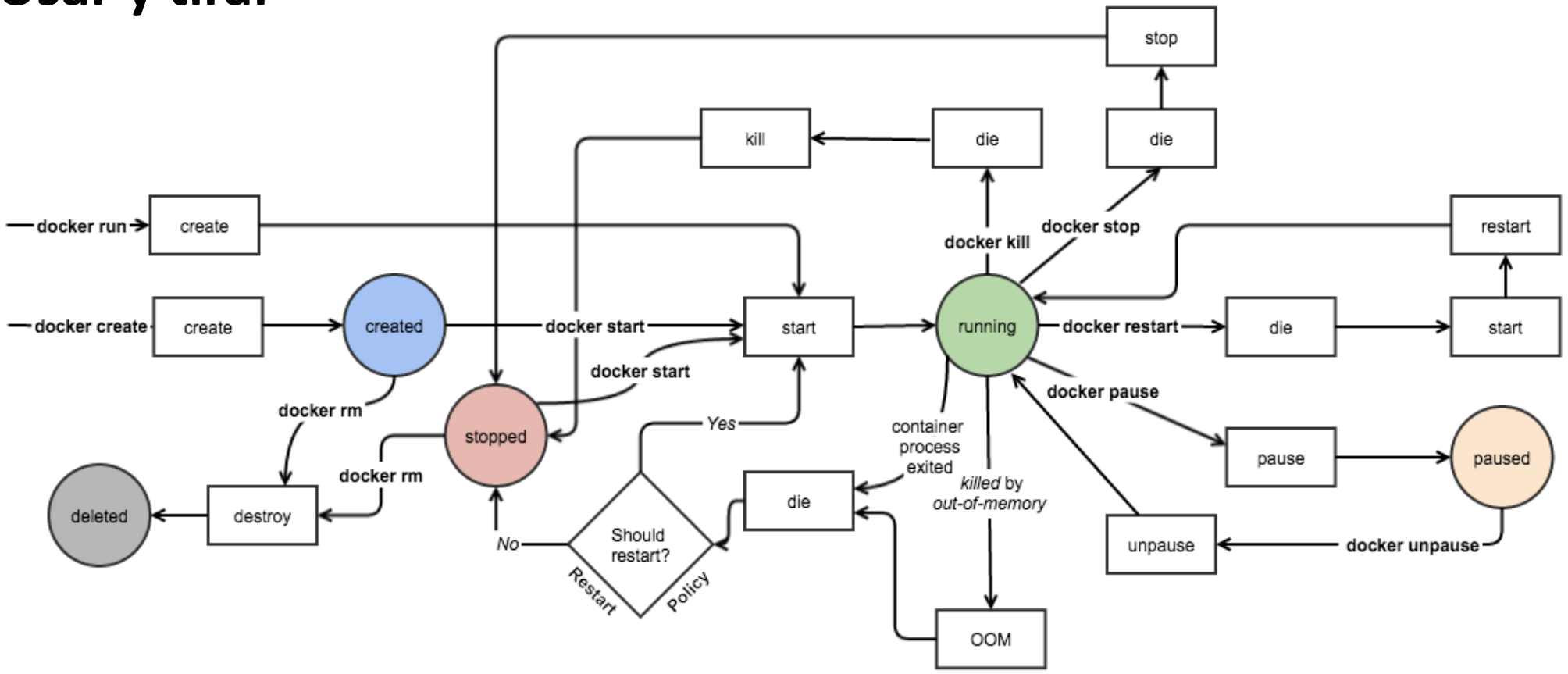
Usar y tirar

Ciclo de vida de un contenedor

- *Created*
- *Restarting*
- *Running*
- *Removing*
- *Paused*
- *Exited*
- *Dead*

- *docker container create*
- *docker container run*
- *docker container start*
- *docker container pause*
- *docker container unpause*
- *docker container stop*
- *docker container restart*
- *docker container kill*

Usar y tirar


stop
kill
die
die
docker run
create
docker kill
docker stop
restart
docker create
create
created
docker start
start
running
docker restart
die
start
docker start
docker rm
docker pause
stopped
Yes
container process exited
killed by out-of-memory
pause
paused
deleted
destroy
docker rm
die
No
Should restart?
Restart Policy
unpause
docker unpause
OOM

Configurando contenedores

*$ docker container run **--name my-web-server** -d nginx*

$ docker container run -it --hostname=my-server ubuntu

$ docker container rename wizardly_minsky my-server

$ docker container run --dns=8.8.8.8 -it ubuntu

$ docker container run --add-host=NOMINAS_DB:10.23.0.5 -it ubuntu

Interactuando con los contenedores

$ docker container exec my-nginx cat /etc/nginx/nginx.conf

$ docker container exec -it my-nginx /bin/bash
root@5db80718c041:/#

$ docker container cp my-nginx:/etc/nginx/nginx.conf .

$ docker container export -o fs.tar my-nginx

$ docker container attach my-nginx

Capas

Layers (capas)

$ docker image history

Etiquetando imágenes

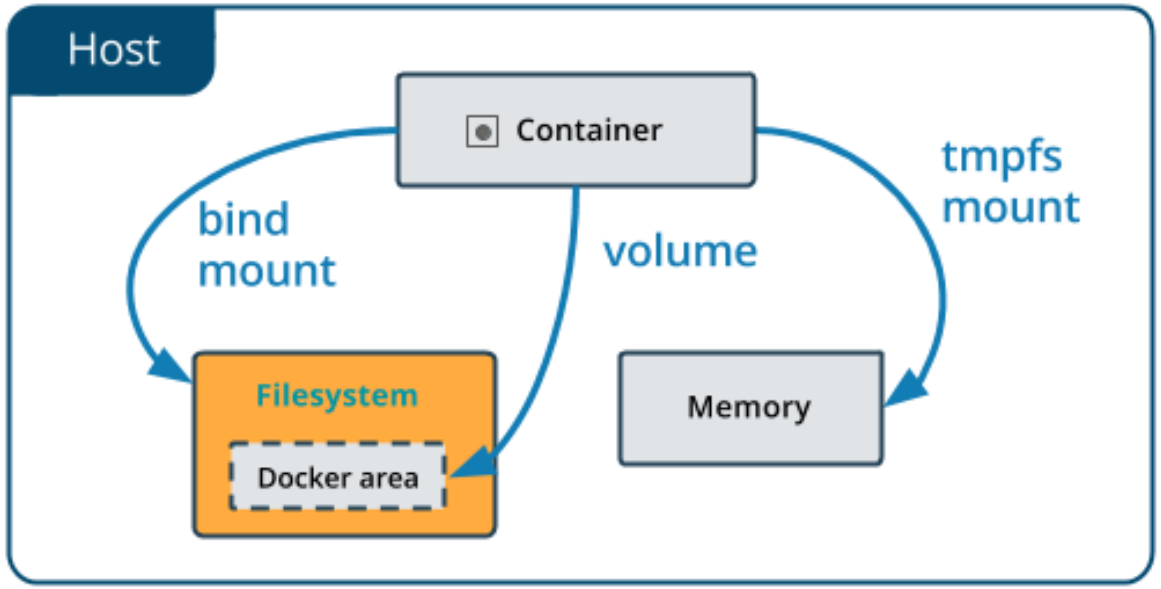
docker image build -t

docker image tag

La importancia de etiquetar correctamente una imagen

Persistencia

Persistencia de los datos



docker container diff

Compartir ficheros del host

```
$ docker run -d -it --mount
type=bind,source="$(pwd)"/fuente,target=/app nginx

$ docker run -d -it --mount
type=bind,source="$(pwd)"/fuente,target=/app, readonly nginx

$ docker run -d -it \
--mount type=bind,source="$(pwd)"/fuente,target=/app, readonly \
--mount type=bind,source="$(pwd)"/fuente2,target=/app2 \
nginx
```

Acceso directo a memoria

$ docker run -d -it --mount type=tmpfs,destination=/logs nginx

Volúmenes

$ docker volume create myvol

$ docker container run -d --mount=type=volume,source=myvol,destination=/app ubuntu

$ docker container run -d --mount=type=volume,source=myvol2,destination=/app ubuntu

Limpieza

Contenedores

$ docker container rm -f my-nginx

$ docker container prune

$ docker container run -it --rm --name my-nginx -p 80:80 nginx

Limpieza

Volúmenes

$ docker volume prune

$ docker volume rm myvol

Contexto

Contexto

- Permite interactuar con otros demonios docker u orquestradores como Swarm o Kubernetes
- SSH con clave pública sólo. No passwords
- Útil para probar y/o ejecutar contenedores en otros entornos
- DOCKER_HOST vs DOCKER_CONTEXT

docker context create nombre --docker "host=ssh://usuario@host:puerto”

docker context create k8s-test --default-stack-orchestrator=kubernetes --kubernetes config-file=/home/ubuntu/.kube/config --docker host=unix:///var/run/docker.sock

Redes

Redes

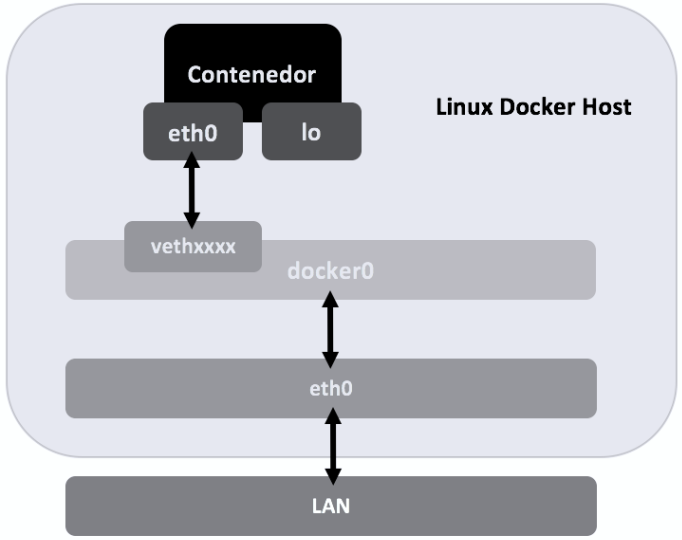
El demonio *Docker* crea por defecto, cada vez que se ejecuta, una interfaz de red virtual *Ethernet* (tipo *bridge*) con el nombre *docker0* y la dirección *IP* 172.17.0.1/24

```
$ docker network ls

NETWORK ID      NAME      DRIVER      SCOPE
feeb67c2f590    bridge    bridge      local
8b5e3e4d5598    host      host        local
5bc062cb719a    none      null        local
```

Redes

Cuando se crea un contenedor, *Docker* también crean dos *interfaces* de red virtuales asociadas al mismo



Redes

- *--net default. La interface bridge definida por defecto será la encargada de conectar entre sí a los contenedores.*
- *--net=none. Crea un contenedor sin ninguna configuración de red.*
- *--net=container1:container2. Esta opción provoca que el container1 comparta su network namespace (este concepto lo veremos más adelante) con el container2.*
- --net=host. En este caso se comparte el network namespace del contenedor con el host.

Redes

$ docker network inspect bridge


```
[
    {
        "Name": "bridge",
        "Id": "feeb67c2f5907bca5eedaed0cbbc9bb9c492d4a4a090459266c2a6377188549d",
        "Created": "2018-01-06T01:40:35.382791057+01:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "abc0ee8ac5c029aaf1c80485a897b25a35393c9902ba510497f28d2f6d2c0f3c": {
                "Name": "unruffled_cray",
                "EndpointID": "890d15933d0d11d296cf88df23a4500307be5a45c14fc0dc5a0731a8ed3475a3",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
```

Redes

docker container inspect ID

```
....
"Networks": {
            "bridge": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "NetworkID":
"feeb67c2f5907bca5eedaed0cbbc9bb9c492d4a4a090459266c2a6377188549d",
                "EndpointID":
"890d15933d0d11d296cf88df23a4500307be5a45c14fc0dc5a0731a8ed3475a3",
                "Gateway": "172.17.0.1",
                "IPAddress": "172.17.0.2",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "MacAddress": "02:42:ac:11:00:02",
                "DriverOpts": null
            }
```

Redes

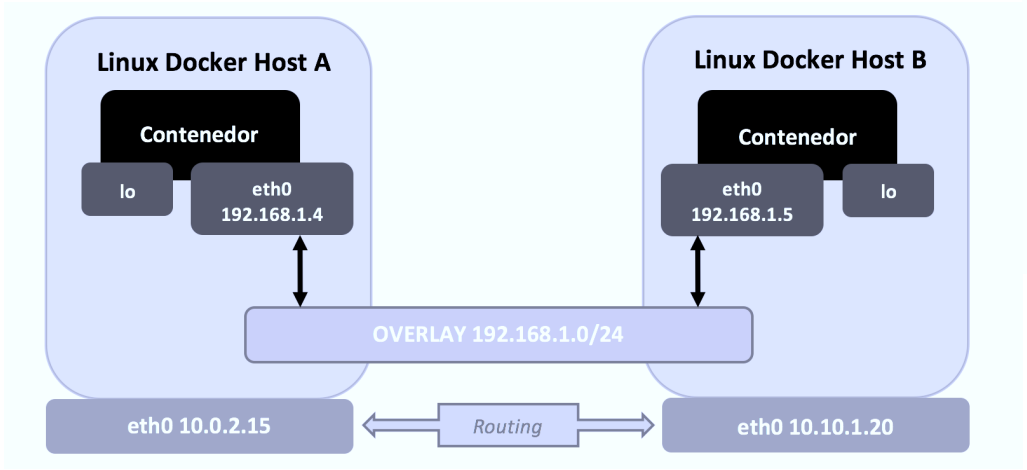
Tipos de redes Docker

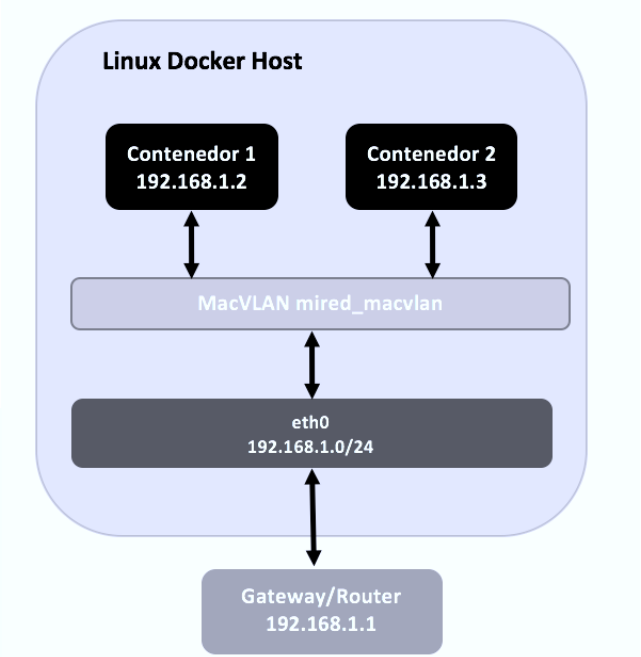
- Bridge
- Host
- None

Creación de Redes

Drivers:

- Overlay
- MacVLAN





Enlazando (link) contenedores

Permite conectar contenedores *Docker* y así poder intercambiar información entre ellos.

```
$ docker run -it --name apache3 --link apache2:alias_apache2 httpd /bin/bash
root@980a591bb8ff:/usr/local/apache2# cat /etc/hosts
127.0.0.1     localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
172.17.0.2   alias_apache2 f077908a13b8 apache2
172.17.0.3   980a591bb8ff
```

Docker Compose

Compose

docker-compose.yml

- Permite la creación de entornos aislados en un mismo *host*.
- Preserva los volúmenes de datos cuando los contenedores son creados.
- Sólo se recrean los contenedores que han cambiado, a menos que se fuerza la recreación de los mismos.
- Permite el uso de variables para la configuración de entornos

Compose

docker-compose.yml

```
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
  redis:
    image: "redis:alpine"
```

docker-compose up

docker-compose ps

docker-compose exec web env

docker-compose run web env

Compose. Variables de entorno

Compose permite la declaración de variables de entorno en un fichero llamado *.env*, el cual debe estar localizado en el mismo directorio desde donde se ejecuta *docker-compose*

Las variables definidas en *.env* sólo están disponibles en tiempo de ejecución de *docker-compose*, es decir, dichas variables no estarán disponibles dentro del contenedor.

- COMPOSE_API_VERSION
- COMPOSE_CONVERT_WINDOWS_PATHS
- COMPOSE_FILE
- COMPOSE_HTTP_TIMEOUT
- COMPOSE_TLS_VERSION
- COMPOSE_PROJECT_NAME
- DOCKER_CERT_PATH
- DOCKER_HOST
- DOCKER_TLS_VERIFY

Buenas prácticas de seguridad

Ejecución de contenedores con usuario no root

Los contenedores Docker se ejecutan por defecto con el usuario root y por ello, se dispone de privilegios de root dentro del contenedor

Es una buena práctica el definir un usuario no root en el fichero Dockerfile siempre que sea posible.

Eliminación de los permisos de *setuid* y *setgid*

setuid y ***setgid*** podrían ser utilizados para elevar privilegios usando ataques de elevación de privilegios.

Se recomienda eliminarlos de la imagen

Es aconsejable eliminarlos durante la fase de la construcción de la imagen docker del mismo modo que sucedía con la eliminación de usuarios añadiendo el siguiente comando en el Dockerfile (preferiblemente al final del mismo):

```
RUN find / -perm +6000 -type f -exec chmod a-s {} \; || true
```

COPY en vez de ADD en el Dockerfile

ADD podría descargar ficheros desde URLs remotas y además ejecutar operaciones de desempaquetar o descomprimir, añadiendo estos ficheros no comprobados en el sistema.

ADD sólo puede descomprimir ficheros locales (descargados previamente con ADD o de alguna otra forma)

No almacenar secretos en el Dockerfile

Los ficheros *Dockerfile* son fácilmente accesibles. Listar su contenido es una tarea trivial que no está restringida. De hecho, este contenido debe de ser siempre accesible para demostrar confiabilidad en los contenedores que se van a ejecutar.

```
docker history <Image_ID>
```

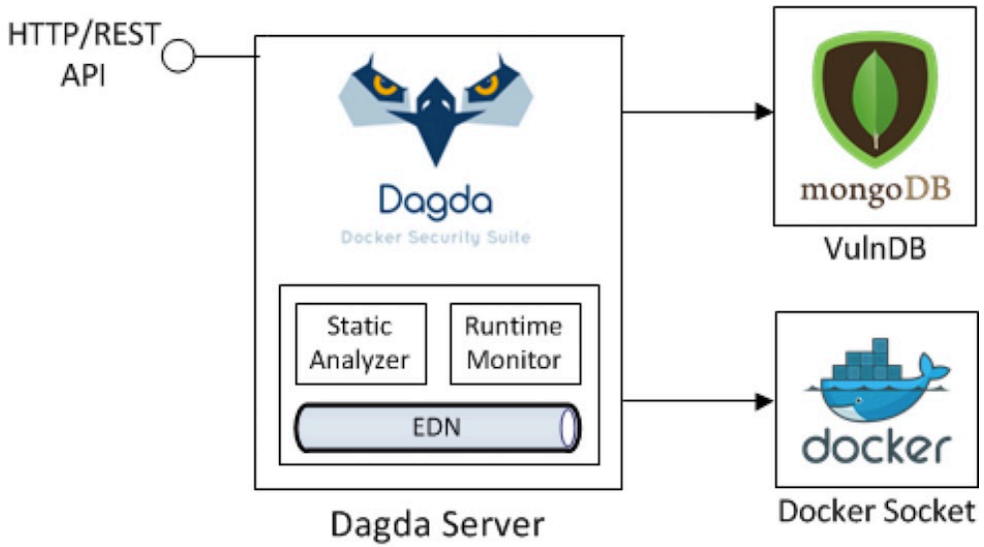
Dagda

https://github.com/eliasgranderubio/dagda

Clair (CoreOS)
https://github.com/coreos/clair

Anchore
https://anchore.io/

Twislock
Aporeto
…



- **Asegúrate que AppArmor está habilitado**
- Docker viene con un perfil ya definido. Docker 1.13.0 o anterior:

  /etc/apparmor.d/docker

  En versiones posteriores Docker genera el perfil en un tmpfs, lo carga en AppArmor y lo borra.

Docker aplica por defecto el perfil docker-default

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}:
AppArmorProfile={{ .AppArmorProfile }}'
```

Asegúrate que SELinux está habilitado

- Por defecto SELinux no se aplica a contenedores

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}:
SecurityOpt={{ .HostConfig.SecurityOpt }}'
```

```
docker run -it --security-opt label=level:TopSecret centos /bin/bash
```

Asegúrate que Linux Kernel Capabilities están restringidas

docker run -it --cap-add={"NET_ADMIN","SYS_ADMIN"} centos /bin/bash

docker run -it --cap-add={"SETUID","SETGID "} centos /bin/bash

docker run -it --cap-drop=all --cap-add={"NET_ADMIN","SYS_ADMIN"} centos /bin/bash

Capability Key	Capability Description
SETPCAP	Modify process capabilities.
MKNOD	Create special files using mknod(2).
AUDIT_WRITE	Write records to kernel auditing log.
CHOWN	Make arbitrary changes to file UIDs and GIDs (see chown(2)).
NET_RAW	Use RAW and PACKET sockets.
DAC_OVERRIDE	Bypass file read, write, and execute permission checks.
FOWNER	Bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file.
FSETID	Don't clear set-user-ID and set-group-ID permission bits when a file is modified.
KILL	Bypass permission checks for sending signals.
SETGID	Make arbitrary manipulations of process GIDs and supplementary GID list.
SETUID	Make arbitrary manipulations of process UIDs.
NET_BIND_SERVICE	Bind a socket to internet domain privileged ports (port numbers less than 1024).
SYS_CHROOT	Use chroot(2), change root directory.
SETFCAP	Set file capabilities.

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: CapAdd={{ .HostConfig.CapAdd }} CapDrop={{ .HostConfig.CapDrop }}'

Capacidades no asignadas por defecto

Capability Key	Capability Description
SYS_MODULE	Load and unload kernel modules.
SYS_RAWIO	Perform I/O port operations (iopl(2) and ioperm(2)).
SYS_PACCT	Use acct(2), switch process accounting on or off.
SYS_ADMIN	Perform a range of system administration operations.
SYS_NICE	Raise process nice value (nice(2), setpriority(2)) and change the nice value for arbitrary processes.
SYS_RESOURCE	Override resource Limits.
SYS_TIME	Set system clock (settimeofday(2), stime(2), adjtimex(2)); set real-time (hardware) clock.
SYS_TTY_CONFIG	Use vhangup(2); employ various privileged ioctl(2) operations on virtual terminals.
AUDIT_CONTROL	Enable and disable kernel auditing; change auditing filter rules; retrieve auditing status and filtering rules.
MAC_ADMIN	Allow MAC configuration or state changes. Implemented for the Smack LSM.
MAC_OVERRIDE	Override Mandatory Access Control (MAC). Implemented for the Smack Linux Security Module (LSM).
NET_ADMIN	Perform various network-related operations.
SYSLOG	Perform privileged syslog(2) operations.
DAC_READ_SEARCH	Bypass file read permission checks and directory read and execute permission checks.
LINUX_IMMUTABLE	Set the FS_APPEND_FL and FS_IMMUTABLE_FL i-node flags.
NET_BROADCAST	Make socket broadcasts, and listen to multicasts.
IPC_LOCK	Lock memory (mlock(2), mlockall(2), mmap(2), shmctl(2)).
IPC_OWNER	Bypass permission checks for operations on System V IPC objects.
SYS_PTRACE	Trace arbitrary processes using ptrace(2).
SYS_BOOT	Use reboot(2) and kexec_load(2), reboot and load a new kernel for later execution.
LEASE	Establish leases on arbitrary files (see fcntl(2)).
WAKE_ALARM	Trigger something that will wake up the system.
BLOCK_SUSPEND	Employ features that can block system suspend.

Evita el uso de contenedores privilegiados

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: Privileged={{ .HostConfig.Privileged }}’

docker run -it --privileged …

No permitas el montaje de directorios sensibles del sistema

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: Volumes={{ .Mounts }}’

No permitas SSH dentro de un contenedor

docker exec *nombre_o_id_contenedor* ps -ef

Evita el mapeo de puertos privilegiados y que sólo se abran los puertos necesarios

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: Ports={{ .NetworkSettings.Ports }}’

Asegúrate que el contenedor no comparte el espacio de red del host

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}: NetworkMode={{
.HostConfig.NetworkMode }}'
```

Limita el uso de la memoria y la CPU

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: Memory={{ .HostConfig.Memory }}’

docker run -it --memory 256m ….

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: CpuShares={{ .HostConfig.NanoCpus }}’

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: CpuShares={{ .HostConfig.CpuShares }}'

Monta el sistema de ficheros raíz del contenedor como sólo lectura

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}:
ReadonlyRootfs={{ .HostConfig.ReadonlyRootfs }}'
```

```
docker run --read-only -it redis
```

```
docker run --read-only -it --mount type=tmpfs,destination=/var/cache/nginx/ --
mount=type=volume,source=myvol,destination=/var/run/ nginx
```

Establece la política de reinicio del contenedor

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: RestartPolicyName={{ .HostConfig.RestartPolicy.Name }} MaximumRetryCount={{ .HostConfig.RestartPolicy.MaximumRetryCount }}'

docker run -d --restart=on-failure:5 nginx

Opciones:
no (valor por defecto)
on-failure (código de salida distinto de 0)
unless-stopped (a menos que haya sido explícitamente parado)
always

No compartas el espacio de procesos con el host

docker container inspect $(docker container ls -aq) --format '{{ .Id }}: PidMode={{ .HostConfig.PidMode }}’

docker run -d —pid=host ubuntu bash

docker run --pid=host ubuntu strace -p 1234

No expongas directamente los dispositivos del host

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}: Devices={{
.HostConfig.Devices }}'
```

```
docker run -it --device=/dev/tty0:/dev/tty0:rwm ubuntu bash
```

r: read

w: write

m: mknod

Asegúrate que *exec* no es abusado

docker container exec --user root *contenedor* ….

docker container exec --privileged *contenedor* ….

ps -ef | grep docker | grep exec | grep privileged

ps -ef | grep docker | grep exec | grep user

Prevén que los contenedores adquieran privilegios nuevos

docker container inspect $(docker container ls -aq) --format '{{ .Id }}:SecurityOpt={{ .HostConfig.SecurityOpt }}'

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

```
--security-opt="label=user:USER"     : Set the label user for the container

--security-opt="label=role:ROLE"     : Set the label role for the container

--security-opt="label=type:TYPE"     : Set the label type for the container

--security-opt="label=level:LEVEL"   : Set the label level for the container

--security-opt="label=disable"       : Turn off label confinement for the container

--security-opt="apparmor=PROFILE"    : Set the apparmor profile to be applied to the container

--security-opt="no-new-privileges:true|false"   : Disable/enable container processes from gaining new privileges

--security-opt="seccomp=unconfined"  : Turn off seccomp confinement for the container

--security-opt="seccomp=profile.json": White listed syscalls seccomp Json file to be used as a seccomp filter
```

Comprueba que HealthCheck esté establecido

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}:Health={{ .State.Health.Status }}'
```

```
docker container run -d --health-cmd='stat /etc/passwd || exit 1' nginx
```


viu Universidad Internacional de Valencia

Asegúrate que el espacio de usuarios del host no es compartido

```
docker container inspect $(docker container ls -aq) --format '{{ .Id }}:Health={{ .State.Health.Status }}'
```

```
docker run --rm -it --userns=host ubuntu bash
```

Docker bench security

```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.3
#
# 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 Fri Jul 14 09:18:42 UTC 2017


[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
[PASS] 1.3  - Ensure Docker is up to date
[INFO]      * Using 17.06.0 which is current
[INFO]      * Check with your operating system vendor for support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:992:vagrant
[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
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]      * File not found
[INFO] 1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO]      * File not found
[INFO] 1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-runc


[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
```

https://github.com/docker/docker-bench-security
https://www.cisecurity.org/cis-benchmarks/

Gracias

viu | Universidad Internacional de Valencia

2.7.2020